sciendo

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **22**, No 4
Sofia • 2022 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2022-0038

Fuzzy Neutrosophic Soft Set Based Transfer-Q-Learning Scheme for Load Balancing in Uncertain Grid Computing Environments

Bhargavi K[1], *Sajjan G. Shiva*[2]

[1]*Department of CSE, Siddaganga Institute of Technology, Tumakuru, Karnataka, India*
[2]*Department of CS, University of Memphis, Memphis, Tennessee, USA*
E-mails: bhargavik@sit.ac.in Sshiva@memphis.edu

***Abstract*:** *Effective load balancing is tougher in grid computing compared to other conventional distributed computing platforms due to its heterogeneity, autonomy, scalability, and adaptability characteristics, resource selection and distribution mechanisms, and data separation. Hence, it is necessary to identify and handle the uncertainty of the tasks and grid resources before making load balancing decisions. Using two potential forms of Hidden Markov Models* (*HMM*)*, i.e., Profile Hidden Markov Model* (*PF_HMM*) *and Pair Hidden Markov Model* (*PR_HMM*)*, the uncertainties in the task and system parameters are identified. Load balancing is then carried out using our novel Fuzzy Neutrosophic Soft Set theory* (*FNSS*) *based transfer Q-learning with pre-trained knowledge. The transfer Q-learning enabled with FNSS solves large scale load balancing problems efficiently as the models are already trained and do not need pre-training. Our expected value analysis and simulation results confirm that the proposed scheme is 90 percent better than three of the recent load balancing schemes.*

***Keywords*:** *Transfer-Q-learning, load balancing, Grid, Performance, Fuzzy Neutrosophic Soft Set, Uncertainty.*

1. Introduction

The grid-computing environment is composed of several clusters of computing devices dispersed over a large geographical area and operates in a coordinated manner as a virtual supercomputing machine to perform computationally intensive tasks [1, 2]. Some of the potential challenges in grid computation include heterogeneity of hardware and software resources, multiple administration of resources, inefficient load balancing, different protocols at different layers, lack of trust in grid data models, poor stability, huge data handling by a single platform, large scale data manipulation, improper pooling of resources, etc., [3, 4]. Among the challenges faced by the grid-computing environment, load balancing is one that affects the effective performance. The main objectives of the paper is to address load balancing issue in the grid computing environment in following steps:

- Handling uncertainty in grid resources and tasks.
- Developing a transfer Q-learning scheme for load balancing.
- Formulating high quality load balancing policies by implementing the proposed scheme using SimGrid software.
- Results obtained are evaluated against performance metrics like Execution time, Response time, Learning rate and Throughput.

Load balancing is tougher in grids compared to other conventional distributed computing platforms due to several factors that include heterogeneity, autonomy, scalability, adaptability, resource selection, resource distribution, data separation, etc, [5, 6]. The uncertainty in this dynamic computing environment also affects the performance. The main sources of uncertainty include variety of incoming data, virtualization of resources, frequent migration of computing tasks, high consumption of energy, dynamic pricing of computation models, grids distributed among wide geographical area, elastic provisioning of grid resources, frequent variation in the task processing time, etc. This uncertainty affects several performance parameters of the grid which include bandwidth consumption, task processing time, memory utilization, capacity of the resources and network, number of computing devices involved in the computation, vulnerable to attacks, computational complexity, etc. Hence, it is necessary to identify and handle the uncertainty of the tasks and grid resources before making load balancing decisions.

The Profile Hidden Markov Model (PF_HMM) and Pair Hidden Markov Model (PR_HMM) are potential forms of Hidden Markov Models (HMM) used in identification of the uncertainty in the system parameters by measuring both visible states and partial observable state parameters of the system model [7, 8]. Some of the potential applications of HMM include speech recognition, digital communication, spicing signal prediction, convergence of multiple user activities, vehicle trajectory projection, stock market prediction, protein family profiling, gene finding and so on [9, 10].

Neutrosophic Soft Set theory (NSS) is a mathematical framework, which solves decision-making problems under uncertainties and imprecision environments. It considers the truth, falsity, and indeterminacy membership functions to solve real world problems. The “neutrosophic” is originated from “neutrosophy” which means neutral. Every component of NSS is explained by considering three contrast estimates of data: inaccurate, absurd, and condensed form [11-13]. Fuzzy Neutrosophic Soft Set theory (FNSS) is one of the parameterized families of neutrosophic soft set theories in the universe that effectively handles parameter uncertainty in distributed computing environment through approximation of the parameters. FNSS handles the uncertainty in measured parameters using three approximation metrics: truth, false and indeterminate values. The FNSS is being widely used in several applications in engineering, economics, medical science, smart framing, and many more as it makes effective decisions by reducing the uncertain parameters to choose the optimal set of parameters [14, 15].

Q-learning is a value-based reinforcement learning algorithm that computes optimal action using Q-function. Once it is enabled with transfer learning it basically stores the optimal actions performed while solving the problem and makes use of

those actions in solving similar kinds of problems [16, 17]. The transfer Q-learning is suitable for solving large scale problems since the models used are already trained and pre-training is not required. Exascale computation is carried out with less computational power and models learn with small amounts of data. The learning rate achieved is very high as uncertainty is handled using Q-function enriched with FNSS [18-20]. In this paper, uncertainties in the tasks and resources are identified using PF-HMM and PR-HMM. FNSS theory handles the identified uncertainties and on top of which we design the transfer Q-learning scheme to perform uncertainty aware load balancing in the grid [21].

The novelty of the contributions made in the paper are as follows.

- Precise identification of uncertainties in the grid resources and tasks using PF_HMM and PR_HMM and FNSS theory. This leads to efficient handling of the parameter uncertainties in dynamic computing setup.
- Possibility of convergence towards suboptimal load balancing solutions is less due to reduction in Bellman error function and efficient smoothing of system parameters.
- The Q states are not overloaded due to efficient utilization of learnt knowledge and reduced error in transferred target Q-function.
- A novel FNSS based transfer Q-learning scheme is designed for load balancing in grid supported by algorithms and mathematical definitions. The use of FNSS theory improves the decision-making ability of the proposed scheme and prevents underutilization and overutilization of resources.
- Expected value analysis of the proposed FNSS based transfer Q-learning scheme for various performance metrics has been done.
- The proposed FNSS based transfer Q-learning scheme is tested against other recent works using SimGrid simulator considering uncertainty of tasks and resources.

The paper is organized as follows. Section 2 discusses related works. Section 3 discusses the system model considered. Section 4 provides definitions for the performance metrics considered for evaluation purposes and high-level view of grid resource and task models. Section 5 presents the proposed architecture for load balancing. Section 6 presents the results and discussion, and finally Section 7 draws the conclusion.

2. Related work

K h a n et al. [22] provide a survey of the load balancing strategies for grid computing environments. During static load balancing, distribution of incoming tasks among the available grid resources is done through a fixed schema. During dynamic load balancing the allocation and reallocation of tasks, are done dynamically during execution time. The strategies reported either support or does not support task migration and involve flat and hierarchical topologies. For formulating load balancing policies four types of policies are involved: information, location, transfer, and selection. Some of the important factors influencing the load balancing solutions are scalability, fine grained tasks, hierarchical topology, heterogeneity of resources, dependencies over the tasks, fault tolerance, and resource processing. Performance

metrics include response time, communication overhead, resource utilization, communication delay, and throughput.

W e n j i e et al. [23] discuss the work-stealing algorithm to balance the load in grid computing environment. This algorithm works in two main stages: resource discovery and scheduling of the workflow tasks. In resource discovery stage, identification of available resources in the grid systems is done by continuously collecting the information on registered grid resources from grid information centre. During scheduling of the workflows, distribution of incoming tasks among the resources is done by comparing the resource requirement and resource availability ratio. Here the main structure used in load balancing is Directed Acyclic Graph (DAG) that depicts the requirements of the tasks. The main drawback of this strategy is mapping of tasks onto the resources without considering the uncertainty factor. The latency incurred in task propagation is also very high due to the use of DAG task representation model.

W u et al. [24] discuss load balancing using intelligent agents across the grid. Scalable scheduling of the tasks onto the resources in the grid environment demands the use of artificial intelligence techniques. Here the multiple-agents approach is used for load balancing with six types of agents i.e., worker agent, load balancing agent, resource agent, migration agent, cluster agent and grid agent. During the load imbalance situation, the cluster agent is responsible for setting up the equilibrium threshold by gathering the resource information from each of the resource agents using a knowledge algorithm. The migration agents transfer the worker agent from underloaded grid resources to the overloaded grid resources. Finally, the grid agents are responsible for transferring the tasks onto the grid resources. Multiple agents are very well coordinated and proper synchronization is established among them. Each agent learns independently by considering local state and reward without explicit communication among one another. This method can explore the large state and action pair and solves the convergence problem under non-stationary environment of grid. Even though the approach is found to be good in handling load imbalance situations while dealing with the computationally intensive problems, the execution time efficiency is poor.

H a j o u i et al. [25] use Q-learning to address the problem of task scheduling on heterogeneous architectures that deals with complicated applications that demand computationally intensive operation. By designing a three-layered framework for scheduling of tasks efficiency are achieved utilizing producer agents in Layer 1, scheduling and load balancing agents in Layer 2 and worker agents in Layer 3. The load balancer uses Q-learning algorithm to handle the load imbalance situation by making use of the knowledge gained during the previous scheduling problem. Load balancing decisions are taken using Q-algorithm. Optimal actions are selected using Markov decision process and maximize the entire reward collection policy. The Q-function is updated by performing each action with optimal value for the discount factor. The core idea of the algorithm is using a simple formula for the reward calculation and computing the threshold value to determine overloaded and underloaded nodes. The learning process involves dynamism of grid structure and service requirement of applications. The approach is good in reducing the tardiness

of tasks distribution and scheduling but often ends up in suboptimal solution due to premature convergence [25].

Garcia-Galan, Prado and Expósito [26] present load balancing based on fuzzy scheduling and use swarm intelligence for gaining the knowledge of the distributed grid environment. The Fuzzy rule-based system is designed to act as the grid middleware system. The grid dynamism is handled in two ways where in that are grid states are characterized using fuzzy notations and learning mechanism is automatically adopted for changing conditions. The main advantage of combining fuzzy rule system with swarm intelligence is its ability to handle vagueness in the computing environment and its easy adaptation to the dynamic changes in the grid conditions. The basic structure of the scheduler consists of three main components that are fuzzification, inference, and defuzzification. The probability of determining good rules keeps increasing with the increase in the number of features in the grid search space. However, the use of fuzzy rule-based system leads to limited acquisition of knowledge due to the predefined fixed number of rules. The stopping condition to arrive at the termination results are completely based on the statistical analysis results. Hence, the ability to self-improve the quality of scheduling policies is totally less [26].

Tang et al. [27] deal with the task scheduling and load balancing policy in grid computing using a memory-based algorithm. The task scheduling plays a very important role in load balancing as it directly affects the response time incurred by the computing system. The dynamic scalability and scheduling of tasks are dependent on the behaviour statistics of the computing nodes. The behaviour and unique characteristics of the heterogeneous tasks are handled through continuous monitoring approach. The distributed Particle Swarm Optimization (PSO) algorithm with the memory function generates dynamic and scalable scheduling policies for heterogeneous computing environment. The PSO algorithm outperforms the stochastic algorithm in producing high quality scheduling policies due to its easily adaptable nature and requires only few parameter adjustments at the time of learning. However, the algorithm ends up in local optimum solution in high dimensional computing space like grid and the convergence rate is slow due to the number of iterations of training and testing.

Patni [28] presents a centralized strategy for load balancing across homogeneous form of grid computing environment. A load balancing solution based on typical client and server architecture is proposed to improve the system performance with maximum utilization of grid resources. A node is designated as monitored node, which gathers information from neighbouring computing nodes and sends it to the resource allocation node that does the load-balancing task. Higher level of optimization is achieved by transferring the tasks from overloaded nodes to underloaded nodes during load imbalance situation. The transferring of tasks happens at high-speed using Gridftp service which offers faster and reliable transfer. The complexity of task transferring is less as the overload and error problems are reduced. However, the approach depends on centralized server for load balancing decisions, which becomes susceptible to single point failures.

Ali and Bouakkaz [29] discuss an agent-based mechanism for load balancing in grid. The self-adaptive and self-sustaining capability of autonomous agents is used to handle load imbalance situations. The agent cluster gathers current load information from the resource agent. The load information gathering happens at several stages, which involve arrival of new resources, withdrawal of resources, termination of load agent, start of local worker agent, receiving incoming agent, departure of mobile agent, and assignment of workload to agent workers. A balance threshold is setup based on the collected information to handle load imbalance situations. The capacity of the agent is determined by assigning credit value for each of the worker agents. The worker agent, which has the highest credit value, will be provided more opportunity for storing the current location. By gathering the global state of each of cluster, the overloaded cluster will transfer the worker agents to underloaded clusters. However, they fail to handle failed entities recognition, and synchronization between multiple agents is not achieved.

In summary, the limitations identified in the existing works are as follows.

- Hidden uncertainties in the tasks and resources parameters go unidentified.
- Most of the load balancing policies produced for large dimensional space like grid end up in local optimum solution and the rate of convergence is too slow, due to high training requirements.
- Delay and latency incurred in mapping of the tasks onto the grid resources are high due to the use of poor representation models.
- The solutions proposed suffer scalability issues due to the formulation of load balancing policies based on the limited knowledge gained over the application requirements.
- The load balancing decisions without considering the uncertainty, leads to low quality of decisions causing over and underutilization of resources.
- Wastage of previously gained knowledge of the load balancer may result if such knowledge is not reused in similar situations.
- Most of the algorithms lead to premature convergence as a result the chances of picking global optimal solution are less.
- Underutilization and overutilization of the resources occur due to inappropriate load balancing decisions, as most of the algorithms are reactive in approach and shift the load only in the case of load imbalance situations.

3. System model

A typical Grid Computing Platform (GCP) is a widely distributed computing domain composed of several regions,

$$\mathrm{GCP} = \{R_i\}_{i=0}^{i=k}. \tag{1}$$

A Region (R_i) is a collection of several groups. Each group consists of infinite amount of grid resources,

$$R_i = \{G_i\}_{i=0}^{i=\infty}. \tag{2}$$

A Group (G_i) is collection of several computing devices and each computing device has infinite amount of grid resources,

(3) $$G_i = \{\mathrm{CD}_i\}_{i=0}^{i=\infty}.$$

The load imbalance can occur at two levels, one is at intra-region level, and the other is at inter-region level. At first the transfer Q-learning agents are deployed within the regions which are referred to as Intra Transfer Q-Learning Agents (IT_TQLAs) to balance the load within the region,

(4) $$R = \{R_i \leftarrow \mathrm{IT_TQLA}_i, R_j \leftarrow \mathrm{IT_TQLA}_2, \ldots, R_k \leftarrow \mathrm{IT_TQLA}_k\}.$$

Then the transfer Q-learning agents are deployed between the regions which are referred to as inter transfer Q-learning agents (IR_TQLAs) to balance the load between the regions,

(5) $$(R_i, R_j) \leftarrow \mathrm{IR_TQLA}_i,\ (R_j, R_k) \leftarrow \mathrm{IR_TQLA}_j, \ldots,\ (R_k, R_l) \leftarrow \mathrm{IR_TQLA}_k.$$

The intra transfer Q-learning agents store the optimal set of actions performed for load balancing within the region and replay it whenever similar load imbalance situation is encountered within the region. Suppose if the load imbalance situation of R_i and R_j are same then the optimal set of actions performed in R_i get replayed in R_j,

(6) $$R_i \leftarrow \mathrm{IT_TQLA}_i \leftarrow \mathrm{Store}\{O(A_i), \ldots, O(A_i)\},$$

(7) $$R_j \leftarrow \mathrm{IT_TQLA}_i \leftarrow \mathrm{Replay}\{O(A_i), \ldots, O(A_i)\}.$$

Similarly inter transfer Q-learning agents store the optimal actions performed for load balancing between the regions and replay it whenever similar load imbalance situations occur. Suppose if the load imbalance situation between (R_i, R_j) is same as (R_j, R_k) then the optimal set of actions performed in $(R_i, R_j)$is replayed in (R_j, R_k),

(8) $$(R_i, R_j) \leftarrow \mathrm{IR_TQLA}_i \leftarrow \mathrm{Store}\{O(A_i), \ldots, O(A_i)\},$$

(9) $$(R_i, R_j) \leftarrow \mathrm{IR_TQLA}_i \leftarrow \mathrm{Replay}\{O(A_i), \ldots, O(A_i)\}.$$

4. Definitions

This section provides the definition for the performance metrics used in the paper.

Execution time. Execution Time (ET) of the Transfer Q-learning Agent $\mathrm{ET(TQA, GCE)}$ is the summation of the time taken by the transfer Q-learning agent in processing the incoming tasks $T_{\mathrm{TP}(t_i)}$ with varying QoS requirements onto the appropriate grid resources for successful completion of the tasks $T_{\mathrm{SC}(t_i \to \mathrm{gr}_j)}$,

(10) $$\mathrm{ET(TQA, GCE)} = \sum_{i=1,j=1}^{i,j=n} [T_{P(t_i)} + T_{\mathrm{SC}(t_i \to \mathrm{gr}_j)}].$$

Response time. Response Time (RT) of the transfer Q-learning agent $\mathrm{RT(TQA, GCE)}$ is the time difference between the arrival time of the incoming tasks $T_{A(t_i)}$ and successful completion time of the tasks $T_{\mathrm{SC}(t_i \to \mathrm{gr}_j)}$,

(11) $$\mathrm{RT(TQA, GCE)} = \sum_{i=1,j=1}^{i,j=n} [T_{A(t_i)} - T_{\mathrm{SC}(t_i \to \mathrm{gr}_j)}].$$

Learning rate. Learning Rate (LR) of the transfer Q-learning agent $\mathrm{LR(TQA, GCE)}$ is the speed at which mapping of the incoming tasks onto the appropriate grid resources happen,

(12) $$\mathrm{LR(TQA, GCE)} = \sum_{i=1,j=1}^{i,j=n} \frac{N_{\mathrm{SC}(t_i \to \mathrm{gr}_j)}}{\mathrm{ET(TQA,GCE)}}.$$

Throughput. Throughput achieved by the transfer Q-learning agent $\mathrm{TH}(\mathrm{TQA}, \mathrm{GCE})$is the measure of number of successfully completed tasks $N_{\mathrm{SC}(t_i \to \mathrm{gr}_j)}$ by the transfer Q-learning agent out of the total number of tasks allocated $T_{A(t_i \to \mathrm{gr}_j)}$.

$$\mathrm{TH}(\mathrm{TQA}, \mathrm{GCE}) = \sum_{i=1, j=1}^{i,j=n} \frac{N_{\mathrm{SC}(t_i \to \mathrm{gr}_j)}}{T_{A(t_i \to \mathrm{gr}_j)}}. \tag{13}$$

4.1. Grid resource model

The availability and capability of the grid resources is modelled using the Profile Hidden Markov Model (PF_HMM) which vary dynamically since the grid is a large scalable computing environment. The uncertainty in grid resources is of various kind including execution failures, overutilization of CPU, variation in the availability of processors, rise and drop in the processor speed, lack of resource integration models and many more [30].

The PF_HMM is a statistical uncertainty-handling model that converts the data obtained from multiple sequence alignments into collection of probability values that is helpful in early identification of the variation levels in the measured grid resource parameters. Some of the potential functions performed by PF_HMM towards uncertainty handling include approximation of missing parameters in the measured data, removal of deleted parameter states, identification of variation in parameters states, alignment of parameter sequence without losing the precision, unbiased investigation of position specific parameter errors and many more. PF_HMM helps in identification and isolation of the uncertainties in the grid resources.

The PF_HMM applied over the grid resource consists of six attributes,

$$\mathrm{PF}_{\mathrm{HMM}} = \langle \mathrm{gr}_i, Q(\mathrm{gr}_i), P_{\mathrm{t}}(\mathrm{gr}_i), P_{\mathrm{d}}(\mathrm{gr}_i), P_{\mathrm{e}}(\mathrm{gr}_i), \pi_0(\mathrm{gr}_i), \pi_{\mathrm{f}}(\mathrm{gr}_i) \rangle, \quad \mathrm{gr}_i \in \mathrm{GR}, \tag{14}$$

where:

$Q(\mathrm{gr}_i)$ = Set of states of the grid resource;

$P_{\mathrm{t}}(\mathrm{gr}_i)$ = Set of state transition probabilities, i.e., $P_{\mathrm{t}}(\mathrm{gr}_i) = Q(\mathrm{gr}_i) * Q(\mathrm{gr}_i) \to R$, R stands for real;

$P_{\mathrm{d}}(\mathrm{gr}_i)$ = Set of duration probabilities, i.e., $P_{\mathrm{d}}(\mathrm{gr}_i) = N * Q(\mathrm{gr}_i) \to R$, N stands for non-negative integer;

$P_{\mathrm{e}}(\mathrm{gr}_i)$ = Set of emission probabilities, i.e., $P_{\mathrm{e}}(\mathrm{gr}_i) = \alpha^* * N * Q(\mathrm{gr}_i) \to R$, α^* stands for strings over α;

$\pi_0(\mathrm{gr}_i)$ = Starting state runof PF_HMM;

$\pi_{\mathrm{f}}(\mathrm{gr}_i)$ = Final state run of PF_HMM.

The final PF_HMM refined grid resource parameters are as follows,

$$\theta^* = \underset{\theta}{\mathrm{argmax}} \left(\sum \frac{\prod_{i=1}^{i=n} P_{\mathrm{e}}(\mathrm{gr}_i) P_{\mathrm{t}}(\mathrm{gr}_i) P_{\mathrm{d}}(\mathrm{gr}_i)}{P(S|\theta)} \right). \tag{15}$$

The Truth, false and indeterminate values of FNSS, i.e., FNSS(GR) are as follows:

$$\mathrm{FNSS}(\mathrm{HMM}(\mathrm{GR})) = \{ \langle \mathrm{gr}_i, T_{\mathrm{FNSS}}(\mathrm{gr}_i), I_{\mathrm{FNSS}}(\mathrm{gr}_i), F_{\mathrm{FNSS}}(\mathrm{gr}_i) \rangle, \mathrm{gr}_i \in \mathrm{GR} \}. \tag{16}$$

The truth, false and indeterminate values of FNSS based grid resources fall within the closed interval of zero and one:

$$T_{\mathrm{FNSS}}(\mathrm{gr}_i), I_{\mathrm{FNSS}}(\mathrm{gr}_i), F_{\mathrm{FNSS}}(\mathrm{gr}_i) \to [0, 1].$$

4.2. Task model

The Quality of Service (QoS) requirement of the tasks varies dynamically over time that demands for constant evaluation of the requirements for readjustment of the load balancing policies [31]. It is modelled using Pair Hidden Markov Model (PR_HMM). The uncertainty in incoming task model is due to sudden variation in the task demands, increasingly demanding applications, poor correlation between the tasks and resources, lack of customizable resource requirement policies, elasticity of the task demands, unpredictable spike in the incoming task traffic, etc.

The PR_HMM is a kind of HMM which provides probability-based distribution over the specific pairs of observations made over the task parameters. The PR_HMM is an extension of PF_HMM in which inferences are drawn based on the two observation sequences of the input parameters instead of one. Some of the potential functions performed by PR_HMM towards uncertainty handling include pair wise alignment of the parameters, making multiple observations to track the mobility of the incoming parameters, liner interpolation of high variability parameters, identification of structural variability, achieving fine-grained control over the linear flow of parameters, etc. The PR_HMM handles uncertainties in the incoming tasks.

The PR_HMM over the task consists of six attributes,

(17) $\text{PR_HMM} = \langle t, Q(t_i), P_{\mathrm{t}}(t_i), \varphi_{\mathrm{d}}(t_i), \varphi_{\mathrm{e}}(t_i), \pi_0(t_i), \pi_{\mathrm{f}}(t_i)\rangle, t_i \in T,$

where:

$Q(t_i)$ = Set of states of the grid resource;

$P_{\mathrm{t}}(t_i)$ = Set of state transition probabilities, i.e., $P_{\mathrm{t}}(\mathrm{gr}_i) = Q(\mathrm{gr}_i) * Q(\mathrm{gr}_i) \rightarrow R$, R stands for real;

$\varphi_{\mathrm{d}}(t_i)$ = Joint distribution of paired duration, i.e., $P_{\mathrm{d}}(\mathrm{gr}_i) = N * N * Q(\mathrm{gr}_i) \rightarrow R$, N stands for non-negative integer;

$\varphi_{\mathrm{e}}(t_i)$ = Joint distribution of paired emission, i.e.,

$$P_{\mathrm{e}}(\mathrm{gr}_i) = \alpha^* * \alpha^* * N * Q(\mathrm{gr}_i) \rightarrow R, \alpha^*$$

stands for strings over α;

$\pi_0(t_i)$ = Starting state run of PR_HMM;

$\pi_{\mathrm{f}}(\mathrm{gr}_i)$ = Final state run of PR_HMM.

The final PR_HMM refined grid resource parameters are as follows:

(18) $$\theta^* = \underset{\theta}{\operatorname{argmax}} \left(\sum \frac{\prod_{i=1}^{i=n} \varphi_{\mathrm{e}}(t_i) P_{\mathrm{t}}(t_i) \varphi_{\mathrm{d}}(t_i)}{P(S|\theta)} \right).$$

FNSS handles the uncertainty in the incoming tasks using true, false, and indeterminate values, i.e.,

(19) $\text{FNSS}(\text{PR_HMM}(T)) = \{\langle t_i, T_{\mathrm{FNSS}}(t_i), I_{\mathrm{FNSS}}(t_i), F_{\mathrm{FNSS}}(t_i)\rangle, t_i \in T\}.$

The true, false, and indeterminate values of FNSS based incoming tasks falls within the closed interval of zero and one $T_{\mathrm{FNSS}}(t_i), I_{\mathrm{FNSS}}(t_i), F_{\mathrm{FNSS}}(t_i) \rightarrow [0, 1]$.

5. Proposed work

Fig. 1 gives the high-level architecture of the FNSS based transfer Q-learning scheme for load balancing. The architecture is composed of three functional components, i.e., Fuzzy Neutrosophic Soft Set Task Uncertainty Handler (FNSS-TUH), Fuzzy Neutrosophic Soft Set Resource Uncertainty Handler (FNSS-RUH) and Inter/Intra

Transfer Q-Learning Agent (I/I_TQLA). The FNSS-TUH handles the uncertainty in the task parameters; FNSS-RUH handles the uncertainty in the grid resource parameters. The I/I_TQLA formulate load-balancing policies within the region and between the regions using TQLA. Fig. 2 shows the flow of the proposed scheme.

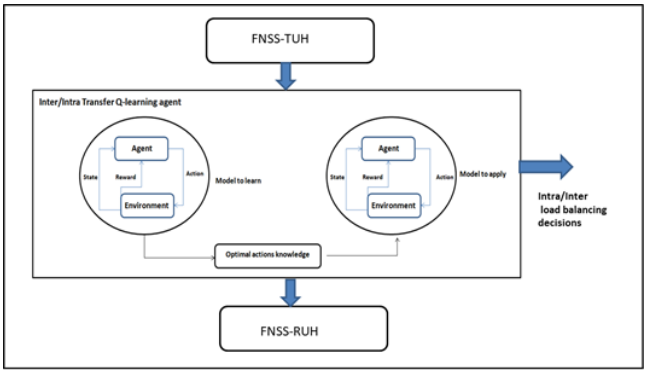


Fig. 1. High-level architecture of the FNSS based transfer Q-learning scheme for load balancing

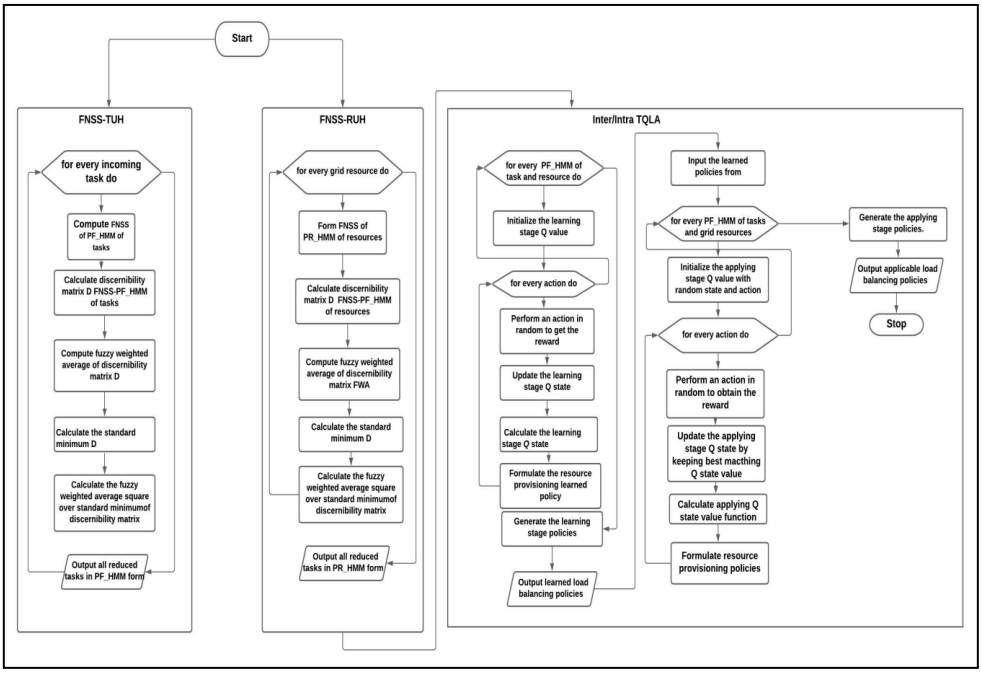


Fig. 2. Flowchart of the proposed FNSS based transfer Q-learning scheme for load balancing

5.1. Fuzzy Neutrosophic Soft Set Task Uncertainty Handler (FNSS-TUH)

The FNS-TUH inputs the PF_HMM enabled tasks to generate reduced PF_HMM enabled tasks by using FNSS theory. The main goal is to use Discernibility matrix and weighted average of discernibility matrix to remove uncertainty in the tasks and normalize it. By performing weighted Average Square over standard minimum of discernibility matrix, the uncertainty free task parameters are obtained. The task

parameters are transferred into set of subsets of task attributes by considering each of the elements of the task matrix. The operation begins by considering partial set of subsets of tasks and through absorption operation, the superset of task parameters is removed. Through absorption and grouping operations the partial subset of tasks is deleted, this modified representation of tasks is repeated in loop until final reluctant of reduced task set is produced. Algorithm 1 gives the working of FNSS-TUH.

Algorithm 1. Working of FNSS-TUH

Step 1. Begin

Step 2. *Input:* PF_HMM$(t_i) = (Q(t_i), P_{\mathrm{t}}(t_i), \varphi_{\mathrm{d}}(t_i), \varphi_{\mathrm{e}}(t_i), \pi_0(t_i), \pi_{\mathrm{f}}(t_i))$

Step 3. *Output:*

PF_HMM$(t_i)^{\mathrm{rd}}(Q(t_i)^{\mathrm{rd}}, P_{\mathrm{t}}(t_i)^{\mathrm{rd}}, \varphi_{\mathrm{d}}(t_i)^{\mathrm{rd}}, \varphi_{\mathrm{e}}(t_i)^{\mathrm{rd}}, \pi_0(t_i)^{\mathrm{rd}}, \pi_{\mathrm{f}}(t_i)^{\mathrm{rd}})$

Step 4. for every $t_i \in T$ **do**

Step 5. Form FNSS of PF_HMM (t_i)

Step 6. PF_HMM (t_i) = $(t_i, T_{\mathrm{FNSS}}(t_i), I_{\mathrm{FNSS}}(t_i), F_{\mathrm{FNSS}}(t_i))$

Step 7. Calculate discernibility matrix D (FNSS-PF_HMM (t_i, t_j)), i.e.,

Step 8. D (FNSS-PF_HMM (t_i, t_j)) =$\{\mu(a) \in A | g(t_i, \mu(a)) \neq g(t_j, \mu(a))\}$

Step 9. Compute fuzzy weighted average of discernibility matrix, i.e., FWA $= \sum_{i=1}^{i=n}(w_i * \mu(a))$, where w_i is the assigned weight.

Step 10. D (FNSS-PF_HMM (t_i, t_j)) =

Φ	$\{a_{1/\mathrm{FWA}}\}$	$\{a_{1/\mathrm{FWA}}, a_{4/\mathrm{FWA}}\}$
....	....	.....
$\{a_{1/\mathrm{FWA}}, a_{3/\mathrm{FWA}}, a_{4/\mathrm{FWA}}\}$	...	Φ

Step 11. Calculate the standard minimum $\Delta^* D$ (FNSS-PF_HMM (t_i, t_j)) =

$\Delta^* D(\mathrm{FNSS} - \mathrm{PF_HMM}(t_i, t_j)) = (\mu(a_i) \wedge \mu(a_j)) \vee (\mu(a_k) \wedge \mu(a_l))$

$\{a_1^*, a_6^*\}$	$\{a_1^*\}$	$\{a_1^*, a_6^*\}$
....	....	.....
$\{a_6^*\}$	...	$\{a_4^*, a_7^*\}$

Step 12. Calculate the fuzzy weighted average square over standard minimum of discernibility matrix, i.e., FWA^2=$\sum_{i=1}^{i=n}(w_i * \mu(a))^2$, i.e.,

Step 13. D (FNSS-PF_HMM (t_i, t_j)) =

$\{a_1^*/\mathrm{FWA}^2, a_6^*/\mathrm{FWA}^2\}$	$\{a_3^*/\mathrm{FWA}^2\}$	$\{\Phi^*/\mathrm{FWA}^2\}$
....	....	.....
$\{a_6^*/\mathrm{FWA}^2\}$	...	$\{a_4^*/\mathrm{FWA}^2, a_7^*/\mathrm{FWA}^2\}$

Step 14. End for

Step 15. Output all reduced tasks in PF_HMM(t_i) to form PF_HMM$(t_i)^{\mathrm{rd}}$

Step 16. End

5.2. Fuzzy Neutrosophic Soft Set Resource Uncertainty Handler (FNSS-RUH)

The FNSS-RUH inputs the PF_HMM enabled grid resources to generate reduced PF_HMM enabled grid resoures by using FNSS theory. By using Discernibility matrix and weighted average of discernibility matrix the uncertainty in the grid resources is normalized. By performing weighted Average Square over standard minimum of discernibility matrix, the uncertainty free grid resource

parameters are obtained. The grid resource parameters are transferred into set of subsets of grid resource attributes by considering each of the elements of the grid resource matrix. The operation begins by considering partial set of subsets of grid resources and through absorption operation, the superset of grid resource parameters is removed. Through absorption and grouping operations the partial subset of grid resources gets deleted, this modified representation of grid resources gets repeated in loop until final reluctant of reduced grid resource set is produced. Algorithm 2 gives the working of FNSS-RUH.

Algorithm 2. Working of FNSS-RUH

Step 1. Begin

Step 2. *Input:* $\mathrm{PF_HMM}(\mathrm{gr}_i) = (Q(\mathrm{gr}_i), P_{\mathrm{t}}(\mathrm{gr}_i), \varphi_{\mathrm{d}}(\mathrm{gr}_i), \varphi_{\mathrm{e}}(\mathrm{gr}_i), \pi_0(\mathrm{gr}_i), \pi_{\mathrm{f}}(\mathrm{gr}_i))$

Step 3. *Output:* $\mathrm{PF_HMM}(\mathrm{gr}_i)^{\mathrm{rd}} = (Q(\mathrm{gr}_i)^{\mathrm{rd}}, P_{\mathrm{t}}(\mathrm{gr}_i)^{\mathrm{rd}}, \varphi_{\mathrm{d}}(\mathrm{gr}_i)^{\mathrm{rd}}, \varphi_{\mathrm{e}}(\mathrm{gr}_i)^{\mathrm{rd}}, \pi_0(\mathrm{gr}_i)^{\mathrm{rd}}, \pi_{\mathrm{f}}(\mathrm{gr}_i)^{\mathrm{rd}})$

Step 4. for every $\mathrm{gr}_{i_i} \in \mathrm{GR}$ **do**

Step 5. Form FNSS of PR_HMM (gr_i)

Step 6. PR_HMM $(\mathrm{gr}_i)=(\mathrm{gr}_i, T_{\mathrm{FNSS}}(\mathrm{gr}_i), I_{\mathrm{FNSS}}(\mathrm{gr}_i), F_{\mathrm{FNSS}}(\mathrm{gr}_i))$

Step 7. Calculate discernibility matrix *D* (FNSS-PF_HMM $(\mathrm{gr}_i, \mathrm{gr}_j)$), i.e.,

Step 8.

D(FNSS-PR_HMM $(\mathrm{gr}_i, \mathrm{gr}_j)) = \{\mu(a) \in A | g(\mathrm{gr}_i, \mu(a)) \neq g(\mathrm{gr}_j, \mu(a))\}$

Step 9. Compute fuzzy weighted average of discernibility matrix, i.e., $\mathrm{FWA} = \sum_{i=1}^{i=n}(w_i * \mu(a))$, where $w_i$is the assigned weight.

Step 10. *D* (FNSS-PR_HMM $(\mathrm{gr}_i, \mathrm{gr}_j)$) =

Φ	$\{a_{1/\mathrm{FWA}}\}$	$\{a_{1/\mathrm{FWA}}, a_{4/\mathrm{FWA}}\}$
....	....	.....
$\{a_{1/\mathrm{FWA}}, a_{3/\mathrm{FWA}}, a_{4/\mathrm{FWA}}\}$	...	Φ

Step 11. Calculate the standard minimum Δ^*D (FNSS-PR_HMM $(\mathrm{gr}_i, \mathrm{gr}_j)$) $= \Delta^* D(\mathrm{FNSS-PR_HMM}(\mathrm{gr}_i, \mathrm{gr}_j)) = (\mu(a_i) \wedge \mu(a_j)) \vee (\mu(a_k) \wedge \mu(a_l))$

$\{a_1^*, a_6^*\}$	$\{a_1^*\}$	$\{a_1^*, a_6^*\}$
....	....	.....
$\{a_6^*\}$	...	$\{a_4^*, a_7^*\}$

Step 12. Calculate the fuzzy weighted average square over standard minimum of discernibility matrix, i.e., $\mathrm{FWA}^2=\sum_{i=1}^{i=n}(w_i * \mu(a))^2$, i.e.,

Step 13. *D* (FNSS-PR_HMM (t_i, t_j)) =

$\{a_1^*/\mathrm{FWA}^2, a_6^*/\mathrm{FWA}^2\}$	$\{a_3^*/\mathrm{FWA}^2\}$	$\{\Phi^*/\mathrm{FWA}^2\}$
....	....	.....
$\{a_6^*/\mathrm{FWA}^2\}$	...	$\{a_4^*/\mathrm{FWA}^2, a_7^*/\mathrm{FWA}^2\}$

Step 14. End for

Step 15. Output all reduced tasks in $\mathrm{PR_HMM}(t_i)$ to form $\mathrm{PR_HMM}(t_i)^{\mathrm{rd}}$

Step 16. End

5.3. Inter/Intra Transfer Q-Learning Agent (I/I_TQLA)

This module generates load-balancing policies AΠ by accepting uncertainty free tasks and grid resource parameters in reduced form $\text{PF_HMM}(t_i)^{\text{rd}}$, $\text{PF_HMM}(\text{gr}_i)^{\text{rd}}$. This component generates high quality policies by updating the Q-value twice in both learning stage and applying stage using state value function. It mainly consists of two sub-stages one to learn Inter/Intra TQLA and the other to apply Inter/Intra TQLA. During learning stage, a random action is performed to obtain the reward and the learning state Q-value is computed for each state of the agent. During apply stage the Q-value is updated by considering the best matching Q-value state in the leaning stage. Finally, the applied resource provisioning policies are formulated by keeping the basis of the learned resource provisioning policies. Algorithm 3 provides the working of IT_TQLA/IR_TQLA.

Algorithm 3. Working of the Inter/Intra Transfer-Q-Learning Agent

Step 1. Start

Step 2. *Input:* $\text{PF_HMM}(t_i)^{\text{rd}} = (Q(t_i)^{\text{rd}}, P_{\text{t}}(t_i)^{\text{rd}}, \varphi_{\text{d}}(t_i)^{\text{rd}}, \varphi_{\text{e}}(t_i)^{\text{rd}}, \pi_0(t_i)^{\text{rd}}, \pi_{\text{f}}(t_i)^{\text{rd}})$,

$\text{PF_HMM}(\text{gr}_i)^{\text{rd}} = (Q(\text{gr}_i)^{\text{rd}}, P_{\text{t}}(\text{gr}_i)^{\text{rd}}, \varphi_{\text{d}}(\text{gr}_i)^{\text{rd}}, \varphi_{\text{e}}(\text{gr}_i)^{\text{rd}}, \pi_0(\text{gr}_i)^{\text{rd}}, \pi_{\text{f}}(\text{gr}_i)^{\text{rd}})$

Step 3. *Output:* Set of applied load balancing policies

$$\text{A}\Pi = \{\text{A}\Pi_1, \text{A}\Pi_2, \text{A}\Pi_3, \dots, \text{A}\Pi_{\text{p}}\}$$

Step 4. Model to learn: Inter/Intra TQLA

Step 5. for every $\text{PF_HMM}(t_i)^{\text{rd}}$ and $\text{PF_HMM}(\text{gr}_i)^{\text{rd}}$ **do**

Step 6. Initialize the learning stage Q-value with the random state and action $\text{LQ}(S, A) = \emptyset$

Step 7. **for** every action $A_i \in A$ **do**

Step 8. Perform an action in random to get the reward in the state

Step 9. Update the learning stage Q-state

$$\text{LQ}(S, A) = \text{LQ}(S, A) + \delta R(S, A) + \delta \text{LQ}(\arg\max \text{LQ}(S, A)) - \text{LQ}(S, A)$$

Step 10. Calculate the Learning stage *Q*-state, i.e., LQ-state value function $\text{LV}(S, A) = E(\pi)$, Include the reward given for each state

Step 11. Formulate the resource provisioning learned policy

$$\text{L}\pi_i = \pi r^2 = \delta + \delta \text{LQ}(\arg\max(\text{LQ}(S, A))$$

Step 12. **End for**

Step 13. **End for**

Step 14. Generate the Learning stage policies.

$$\text{L}\Pi ::= \text{L}\Pi \cup \text{L}\pi_i$$

Step 15. Output learned Load-balancing policies

$$\text{L}\Pi = \{\text{L}\Pi_1, \text{L}\Pi_2, \text{L}\Pi_3, \dots, \text{L}\Pi_p\}$$

Step 16. Model to Apply: Inter/Intra TQLA

Step 17. Input the learned policies from the model

$$\text{L}\Pi = \{\text{L}\Pi_1, \text{L}\Pi_2, \text{L}\Pi_3, \dots, \text{L}\Pi_p\}$$

Step 18. **for** every $\text{PF_HMM}(t_i)^{\text{rd}}$ and $\text{PF_HMM}(\text{gr}_i)^{\text{rd}}$ **do**

Step 19. Initialize the applying stage Q-value with the random state and action $\mathrm{AQ}(S, A) = \emptyset$

Step 20. **for** every action $A_i \in A$ **do**

Step 21. Perform an action in random to get the reward in the state

Step 22. Update the applying stage *Q*-state by keeping the best matching learning Q-state value

$$\mathrm{AQ}(S, A) = \mathrm{AQ}(S, A) + \delta \mathrm{R}(S, A) + \delta \arg\max(\mathrm{AQ}(S, A), \mathrm{LQ}(S, A))$$

Step 23. Calculate the applying *Q*-state AQ state value function $\mathrm{AV}(S, A) = E(\pi)$, Include the reward given for each state.

Step 24. Formulate the resource provisioning applied policy

$$\mathrm{A}\pi = \delta + \delta \mathrm{AQ}(\arg\max(\mathrm{AQ}(S, A)))$$

Step 25. **End for**

Step 26. **End for**

Step 27. Generate the applying stage policies.

$A\Pi ::= A\Pi \cup \mathrm{A}\pi_i$

Step 28. Output applicable load balancing policies

$$\mathrm{A}\Pi = \{\mathrm{A}\Pi_1, \mathrm{A}\Pi_2, \mathrm{A}\Pi_3, \ldots, \mathrm{A}\Pi_p\}$$

Step 29. **Stop**

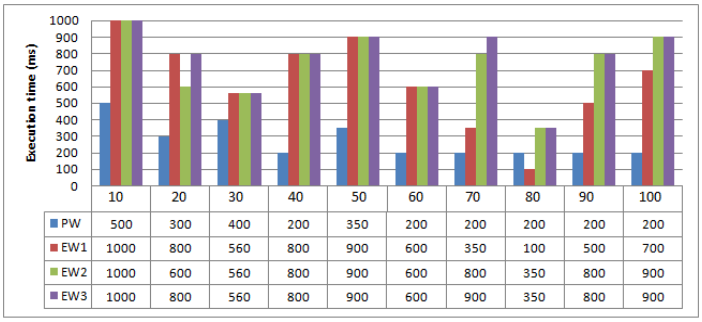
6. Results and discussion

This section provides the experimental setup for comparison of the Proposed Work (PW) with three of the recent Existing Works (EW1 [24], EW2 [25], EW3 [26]) based on Execution time, Response time, Learning rate and Throughput using SimGrid open-source simulation software. The SimGrid software allows simulation of grid computing environment by modelling heterogeneous grid resources under uncertainty of tasks and grid resources. The topology of the grid-computing environment, content present in the computing nodes, and status of resource availability in grid resources is detailed inside configuration file. The task and resource configuration file includes the distributed data files content. Application, containing benchmark dataset is considered for simulation purpose and initialization of the parameters are as follows: grid dimension=28∗19∗5; active cells=1761; cell dimension=40∗40∗40; control steps=5; volume=1 PV; simulation time=5 min; packet size=80 Bytes; packet interval=20 ms, number of computing nodes=1500, InitEng=4.0 J; grid size=100 m; distance between the neighbours=10 m; routing protocol=AODV; MAC protocol=MAC/802_15_4; system loss=1.0; gain transmitter antenna=1.0; gain receiver antenna=1.0; transmitter height = 1.5 m; radio model=TwoRayGround; and Antenna type=Omni antenna [32, 33]. The grid topology considered for evaluation contains three computing clusters composed of 27 nodes in which 13 nodes contain both computing elements and storage elements. Remaining nodes are network nodes that do not contain computing elements and storage elements.

6.1. Uncertainty of tasks

The incoming tasks in grid computing environment exhibit highly dynamic behaviours due to several factors, which cause uncertainty. These factors include inefficient distribution of tasks, uneven arrival rate of tasks, heterogeneous real-time tasks, uncertain duration for task offloading, frequent suspension of tasks computation, and proper trajectory positions of the tasks [34].

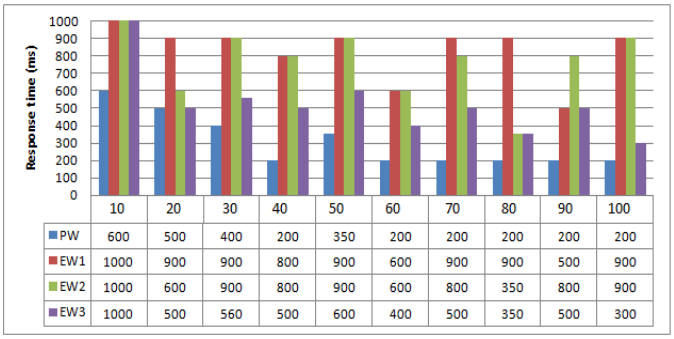
Execution time. Fig. 2 depicts the execution time incurred over the varying uncertainty of tasks. The execution time of the PW is very low throughout with the increase in the uncertainty of the tasks as the transfer Q-learning uses the already learnt knowledge about previous similar load imbalance situations. The execution time of the EW1 is higher during lower uncertainty of tasks and remained to be higher even with the increase in the uncertainty of the tasks as the conventional Q-learning takes longer time to arrive at optimal load balancing policies with overloaded Q-states. The execution time of the EW2 kept increasing with the increase in the uncertainty of the tasks due to the combination of swarm intelligence and limited number of fuzzy rules. The execution time of the EW3 is very high during lower uncertainty of tasks and even during higher uncertainty of tasks as the PSO Algorithm easily ends up into suboptimal solution when exposed to high dimensional grid computing environment.



	10	20	30	40	50	60	70	80	90	100
PW	500	300	400	200	350	200	200	200	200	200
EW1	1000	800	560	800	900	600	350	100	500	700
EW2	1000	600	560	800	900	600	800	350	800	900
EW3	1000	800	560	800	900	600	900	350	800	900

Fig. 2. Execution time incurred over the varying uncertainty of tasks

Response time. Fig. 3 shows the response time incurred over the varying uncertainty of tasks.

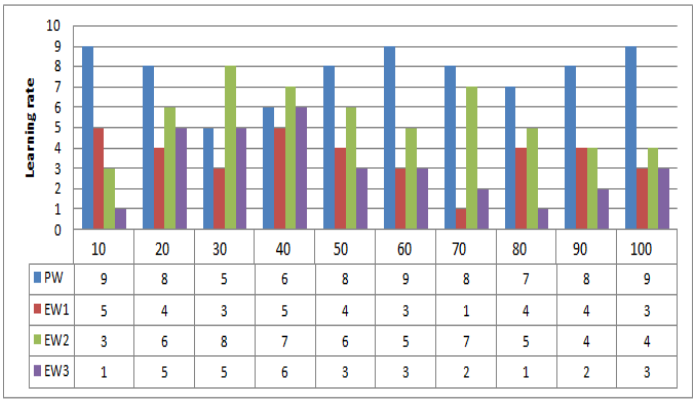


	10	20	30	40	50	60	70	80	90	100
PW	600	500	400	200	350	200	200	200	200	200
EW1	1000	900	900	800	900	600	900	900	500	900
EW2	1000	600	900	800	900	600	800	350	800	900
EW3	1000	500	560	500	600	400	500	350	500	300

Fig. 3. Response time incurred over the varying uncertainty of tasks

The response time of the PW is found to decrease gradually with the increase in the uncertainty of the tasks as the uncertainties in the tasks and grid resources are handled properly using FNSS theory before formulating load balancing policies using transfer Q-learning. The response time of the EW1 and EW2 is higher throughout even during lower uncertainty of tasks and higher uncertainty of tasks as the techniques suffer from scattering problem when exposed to high dimensional grid computing environment. The response time of EW3 is moderate during lower uncertainty level of tasks and remained lower during higher uncertainty of tasks due to poor local and global search ability of the technique.

Learning rate. Fig. 4 shows the learning rates observed over the varying uncertainty of tasks. The learning rate of the PW remained very high during lower and higher uncertainty of tasks as the speed of operation of the transfer Q-learning is very high since the initial and target load imbalance situations are very much same in the grid environment. The learning rate of the EW1 and EW3 remained average during lower and higher uncertainty of tasks as the learning speed slows down due to frequent visiting of the state-action pairs of the agent and poor parameter optimization procedure. The learning rate of the EW2 is higher during lower and higher uncertainty of tasks as the technique fail to readily deal with the uncertainty in the tasks and grid resource parameters.

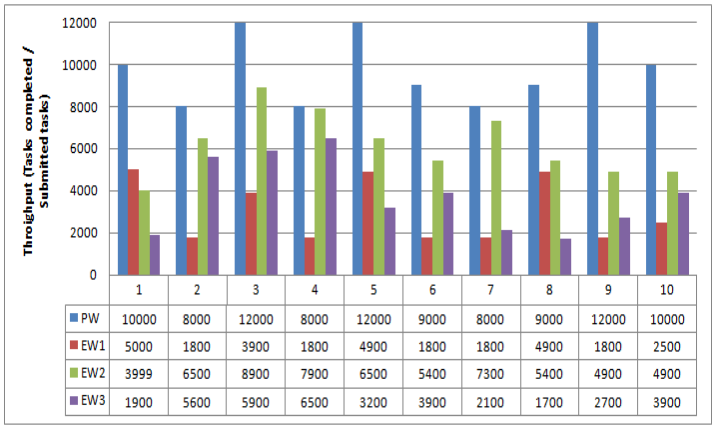


	10	20	30	40	50	60	70	80	90	100
PW	9	8	5	6	8	9	8	7	8	9
EW1	5	4	3	5	4	3	1	4	4	3
EW2	3	6	8	7	6	5	7	5	4	4
EW3	1	5	5	6	3	3	2	1	2	3

Fig. 4. Learning rate observed over the varying uncertainty of tasks

Throughput. Fig. 5 provides a graph of throughputs achieved over the varying uncertainty of tasks. The throughput achieved by the PW is very high during lower and higher uncertainty of tasks as the uncertainty in the tasks, grid resources get identified using PF_HMM, and PR_HMM after that high quality load balancing policies are formulated using transfer learning. The throughput achieved by the EW1 is very low during lower and higher uncertainty of tasks as the quality of load balancing policies formulated are poor due to trial-and-error approach. The throughput achieved by EW2 is moderate during lower uncertainty of tasks and remained lower during higher uncertainty of tasks due to the sequence of random decisions taken at runtime and being highly sensitive to noisy parameters. The throughput achieved by the EW3 is moderate during lower uncertainty of tasks,

reduced still more with the increase in the uncertainty of tasks due to poor generalization performance, and suffers from over-fitting problem.



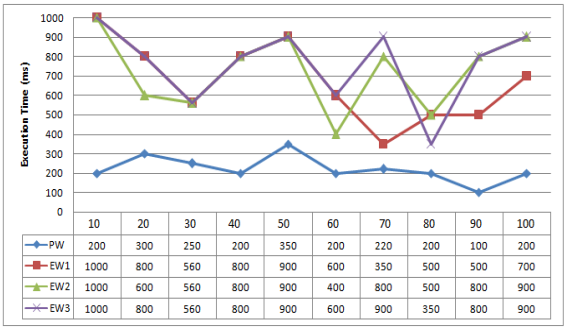
	1	2	3	4	5	6	7	8	9	10
PW	10000	8000	12000	8000	12000	9000	8000	9000	12000	10000
EW1	5000	1800	3900	1800	4900	1800	1800	4900	1800	2500
EW2	3999	6500	8900	7900	6500	5400	7300	5400	4900	4900
EW3	1900	5600	5900	6500	3200	3900	2100	1700	2700	3900

Fig. 5. Throughput achieved over the varying uncertainty of tasks

6.2. Uncertainty of grid resources

The resources in grid computing environment also involve huge number of uncertainties that arise from several factors that include large number of distributed resources, loosely coupled grid resources, inherent variability of grid resources, poor ensemble of grid resources, uneven scaling of resources, distributed ownership problem, and poor stability of resources due to the launch of cyber-attacks [35].

Execution time. Fig. 6 gives a graph of execution times incurred over the varying uncertainty of grid resources. The execution time of the PW is lower during lower and higher uncertainty of grid resources as the transfer Q-learning is capable enough of making high quality load balancing policies through fast iterations with the grid environment. The execution time of the EW1, EW2 and EW3 remained to very high during lower uncertainty and higher uncertainty level of grid resources as the models used exhibit higher probability of converging to suboptimal solutions and demands significant effort for fine-tuning of the sensitive outlier parameters.

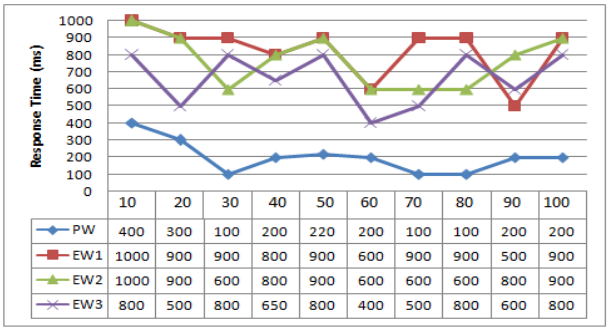


	10	20	30	40	50	60	70	80	90	100
PW	200	300	250	200	350	200	220	200	100	200
EW1	1000	800	560	800	900	600	350	500	500	700
EW2	1000	600	560	800	900	400	800	500	800	900
EW3	1000	800	560	800	900	600	900	350	800	900

Fig. 6. Execution time incurred over the varying uncertainty of grid resources

Response time. Fig. 7 gives a graph of response time incurred over the varying uncertainty of grid resources. The response time of the PW is found to be moderate initially during lower uncertainty of grid resources and found to be lower even with

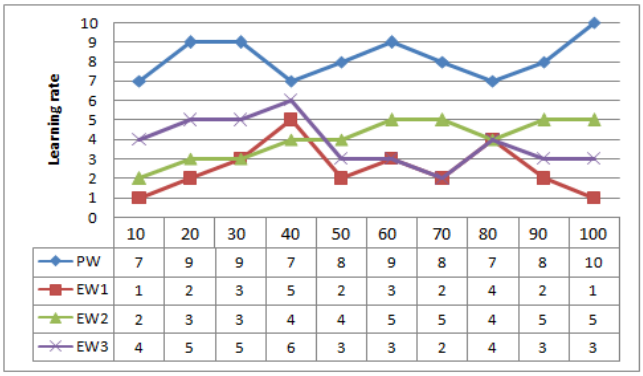
the increase in the uncertainty of the grid resources as the transfer Q-learning model is capable enough choosing actions with high expected utility without the need to exactly model the uncertain grid computing environment. The response time of the EW1 and EW2 is very much higher during lower and higher uncertainties of grid resources as the loads balancing decisions taken lacks rational thinking and are often mistaken because of likelihood hypothesis. The response time of the EW3 is moderate during lower and higher uncertainty of grid resources as the approach suffers from sizing problem and produces diverging value for mean square displacement and infinite value for particle velocity.



	10	20	30	40	50	60	70	80	90	100
PW	400	300	100	200	220	200	100	100	200	200
EW1	1000	900	900	800	900	600	900	900	500	900
EW2	1000	900	600	800	900	600	600	600	800	900
EW3	800	500	800	650	800	400	500	800	600	800

Fig. 7. Response time incurred over the varying uncertainty of grid resources

Learning rate. Fig. 8 gives a graph of learning rates observed over the varying uncertainty of tasks.



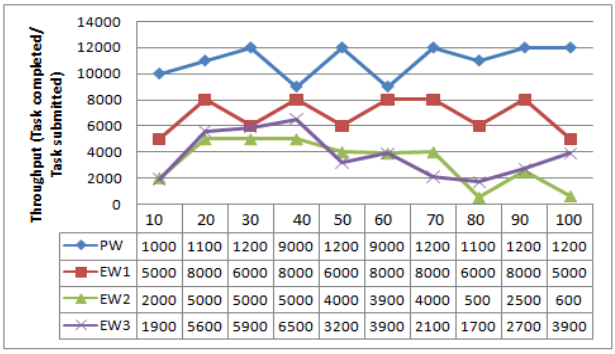
	10	20	30	40	50	60	70	80	90	100
PW	7	9	9	7	8	9	8	7	8	10
EW1	1	2	3	5	2	3	2	4	2	1
EW2	2	3	3	4	4	5	5	4	5	5
EW3	4	5	5	6	3	3	2	4	3	3

Fig. 8. Learning rate observed over the varying uncertainty of grid resources

The learning rate of the PW is moderate during lower uncertainty of grid resources and keeps increasing with the increase in the uncertainty of the grid resources due to proper transfer of learnt knowledge composed of weights and features from trained model to training model to tackle newer problems with few training data. The learning rate of the EW1 is lower with lower uncertainty of grid resources and remains lower during higher uncertainty of grid resources as the as the Q-learning behaves poorly in stochastic environment. The learning rate of EW2 is lower during lowered uncertainty of grid resources and remains moderate with the increase in the uncertainty of grid resources due to poor uncertainty handling using fuzzy membership function. The learning rate of the EW3 is higher during lower

uncertainty of grid resources but starts to decline with the increase in the uncertainty of grid resources as the convergence speed is low due to the approach being highly sensitive to velocity parameters.

Throughput. Fig. 9 gives a graph of throughput achieved over the varying uncertainty of grid resources. The throughput of the PW remains higher during lower and higher uncertainty of grid resources as the transfer Q-learning algorithm is more goal oriented and predictions of the Q-function after gaining knowledge through transfer learning is highly accurate. The throughput of EW1 is moderate during lower and higher uncertainty of grid resources as the approach suffers from high instability due to correlated updating of sequential training data and parameters affects the estimator target and causes high divergence between them. The throughput of EW2 and EW3 are found to be higher during lower uncertainty of grid resources but start to decline with the increase in the uncertainty of grid resources as both the approaches lack wider exploration capability and fail to achieve proper trade-off between exploration and exploitation phases for performance enhancement.



	10	20	30	40	50	60	70	80	90	100
PW	1000	1100	1200	9000	1200	9000	1200	1100	1200	1200
EW1	5000	8000	6000	8000	6000	8000	8000	6000	8000	5000
EW2	2000	5000	5000	5000	4000	3900	4000	500	2500	600
EW3	1900	5600	5900	6500	3200	3900	2100	1700	2700	3900

Fig. 9. Throughput achieved over the varying uncertainty of grid resources

7. Conclusion

This paper presents a novel FNSS based Transfer-Q-Learning scheme for load balancing in uncertain grid computing environments. The hidden uncertainties in the tasks and grid resource parameters are identified precisely using PF_HMM and PR_HMM. FNSS handles the identified uncertainty of the tasks and grid resource parameters using FNSS theory. The transfer Q-learning agent with pre-trained knowledge solves the large-scale load-balancing problem. The transfer Q-learning agents can automatically learn with small amounts of data by efficiently handling the uncertainty in the system parameters using FNSS. The error encountered while computing target transfer Q-function value is negligible as it reduces possible harm to the incoming task and resource parameters. From the results obtained it is inferred that the proposed work outperforms three of the existing recent works by about 90 percent with respect total execution time, response time, learning rate and throughput. We plan to extend the proposed work to develop secure and robust load balancing scheme for other high-performance computing domains like cloud and fog.

However the proposed FNSS based Transfer-Q-Learning scheme for load balancing include several limitations, which need to be handled efficiently. Some of the potential challenges encountered are as follows: Practical implementation of FNSS in real-world applications becomes difficult as the fuzzy neutrosophic components exhibit nonstandard intervals. The PF_HMM and PR_HMM frameworks used to identify parameter uncertainties often suffer evaluation and decoding problems.

References

1. S i n g h, M. An Overview of Grid Computing. – In: Proc. of International Conference on Computing, Communication, and Intelligent Systems (ICCCIS'19), 2019.
2. S u n g k a r, A., T. K o g o y a. A Review of Grid Computing. – Computer Science & IT Research Journal, Vol. **1**, 2020.
3. D a k k a k, O., S. A. N o r, S. A r i f, Y. F a z e a. Improving QoS for Non-Trivial Applications in Grid Computing. – In: Proc. of International Conference of Reliable Information and Communication Technology, 2019.
4. F o s t e r, I., C. K e s s e l m a n. Translating the Grid: How a Translational Approach Shaped the Development of Grid Computing. – Journal of Computational Science, Vol. **52**, 2021.
5. A s w a l, M. S. VM Consolidation Plan for Improving the Energy Efficiency of Cloud. – Cybernetics and Information Technologies, Vol. **21**, 2021, No 3, pp. 145-159.
6. D h i n g r a, S., P. B a n s a l. Employing Divergent Machine Learning Classifiers to Upgrade the Preciseness of Image Retrieval Systems. – Cybernetics and Information Technologies, Vol. **20**, 2020, No 3.
7. K a r a, N., H. G. K o c k e n. A Fuzzy Approach to Multi-Objective Solid Transportation Problem with Mixed Constraints Using Hyperbolic Membership Function. – Cybernetics and Information Technologies, Vol. **21**, 2021, No 4, pp. 158-167.
8. K o u a d r i, A., M. H a j j i, M. F. H a r k a t, K. A b o d a y e h, M. M a n s o u r i, H. N o u n o u, M. N o u n o u. Hidden Markov Model Based Principal Component Analysis for Intelligent Fault Diagnosis of Wind Energy Converter Systems. – Renewable Energy, Vol. **150**, 2020.
9. G o h, C. Y., J. D a u w e l s, N. M i t r o v i c, M. T. A s i f, A. O r a n, P. J a i l l e t. Online Map-Matching Based on Hidden Markov Model for Real-Time Traffic Sensing Applications. – In: Proc. of 15th International IEEE Conference on Intelligent Transportation Systems, 2012.
10. M o r, B., S. G a r h w a l, A. K u m a r. A Systematic Review of Hidden Markov Models and Their Applications. – Archives of Computational Methods in Engineering, Vol. **28**, 2021.
11. D e l i, I., S. B r o u m i. Neutrosophic Soft Matrices and NSM-Decision Making. – Journal of Intelligent & Fuzzy Systems, Vol. **28**, 2015.
12. K o k o ç, M., S. E r s o z. New Ranking Functions for Interval-Valued Intuitionistic Fuzzy Sets and Their Application to Multi-Criteria Decision-Making Problem. – Cybernetics and Information Technologies, Vol. **21**, 2021, No 1, pp. 3-18.
13. D e l i, I., S. E r a s l a n, N. Ç a g m a n. IVNPIV-Neutrosophic Soft Sets and Their Decision Making Based on Similarity Measure. – Neural Computing and Applications, Vol. **29**, 2018.
14. A l i, M., L. H. S o n, I. D e l i, N. D. T i e n. Bipolar Neutrosophic Soft Sets and Applications in Decision Making. – Journal of Intelligent & Fuzzy Systems, Vol. **33**, 2017.
15. D e l i, I., S. B r o u m i. Neutrosophic Soft Relations and Some Properties. – Annals of Fuzzy Mathematics and Informatics, Vol. **9**, 2015.
16. S i n g h, S., S. L a l o t r a, A. H. G a n i e. On Some Knowledge Measures of Intuitionistic Fuzzy Sets of Type-Two with Application to MCDM. – Cybernetics and Information Technologies, Vol. **20**, 2020, No 1, pp. 3-20.
17. N a e e m, K., M. R i a z, D. A f z a l. Fuzzy Neutrosophic Soft σ-Algebra and Fuzzy Neutrosophic Soft Measure with Applications. – Journal of Intelligent & Fuzzy Systems, Vol. **39**, 2020.
18. F a n, J., Z. W a n g, Y. X i e, Z. Y a n g. A Theoretical Analysis of Deep Q-Learning. – In: Learning for Dynamics and Control, 2020.

19. S a m m a, H., J. M o h a m a d-S a l e h, S. A. S u a n d i, B. L a h a s a n. Q-Learning-Based Simulated Annealing Algorithm for Constrained Engineering Design Problems. – Neural Computing and Applications, Vol. **32**, 2020, pp. 5147-5161.
20. W a n g, Y., Y. L i u, W. C h e n, Z. M. M a, T. Y. L i u. Target Transfer Q-Learning and Its Convergence Analysis. – Neurocomputing, Vol. **392**, 2020.
21. J e o n g, G., H. Y. K i m. Improving Financial Trading Decisions Using Deep Q-Learning: Predicting the Number of Shares, Action Strategies, and Transfer Learning. – Expert Systems with Applications, Vol. **117**, 2019.
22. K h a n, S., B. N a z i r, I. A. K h a n, S. S h a m s h i r b a n d, A. T. C h r o n o p o u l o s. Load Balancing in Grid Computing: Taxonomy, Trends and Opportunities. – Journal of Network and Computer Applications, Vol. **88,** 2017.
23. W e n j i e, T., Y. Y i p i n g, Z. F e n g, L. T i a n l i n, S. X i a o. A Work-Stealing Based Dynamic Load Balancing Algorithm for Conservative Parallel Discrete Event Simulation. – In: Proc. of Winter Simulation Conference (WSC'17), 2017.
24. W u, J., X. X u, P. Z h a n g, C. L i u. A Novel Multi-Agent Reinforcement Learning Approach for Job Scheduling in Grid Computing. – Future Generation Computer Systems, Vol. **27**, 2011.
25. H a j o u i, Y., O. B o u a t t a n e, M. Y o u s s f i, E. I l l o u s s a m e n. Q-Learning Applied to the Problem of Scheduling on Heterogeneous Architectures. – International Journal of Computer Science and Network Security, Vol. **18**, 2018.
26. G a r c i a-G a l a n, S., R. P. P r a d o, J. M. E x p ó s i t o. Fuzzy Scheduling with Swarm Intelligence-Based Knowledge Acquisition for Grid Computing. – Engineering Applications of Artificial Intelligence, Vol. **25**, 2012.
27. T a n g, K., W. J i a n g, R. C u i, Y. W u. A Memory-Based Task Scheduling Algorithm for Grid Computing Based on Heterogeneous Platform and Homogeneous Tasks. – International Journal of Web and Grid Services, Vol. **16**, 2020.
28. P a t n i, J. C. Centralized Approach of Load Balancing in Homogenous Grid Computing Environment. – In: Proc. of 3rd International Conference on Computers in Management and Business, 2020, pp. 151-156.
29. A l i, W., F. B o u a k k a z. Agent Based Load Balancing in Grid Computing. – In: Proc. of Multi-Agent Systems-Theory, Implementation and Applications. IntechOpen, 2020.
30. L i u, F., D. J a n s s e n s, J. C u i, G. W e t s , M. C o o l s. Characterizing Activity Sequences Using Profile Hidden Markov Models. – Expert Systems with Applications, Vol. **42**, 2015.
31. W a l k e r, C. R., A. S c a l l y, N. De M a i o, N. G o l d m a n. Short-Range Template Switching in Great Ape Genomes Explored Using Pair Hidden Markov Models. – PloS Genetics, Vol. **17**, 2021.
32. B r a u n, T. D., et al. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. – J. Parallel Distrib. Comput., Vol. **61**, 2001, No 6, pp. 810-837.
33. L e b r e, A., A. L e g r a n d, F. S u t e r, P. V e y r e. Adding Storage Simulation Capacities to the Simgrid Toolkit: Concepts, Models, and Api. – In: Proc. of 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2015, pp. 251-260.
34. C o r d e r y, J. L., D. M o r r i s o n, B. M. W r i g h t, T. D. W a l l. The Impact of Autonomy and Task Uncertainty on Team Performance: A Longitudinal Field Study. – Journal of Organizational Behavior, Vol. **31**, 2010.
35. R e a l, R., A. Y a m i n, L. da S i l v a, G. F r a i n e r, I. A u g u s t i n, J. B a r b o s a, C. G e y e r. Resource Scheduling on Grid: Handling Uncertainty. – In: Proc. of 1st Latin American Web Congress, 2003.

Received: 25.03.2022; Second Version: 07.10.2022; Accepted: 12.10.2022