Decision Science Letters 3 (2014) 275–284

Contents lists available at GrowingScience

Decision Science Letters

homepage: www.GrowingScience.com/dsl

An application of TOPSIS method for task scheduling algorithm in grid computing environment

Sasan Kohzadian[a*], Ali Harounabadi[b] and Mehdi Sadeghzadeh[c]

[a]*Department of Computer Engineering, Science and Research branch, Islamic Azad University, Ilam, Iran*
[b]*Faculty Member, Department of Computer Science, Central Branch, Islamic Azad University, Tehran, Iran*
[c]*Faculty member, Department of Computer Science, Mahshahr Branch, Islamic Azad university, Mahshahr, Iran*

C H R O N I C L E

Article history:
Received October 15, 2013
Received in revised format
March 2 2014
Accepted April 17, 2014
Available online
April 23 2014

Keywords:
Grid System
Grid Scheduling
Multi Criteria Decision Making
Run Time
Load Balancing

A B S T R A C T

Today, the world facing with huge flood of data and the recent advances in computer technology have provided the capability to process significant amount of data. On the other hand, analyzing the information requires resources that most institutions do not have, independently. To handle such circumstances, grid computing has emerged as an important research area where the calculation of distributed computing and clustering are different. In this study, we propose a grid computing architecture as a set of protocols that use the cumulative knowledge of computers, networks, databases and scientific instruments based on the implementation of Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) technique. The results of the implementation of the proposed algorithm on grid systems indicate the superiority of the proposed approach in terms of validation criteria scheduling algorithms, such as task completion time and the performance compared with some alternative method.

1. Introduction

In the past, one of the biggest problems associated with complex calculations is associated with lack of appropriate hardware and software to perform the calculations in a reasonable amount of time. In many cases, it is possible that the code would not yield proper results because of existing bugs in codes. Applying supercomputers may reduce the burden of the computations but it cannot handle many scheduling problems. Therefore, one alternative is to handle the problem using the grid computing networks (Xueguang & Haigang, 2004; Plestys et al., 2007). In grid computing, many computational resources are shared by various networks of computers, geographically. Today, there is an increasing trend in solving problems using computing resources distributed strict geographical terms, called grid computing (Lan & Li, 2006; Wang et al., 2010; Yousif et al., 2011). The

* Corresponding author. Tel:+989183402029
E-mail addresses: s.kohzadian@yahoo.com (S. Kohzadian)

doi: 10.5267/j.dsl.2014.5.001

computational grid is a hardware and software infrastructure, which utilizes local resources. They provide the capability to enable the virtual enterprise to share and to integrate millions of geographically places. Grades are formed based on a set of heterogeneous resources, management systems, policies and requirements of various applications. Grid resources are heterogeneous and they are distributed, jointly. Thus, scheduling algorithms need changing workload and resources' availability to adapt the grid to achieve acceptable performance at the same time to observe the time limit. Scheduling various tasks on different servers is also considered as an interesting research work (Bouyer et al., 2008; Zhai, 2010).

According to Sarhadi and Meybodi (2009) and Shih et al. (2008), one of the main challenges in Grid computing is to select resources such as CPU-hours, network bandwidth, etc. appropriately for jobs in resource discovery phase. Since there is normally no centralized control and the dynamic/stochastic nature of resource availability, an appropriate selection mechanism needs to be highly distributed against the changes in the Grid environment. In addition, it is desirable to have a selection mechanism, which would not depend on the availability of coherent global information.

Sarhadi and Meybodi (2009) studied a minimalist decentralized method for resource selection in a simplified Grid-like environment. They considered a system consisting of large numbers of heterogeneous learning automata connected to tasks that choose the best resources for their computational requirements. In their approach, there was no communication between the learning automata. They reported that reinforcement learning could be implemented to improve the quality of resource selection in large-scale heterogeneous system.

Hamscher et al. (2000) discussed typical scheduling structures, which happen in computational grids and applied simulation techniques to make assessment on these characteristics based on combinations of various Job and Machine Models.

Dai and Levitin (2007) studied a grid computing systems in which the resource management systems (RMS) could divide service tasks into execution blocks (EB), and send these blocks to various resources. The RMS is capable of assigning the same EB to various independent resources for parallel (redundant) execution to reach a desired level of service reliability. Based on the optimal schedule for service task partition, and distribution among resources, it is possible to reach the highest possible expected service performance or reliability. Dai and Levitin (2007) presented an algorithm, which was based on graph theory, Bayesian approach, and the evolutionary optimization approach.

Plestys et al. (2007) emphasized on the relative importance of Grid quality of service (QoS) along with the development of GRID technology because of the concept of GRID service, the multiplicity of users' demands, and the heterogeneity of Grid resources. Sun and Wu (2007) conducted a comprehensive study of QoS (Wang & Luo, 2004) of distributed computing, specifically on grid computing where the necessity of distributed sharing and coordination tends to the extreme. They started at QoS policies, and then concentrated on technical characteristics of the enforcement of the policies and performance optimization under each policy. They also provided a classification of existing software system based on their underlying policies, a systematic understanding of QoS, and a framework for QoS of grid computing.

2. The proposed method

In this part, we present the proposed method and the definition of the problem along with its assumptions. In this paper, we consider Grid environment, and consequently, we assume a star topology grid management. Therefore, the users and their tasks are considered into independent sub-tasks and sub-tasks, which allocate the available resources. Redundancy techniques are normally used in order to increase reliability and better performance tasks such that a task can be processed by two

or more sources although each source needs only one sub-task processing. Let *m* be the number of sources and *n* be the number of sub-tasks with *n* <*m*.

3.1. Run time calculation

For the proposed method, to run things, redundancy techniques are used. The functions presented in Fig. 1 demonstrate the most appropriate sources for the proposed method, based on the data required for the execution of the task and its computational complexity, and bandwidth resources as well as the processing speed of each resource.

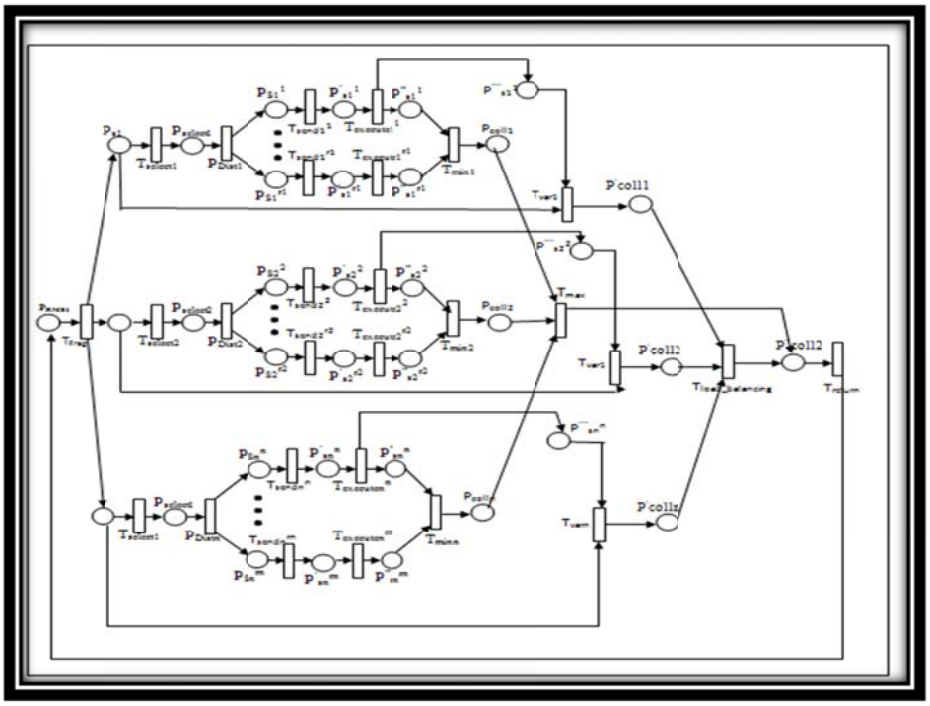

Fig. 1. The proposed model

To calculate the run time, we act as follows: The data required to run the task S_i should be sent to R_J RMS power, with such scoring rd_i is shown and the results are sent to RMS resource R_j, which is defined by I_I. Then all data transferred between the RMS and the power to carry out the task S_I R_J can be shown as A_i, which is calculated as follows:

$$A_i=rd_i+I_i \tag{1}$$

If bandwidth communication lines between RMS and bw_j represented by R_J source, the data transfer time is calculated as follows:

$$\tau_{ij} = \frac{a_i}{bw_j} \tag{2}$$

Top relationship between RMS and power transmission is shown as R_J when we pass through tick trackers pair (τ_I_J, q_ij), which is added to them. Let *C* be the imported duty with the computational complexity, the computational complexity of each task is calculated as follows,

$$\sum_{i=1}^{n} C_i = C \tag{3}$$

where n represents the number of sub-tasks belonging to a specific task, c_i is the computational complexity of each task S_i. Let Ps_j be the processing speed for R_j source, then the processing time of task S_i is obtained as follows,

$$T_{ij} = \frac{C_i}{ps_j} \tag{4}$$

As we can observe from Fig. 1, the transition *j, k, l, m, n, o* is associated with the task of processing time (T_{ij}) and data transfer (t_{ij}) and the resulting sum is added to the token. The probability of failures encountered during the processing of a task (p_{ij}) and the probability of failure encountered during data transfer (q_{ij}) are multiplied and added to the token. So that the transfer token T (execute) their transition-pair ($t_{ij} \times T_{ij}$), (p_{ij}, q_{ij}), which indicates successful execution times to carry. When the redundancy technique is used to improve, the reliability of the simulation was carried out for each sub-task is processed by several sources. Therefore, T_i, R_{ri}, S_i represent the minimum execution time following a task in the resources, which is calculated as follows,

$$T_{i,Rri} = \min\left(t_{ij} + T_{ij}\right); \quad k \in R_{ri} \tag{5}$$

According Eq. (5) and the pair ($t_{ij} \times T_{ij}$), (p_{ij}, q_{ij}), perform the following task in less time and probability S_i (P_i, Rr_i), after passing through the transmission T (min) located on the token in place P (coll) is placed. In order for a task to be completed, the following tasks must be performed.

$$T = \sum_{i=1}^{n} T_i \tag{6}$$

where n represents the number of sub-tasks. In other words, to perform a task, the task is divided into a number of sub-tasks. Then for all the sub-tasks, completion time on a single source is calculated. For each sub-task, the minimum completion time is determined based on resources. After doing this for all the tasks, we had little time to complete all the tasks, and it will choose the least resources to run to the side send. After passing through the transmission T (max) and get a token in place P (RMS2) entered into operation duty ends with fire T (return), other tasks can be logged. In order to run the proposed method in CPNTOOLS we need to repeat our proposed algorithm and enormous tasks should be logged. For this reason, we assume that the task arrival rate is uniformly distributed between 0 and 1.

3.2. Calculation of Load Balancing

To achieve the maximum balance time between resources, utilization rates must be equal. To calculate the amount of load balancing, we compute the sum of the standard deviation of productivity. As a result, we first calculate resource utilization. Utilization rate is calculated as follows:
The largest run-time gain:

$$E = MAX\{T_{ij}, \tau_{ij}\} \tag{7}$$

To calculate the value of e at the top of the track, the two places, a_{s1}, a_{s1}, respectively RMS data transmission times and processing times are kept by resource-use. The efficiency of resources i and j have the following implementation tasks,

$$\mu_i = \frac{T_{ij} + \tau_{ij}}{e} \tag{8}$$

According to Eq. (8), the amount of each resource utilization is calculated and it should be noted at this point that efficiency rate based on the proposed model Azgomy for additional resources (resources that have a running time of more than a source) is equal to zero. To obtain the standard deviation, we first calculate the average productivity:

$$\mu = \frac{\sum_{i=1}^{m} \mu_i}{m} \tag{9}$$

and the standard deviation is calculated as follows,

$$\sigma_i = |\mu_i - \mu|. \tag{10}$$

and the load balancing (lb) is computed as follows,

$$lb = \sum_{i=1}^{m} \sigma_i \tag{11}$$

Finally, to calculate the load balancing system, the transaction *W* is used. In other words, all values of *W* imported transaction efficiency, and then the final amount of the transaction will bring balance to the system load. So if lb = 0, the maximum load balancing will happen. If we assume that we have several different versions for all kinds of reliabilities and load balancing calculated by Eq. (11), we may use some multiple criteria decision making such as Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) (Hwang & Yoon, 1981). The idea of TOPSIS is that the chosen alternative maintains the shortest distance from the ideal solution and the farthest from the negative-ideal solution. The following summarizes the implementation of the TOPSIS method.

First step: The decision matrix is transformed into an amorphous matrix scale as follows,

$$n_{ij} = \frac{r_{ij}}{\sqrt{\sum_{i=1}^{m} r_{ij}^{\,2}}} \tag{12}$$

Second step: Creation Matrix Scale weight vector w is given as input to the algorithm. Namely:

$$W=\{w_1,w_2,\ldots,w_n\}$$

$$V = N_D.W_{n*n} = \begin{vmatrix} V_{11} \ldots\ldots & V_{1j} \ldots\ldots & V_{1n} \\ \vdots & \vdots & \vdots \\ V_{m1} \ldots\ldots & V_{mj} \ldots\ldots & V_{mn} \end{vmatrix} \tag{13}$$

where N_d a matrix so that it points in the Scale and comparable indicators and $w_{n \times n}$ is a diagonal matrix whose only non-zero elements of its original diameter.

Step Three: Determine the ideal solution and negative ideal solution.

$$\begin{aligned} A^+ &= \left\{v_1^+,\cdots,v_n^+\right\}, \text{ where } v_j^+ = \left\{\max(v_{ij}) \text{ if } j \in J; \min(v_{ij}) \text{ if } j \in J'\right\} \\ A^- &= \left\{v_1^-,\cdots,v_n^-\right\}, \text{ where } v_j^* = \left\{\min(v_{ij}) \text{ if } j \in J; \max(v_{ij}) \text{ if } j \in J'\right\} \end{aligned} \tag{14}$$

Step Four: Calculate the size of the separation (distance).

$$d_{i+} = \{\sum_{j=1}^{n} (V_{ij} - V_J^+)^2\}^{0.5}; \quad i = 1,2,...,m \tag{15}$$

$$d_{i-} = \{\sum_{j=1}^{n} (V_{ij} - V_j^-)^2\}^{0.5}; \quad j = 1,2,...,m \tag{16}$$

Step Five: Calculate the relative closeness to the ideal solution Ai. The relative closeness can be defined as follows:

$$l_{i+} = \frac{d_{i-}}{(d_{i+} + d_{i-})}; \quad 0 \le cl_{i+} \le 0; \quad i = 1,2,...,m \tag{17}$$

Note that if $A_i = A^+$, then $d_{i+} = 0$ and we have $cl_{i+} = 1$ if $A_i = A_-$ is and then $d_{i-} = 0$ and will $cl_{i+} = 0$.

Step Six: Ranking of options. Descending order based on the issue of cli options available can be ranked.

4. Case study

In this section, we consider the implementation of a system on its own and load balancing, and response times. The algorithm is based on TOPSIS and selects the best scenario among possible scenarios. We consider several case studies and comprehensive examples. For instance, a bank ATM systems is considered because of its complexity and the lack of possibility of duplicate system. In an ATM bank system, a customer (User) and Bank (Bank) are interacting. In practice, users take the money and this is accomplished when the client enters card inside the ATM system and the information is entered into a database.

The bank then examines the validity of the card and ATM systems, the password request is issued by the bank and the customer enters the password. If the card is invalid, the system returns the customer's bank ATM card, otherwise the customer needs to investigate the validity password. If the password is invalid, the card system for is returned to customer, otherwise, customer can view the different options such as cash withdrawals and transfers. When customer selects “taking the money from the account”, the system will issue the requested withdrawal amount. The system first tries to see whether there are sufficient funds to be given to client. When there is insufficient fund, the customer receives a warning. Otherwise, the system allows the users to withdraw money. We assume that the input task arrival rate is uniformly distributed between 0 and 1. Now, according to the proposed model, we have simulated ATM systems shown in the previous section do. According to the description provided in the proposed algorithm, we first need to specify the initial system specification. These characteristics, system bandwidth, computational complexity of each task are entered into the system. After specifying the above, the present system can be simulated. Fig. 3 shows details of our results.

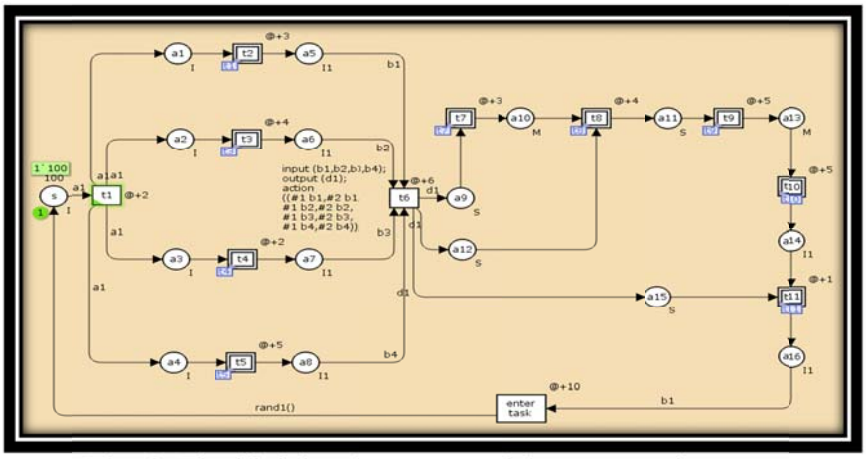

Fig. 2. The high level diagram of the proposed system

Similarly Fig. 3 shows sub-systems to obtain run-time and load balancing for each route.

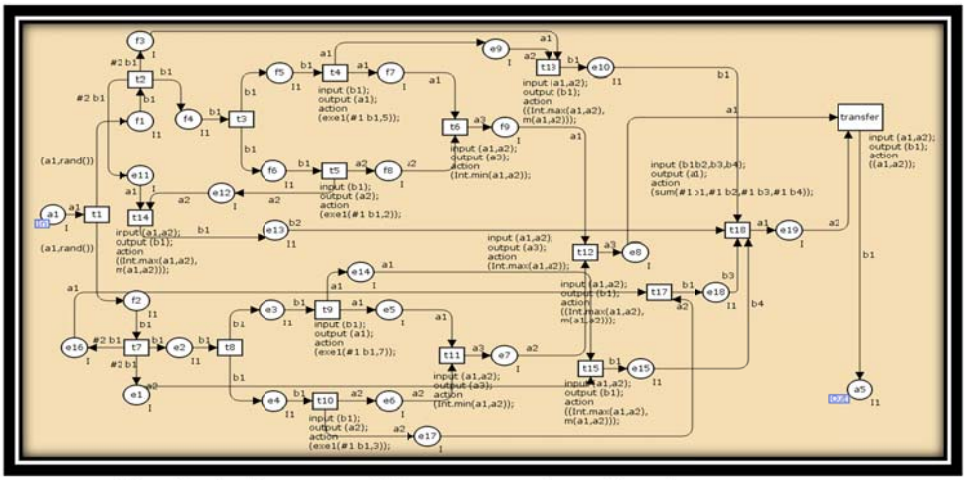

Fig. 3. A diagram of the proposed runtime load balancing

After the execution time and balance the load for each of the routes proposed, we see there are four scenarios and the proposed TOPSIS method needs to find the efficient solution.

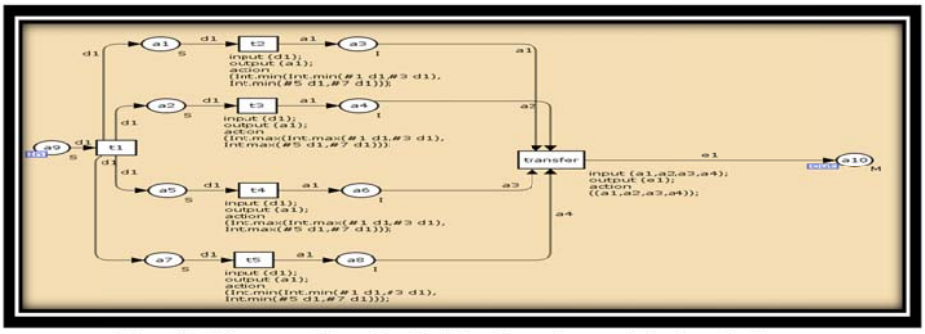

Fig. 4. The results of sub-ideal and non-ideal solutions

Next, we present subsystem integral to obtain the size of each scenario in Fig. 5 as follows,

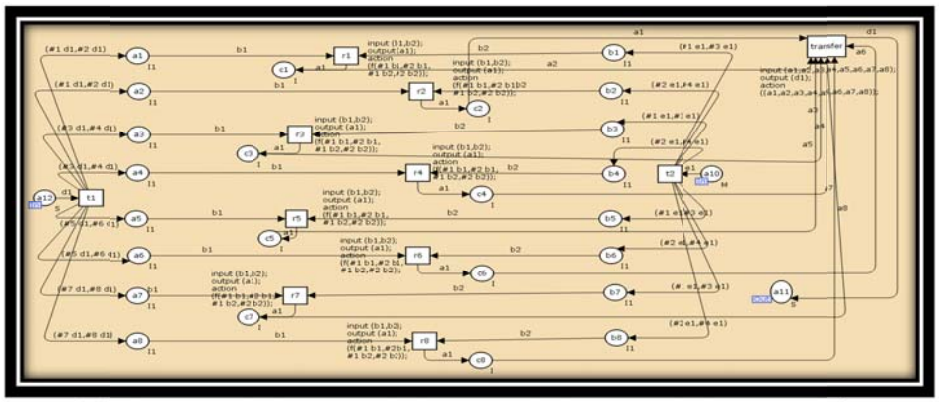

Fig. 5. Subsystem integral to obtain the size of each scenario

After obtaining separate measurements for each scenario, we need to measure the relative closeness to the ideal solution for any scenario, to measure. Fig. 6 shows the process as follows,

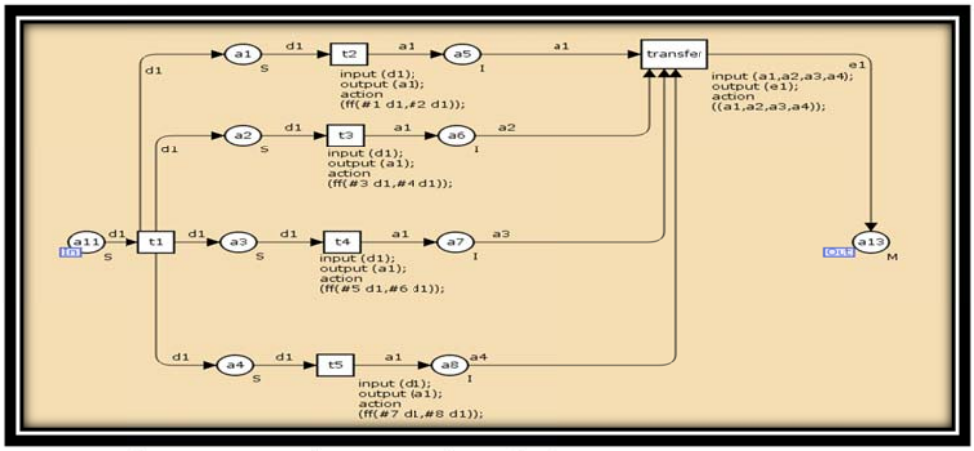

Fig. 6. Subsystem get close enough to the ideal solution for every scenario

Finally Fig. 7 shows the process of selecting the best-case scenario:

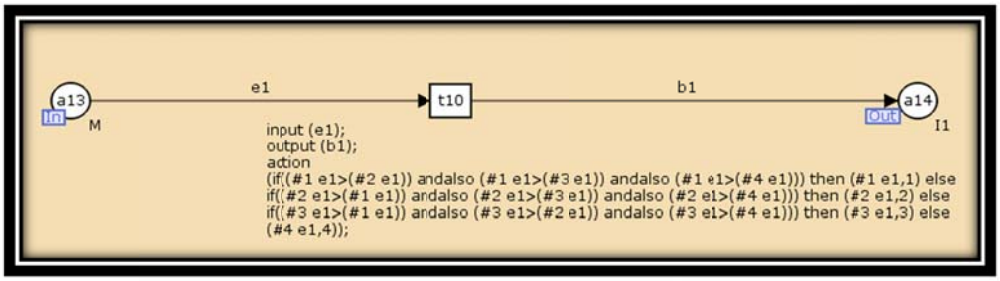

Fig. 7. Subsystem get the best possible scenario for the given scenario

In order to demonstrate the performance of the proposed method, we have compared the results with an alternative method, Azgomi, and Fig. 8 and Fig. 9 show the summary of our findings.

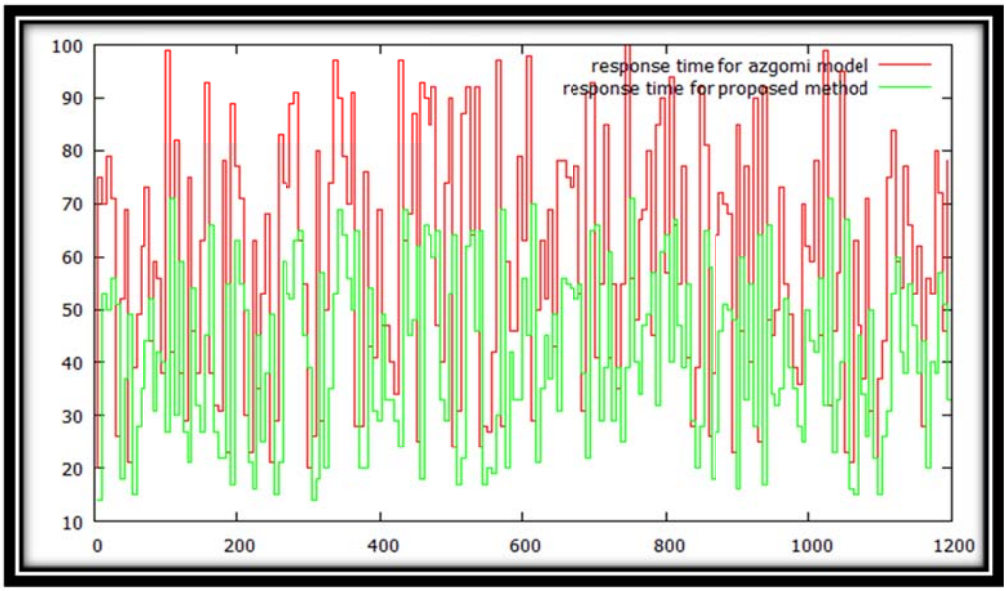

Fig. 8. The comparison of the runtime of the proposed algorithm versus Azgomy model

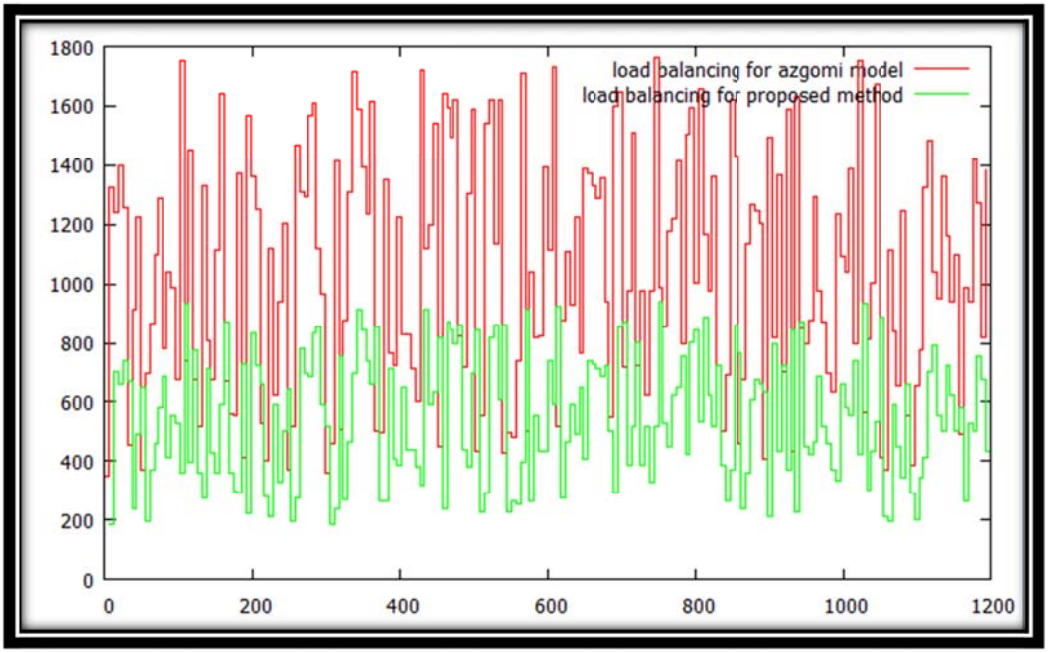

Fig. 9. Comparison for load balancing algorithm of the proposed method versus Azgomy model

According to the results of Fig. 8 and Fig. 9, we can see that the proposed model performed better than the Azgomy model in terms of run-time and load balancing.

5. Conclusion

In this paper, we have evaluated the performance of an algorithm for scheduling tasks and tasks in Grid environment. In this algorithm, the concept of the multi criteria decision-making has been implemented to find the best possible solution in terms of run-time and load balancing system. A significant point in our proposed algorithm is that it can distinguish earlier, taking into account both runtime and load balancing with two parameters for scheduling tasks on the grid. The results obtained from implementing the proposed algorithm on a case study shows the superiority of the proposed method compared with other existing methods such as Mr. Azgomy.

References

Bouyer, A., Karimi, M., Jalali, M., Sap, M., & Noor, M. (2008). A new approach for selecting best resources nodes by using fuzzy decision tree in grid resource broker. *International Journal of Grid and Distributed Computing*, *1*(1), 49-62.

Dai, Y. S., & Levitin, G. (2007). Optimal resource allocation for maximizing performance and reliability in tree-structured grid services. *Reliability, IEEE Transactions on*, *56*(3), 444-453.

Lan, Z., & Li, Y. (2006, June). Failure-aware resource selection for grid computing. In *International Conference on Dependable Systems and Networks (DSN), Philadelphia* (pp. 186-187).

Hamscher, V., Schwiegelshohn, U., Streit, A., & Yahyapour, R. (2000). Evaluation of job-scheduling strategies for grid computing. In *Grid Computing—GRID 2000* (pp. 191-202). Springer Berlin Heidelberg.

Hwang, C.L. and Yoon, K. (1981). Multiple attribute decision making: Methods and applications, Springer-Verlag, Berlin.

Plestys, R., Vilutis, G., Sandonavicius, D., Vaskeviciute, R., & Kavaliunas, R. (2007, June). The measurement of grid QoS parameters. In *Information Technology Interfaces, 2007. ITI 2007. 29th International Conference on* (pp. 703-707). IEEE.

Sarhadi, A., & Meybodi, M. R. (2009). New algorithm to resource selection in computational grid using learning automata. *International journal of Intelligent Information Technology Application*, *2*(5), 204-208.

Shih, P. C., Chen, H. M., Chung, Y. C., Wang, C. M., Chang, R. S., Hsu, C. H., ... & Yang, C. T. (2008, March). Middleware of Taiwan UniGrid. In *Proceedings of the 2008 ACM symposium on Applied computing* (pp. 489-493). ACM.

Sun, X. H., & Wu, M. (2007, August). Quality of service of grid computing: resource sharing. In *Grid and Cooperative Computing, 2007. GCC 2007. Sixth International Conference on* (pp. 395-402). IEEE.

Wang, X., & Luo J. (2004). Architecture of grid resource allocation management based on QoS, Book chapter of "Grid and Cooperative Computing", Springer Berlin / Heidelberg.

Wang, C. M., Chen, H. M., Hsu, C. C., & Lee, J. (2010). Dynamic resource selection heuristics for a non-reserved bidding-based Grid environment. *Future Generation Computer Systems*, *26*(2), 183-197.

Xueguang, C., & Haigang, S. (2004, March). Further extensions of FIPA contract net protocol: threshold plus DoA. In *Proceedings of the 2004 ACM symposium on Applied computing* (pp. 45-51). ACM.

Yousif, A., Abdullah, A. H., & Ahmed, A. A. (2011). A bidding-based grid resource selection algorithm using single reservation mechanism. *International Journal of Computer Applications*, *16*(4), 39-43.

Zhai, Z. (2010, October). Grid resource selection based on reinforcement learning. In *Computer Application and System Modeling (ICCASM), 2010 International Conference on* (Vol. 12, pp. V12-644). IEEE.