# New Hybrid Task Scheduling Algorithm with Fuzzy Logic Controller in Grid Computing

Younes Hajoui, Omar Bouattane, Mohamed Youssfi, Elhocein Illoussamen
Laboratory SSDIA
ENSET Mohammedia, Hassan II University of Casablanca Mohammedia 28999, Morocco

***Abstract*—Distributed heterogeneous architecture is extensively applied to a diversity of large scale research projects conducive to solve complex computational problems. Mentioned distributed systems consist of multiple heterogenous linked processing units used to handle the continuous arrival jobs. The tasks scheduling problem is concerned with resource allocation strategies to assign jobs to available computing resources. The load balancing of linked resources becomes a main issue to select in each task schedule the adequate computing resource. Our proposal consists of combining Q-learning with ACO (Ant Colony Optimization) to solve the tasks allocation dilemma. In our proposed Fuzzy Hybrid Framework, Fuzzy ants are used to calculate at each scheduling operation, the novel reward values whereas Q-learning is used to select the suitable Worker Machine. The simulation findings confirmed the efficiency of the proposed framework due to the significant decrease of the makespan.**

***Keywords—Distributed systems; computational problems; load balancing; Q-learning; ACO; fuzzy hybrid framework***

## I. Introduction

The tasks scheduling problem is concerned with resource allocation strategies to assign jobs to available computing resources. The load balancing of linked resources becomes a main issue to select in each task schedule the adequate computing resource.

Due to the heterogeneity of arrival tasks and uneven nodes performance, some nodes work more than others. Therefore, to achieve equal distribution and optimal use of resources, scheduling need to be fair, well studied and strategic [1,2].

In [27], the authors propose a global taxonomy which is used to classify frequently encountered types of job scheduling, facilitate researchers to build on prior art, increase new research visibility, and minimize redundant effort.

In the literature, load balancing algorithms can be classified into centralized, decentralized or hierarchical categories [3].

In centralized scheme, tasks are scheduled first to a central resource then this central node decides how to assign received tasks to executers. A major disadvantage of using central node is that it must not fail because it should ensure the allocation of tasks.

Decentralized scheme does not contain a central scheduler, scheduling decisions is done by all resources in the distributed system [4, 9, 11, 29]. This model suffers from several weaknesses and especially the cost resulting from the involvement of all resources in the balancing procedure.

In hierarchical model [30], the responsible schedulers are ordered in a hierarchy. This model results from the hybridization of the centralized and decentralized model. Each scheduler is responsible on the schedulers which are below at lower levels and is under the orders of the schedulers from above at higher levels.

Further, load balancing algorithms can be classified into three categories: static, dynamic or adaptive algorithms.

The approach in the static balancing system assigns the tasks by unique and definitive allocation, to the processors or nodes in parallel architectures [5], [6], [28], [30]. Furthermore, the static algorithms don’t have the ability to deal with the dynamic changes of such environments. This problem especially arises in distributed systems, where some external variables such as network load and waiting for results of other tasks, make difficult the effective scheduling of tasks. Also, the continuous arrival of new tasks makes the scheduling difficult by a static load balancing approach. In dynamic environments, it is even possible that a static balancing creates major imbalances greater than the balancing produced by a random distribution of tasks. So, the need to adapt the initial machine performance estimation is justified. Dynamic load balancing approach considers, for task scheduling, the current processor load [7, 9,12].

Recently, many load-balancing schemes based on mobile agents have been proposed. The MAS [13] (Mobile Agent Systems) are widely used to offer solutions to dynamic and complex application domains. The main characteristic of these intelligent systems is the migration. The ability of agent migration facilitates the implementation of strong dynamic load balancing strategy. The migrant agent selection is relied on the strategy adopted by the load balancer while the destination is related to the lightly loaded machines.

The migration decision is taken by a centralized load balancer agent that activates the migration process when it is obligatory. The centralized control is not suitable for a dynamic scheme since it must collect data more regularly than the non-centralized one, leading to the overload of the network traffic [14].

In [15], a centralized load balancing scheme is proposed. The principal measure for selecting a node is constructed on job’s execution time, while location rule is constructed on cooperation with cluster nodes. A special agent in each node is in charge for gathering the occupancy rate and the local resource usage quantity. The migration choice is founded on the comparison given to an assumed load threshold value.

In [16] authors suggest a new framework for job scheduling founded on mobile agents. Their proposed model uses a dispatcher agent to schedule parallel jobs to worker agents. Each worker agent is installed in a node of the distributed system giving to a load balancing strategy. A test of application, associated to the distributed image processing, was presented to judge the performance of the framework. Additional work in [17] used a mobile agent, to migrate the jobs from overloaded nodes to the under loaded ones. In the used distributed system, each job should be allocated to a VPU (virtual processing unit). The VPUs connected with each other asynchronously by exchanging through their ports ACL messages (FIPA-ACL). Exchanged messages contain data and jobs to be performed.

In [18,19] the authors centered their research on studying load balancing necessities in a distributed system and planned a design and implementation of an advanced load balancing scheme for grid environment via machine learning. Their method is equilibria to the load dynamically. It uses initial load data kept in the database at the primary level of the procedure. Once a load imbalance arises, the recent load data is collected and warehoused as raw data. Later, numerous machine-learning algorithms have been used to process and investigate the logged data. As a final step, the rules are automatically engendered by data mining methods and used for migrating jobs to rebalance loads.

Recently Multi-agent learning methods have been extensively used in the problematic of resource allocation in the Grid. In [20] the authors present Reinforcement learning in which the agents learn through a trial and error to familiarize to all variations such as the changing resource capacities, latencies, or resource failure, by getting rewards for its actions. The Agents give a score rewarding each machine based on its role to reduce the maximum completion time (makespan).

The tasks scheduling has been proved as a NP-hard problem accordingly [21, 31, 32]. Hence, the use of swarm intelligence systems has become very suitable to deal with the difficulty of such problems [22]. Ant colony optimization is one of the well-known meta-heuristics that is largely used in both path finding and load balancing [23, 22]. In [23] Authors suggest two new distributed swarm intelligence inspired load-balancing algorithms. The first and the second algorithm are correspondingly based on ant colony and on particle swarm optimization. The test of their proposed model is conducted by means of GridSim, which is a platform of simulation based on Java [24]. The robustness of their two strategies is assessed using performance criteria such as makespan and load balancing ratio.

In [25], the authors suggest a new scheme inspired load-balancing algorithms founded on the use of ant colony optimization. In the setting of their exploration, the load balancer is used as an ant which selects, for the recent job, the worker machine having the higher amount of pheromone.

Recently Multi-agent learning for load balancing problems has been extensively treated in the literature. In [26] the authors present machine learning in which the agents learn through the previous experiments completed by the scheduler. It is through test and mistake that the agent learns and progresses his tactic. The Agents allocate a score rewarding each worker machine based on its performance in the past. The principal goal of these teams of cooperative agents is maximizing the global reward, which will later reduce the overall execution time (makespan).

In this paper, we propose a new Framework for task scheduling based on hybridization of Q-Learning and ant-colony optimization technique. Ants are used to calculate reward and Q-Learning is used to schedule the current task to the appropriate worker. In the planned model, a grid manager agent is involved to allocate received jobs to the available worker agents according to the precise decisions to minimalize the total execution time (makespan). The proposed framework is constructed by means of three layers, which are the user task producer layer, the scheduling load balancing layer and the workers layer. The implementation of the proposed method uses the agents based middleware for distributed programming JADE tool [8].

The structure of this paper is as follows: In Section II, we formulate and describe the problem presented in this work. Next in Section III we present the technical backgrounds used to develop the proposed scheme. In Section IV, we present the load balancing system used in task routing. In Section V, an example of application using Multiple Program Multiple Data (MPMD) architecture for the distributed image processing, is presented to assess the performance of the proposed framework. In the last section, a conclusion and perspectives are presented.

## II. Problem Description

Basic assumptions and notations used in this paper are listed below in Table I:

In our study, the total execution time: makespan can be expressed as follows:

$$\theta P(t) = \underset{i=1}{\overset{N}{Max}}(\theta P_i(t)) \tag{2}$$

Where:

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

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

$TE_i(t)$ can be formulated as follows:

$$TE_i(t) = \sum_{i=1}^{|Q_i|} \tau_i \tag{4}$$

TABLE I. BASIC ASSUMPTIONS AND NOTATIONS

| Notation | Meaning |
|---|---|
| [N] = {1, 2, . . . , n} | Array of available resource workers. |
| [T] = {1, 2, . . . , m} | Array of Tasks to be executed. |
| $L_i$ | Speed of the network linking the node $M_i$ with the Dispatcher.(see section V. A for further details) $L_i = \theta_i$:- $\theta PN_i$ (1 ) |
| $P_i$ | Computational power of the worker machine $M_i$. |
| $C_i$ | Complexity of task $T_i \in [T]$. |
| $\tau_i$ | Estimation execution time of $T_i \in [T]$ . |
| $TE_i(t)$ | Estimated times of all Tasks wait in line on node ($M_i$) at time t. |
| $\|Q_i\|$ | Number of the wait in line tasks in machine $M_i$. |
| $NBC_i$ | Number of cores of node $M_i$ (CPUs). |
| $NbT_i(t)$ | Number of the wait in line tasks on node $M_i$ at time t. |
| $r_i(t)$ | Reward of machine i at time t. |
| $s_i(t)$ | State of the machine (Mi) at time t. $s_i(t)=\{L_i, P_i, NBC_i, \tau_i, TE_i(t), NbT_i(t), r_i(t)\}$. |
| S(t) | $S(t)=\{s_1(t), s_2(t), .., s_N(t)\}$. |
| R(t) | $R(t)= \{r_i(t)\}i=1..N$. |

Finally, the scheduling problem can be formulated as minimization problem as given below:

$$Min[\theta P(t)] = Min_{\tau \le t} (\underset{i=1}{\overset{N}{Max}}(\theta P_i(t))) = Min_{\tau \le t}\left[\left[\left(\frac{\sum_k \tau_k}{\tau_0}\right) * L_i + TE_i(t)\right] / NBC_i\right] \quad (5)$$

## III. TECHNICAL BACKGROUND

### A. Ant Colony Optimization for the Travelling Salesman Probmem (TSP)

The TSP is a complex problem widely studied in operations research discipline. The problem consists in finding a shorter path on a map that a traveler can choose to visit a list of cities. Each city must be visited once. At the end of the trip, the traveler must return to the starting city. The TSP problem is solved by Dorigo et al. [20] by using the ant colony optimization method (ACO), which engages a set of artificial ants acting parallel searches on a map. The engaged ants choose the following city based on inter-city distances and the amount of pheromone deposited on the paths connecting the cities. However, the pheromone evaporation is also must be taken into consideration to deviate from the local optimum solution.

For ant K, the probability of traveling from city i to city j is given by the next formula [13, 14].

$$P_{ij}^k = \begin{cases} \frac{(\tau_{ij})^a.(\eta_{ij})^b}{\sum_{l \in J_i^k}(\tau_{il})^a.(\eta_{il})^b} & if\ j \in J_i^k \\ 0 & jf\ j \notin J_i^k \end{cases} \quad (6)$$

Where:

$J_i^k$: Represent all neighbor cities of i of the kth ant.

$\tau_{ij}$: Represent the value of pheromone on the path linking city i by city j, a and b control the importance of $\eta_{ij}$ and $\tau_{ij}$.

$$\eta_{ij} = \frac{1}{d_{ij}} \quad (7)$$

Where:$d_{ij}$: Is the distance between cities i and j.

$\tau_{ij}$ is continuously updated between i and j by the following formula:

$$\tau_{ij}(t) = (1 - \rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k(t) \quad (8)$$

Where:
$\rho$ : Pheromone evaporation coefficient.

$\Delta\tau_{ij}^k$ : Deposited Quantity of pheromone by kth ant.

### B. Q-learning

Reinforcement learning (RL) is a recent technique to deal with stochastic and complex problems. By the successive experiences and by trial and error exchanges between executers agent and their residing environment, the agents learn how to act optimally and to cope with more complex situations. This Reinforcement learning model consist of an agent observing its environment, choosing an action from the present state and at that moment winning a positive or negative reward to the action selected. The intention of the agent, during its exploration, is to maximize its total upcoming positive reward and avoid as possible any penalties. Q-learning [25, 26] figure among the most known RL algorithms. Following the same RL philosophy, Q-learning seeks to solve optimally any specified Markov decision problem via a Q function. Q-value can be calculated by (9) listed in the algorithm below:

**Algorithm 1: Q-learning Algorithm**

```
Begin
Initialize Q (s, a) arbitrarily
Initialize s
  Repeat
    Select a' From s' using rule resulting from Q;
    Take action a, observe r, s';
    Q (s, a) = Q(s, a) + α[r +γ(Q(s' ,a') - Q(s, a))];        (9)
    s = s';
  Until s is final state;
End.
```

## IV. Proposed Work

In this section, we propose the fuzzy hybrid algorithm for solving the problem of scheduling tasks on heterogenous distributed machines. Next, we present a case study and show the efficiency of the proposed hybrid method.

### A. Architecture Description

This study purposes to solve a scheduling problem on heterogenous parallel machines. An effective scheduling algorithm plays a significant role in an effective supervision of the grid and so in reducing the maximum completion time. In our suggestion, the load balancing is reached by using a fuzzy hybrid algorithm and a hierarchical mobile agent system.

As shown in Fig. 1, five types of mobile agents are engaged in our proposal: The Producer-Agent, the Tester-Agent, the Dispatcher-Agent, the Controller-Agent and the Worker-Agent. The Producer agent is the one that characterizes the creators of tasks as: web application, mobile application, embedded system (IOT), expert agent (human) and so. The Tester-Agent approximate by previous experiments the execution time of novel tasks. The Dispatcher agent is a type of central manager of the grid and is in charge for distributing new arrival tasks among available workers. Controller-Agent is responsible for continuously monitoring the status of the workers and Worker-Agent performs the tasks received from the dispatcher.

Each machine that seeks to join the grid, at the 3rd layer, as a worker node to participate on the computation must follow these three steps called referencing phase [26]:

*1)* Send a request to the dispatcher to join the workers.

*2)* Receive and perform the referencing task: T0.

*3)* Communicate to the Dispatcher the results of T0 execution: Li, Pi, NBCi.

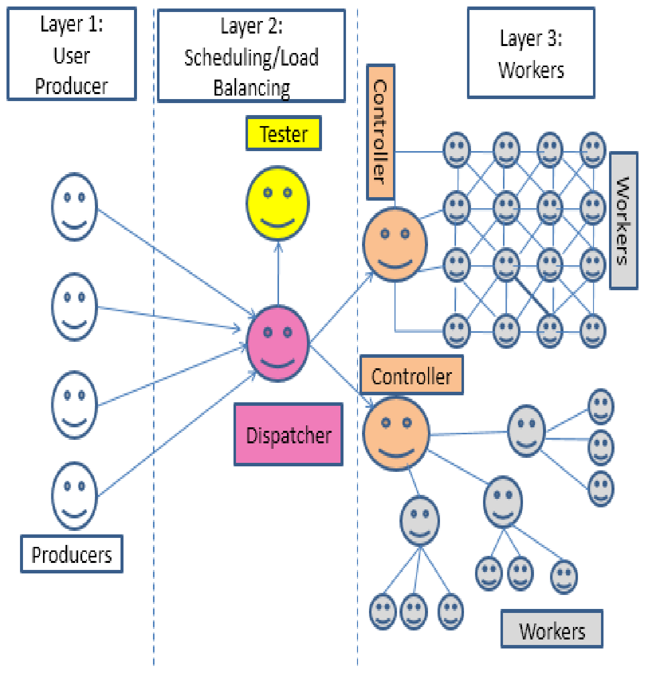

Fig. 1. Framework Architecture [26].

### B. Proposed Fuzzy Hybrid Algorithm: FHA

The design of the proposed fuzzy hybrid algorithm is inspired from the combination of Q-Learning and Ant Colony algorithms. In this method, the hybrid algorithm performs in two parallel phases. In the first phase, ant colony algorithm involves a set of artificial ants acting parallel searches on network links between Dispatcher and resources. Each ant $A_i$ receives from the controller machine, belonging on its network link, the state $s_i(t)$ of the worker $M_i$. All ants $<A_i>_{i=1..N}$ are inside the Dispatcher node and calculate $P_{di}(t)$ according to (10).

$$P_{di}(t)=\frac{\left[r_i(t)^{\alpha}*\frac{1}{NbT_i(t)^{\beta}}*P_i^{\gamma}*\frac{1}{L_i^{\theta}}*NbC_i^{\delta}*\frac{1}{TE_i(t)^{\vartheta}}\right]}{\sum_{j=1}^{N}\left[r_j(t)^{\alpha}*\frac{1}{NbT_j(t)^{\beta}}*P_i^{\gamma}*\frac{1}{L_j^{\theta}}*NbC_j^{\delta}*\frac{1}{TE_j(t)^{\vartheta}}\right]} \qquad (10)$$

Where:

$$\tau_{di}=r_i(t) \qquad (11)$$

$$\eta_{di}{}^{b}=\frac{1}{NbT_i(t)^{\beta}}*P_i^{\gamma}*\frac{1}{L_i^{\theta}}*NbC_i^{\delta}*\frac{1}{TE_i(t)^{\vartheta}} \qquad (12)$$

*$R_i(t)$* refers to the pheromone deposited on the network link connecting the Dispatcher with the worker machine $M_i$. It's also refers to the reward of the worker machine $M_i$ at time t. The Dispatcher rewards powerful machines by a positive value and discourages weak machines by a negative value. The rewards and penalties are continuously calculated in order to supply the Q-Learning by the updated values to select the appropriate worker.

To measure the worker reward, fuzzy logic process is implemented by following in order of these three steps:

*1) Fuzzification of Input*: The first step in the fuzzy inferencing method is the fuzzification. It consists to convert crisp inputs into fuzzy inputs. Crisp inputs are precise inputs calculated in real time by the Dispatcher. As shown in Fig. 2, $P_{di}(t)$ and $\theta P_i(t-1)$ are the two measured crisp inputs passed into the Fuzzy system for treatment to calculate associated crisp output: reward.

In addition, it is important to note that $\theta P_i\,(t-1)$ refers to the execution time of each node in the last previous schedule. This parameter shows the performance of each node in the history and is considered important next to $P_{di}\,(t)$ to calculate the value of the worker's reward.

For each crisp input, a membership function is associated. The two following figures: Fig. 3 and 4 shows the curve of membership functions for : “Node rapidity” and “Node state”.

Regardless of the worker historical, a value of $P_{di}(t)$ that tends to 1 indicates that the worker $M_i$ is the most candidate likely to receive the current task. This also shows that $M_i$ is the most under loaded among all the machines in the grid.

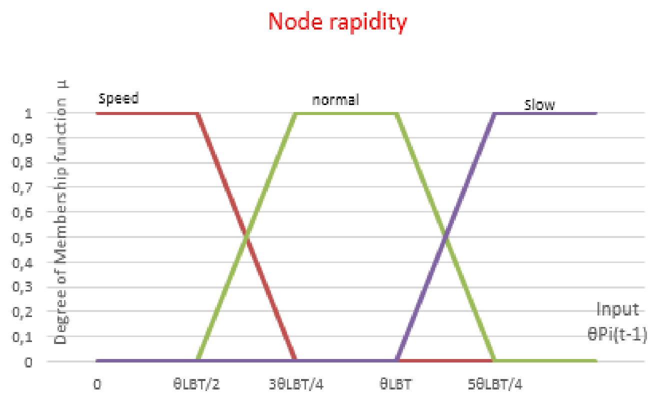

Fig. 2. Input-Node Rapidity.

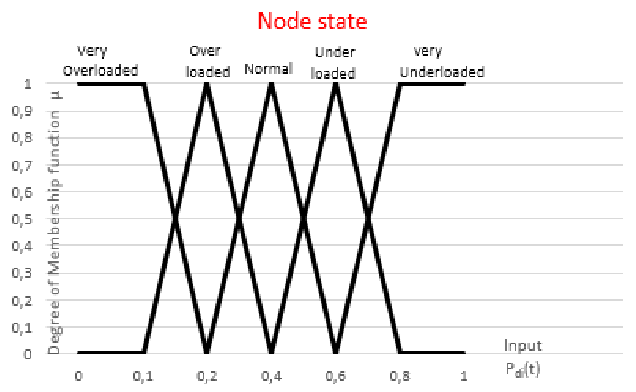

Fig. 3. Input-Node State.

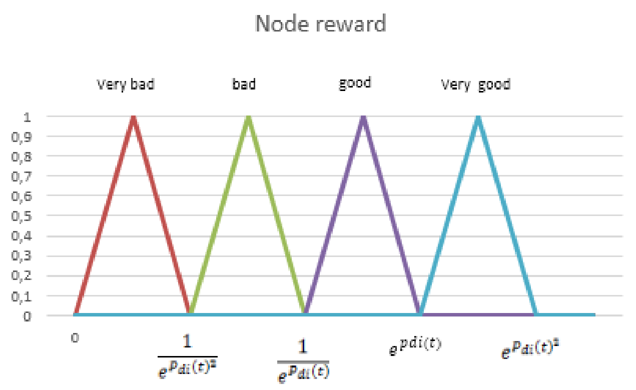

Fig. 4. Output-Node Reward.

The fuzzy system output is a result of all the inputs and the rules. Fig. 4 shows the curve of membership functions for the output: Node reward.

*2) Fuzzy Inference Process-Rules*: Fuzzy inference is the procedure of expressing from specific inputs, an output by using fuzzy logic. This process implies membership Functions, Logical operations, and If-Then Rules. The nature of the rules is the most important parametrization of fuzzy logic systems. It allows concluding the output based on the inputs and the rules. Table II shows the list of fuzzy rules employed to capture the imprecise methods of worker rewarding.

TABLE II. IF THEN-RULES

| **If-Then Rules** | |
|---|---|
| 1 | if node state is (very overloaded or overloaded) and node rapidity is (slow or normal) then node reward is very bad |
| 2 | if node state is normal and node rapidity is (slow or normal) then node reward is bad |
| 3 | if node state is normal and node rapidity is speed then node reward is good |
| 4 | if node state is underloaded and node rapidity is slow then node reward is bad |
| 5 | if node state is underloaded and node rapidity is normal then node reward is good |
| 6 | if node state is very underloaded and node rapidity is (slow or normal) then node reward is good |
| 7 | if node state is (underloaded or very underloaded) and node rapidity is speed then node reward is very good |

*3) Defuzzification*: Defuzzification is the last step that returns crisp output from the fuzzy sets. There are several types of defuzzification methods. The following are the well-known methods: Center of Sums Method (COS), Center of gravity (COG) / Centroid of Area (COA) Method, Center of Area / Bisector of Area Method (BOA), Weighted Average Method and Maxima Methods.

The COG defuzzification is more commonly used fuzzy mathematics method; it defines the output as corresponding to the abscissa of the center of gravity of the surface resulting from the combination of the conclusions and the rules in order to apply the output found to the original problem.

At each computation of $P_{di}(t)$, the reward is calculated by using COG defuzzification as follows:

$$\text{Reward}^* = \frac{\int_S x\mu(x)dx}{\int_S \mu(x)dx}$$

Hence:

$$\tau_{di} = r_i(t) = \frac{\int_S x\mu(x)dx}{\int_S \mu(x)dx} \qquad (13)$$

In the inspired ACO algorithm, the smell of pheromone, as shown in (13), has a direct effect on the calculation of future rewards by the ants. Updating pheromone means that even if a machine has a favorable reward, it does not prevent the dispatcher from continuously reviewing its workload in order to assign it a new value proportional to its state among the other resources. In the second phase, Q-Learning is used to search an optimal action-selection strategy for the tasks allocation. Based on the immediate reward $f(r_i(t))$, the agent Dispatcher updates its estimate for Q by its latest observation from the grid environment. It calculates $Q_i$ for each machine $M_i$ according to (14). Then, it selects the machine having the great value of $Q_i$ for the current task.

$$Q_{t+1}(s_i,a_i) = Q_t(s_i,a_i) + \alpha(r_i(t) + \gamma \max_{a'}(Q_t(s',a') - Q_t(s_i,a_i))) \qquad (14)$$

The grid manager executes parallelly both collaborative algorithms to solve the problem described in this work. Ants are used to calculate reward and Q-Learning is used to schedule the current task to the appropriate worker.

## V. Performance Evaluation and Discussions

In this section, the configuration of the used resources and the results obtained from the fuzzy hybrid algorithm test are described and presented.

The entire system is developed in Java and is tested on a cluster of 10 heterogenous workers. The proposed load distribution process purposes to deploy the agent features to develop a self-directed organization by means of a real multi-agent system based on the JADE platform (Java Agent Development Framework).

The associated scheme aims to build a multi-agent system to schedule jobs on a cluster of 10 heterogeneous machines. The used method is based on the intelligent agents, which must be able to delegate specific tasks.

The configurations of the 10 heterogenous workers is determined by executing a referential task $T_0$ by each machine before joining the grid. The machines have four distinct capacities: $P_i$ ,$L_i$ ,$NBC_i$ .The scheduler must take into account the heterogeneity of node capacities before making scheduling decisions.

### A. Referencing Phase

In this test example, there are ten heterogenous worker nodes that are shown in Table III. These worker configurations, shown below, are determined by executing a referential task $T_0$ [26].

Note that:

$\theta_i$: Total Time required to perform $T_0$ .

$\theta PNi$: Total Time required to perform $T_0$ on Node $N_i$(i=1..n).

$L_i= \theta_i:- \theta PN_i$ (15)

$P_i$ is inversely proportionate to $\theta PN_i$ . $P_i=10^3/ \theta PN_i$ (16)

TABLE III. Parameters Calculation by Referencing Phase

| Node i | Pi (ms) | Li (ms) | NbCi | Reward: |
|---|---|---|---|---|
| 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 |

### B. Q-Learning and ACO Parametrization

The performance metric in searching optimum results depends principally on the parameterization of both Q-Learning and ACO operators.

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

TABLE IV. Best Parameter Settings of the Q-Learning Operators

| α | Υ |
|---|---|
| 0.3 | 0.4 |

The best ACO operators found are shown in Table V.

TABLE V. Best Parameter Settings of the ACO Operators

| α | β | γ | θ | δ |
|---|---|---|---|---|
| 3 | 3 | 3 | 5 | 2 |

### C. Load Balancing Theoretic (LBT)

Theoretically, the load balancing can be calculated for a giving 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 [26].

### D. Scheduling Test by using Fuzzy Hybrid Algorithm (FHA)

The test experiments were generated using a set of *NbT* heterogenous tasks (*NbT= 1000*). To evaluate and measure the system performance, FHA algorithm is put under dissimilar system loads complexity, we select randomly for each task $T_i$:

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

Our hypothesis is tested by making a comparative experiment with the results obtained with scheduling by using Ant colony optimization ACO [25] and by Q-Learning QL [26] under the same controlled configurations.

The scheduling findings are as follows:

Table VI shows the distribution results by using ACO, Q-learning, FHA method, whereas Fig. 5 shows a comparison between their curves duration.

TABLE VI. Distribution Results by Using ACO, Q-Learning, FHA

| Node Ni | θPi(ms) FHA | θPi(ms) Q-L | θPi(ms) ACO |
|---|---|---|---|
| 0 | 70694609 | 91214279,5 | 50694609 |
| 1 | 90009970 | 56255662 | 40009970 |
| 2 | 40009970 | 37788468 | 70171256 |
| 3 | 60171256 | 34039604,5 | 90171256 |
| 4 | 60171256 | 36351290 | 41397431 |
| 5 | 143171256 | 42744272 | 110171256 |
| 6 | 78018383 | 148449863 | 58018383 |
| 7 | 129563908 | 89091486,9 | 149563908 |
| 8 | 55679447 | 165171256 | 65679447 |
| 9 | 85106444 | 99118657,5 | 55106444,3 |

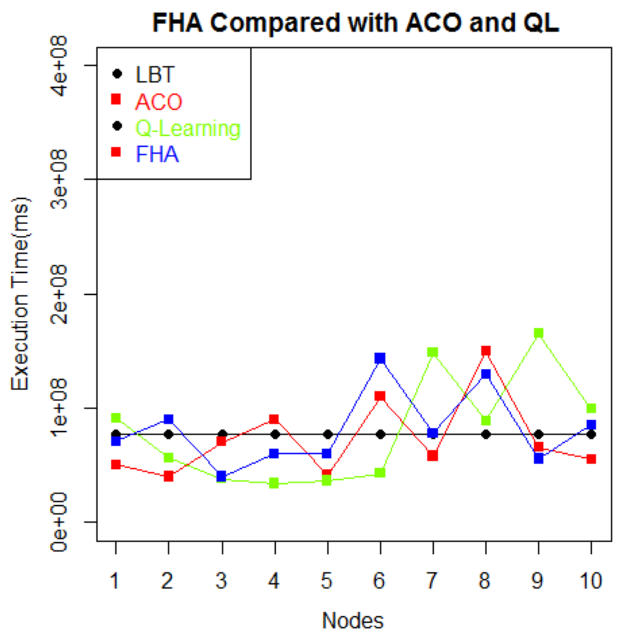

Fig. 5. Comparison Between ACO, Q-learning and FHA Curves Duration.

We use the ratio $\lambda$ to assess the performance of proposed method FHA. The $\lambda$ is expressed as follows.

$$\lambda_{FHA} = \frac{\theta_{FHA}}{\theta_{LBT}} \tag{17}$$

In this section, as shown in Fig. 5, the performance of FHA algorithm is tested in comparison with ACO and QL algorithms, which are two of the well-known scheduling methods for Heterogenous distributed systems as mentioned in [25, 26].

Table VII shows the ratios calculated at the end of each scheduling of 1000 tasks respectively by FHA, ACO, QL.

Clearly, it is shown that the proposed FHA method allows the dispatcher to schedule tasks among the resources much more efficiently than the ACO and QL methods.

The relevant question is whether the system will react in the same way and the same optimality to more complex and more heterogeneous tasks. The answer is of course no, we will not always find the same ratios since it is an np hard problem. Hence the need to develop the practice of artificial intelligence to avoid the repetition of the same scheduling errors and to master as possible the optimal control of all available resources. Looking forward, our goal is to be focused on deep learning through the experience accumulated during scheduling already realized, which allows the dispatcher to review continuously its strategy and then to reduce the total tardiness.

TABLE VII. CALCULATED RATIOS: $\lambda_{FHA}$, $\lambda_{ACO}$, $\lambda_{QL}$

| $\lambda_{FHA}$ | $\lambda_{ACO}$ | $\lambda_{QL}$ |
|---|---|---|
| 1.85 | 1.95 | 2.14 |

## VI. CONCLUSION

In this paper, we have developed a Fuzzy hybrid method called FHA to solve the problem of tasks scheduling. For efficiency purpose, the proposed Framework simultaneously applies two algorithms that solve the same problem. The aim of the proposed hybrid model is to combine the effectiveness of Q-Learning and ACO to reduce the overall execution time and then comes imbalance among available resources. The experiment results showed that the proposed hybrid algorithm has achieved a perfect convergence in terms of the load balancing among available nodes in grid as well as improving the optimal solution.

The proposed method can be extended to hybridize other metaheuristics with RL algorithms in order to minimize as possible the total tardiness.

## REFERENCES

[1] X. Tang, S. Chanson, "Optimizing static job scheduling in a network of heterogeneous computers," the 29 International Conference on Parallel Processing, 2000.

[2] K. Li. Optimal load distribution in nondedicated heterogeneous cluster and grid computing environments. Journal of Systems Architecture, 54(1):111–123, 2008.

[3] Kandagatla, C. (2003). Survey and Taxonomy of Grid Resource Management Systems, University of Texas, Austin. [Online] Available: http://www.cs.utexas.edu/users/browne/cs395f2003 /projects/ File: KandagatlaReport.pdf.

[4] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," IEEE Trans. Ind. Informatics, vol. 9, no. 1, pp. 427–438, 2013.

[5] Kameda, H., Li, J., Kim, C., Zhang, Y.: Optimal Load Balancing in Distributed Computer Systems. Springer,London (1997)

[6] Penmatsa, S., Chronopoulos, A.T.: Price-based useroptimal job allocation scheme for Grid systems. In: Proceedings of 20th IEEE International Parallel and Distributed Processing Symposium, Rhodes, 2006

[7] 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 regenerationtheory approach. IEEE Trans. Parallel Distrib. Syst. 18(4), 485–497 (2007)

[8] 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

[9] 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

[10] 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)

[11] 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)

[12] Zheng, Q.: Dynamic load balancing and pricing in grid computing with communication delay. J. Grid Comput. 6, 239–253 (2008)

[13] F. L. Bellifemine, G.Caire, and D. Greenwood, "Developing Multi Agent Systems with JADE". Wiley, 2007.

[14] 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)

[15] Cho ChoMyint, Khin Mar LarTun, 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.

[16] 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.

[17] 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.

[18] 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

[19] TarekHelmy ,Hamdi Al-Jamimi, Bashar Ahmed, HamzahLoqman .'Fuzzy Logic–Based Scheme for Load Balancing in Grid Services', A Journal of Software Engineering and Applications, pp. 149-156. (December 2012)

[20] A. Galstyan, K. Czajkowski, K. Lerman, Resource allocation in the grid with learning agents, Journal of Grid Computing 3 (2005) 91–100.

[21] Coffman Jr EG, Garey MR, Johnson DS. Approximation algorithms for bin packing: a survey. Approximation algorithms for NP-hard problems, PWS Publishing Co., 1996; 46–93.

[22] Ludwig, S.A., Moallem, A. : Swarm Intelligence Approaches for Grid Load Balancing.J Grid Computing 9, 279–301 (2011)

[23] Kwang, M.S., Sun, H.W.: Ant colony optimization for routing and load-balancing: survey and new directions.IEEE Trans. Syst. Man Cybern. Part A33(5), 560–572(2003)

[24] Buyya,R.,Murshed,M. :GridSim :a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing.Journal of concurrency and computation practice and experience 14(13-15),1175-1220(2002)

[25] Y. Hajoui, O. Bouattane, M. Youssfi, and E. Illoussamen, "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).

[26] Y. Hajoui, O. Bouattane, M. Youssfi, and E. Illoussamen, "Q-Learning applied to the problem of scheduling on heterogeneous architectures". International Journal of Computer Science and Network Security, vol. 18, no. 2, pp. 153–159, 2018.

[27] R. V. Lopes and D. Menasce, "A Taxonomy of Job Scheduling on Distributed Computing Systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 12, pp. 3412–3428, 2016.

[28] M. I. Daoud and N. Kharma, "A high performance algorithm for static task scheduling in heterogeneous distributed computing systems,"J. Parallel Distrib. Comput., vol. 68, no. 4, pp. 399–409,2008

[29] W. E. Walsh, G. Tesauro, J. O. Kephart, and R. Das, "Utility functions in autonomic systems," inProc. Int. Conf. Autonomic Comput.,May. 2004, pp. 70–77.

[30] J. Koodziej and S. U. Khan, "Multi-level hierarchic genetic-based scheduling of independent jobs in dynamic heterogeneous grid environment,"Inform. Sci., vol. 214, pp. 1–19, 2012.

[31] J. Ullman, "NP-complete scheduling problems," J. Comput. Syst. Sci., vol. 10, no. 3, pp. 384–393, 1975.

[32] M. Drozdowski, Scheduling for Parallel Processing, 1st ed. New York, NY, USA: Springer, 2009.