

# Q-Learning applied to the problem of scheduling on heterogeneous architectures

Younes Hajoui <sup>†</sup> Omar Bouattane <sup>†</sup> Mohamed Youssfi <sup>†</sup> and Elhocein Illoussamen <sup>††</sup>,

Laboratory SSDIA, ENSET Mohammedia, Hassan II University of Casablanca  
Mohammedia 28999, Morocco

## Summary

Grid computing exploits linked heterogeneous resources to deal with greedy applications or complicated executing tasks. Hence, an efficient resource allocation algorithm is extremely important for the sake of reducing and minimizing the overall execution time. Our approach consists in developing a novel collaborative Q-Learning scheduler using a Holonic Multi Agent System to solve the job scheduling problems on heterogeneous distributed systems. The results show that for many tested dynamic environments, the proposed load balancer optimizes the distribution of tasks and reduces the total tardiness.

### Key words:

*Job scheduling, Grid Computing, Distributed System, Multi-Agent system, Load balancer, Q-learning.*

## 1. Introduction

This paper deals with the problem of scheduling jobs on heterogeneous parallel machine in order to optimize an utilization of distributed system and minimize the makespan. An efficient strategy for scheduling tasks in grid plays a key role in assigning tasks among available machines as equally as possible. This work aims at evolving an exact model to solve this kind of problem.

The resource allocation problem is extensively studied in the literature [1,2,3,4,5] given that greedy calculations is frequently appears in many areas and in data intensive application [6] such as: weather forecasts, financial projects, industrial production, aerodynamic, scientific simulations, molecular biology problems, image processing, video games and so. Hence the need for more efficient and cost-effective strategies to cope with this issues in resource allocation.

Load balancing can be classified into two types: static and dynamic algorithms.

In general, according to [7-11], the used algorithms to deal with the static balancing method allocates the tasks by unique and final allocation, to the resources in a distributed system. It doesn't take into consideration the current system status and the node properties.

The main disadvantage of the static load balancing algorithm is that it creates major imbalances greater than the balancing produced by an arbitrary distribution.

Dynamic load balancing approach considers, for task scheduling, the current status of the system [12-16,17]. Dynamic load balancing algorithms are more complicated to implement but efficient and balances the load in an optimal manner.

Recently, many load-balancing schemes are developed based on mobile agents. The MAS [18] (Mobile Agent Systems) is a system composed of a set of collaborating agents to solve difficult problems that are hard or impossible for an individual agent. Further, in multi-agent systems the concept of agent autonomy and mobility helps the load balancing developers to conceive and implement the migration strategy from the overloaded nodes to the lightly ones [19,20].

Many mobile agent methods have been proposed to deal with the load balancing problems. In [4,5] authors developpe an innovative multi agent Framework for grid computing. Their model engages a set of collaborative mobile agents to perform parallelly the continuous coming tasks. Dispatcher agent is used to schedule tasks received from producers to the workers agent. workers are the executers residing in nodes of the grid under the oversight and guidance of the controller's agent. In [4], the authors propose a new model that can launch the migration process from container to other. the goal is to lighten the overloaded nodes.

Meta-heuristics methods are broadly used to solve complex problems and especially the load balancing dilemma [21, 22]. In [23] Authors propose a new method inspired load-balancing algorithms based on the use of ant colony optimization. In the context of their research, the scheduler is used as an ant which chooses, for the current job, the machine having the higher rate of pheromone.

The load balancing experiments were therefore satisfactory and the performance was as expected during the design phase.

In this paper, we propose a new framework for load balancing based on mobile agents and an Q-Learning algorithm. In this proposal, a scheduler agent is involved to distribute received jobs to the executers according to the exact decisions. The implementation of the proposed scheme uses the agents based middleware for distributed programming JADE tool [24].

Section II defines and formulate the problem presented in this work. Section III describes the architecture of the proposed computational model. Section IV and V presents respectively the technical background and the load balancing system used in task routing. Section 5 show the experimental setup and results accomplished. In conclusion, section 6 recaps the contribution of this paper and presents some perspectives for upcoming extensions.

## 2. Model

Let  $[T] = \{1, 2, \dots, m\}$  be the set of jobs and let  $[N] = \{1, 2, \dots, n\}$  be the set of machine workers. As shown in fig.1, the resources are connected to the dispatcher of the layer 2 through heterogeneous network links. Let  $L_i$  and  $P_i$  denotes respectively the speed of the network link connecting the machine  $M_i$  with the Dispatcher and the velocity of the worker machine  $M_i$ . Each task:  $T_i \in [T]$  has an associated complexity  $C_i$  and an estimation execution time  $\tau_i \in \mathbb{R}$ .

For  $t \geq 0$  let  $TE_i(t)$  represent the estimated times of all jobs waiting in line on machine ( $M_i$ ) at time  $t$ .

$$TE_i(t) = \sum_{i=1}^{|\mathcal{Q}_i|} \tau_i \quad \text{Where:} \quad (1)$$

$|\mathcal{Q}_i|$ : is the number of the queued tasks in machine  $M_i$ .

Each worker machine is monitored by a controller agent. The controller's agent communicates continuously to the dispatcher the state of their nodes:  $s_i(t)$ .

$$s_i(t) = \{L_i, P_i, NBC_i, \tau_i, TE_i(t), NbT_i(t), r_i(t)\} \quad \text{Where :}$$

$NBC_i$  : Is the number of cores (CPUs) of node  $M_i$ .

$NbT_i(t)$ : Is the number of the queued tasks on node  $M_i$  at time  $t$ .

$r_i(t)$ : Represents the reward of machine  $i$  at time  $t$ .

Let  $S(t) = \{s_1(t), s_2(t), \dots, s_N(t)\}$  represents all the status of all worker machines.

The mentioned gathered parameters by the dispatcher, allows in each task distribution, to choose the suitable machine to execute the current task.

Therefore, at any given moment, the overall execution time to perform the queued jobs depends essentially on the latest executer machine. In the literature it is also called the makespan, or the total tardiness. In our proposal, the makespan can be formulated as follows:

$$\theta P(t) = \text{Max}_{i=1}^N(\theta P_i(t)) \quad (2)$$

$$\text{Where: } \theta P_i(t) = \left[ \left( \frac{\sum_k \tau_k}{\tau_0} \right) * L_i + TE_i(t) \right] / NBC_i \quad (3)$$

$k$  : Is the index of task  $T_k$  listed on the queue of the node  $M_i$  at time  $t$ .

Finally, the scheduling problem can be mathematically expressed as optimization problem as given below:

$$\begin{aligned} \text{Min}[\theta P(t)] &= \text{Min}_{\tau \leq t} (\text{Max}_{i=1}^N(\theta P_i(t))) = \\ \text{Min}_{\tau \leq t} &\left[ \left[ \left( \frac{\sum_k \tau_k}{\tau_0} \right) * L_i + TE_i(t) \right] / NBC_i \right] \end{aligned} \quad (4)$$

## 3. Computational model

### 3.1 Architecture description

This work aims to solve a scheduling problem on heterogenous parallel machines. An efficient scheduling algorithm plays an important role in an effective management of the resources and consequently in reducing the overall execution time. In our proposal, the load balancing is achieved by using Q-learning and a hierarchical mobile agent system.

As shown in fig.1 ,5 types of mobile agents are used in our method: The Producer-Agent, the Tester-Agent, the Dispatcher-Agent, the Controller-Agent and the Worker-Agent. The Producer agent is the one that represents the producers of tasks as: web application, mobile application, embedded system(IOT), expert agent(human) and so. The Tester-Agent estimate by learning the execution time of tasks. The Producer agent is a type of central unique agent and is responsible for scheduling tasks among available resources. Controller-Agent is responsible for continuously monitoring the status of the resources and Worker-Agent executes the tasks received from the dispatcher.

The main idea of this three-layer model is to build a Framework able to execute parallelly the continuous arrival tasks in such a way as to make the system in the balanced state.

### 3.2 Monitoring model

#### 3.2.1 Referencing phase

The aim of the referencing phase is to provide the scheduler by the performance parameters of each worker machine. Consequently, these settings are used to avoid overloading the weak machines, underloading the strong machines and then to reduce the total tardiness.

Each machine that seeks to join the Framework, at the 3rd layer, as a worker node to participate on the computation must follow these three steps as shown in fig.2:

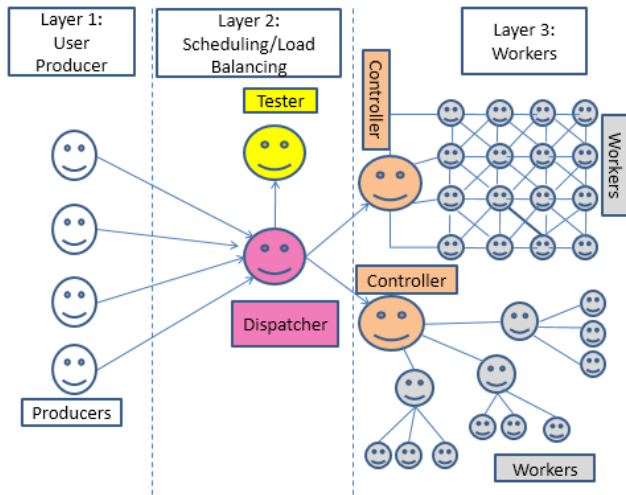


Fig. 1 Framework Architecture .

1. Send a request to the dispatcher to join the workers.
2. Receive and perform the referencing task:  $T_0$ .
3. Communicate to the Dispatcher the results of  $T_0$  execution:  $L_i, P_i, NBC_i$ .

Note that:

- $\theta_i$  : Represent the total Time required to perform  $T_0$  since its departure from the dispatcher, its execution on node  $N_i$  and its return to the dispatcher.
- $\theta P_i$  : Represent the total Time required to perform  $T_0$  on Node  $N_i$  ( $i=1..n$ ).
- $L_i$  : The communication Latency of each node  $N_i$ . It corresponds to:  $L_i = \theta_i - \theta P_i$  (4)

#### 3.2.2 Load balancing process

Two methods are presented to solve the addressed scheduling problem:

- Q Learning Algorithm:

The used Framework is endowed by a load balancing process which is performed by the dispatcher agent. This agent takes decisions about scheduling tasks using Q

Learning algorithm. This means that the load balancer learns from the previous scheduling. Hence, the imbalance of the system will be reduced by the experience accumulated by the Dispatcher.

- Migration Task Strategy:

This second strategy consists of classify periodically nodes into 3 groups: Overloaded nodes(ON), Under loaded nodes (UN) and Normal nodes (NN). Then, start the process of tasks migration from overloaded nodes to under loaded ones.

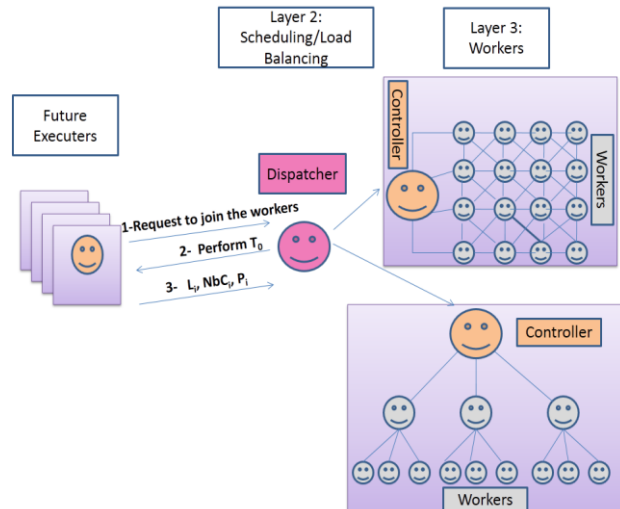


Fig. 2 Referencing phase.

## 4. Technical background

### 4.1 Q Learning

Reinforcement learning (RL) is an actual method allowing algorithms to act optimally on very dynamic and stochastic environments. By trial and error interactions and exchanges between agent and its environment, the algorithm of RL learn an intelligent optimal strategy to cope with the more difficult circumstances. These situation model consist of an agent observing its environment, selecting an action from the current state and then taking a reward or penalty to the action chosen. The goal of the agent is to maximize its entire upcoming reward. Q-learning [25, 26] is one of the well-known RL methods. Its goal is to find an optimal action-selection strategy for any specified Markov decision process.

The core of the Q-learning consists of a simple update of the function Q. Q is updated at each action-selection by the succeeding formula:

$$Q_{t+1}(s_t, a_t) = (1 - \alpha)Q_t(s_t, a_t) + \alpha \left( r_t + \gamma \cdot \max_a Q_t(s_{t+1}, a) \right) \quad (5)$$

learned value  
learning rate  $\alpha$   $\left( \begin{array}{c} \text{reward} \\ r_t \end{array} + \begin{array}{c} \text{discount factor} \\ \gamma \end{array} \cdot \begin{array}{c} \text{estimate of optimal future value} \\ \max_a Q_t(s_{t+1}, a) \end{array} \right)$

Where:

$\alpha$ : is the learning speed, it controls how much new calculated data will exceed the old one.

$\gamma$ : is called: discount factor. it controls the weight of future rewards.

## 5. Proposed approach

In our proposed Framework, the controller agents continuously communicates to the second layer the following calculated parameters:  $NbT_i(t), TE_i(t), r_i(t)$ . Then, the Dispatcher stock these collected data beside  $P_i, L_i, NbC_i$  in  $s_i(t)$ . In real time, the Dispatcher updates  $S(t)$  by the novel gathered values  $\{s_i(t)\}_{i=1..N}$ .

As shown in fig.3, the Dispatcher uses the QL to schedule, in an optimum way, the received tasks from Producers.

The Dispatcher agent receives in real time the list of tasks from the Producers agent. It analyzes the estimation time of each task and then it uses Q-learning to send each task to the suitable worker. The controllers communicate directly to the second layer the gathered state resource  $S(t)$  in order to provides the Dispatcher by the efficiency of each worker machine  $M_i$  in the past.

The situation consists of a QL-Dispatcher agent that schedules tasks over available nodes, a set ( $S$ ) that represent the workers states, and a set of actions  $A$  that correspond to the list of all possible schedules. By executing an action  $a$ , the agent switches from a state to a new one. Choosing an action  $a$  rewards the selected worker by a numerical value  $r_i(t)$  that can be calculated based on the other parameters of  $s_i(t)$  by following this algorithm:

### Begin

```

For the first scheduling: InitReward();
For future scheduling: for each
    WMi :CalculReward(WMi);
InitReward()
For( i=1 to N ) WMi. ri(t) ← 1; }
CalculReward(WMi) {
If( WMi. TEi(t) > (1+Ts)*θT(t)/N ) {
    (WMi. ri(t)) ← (WMi. ri(t)) - reward ; }
If( WMi. TEi(t) < (1+Ts)*θT(t)/N ) {
    (WMi. ri(t)) ← (WMi. ri(t)) + reward ; }

```

### End.

Algo. 1 Reward calculation algorithm.

The core of this algorithm consists of a simple calculation of  $r_i(t)$ . The reward calculation is based on the comparison to a load threshold value:  $T_s$ . This parameter indicates whether each node belongs to the overloaded machines or to the underloaded ones.

In our case,  $T_s$  is randomly fixed at 20%. We can easily change this value for another possible configuration.

Rewarding a machine by a positive value means encouraging it to receive more tasks in future scheduling.

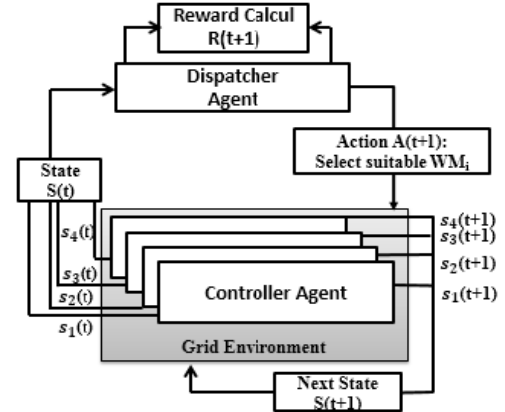


Fig. 3 Q-learning model

Contrary to the negative rewarding, it discourages weak machines to receive more tasks.

Note that  $Q$  is initialized to 0 and  $R$  to 1 where:

$$R(t) = \{r_i(t)\}_{i=1..N} \text{ and } Q(t) = \{Q_t(s_i, a_i)\}_{i=1..N}$$

Next, the QL-Dispatcher adjusts the  $q$ -values according to the actions taken and the reward received.

The dispatcher agent uses the following equation to update  $Q_t(s, a)$  at each time interval:

$$Q_{t+1}(s_1, a_1) = Q_t(s_1, a_1) + \alpha(r_1(t) + \gamma \max_{a'} Q_t(s', a') - Q_t(s_1, a_1)) \quad (6)$$

It means that it calculates for each worker the new value  $Q_{t+1}(s, a)$  based on its old  $Q_t(s, a)$ , its reward  $r(t)$ , and the maximum value  $Q_t$  of all worker machines.

The figure 4 and the following example demonstrates an example of  $Q$  calculation: calculation of the novel  $Q_{t+1}(s_1, a_1)$ :

$$Q_{t+1}(s_1, a_1) = Q_t(s_1, a_1) + \alpha(r_1(t) + \gamma \max_{a'} Q_t(s', a') - Q_t(s_1, a_1))$$

$$\{ \max_{a'} Q_t(s', a') = \max\{Q_t(s_1, a_1), Q_t(s_2, a_2), Q_t(s_3, a_3), Q_t(s_4, a_4)\}$$

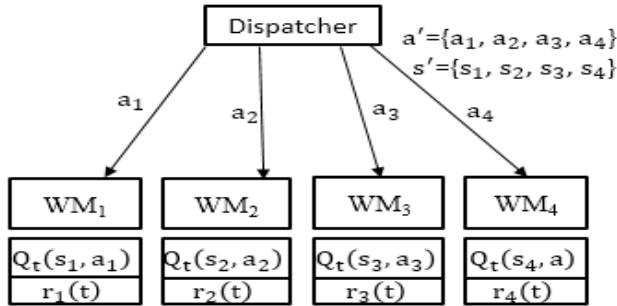


Fig. 4 Q-calculation

Next, the dispatcher selects the machine having the greatest Q value to send it the current task and so forth on each new received task from producers.

The following algorithm summarizes the load balancing steps proposed in this paper.

```

Begin
Repeat
    Controllers communicates S(t) to the Dispatcher.
    Dispatcher calculates R(t+1) and then Q(t+1).
    Dispatcher sends the current task to the
        worker having the greatest Q value.
    Controllers update S(t+1).
While <true>
End.
    
```

Algo. 2 Load balancing algorithm operation.

## 6. Experiments

The Q-Learning scheduler algorithm was implemented and tested on a grid of N (N=10) heterogeneous machines worker. The test experiments were generated using a set of NbT (NbT= 1000) heterogenous tasks. This section presents the computational findings.

### 6.1 Parameter's determination phase

Resources referencing is realized by executing a special task T0 as already mentioned in section 3.2.1. Table 1 shows the parameters calculated during this referencing phase:

Table 1: Parameters calculation by referencing phase

Node i	Pi (ms)	Li (ms)	NbCi	Reward: Mi
0	80	2	4	1
1	150	10	2	1
2	120	8	1	1
3	70	10	4	1
4	100	8	8	1
5	90	10	2	1
6	160	3	1	1
7	170	5	16	1
8	120	1	2	1
9	145	3	4	1

### 6.2 Q-Learning parametrization

The performance metric in finding optimal solutions depends essentially on the parameterization of the Q-Learning operators.

The best Q-Learning operators found are shown in Table 2.

Table 2: Best parameter settings of the Q-Learning

$\alpha$	$\gamma$
0.2	0.4

Theoretically, for a given system S, having at time t, N distributed resources and an overall execution time T, each resource must have a workload of execution time around the theoretical value:  $LBT=T / N$  which is impossible experimentally.

### 6.3 Performance evaluations under different system loads

#### 6.3.1 Tasks having ordered complexity

In this section, we arbitrarily indicate:  $\tau_i = 20i$  for each Task i. Table 3 shows the scheduling results using Round Robin Algorithm and Q-Learning, while figure 5 represents a comparison between these three durations:

- Distribution using Round Robin algorithm.
- Load balancing Theoretic: LBT.
- Distribution using Q-Learning.

Round robin algorithm distributes jobs over all available nodes in circular order Without any type of priority.

Table 3: Parallel execution time of 1000 Tasks using Round robin and Q-Learning

Node Ni	NbT Q-L	$\theta P_i(t)$ (ms) Q-L	Round robin (ms)
0	103	266686,5	157550
1	48	328430	496500
2	24	199220	594800
3	61	172357,5	159250
4	172	214964,5	74850
5	36	181510	500500
6	28	277144	602300
7	346	216619,375	62781,25
8	86	432223	503050
9	96	239207	252075

The chosen metrics to evaluate and measure the system performance at each instant, as already cited, are based on the calculation of the ratio:

$$\lambda = \frac{\theta_{Q-L}}{\theta_{LBT}} \tag{7}$$

In this case  $\lambda = 1.82$ .

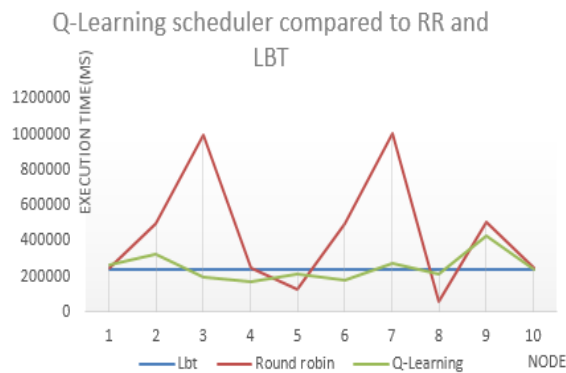


Fig. 5 Curve of Q-Learning distribution compared to the Round robin and LBT.

### 6.3.2 Tasks having unordered complexity

For the additional category of tests related to assessments under dissimilar system loads, we select arbitrarily for each task  $T_i$ :

$$\begin{cases} \tau_i = 20i & : \text{if } i \text{ is an odd number.} \\ \tau_i = 20i^2 & : \text{Else} \end{cases}$$

Giving to this dissimilar system loads example, a novel test is performed. The distribution findings are as follows:

Table 6 shows the distribution results by using Q-learning.

Table 4: Parallel execution time of 1000 Tasks using q-learning of unordered tasks complexities.

Node	NBT <sub>i</sub>	$\theta P_i$ (ms)
0	10	91214279,5
1	104	56255662
2	118	37788468
3	32	34039604,5
4	119	36351290
5	66	42744272
6	41	148449863
7	373	89091486,9
8	46	165171256
9	91	99118657,5

In this case  $\lambda = 2.13$ .

## 7. Conclusion

This paper deals with the problem of scheduling jobs on heterogenous parallel machine in order to optimize a utilization of distributed system and minimize the makespan. The problem is formulated as optimization mathematical model. Based on this exact model, we have developed a solution that consists of an Q-Learning scheduler that learns from the interaction with its environment. The results show that for many tested dynamic environments, the proposed load balancer optimizes the distribution of tasks and reduces the total tardiness. In perspective relatively to this study, another possible contribution of this work would be the hybridization of Q-Learning and Ant colony optimization in order to gradually eliminate the cost of tasks migration strategy.

## References

- [1] S. Sharma, S. Singh, and M. Sharma, "Performance Analysis of Load Balancing Algorithms," World Academy of Science, Engineering and Technology, vol. 38, 2008.
- [2] L. M. Khanlil and BehnazDidevar, "A New Hybrid Load Balancing Algorithm in Grid Computing Systems," Journal of Computer Science Vol-2 No 5 October, 2011.
- [3] A. Revar, M. Andhariya, D. Sutariya, "Load Balancing in Grid Environment using Machine Learning - Innovative Approach," International Journal of Computer Applications (0975 – 8887), Volume 8– No.10, October 2010
- [4] M. Youssfi and O. Bouattane, "Efficient Load Balancing Algorithm for Distributed Systems Using Mobile Agents," Advanced Studies in Theoretical Physics Vol. 9, 2015, no. 5, pp.245 - 253.
- [5] Y.Hajoui, M. Youssfi, O. Bouattane and E.Illoussamen "NEW MODEL OF FRAMEWORK FOR TASK SCHEDULING BASED ON MOBILE AGENTS," Journal of Theoretical & Applied Information Technology . Vol. 81 Issue 1, p65-72 ; October,2015.
- [6] M. D. Beynon, T. Kurc et al. "Efficient Manipulation of Large Datasets on Heterogeneous Storage Systems", Proceedings of the 16th International Parallel & Distributed Processing Symposium, IEEE, Los Alamitos, California, 2002.



- [7] Kameda, H., Li, J., Kim, C., Zhang, Y.: *Optimal Load Balancing in Distributed Computer Systems*. Springer, London (1997)
- [8] Tang, X., Chanson, S.T.: *Optimizing static job scheduling in a network of heterogeneous computers*. In: Proceedings of Intl. Conf. on Parallel Processing, pp.373–382. IEEE, Piscataway (2000)
- [9] Grosu, D., Chronopoulos, A.T.: *Noncooperative load balancing in distributed systems*. *J. Parallel Distrib. Comput.* 65(9), 1022–1034 (2005)
- [10] Penmatsa, S., Chronopoulos, A.T.: *Job allocation schemes in computational Grids based on cost optimization*. In: Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium, Denver, (2005)
- [11] Penmatsa, S., Chronopoulos, A.T.: *Price-based user optimal job allocation scheme for Grid systems*. In: Proceedings of 20th IEEE International Parallel and Distributed Processing Symposium, Rhodes, 2006
- [12] Dhakal, S., Hayat, M.M., Pezoa, J.E., Yang, C., Bader, D.A.: *Dynamic load balancing in distributed systems in the presence of delays: a regeneration theory approach*. *IEEE Trans. Parallel Distrib. Syst.* 18(4), 485–497 (2007)
- [13] Dobber, M., Koole, G., Mei, R.: *Dynamic load balancing experiments in a Grid*. In: Proceedings of IEEE International Symposium on Cluster Computing and the Grid, Cardiff, 2005
- [14] Penmatsa, S., Chronopoulos, A.T.: *Dynamic multi-user load balancing in distributed systems*. In: Proceedings of 21st IEEE International Parallel and Distributed Processing Symposium, Long Beach, 2007
- [15] Shah, R., Veeravalli, B., Misra, M.: *On the design of adaptive and de-centralized load balancing algorithms with load estimation for computational Grid environments*. *IEEE Trans. Parallel Distrib. Syst.* 18, 1675–1686 (2007)
- [16] Arora, M., Das, S.K., Biswas, R.: *A de-centralized scheduling and load balancing algorithm for heterogeneous Grid environments*. In: Proceedings of International Conference on Parallel Processing Workshops, pp. 499–505. IEEE, Piscataway (2002)
- [17] Zheng, Q.: *Dynamic load balancing and pricing in grid computing with communication delay*. *J. Grid Comput.* 6, 239–253 (2008)
- [18] Kim YH, Han S, Lyu CH, Youn HY. *An efficient dynamic load balancing scheme for multi-agent system reflecting agent workload*. In: The 12th IEEE international conference on computational science and engineering; 2009.
- [19] Cho ChoMyint, Khin Mar Lar Tun, *A Framework of Using Mobile Agent to Achieve Efficient Load Balancing in Cluster*. In: Proc. 6th Asia Pacific symposium on information and telecommunication technologies; 2005.
- [20] Maha A. Metawei, Salma A. Ghoneim, Sahar M. Haggag, Salwa M. Nassar. *Load balancing in distributed multi-agent computing systems*, *Ain Shams Engineering Journal*, (), pp. 237–249. ( 23 May 2012)
- [21] Kwang, M.S., Sun, H.W.: *Ant colony optimization for routing and load-balancing: survey and new directions*. *IEEE Trans. Syst. Man Cybern. Part A* 33(5), 560–572 (2003)
- [22] Ludwig, S.A., Moallem, A. : *Swarm Intelligence Approaches for Grid Load Balancing*. *J Grid Computing* 9, 279–301 (2011)
- [23] H. Younes, O. Bouattane, M. Youssfi, and E. Iloussamen, “New load balancing Framework based on mobile AGENT and ant-colony optimization technique”. In: Proceedings of International Conference on Intelligent Systems and Computer Vision (ISCV), IEEE, Fez-Morocco (2017).
- [24] F. L. Bellifemine, G. Caire, and D. Greenwood, “Developing MultiAgent Systems with JADE”. Wiley, 2007.
- [25] J.W. Christopher, D. Peter, *Q-learning*, *Machine Learning*, 8 (1992) 279-292.
- [26] K. Hwang, H. Lin, Y. Hsu, H. Yu, *Self-organizing state aggregation for architecture design of Q-learning*, *Information Sciences*, 181 (2011) 2813-2822.