BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **19**, No 3
Sofia • 2019 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2019-0028

Uncertainty Aware Resource Provisioning Framework for Cloud Using Expected 3-SARSA Learning Agent: NSS and FNSS Based Approach

Bhargavi K.[1], *B. Sathish Babu*[2]

[1]*Department of CSE, Siddaganga Institute of Technology, Tumkur, Karnataka, India*
[2]*Department of CSE, R V College of Engineering, Bangalore, Karnataka, India*
E-mails: bhargavi.tumkur@gmail.com bsbabu@rvce.edu.in

Abstract**:** *Efficiently provisioning the resources in a large computing domain like cloud is challenging due to uncertainty in resource demands and computation ability of the cloud resources. Inefficient provisioning of the resources leads to several issues in terms of the drop in Quality of Service* (*QoS*)*, violation of Service Level Agreement* (*SLA*)*, over-provisioning of resources, under-provisioning of resources and so on. The main objective of the paper is to formulate optimal resource provisioning policies by efficiently handling the uncertainties in the jobs and resources with the application of Neutrosophic Soft-Set* (*NSS*) *and Fuzzy Neutrosophic Soft-Set* (*FNSS*)*. The performance of the proposed work compared to the existing fuzzy auto scaling work achieves the throughput of 80% with the learning rate of 75% on homogeneous and heterogeneous workloads by considering the RUBiS, RUBBoS, and Olio benchmark applications.*

Keywords**:** *SARSA* (*State-Action Reward-State-Action*)*, Resource provisioning, Uncertainty, Soft-set, elasticity, throughput, learning rate.*

1. Introduction

The cloud resource demands of the complex computational applications in the area of engineering, economics, environmental science, and so on, are highly fluctuating in nature and consist of data that are uncertain and imprecise, elastic resource provisioning becomes one of the critical requirements of such applications. The elastic resource provisioning mechanism allows the user to scale up or down the resources dynamically at run-time, this feature reduces infrastructure cost and then models the application to attain high Quality of Service (QoS) requirement by meeting the Service Level Agreements (SLAs). The existing resource provisioning approaches can be classified into two types i.e., reactive or proactive, reactive approaches take resource provisioning decisions when the load on the system resources are high, whereas the proactive approaches estimate the probable load on the system resources and then lease the resources in advance [1-3]. The elastic resource provisioning in cloud involves several challenges in terms of existence of heterogeneous hardware, maintenance of virtual machine compatibility table, periodic updating of states of the virtual machines, frequent failures of nodes during scaling, long-term irregular workload parameters, sudden changes in processing capability of the resources, frequent violation of SLA, and so on [4-10]. Methodologies based on thresholds, time series analysis, queuing theory and control theory have failed to provide satisfactory solutions to resource provisioning problem as those solutions are affected by the undetermined and erratic changes in the processing requirements of jobs and unstable processing behavior exhibited by the resources [11-13]. Hence there is a necessity to handle uncertainty in the job and resource parameters before taking resource provisioning decisions.

Many mathematical models are available to handle uncertainty like probability theory, interval mathematics, and fuzzy sets. But these techniques have several limitations like probability theory is suitable only for stochastically stable phenomena and usually takes more trials to provide a solution; interval mathematics fails to handle uneven changing in the workload parameters; in the fuzzy set computing membership function is tedious as it is not general and is individual specific which cannot handle the dynamics of large state space. These drawbacks motivated towards soft-set which is parameterized family of a set and does not put any restriction on the approximate description as it puts soft boundary depending on the parameters. The conventional reinforcement learning techniques draw policies with the assumption that the underlying environment is static and do not consider the changing dynamics into account but this assumption fails in a highly dynamic environment like cloud, this motivated to use soft-set enabled reinforcement learning [14-17].

The objectives of the paper are as follows.

Identify the uncertainty in the jobs and resources by representing their states in the form of Partially Observable Markov Decision Process model (POMDP) and Hidden Markov Model (HMM) model.

Handle the uncertainties of the jobs and resources using NSS and FNSS as they provide practical frameworks to measure the truth, indeterminacy, and falsehood of the data associated with the resource provisioning decisions.

Design expected 3-SARSA (State-Action Reward-State-Action) learning agent empowered with the NSS and FNSS model, which controls the exploration during action selection state.

Evaluate the resource provisioning policies with respect to successful job completion rate and learning rate, as SARSA agent updates the resource provisioning policies by considering three adjacent expected action-value pairs, which increases the learning stability of the agent and even increases the successful job completion rate.

The remaining part of the paper is organized as follows, Section 2 deals with related work; Section 3 briefs about the system model; Section 4 gives the high-level view of the proposed work; Section 5 does interval-valued NSS analysis of the proposed work; and Section 6 deals with result and discussion; and finally Section 7 draws the conclusion.

2. Related works

The [19] proposed a resource allocation scheme under job uncertainty. Here the execution delay of the incoming jobs is predicted using a self-similar long tail process, where the similar properties are repeated at different time scales. Then the Pareto fractal flow prediction model is used for resource allocation purpose. However the basis for allocating the resources is on the assumption that the jobs exhibit similar properties but the complex computational jobs are highly random in nature and always exhibit an uneven pattern of workload, so the efficiency of the resource allocation is found to be below average.

In [20], a deep reinforcement learning based resource provisioning scheme is proposed to minimize the energy consumption of the data centers. Here deep reinforcement learning is employed using multiple layers of computational nodes, which tries to learn from changing cloud environment to draw optimal resource provisioning policies. The scheme is found to be good with respect to energy reduction in the large data centers as it effectively handles the sudden burst of the workload but the time by the network to convergence is high as it takes too long time to balance between exploration and exploitation.

A reinforcement learning based auto resource scaling system is proposed in [21], here multiple reinforcements learning agents with parallel learning policy is used to allocate the resources. Each agent has different learning experience and every agent share the information learned from the other agents. The parallel learning process is found to be good with respect to the rate of learning and Q-Value table updating. However, this increases the interaction rate between the agents as huge state space need to be considered while deciding the actions, which in turn increases the response time of the agents and leads to improper utilization of the resources.

The [22] proposes a new predictive resource scaling approach for cloud systems. The approach extracts the fine-grained pattern from the workload demands and then adjusts the resources accordingly. To extract the pattern, signal processing, and statistical methods are used. Here the workload patterns are analyzed as it is, i.e., uncertainty is not handled, so there was the drop in prediction accuracy, which resulted in the increased rejection rate of the jobs.

An analytical model based auto scaling mechanism is used in [23]. Here an analytical model is developed to characterize the workload and to analyze its impact on the efficiency of the scale-out or scale-in decisions in the cloud. An inference is drawn that scale up is suitable when SLA is strict and scale down is suitable when the workload is high. The Kalman filtering-based auto-scaling solution is applied for scaling of infrastructure services, as its topology is available. But the model does not fit for scaling of software applications as they lack fixed topology.

A comparison of fuzzy SARSA and fuzzy Q-Learning towards auto-scaling of resources in the cloud environment is given in [24]. Both approaches are used to efficiently scale the resources under varieties of workload and even maximize the resource utilization rate. However the performance of the fuzzy Q-Learning is low with respect to learning rate as it always try to compare the actual state with the best possible next state while taking actions using fuzzy rule base and the performance of the fuzzy SARSA learning is low with respect to adaptability towards heterogeneous workload as the policy formed after the learning phase is not optimized further to adapt to uneven pattern in the workload.

The [25] proposes a self-managed virtual machine scheduling technique for the cloud environment. The placement of virtual machines in cloud is one of the computation intensive activities, so in this approach, the history of the virtual machine's resource (CPU, memory, hard disk, RAM, network and so on) utilization ratio is taken into account to predict the resource utilization level then the decisions about virtual machines placement is made. However, the state of virtual machines inside the physical machines is not directly visible and it consists of several hidden states; as a result, the accuracy of the predicted resource utilization level using various machine learning models is less, this resulted in improper placement of virtual machines inside the physical machines which leads to a drop in physical machine throughput.

In [26] the heuristic approach is used to schedule the tasks through proper distribution of the resources. In this approach every incoming task is processed using modified analytic hierarchy process then the resources are scheduled using differential evolution algorithm. The analytic hierarchy process ranks the tasks based on the requirements of the tasks, however it is not possible to directly rank the tasks in the cloud environment as the jobs are usually malleable they start with very few resource requirements and then gradually expand to higher resource requirements. As a result the application of analytic hierarchy process to malleable jobs leads to improper ranking of jobs and the chances of pre-empting the higher priority jobs are more which leads to improper utilization of resources.

The [27] discusses machine learning based resource provisioning techniques for the cloud environment. Automated self-learning enabled resource provisioning is most important to deliver elastic services to customers by satisfying their needs. Here the time series forecasting technique is to predict the number of resources to be sanctioned for the incoming client requests and support vector regression model is used to forecast the processing capability of the servers. The use of time series model in combination with support vector machine is one of the biggest limitations of the approach as it fails to capture the chaotic and non-deterministic behaviors of the servers and client requests due to the use quadratic programming approach.

In [28], a deep learning based elastic resource provisioning scheme is proposed for the cloud environment. Here three different approaches of deep reinforcement learning techniques, i.e. simple deep Q-Learning, full deep Q-Learning and double deep Q-Learning are proposed to achieve elasticity in resource provisioning which is trained to converge to optimal elasticity policies. All three deep reinforcement-learning techniques are capable of learning in a large state space environment and are able to collect a sufficient amount of rewards. However, the training of the models is computationally expensive and accuracy of the elastic resource provisioning policies formed is weak as it operates directly on the partial information exhibited by the jobs and resources without the use of any membership functions to handle uncertainties in the nested layers of the deep reinforcement techniques.

A survey of prediction models based resource provisioning techniques available for cloud environment is discussed in [29]. Resource provisioning is one of the key issues in the cloud environment as the behavior pattern of workloads keeps varying which leads to frequent violations of service level agreements. Various prediction models like a neural network, fuzzy logic, linear regression, Bayesian theory model, support vector machine, and reinforcement learning are used to estimate the future demands of resources. The pros and cons of each of these models are discussed and the performance of reinforcement learning technique enriched with the fuzzy logic model is good in terms of speed and accuracy of the resource mapping as it is proactive in nature and exactly mines the correlation among the variety of resources.

A reinforcement-enabled technique for energy efficient resource provisioning is discussed in [30] to achieve maximum revenue. Here based on the read user requirements, the virtual machines and physical machines are hosted in the cloud. The resource allocation policy is updated based on the reward collected for energy utilization factor of every virtual machine and physical machine in the data center. However, in this technique, while predicting the future resource demands, the resource like CPU utilization, amount of memory, system availability and system performance are assumed to static and transparent in nature which becomes the major limiting factor.

The approach of load balancing among the virtual machines in the cloud data center using the Pareto principle is mentioned in [31]. As the computation requirement of the applications keeps varying, there is a necessity to scale up and scale down the virtual machines so Pareto based genetic algorithm is used to generate a large number of solutions and then select one of the solutions as the best one. Here the workload requirement of the user is taken directly for analysis without any pre-processing; hence there will be an influence of uncertainty over the load balancing solutions formed. Moreover, the stringent nature of the genetic algorithm increases the time taken to convergence towards an optimal solution and even it fails to arrive at the global optimum solution.

In [32], fuzzy logic based hybrid bio-inspired techniques like Ant-colony, and Firefly is developed for placement of the virtual machines within the data center and to consolidate the server. Here the basic principle used for server consolidation is to pack as many virtual machines as possible within the data center, this works fine on steady-state workloads but during the heavy burst of the workloads, it leads to over-utilization of the resources. The uncertainty factor is handled through the use of fuzzy membership functions inside ant-colony and firefly algorithms but to achieve more accuracy there will be an exponential increase in the fuzzy rules and these algorithms are old enough and their performance is weak compared to the recent bio-inspired techniques like a whale, crow, squirrel, and raven roosting.

By considering the demand uncertainty, dynamic resource allocation for cloud environment is discussed in [33]. The cloud providers allocate resources on the reservation basis or on-demand basis, reservation-based allocation of resources are carried out on long-term duration which involves lower uncertainty, whereas on-demand based allocation of resources is carried out on short-term or long-term duration which involves higher uncertainty. In this work the uncertainty in user demands are modelled as random variables using stochastic optimization approach and an algorithm with two phases is developed, the first phase does the reservation for the resources and the second phase does the dynamic allocation of the resources. However, modelling the demand uncertainty using random variables is very difficult as it does not possess exact stopping criteria and the uncertainty involved in the processing capability of the resources is also ignored which leads to under-utilization or over-utilization of the resources.

To summarize most of the existing works exhibits the following drawbacks.

- Unable to determine the uncertainty involved in the job processing requirement.
- Unable to determine the uncertainty involved in the resource computation ability.
- Drop in prediction accuracy due to the failure in determining the exact pattern in processing requirement in the malleable jobs.
- By ignoring the hidden states and partially observable states while making resource provisioning decisions; the chances of over-provisioning or under-provisioning of the resources is high.
- The conventional reinforcement learning algorithms fail to form robust resource provisioning policies, as it cannot capture the chaotic behaviours of the servers and client using a deterministic approach.
- Lack of proactiveness while taking resource provisioning decisions leads to a decrease in accuracy and speed to learning.
- The bio-inspired algorithms fail to arrive at a global optimum solution, and even the convergence rate is high due to their harsh approach involved in workload analysis.

3. System model

This section provides mathematical modeling of the system under consideration. A cloud is assumed to be ∞ collection of pool of resources,

(1) $$C = \{ R_i \}_{i=0}^{i=k}.$$

Every R_i consists of unlimited set of heterogeneous resources,

(2) $$R_i = \{ r_i \}_{i=0}^{i=k}.$$

The capacity of every resource is subset of resource pool R^+,

(3) $$C(r_i) \epsilon R^+.$$

The price associated with every resource is subset of price pool P^+,

(4) $$P(r_i) \in P^+.$$

The resources r_i and r_j of cloud are connected through a network with link L_{r_i, r_j} and the rental time of each resource to process the incoming jobs is limited.

The jobs are classified into various categories according to their resource requirements, i.e., low (*l*), medium (*m*), and high (*h*),

(5) $$J = \{ J_i \leftarrow \{ R_i(l),\ R_i(m), \ldots,\ R_i(h) \} \ldots J_k \leftarrow \{ R_k(l), R_k(m),\ R_k(h) \} \}.$$

The jobs and resources are associated with uncertainties in terms of their resource requirements and processing capabilities, which dynamically vary within the given time frame $T(J_m^n)$,

(6) $$T(J_m^n) = \langle T(J_i^0), \ldots, T(J_p^k) \rangle, \text{ and } T(R_i) = \langle T(R_i^0), \ldots, T(R_p^k) \rangle.$$

The uncertainties of the resources are handled using Neutrosophic Soft-Set (NSS) theory and the uncertainties of the jobs are handled using Fuzzy Neutrosophic Soft-Set (FNSS) theory.

Let NSS be the neutrosophic soft-set on the universe of discourse *U* and *E* is the set of parameters:

(7) $$\mathrm{NSS} = \{\langle u, T_{\mathrm{NSS}}(u), I_{\mathrm{NSS}}(u), F_{\mathrm{NSS}}(u)\rangle u \in U\},$$

where *T*, *I*, *F* are Truth value, Indeterminate value and False value, and *T*, *I*, $F \to\,]^{-}0, 1^{+}[$ and $0^{-} \langle T_{\mathrm{NSS}}(u) + I_{\mathrm{NSS}}(u) + F_{\mathrm{NSS}}(u)\rangle \le 3^{+}$, and $E = \{E_1, E_1, \dots, E_k\}$ then the collection (*F*, NSS) is referred as neutrosophic soft-set of resources over *U*.

Let FNSS be the fuzzy neutrosophic soft-set on the universe of discourse *U* and *E* is the set of parameters:

(8) $$\mathrm{FNSS} = \{\langle u, T_{\mathrm{FNSS}}(u), I_{\mathrm{FNSS}}(u), F_{\mathrm{FNSS}}(u)\rangle,\ u \in U\},$$

where $T, I, F \to [0, 1]$ and $0 \le T_{\mathrm{FNSS}}(u) + I_{\mathrm{FNSS}}(u) + F_{\mathrm{FNSS}}(u) \le 3$ then the collection $(F,$ FNSS) is referred as fuzzy neutrosophic soft-set of jobs over *U*.

Later resource provisioning decisions for the jobs are taken using expected 3-SARSA model,

(9) $$(F, \mathrm{NSS}), (F, \mathrm{FNSS}) \propto \{Q^A(S_t, A_t), Q^B(S_t, A_t), Q^C(S_t, A_t)\},$$

where

$Q^A(S_t, A_t)^{+} = \alpha[r_t + [r_{t-1} * Q^B(S_{t-1}, A_{t-1}) - r_{t-2} * Q^C(S_{t-2}, A_{t-2})]]$, in which $r_{t,}$ r_{t-1}, and r_{t-2} are the reward obtained at states S_t, S_{t-1}, and S_{t-2} and A_t, A_{t-1}, and A_{t-2} are the action taken at the states S_t, S_{t-1}, and S_{t-2}.

3.1. Resources model

The state of the resources in cloud is dependent on the hidden state of the virtual machines mounted on top of every physical machine in the cloud resource pool. Hence the resources are modeled using HMM,

(10) $$\mathrm{HMM}(R_i) = \big(S(R_i),\ V(R_i),\ B(R_i),\ A(R_i),\ I(R_i)\big),$$

where: $S(R_i) = \{S_1(R_i),\ S_2(R_i), \dots, S_n(R_i)\}$ represent the states of the resource;

$V(R_i) = \{V_1(R_i), V_2(R_i), \dots, V_n(R_i)\}$ is the value symbols associated with the resource;

$B(R_i) = \{b(V_i(R_i))\}$ indicates the output state probability, where $\sum_{i=1}^{i=m} b(V_i(R_i) = 1$;

$A(R_i) = \{a_{ij}\}$ is the probability of transition from state *i* to state *j*, where $\sum_{i,j=1}^{i,j=n} a_{ij}$=1, $i \ge 1$ and $j \le n$;

$I(R_i) = \{I_1(R_i), I_2(R_i), \dots,\ I_n(R_i)\}$ is the initial probability states of the resource, where $\sum_{i=1}^{i=n} I_i(R_i)$=1.

A sample MDP model of resources is shown in Fig. 1. The hidden states of the resources are handled using NSS as it characterizes every hidden state of the resources while processing the job requests by perceiving all information of the unobserved hidden states to optimally switch between exploration and exploitation dilemma of the resources while processing jobs requirements using neutrosophic truth, indeterminate, and falsehood membership function.

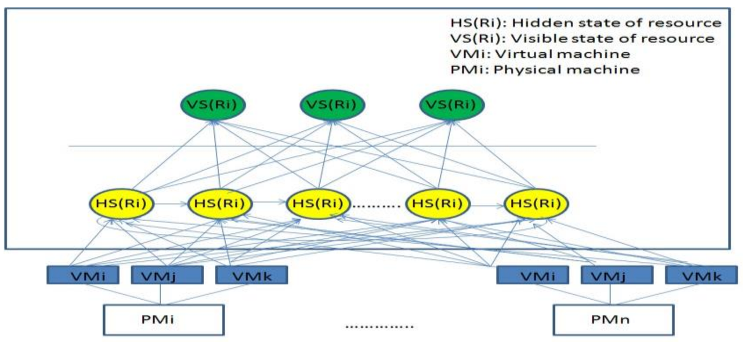


Fig. 1. A sample HMM model of resources

3.2. Jobs model

Based on the elasticity of resource usage, the jobs are broadly classified into two types; they are evolving, and malleable. Evolving jobs operate on the inflatable range of computing nodes and the malleable jobs starts with the available fewer nodes and gradually expands to more number of nodes. Only partial information is available about the resource requirement of the jobs. Hence the jobs are modelled using POMDP,

(11) $$\text{POMDP}(J_i) = \left(S(J_i), A(J_i), P(J_i), R(J_i), \Omega(J_i), O(J_i)\right),$$

where:

$S(J_i) = \{S_1(J_i), S_2(J_i), \ldots, S_n(J_i)\}$ represent the states of the job;

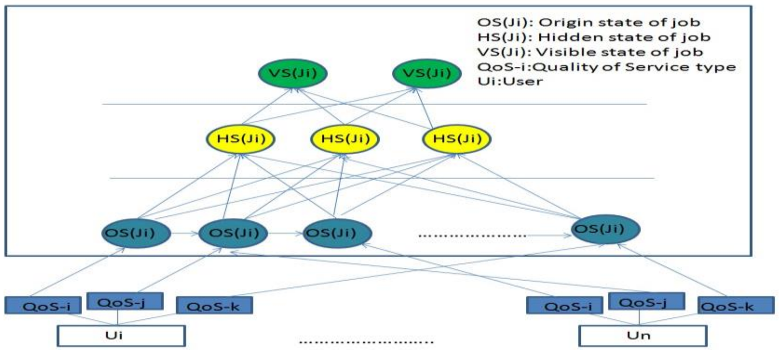
$A(J_i) = \{A_1(J_i), A_2(J_i), \ldots, A_n(J_i)\}$ represent finite set of actions;

$P(J_i) = \{a_{ij}\}$ is the probability of transition from state *i* to state *j*, where $\sum_{i,j=1}^{i,j=n} a_{ij}$=1, *i*≥1, and *j*≤*n*;

$R(J_i) = R(S_i(J_i), A_i(J_i))$ is the reward model of the jobs;

$\Omega(J_i) = \{\Omega_1(J_i), \Omega_2(J_i), \ldots, \Omega_n(J_i)\}$ is the finite set of observations;

$O(J_i) = O\left(\frac{o}{(S_i(J_i), A_i(J_i))}\right)$ is the observation model of the jobs.



OS(Ji): Origin state of job
HS(Ji): Hidden state of job
VS(Ji): Visible state of job
QoS-i:Quality of Service type
Ui:User
VS(Ji)
VS(Ji)
HS(Ji)
HS(Ji)
HS(Ji)
OS(Ji)
OS(Ji)
OS(Ji)
OS(Ji)
QoS-i
QoS-j
QoS-k
QoS-i
QoS-j
QoS-k
Ui
Un

Fig. 2. A sample POMDP model of jobs

A sample POMDP model of jobs is shown in Fig. 2. The partial state of the jobs are handled using FNSS as it characterizes every partial state of the jobs by giving equal importance to varying resource requirements of the jobs and precisely solves hugely intractable uncertainties in the partial states of the jobs using fuzzy neutrosophic truth, indeterminate, and falsehood membership function.

4. Proposed work

The proposed NSS and FNSS model based expected 3-SARSA learning resource provisioning framework is shown in Fig. 3.

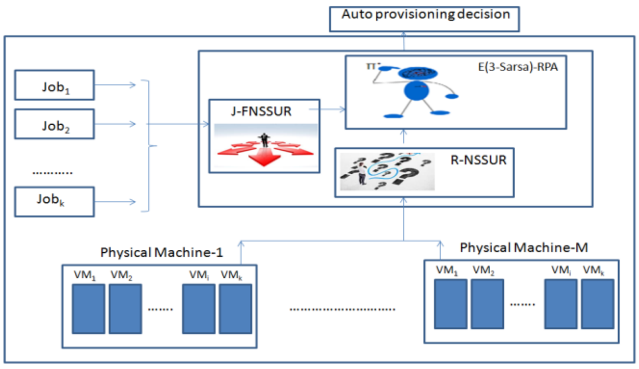
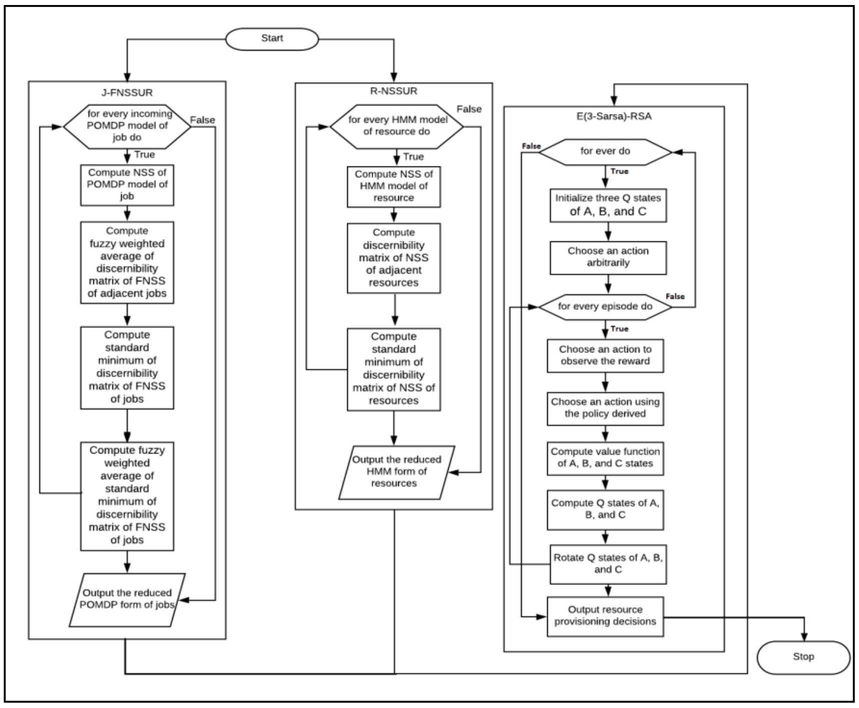


Fig. 3. Proposed NSS enabled expected 3-SARSA learning Resource provisioning framework



Start
J-FNSSUR
for every incoming POMDP model of job do
False
True
Compute NSS of POMDP model of job
Compute fuzzy weighted average of discernibility matrix of FNSS of adjacent jobs
Compute standard minimum of discernibility matrix of FNSS of jobs
Compute fuzzy weighted average of standard minimum of discernibility matrix of FNSS of jobs
Output the reduced POMDP form of jobs
R-NSSUR
for every HMM model of resource do
Compute NSS of HMM model of resource
Compute discernibility matrix of NSS of adjacent resources
Compute standard minimum of discernibility matrix of NSS of resources
Output the reduced HMM form of resources
E(3-Sarsa)-RSA
for ever do
Initialize three Q states of A, B, and C
Choose an action arbitrarily
for every episode do
Choose an action to observe the reward
Choose an action using the policy derived
Compute value function of A, B, and C states
Compute Q states of A, B, and C
Rotate Q states of A, B, and C
Output resource provisioning decisions
Stop

Fig. 4. Flowchart of the proposed framework

It is composed of three modules; one is Resource Neutrosophic Soft-Set Uncertainty Reducer (R-NSSUR*)*, Job Neutrosophic Soft-Set Uncertainty Reducer (J-FNSSUR), and Expected 3-SARSA learning agent. The R-NSSUR module handles resource uncertainty, J-FNSSUR module handles job related uncertainty, and Expected 3-SARSA Resource provisioning Agent (E(3-SARSA)-RPA) forms optimal policies for resource provisioning by triple coupling the action-value pair of the agent. Fig. 4 gives the flowchart representation of the proposed framework.

4.1. Resource Neutrosophic Soft-Set Uncertainty Reducer (R-NSSUR)

The R-NSSUR inputs the HMM model of the resource HMM(R_i) to construct NSS of the HMM model of the resource NSS-HMM(R_i) by calculating truth-membership function TNSS(R_i), indeterminacy function INSS(R_i), and falsity membership function FNSS(R_i) of the resources. The Discernibility matrix is constructed to obtain minimal reduct of resources by removing the irrelevant parameters from the hidden state of resources D(NSS-HMM(R_i, R_j)) then the entries in the matrix are normalized using standard minimum of the discernibility matrix. Finally the reduced soft-set of HMM model of resource HMM(R_i)$^{\text{rd}}$ is produced as output. The working of R-NSSUR is shown in Algorithm 1 and satisfies the Theorems 4.1 and 4.2.

Algorithm 1. Working of R-NSSUR

```
Step 1. Begin
Step 2. Input: $\text{HMM}(R_i) = (S(R_i),\ V(R_i),\ B(R_i),\ A(R_i),\ I(R_i))$
Step 3. Output: $\text{HMM}(R_i)^{\text{rd}}=(S(R_i^{\text{rd}}), V(R_i^{\text{rd}}), B(R_i^{\text{rd}}), A(R_i^{\text{rd}}), I(R_i^{\text{rd}}))$
Step 4. for every $R_i \in R$ do
Step 5.   Form NSS of HMM($R_i$), i.e.,
Step 6.        NSS-HMM($R_i$)=( $R_i$, TNSS($R_i$), INSS($R_i$), FNSS($R_i$))
Step 7.     Calculate the Discernibility matrix $D$(NSS-HMM($R_i$, $R_j$)), i.e.,
          $D$(NSS-HMM($R_i$, $R_j$))=$\{a \in A | g(R_i, a) \neq g(R_j, a)\}$
                 $\Phi$          $\{a_1\}$    $\{a_1, a_4\}$
                 ...          ...          ...
             $\{a_1, a_3, a_4\}$     ...          $\Phi$
Step 8.        Calculate standard minimum $\Delta^* D$(NSS-HMM($R_i$, $R_j$)), i.e.,
Step 9.             $\Delta^* D$(NSS-HMM($R_i$, $R_j$))= $(a_i \wedge a_j) \vee (a_k \wedge a_l)$
             $\{a_1^*, a_4^*\}$    $\{a_3^*\}$          $\Phi$
                 ...          ...          ...
               $\{a_2^*\}$         ...       $\{a_2^*, a_5^*\}$
Step 10. End for
Step 11. Output all reduced resources in HMM form HMM($R_i$)$^{\text{rd}}$
Step 12. End
```

Theorem 4.1. Let HMM(R_i) be the input model of the resource parameters and HMM(R_i)$^{\text{rd}}$ is the reduced model of the resource parameters, which is obtained by eliminating irrelevant parameters γ from HMM(R_i). Then γ is dispensable in HMM(R_i) if and only if HMM(R_i) – $\gamma \Longrightarrow$ HMM(R_i)$^{\text{rd}}$.

Theorem 4.2. Let (NSS, E) be the input neutrosophic soft-set, where $E = \{e_1, e_2, \ldots, e_j\}$ is the parameter set representing NSS. If there exists irrelevant parameters like e_j^0 and e_j^i, they are added into reduced parameters set $E' = \{e_j^0, e_j^i\}$

to find subset A, i.e., $A \subseteq E'$then $E - A - E'$ produces reduced (NSS, E') excluding e_j^0 and e_j^i.

4.2. Job Fuzzy Neutrosophic Soft-Set Uncertainty Reducer (J-FNSSUR)

The J-FNSSUR inputs the POMDP model of the job $\mathrm{POMDP}(J_i) = (S(J_i), A(J_i), P(J_i), R(J_i), \Omega(J_i), O(J_i))$ to construct the FNSS of the POMDP model of the job $\mathrm{FNSS} - \mathrm{POMDP}(J_i)$ by calculating Truth fuzzy membership function $T_{\mathrm{FNSS}}\,(J_i)$, Indeterminacy fuzzy function $I_{\mathrm{FNSS}}\,(J_i)$, and Falsity fuzzy membership function $F_{\mathrm{NSS}}\,(J_i)$ of the jobs. The discernibility matrix is constructed to obtain minimal residue of resources by removing the irrelevant parameters from the hidden state of jobs $\left(\mathrm{NSS} - \mathrm{POMDP}(J_i, J_j)\right)$. Then the entries in the discernibility matrix are normalized in three levels, in the first level the fuzzy weighted average of discernibility matrix is calculated, in the second level standard minimum of discernibility matrix is calculated, and in the third level fuzzy weighted average square over standard minimum of discernibility matrix is formulated. Finally the reduced soft-set of POMDP model of jobs POMDP(J_i)rd is generated as output. The working of J-FNSSUR is shown in Algorithm 2 and satisfies the Theorems 4.3 and 4.4.

Algorithm 2. Working of J-FNSSUR

```
Step 1.  Begin
Step 2.  Input: $\mathrm{POMDP}(J_i) = (S(J_i), A(J_i), P(J_i), R(J_i), \Omega(J_i), O(J_i))$
Step 3.  Output: $\mathrm{POMDP}(J_i)^{\mathrm{rd}} =$
             $= (S(J_i)^{\mathrm{rd}}, A(J_i)^{\mathrm{rd}}, P(J_i)^{\mathrm{rd}}, R(J_i)^{\mathrm{rd}}, \Omega(J_i)^{\mathrm{rd}}, O(J_i)^{\mathrm{rd}})$
Step 4.  for every $J_i \in J$ do
Step 5.          Form FNSS of POMDP($J_i$)
Step 6.          FNSS-POMDP($J_i$)=($J_i$, $T_{\mathrm{FNSS}}(J_i), I_{\mathrm{FNSS}}(J_i)$, $F_{\mathrm{NSS}}(J_i)$)
Step 7.          Calculate discernibility matrix D(FNSS-POMDP($J_i$,$J_j$)), i.e.,
Step 8.        $D$(FNSS-POMDP($J_i$,$J_j$))=$\{\mu(a) \in A | g(J_{i_i}, \mu(a)) \neq g(J_j, \mu(a))\}$
Step 9.          Compute fuzzy weighted average of discernibility matrix, i.e.,
             FD $= \sum_{i=1}^{i=n}(w_i * \mu(a))$, where $w_i$ is weight assigned
Step 10.           D(FNSS-POMDP($J_i$,$J_j$))=
                   $\Phi$              {$a_1$/FD}    {$a_1$/FD, $a_4$/FD}
                   ...                 ...           ...
             {$a_1$/FD, $a_3$/FD, $a_4$/FD}    ...           $\Phi$
Step 11.         Calculate the standard minimum $\Delta^*D$(FNSS-POMDP($J_i$, $J_j$))
             $\Delta^*D$(FNSS-POMDP($J_i$, $J_j$))= $(\mu(a_i) \wedge \mu(a_j)) \vee (\mu(a_k) \wedge \mu(a_l))$
                   {$a_1$*, $a_6$*}    {$a_1$*}    {$a_1$*, $a_6$*}
                        ...            ...         ...
                     {$a_6$*}          ...       {$a_4$*, $a_7$*}
Step 12.   Calculate the fuzzy weighted average square over standard minimum
             of discernibility matrix, i.e., FD$^2$=$\sum_{i=1}^{i=n}(w_i * \mu(a))^2$, i.e.,
Step 13.           $D$(FNSS-POMDP($J_i$,$J_j$))=
             {$a_1^*$ /FD$^2$, $a_6^*$ /FD$^2$}   {$a_3^*$ /FD$^2$}   {$\Phi^*$/FD$^2$}
```

```
        ...                    ...            ...
        $\{a_6^*/\text{FD}^2\}$          ...            $\{a_4^*/\text{FD}^2, a_7^*/\text{FD}^2\}$
Step 14.  End for
Step 15.  Output all reduced jobs in POMDP form POMDP($J_i$)rd
Step 16.  End
```

Theorem 4.3. Let (FNSS, $\mu(E)$) be the input fuzzy neutrosophic soft-set, where $\mu(E) = \{\mu(e_1)\mu(e_1), \dots, \mu(e_j)\}$ is the membership value of the parameter set representing FNSS. If there exists membership values of irrelevant parameters like $\mu(e_j^0)$ and $\mu(e_j^i)$ which exhibits the probability of greater than 0.5, they are added into reduced parameters set $\mu(E') = \{\mu(e_j^0), \mu(e_j^i)\}$ to find subset *A*, i.e., $A \subseteq \mu(E')$ then $\mu(E) - A - \mu(E')$ produces reduced FNSS excluding $\mu(e_j^0)$ and $\mu(e_j^i)$, i.e., $(\text{FNSS}, \mu(E'))$.

Theorem 4.4. Let *f* be a function mapping from $f: (J_i, E) \to (J_k, E')$ then for any FNSS, i.e., FNSS(J_i, E) in *f* the following conditions hold good:

$$f(\emptyset) = \emptyset,$$
$$f(J_i, E) \subseteq f(J_k, E'),$$
$$f\big((J_i, E) \cup (J_k, E')\big) = f(J_i, E) \cup f(J_k, E'),$$
$$f\big((J_i, E) \cap (J_k, E')\big) \subseteq f(J_i, E) \cap f(J_k, E').$$

4.3. Expected 3-SARSA resource provisioning agent (*E*(3-SARSA)-RSA)

The *E*(3-SARSA-RSA) model inputs three states among which one is current state Q^A(*S*, *A*) and other two are expected states Q^B(*S*, *A*), and Q^C(*S*, *A*), the intention behind considering three states are it increases the probability of selecting action with highest action value. All three states are initialized to null value, an action is chosen in the beginning using an arbitrarily generated policy, the value function is computed for expected next two states $V_{S'}^B$ and $V_{S'}^C$ using which agent updates the states and actions, which allows the agent to converge at different values and move towards goal by maintaining safe distance from cliff. Later rotate operation is performed on all the three *Q* states with probability *P* to derive optimal policy for resource provisioning, this increases speed of learning and convergences to optimal solution. The working of *E*(3-SARSA)-RSA is given in Algorithm 3 and satisfies the Theorems 4.5, 4.6, and 4.7.

Algorithm 3. Working of *E*(3-SARSA)-RSA

```
Step 1. Begin
Step 2. Input: Q(S, A)= $Q^A(S, A)$, $Q^B(S, A)$, and $Q^C(S, A)$
Step 3. Output: Optimal Q(S, A)* for every (S, A) pair
Step 4. for ever do
Step 5.     Initialize Q(S, A)={$\Phi$}
Step 6.     Choose A from S using the arbitrary policy derived from Q(S, A)
Step 7.    for every episode do
Step 8.   Choose A to observe reward and next state ($r, S'$).
Step 9.  Choose $A'$ in $S'$ using $\prod$ derived from $Q^A(S,A)$, $Q^B(S,A)$, and $Q^C(S,A)$.
Step 10.      Compute value function $V_{S'}^A$=$\sigma_{A'}\pi(A'/S')\ A(S', A')$
Step 11.   Compute value function $V_{S'}^{B}$=$\sigma_{B'}\pi(B'/S')$ $Q^B(S', B')$
Step 12.   Compute value function $V_{S'}^{C}$=$\sigma_{C'}\pi(C'/S')$ $Q^C(S', C')$
Step 13.   Compute $Q^A(S,A)$= $Q^A$(S, A)+α[$\gamma + \gamma V_{S'}^{B} + \gamma V_{S'}^{C} - Q^A(S,A)$]
Step 14.   Compute $Q^B(S,A)$= $Q^B(S,B)$+α[$\gamma + \gamma V_{S'}^{c} + \gamma V_{S'}^{A} - Q^B(S,A)$]
Step 15.   Compute $Q^C(S,A)$= $Q^C(S,B)$+α[$\gamma + \gamma V_{S'}^{A} + \gamma V_{S'}^{B} - Q^C(S,A)$]
Step 16.   Update $S \leftarrow S'$and $A$← $A'$
Step 17.   Rotate $Q^A(S,A)$, $Q^B(S,A)$, and $Q^C(S,A)$ with probability $P$
Step 18.
               $Q^{A_1}{}'$   $Q^{A_2}{}'$   $Q^{A_n}{}'$
               $Q^{B_1}{}'$   $Q^{B_2}{}'$   $Q^{B_n}{}'$
               $Q^{C_1}{}'$   $Q^{C_2}{}'$   $Q^{C_n}{}'$
Step 19.   End for
Step 20. Compute resource provisioning decisions AD$^{A^*}$, AD$^{B^*}$, and AD$^{C^*}$, i.e.,
      AD$^{A^*}$ ← $\sum_{i=1}^{i=n}[Q^{Ai^*}]$, AD$^{B^*}$ ← $\sum_{i=1}^{i=n}[Q^{Bi^*}]$, and AD$^{C^*}$ ← $\sum_{i=1}^{i=n}[Q^{Ci^*}]$
Step 21.  End for
Step 22. Output resource provisioning decisions AD= {AD$^{A^*}$, AD$^{B^*}$, AD$^{C^*}$}
Step 23.  End
```

Theorem 4.5. For any $\text{HMM}(R_i)^{\text{rd}}$ of resources and $\text{POMDP}(J_i)^{\text{rd}}$ of jobs, the computed $Q(S, A)$ of E(3-SARSA)-RSA agent is always greater than computed value function of the agent at state S'.

Theorem 4.6. If $Q(S, A)$ is the Q state of single SARSA, $Q'(S,A)$ is the Q state of double SARSA and $Q''(S,A)$ is the Q state of triple SARSA then the learning rate α of $Q''(S,A) \geq \max(Q(S,A), Q'(S,A))$.

Theorem 4.7. The update rule of SARSA does not converge unless the learning rate drops to zero and exploration rate tends to zero, i.e., $Q(S,A) = Q(S,A)\alpha[\gamma + \gamma V_{s'} - Q(S,A)]$. Whereas, expected three SARSA does not wait till the next state action is performed, it converges as soon as the expected value of next state and action is obtained $Q^A(S,A) = Q^A(S,A) + \alpha[\gamma + \gamma V_{s'}^{B} + \gamma V_{s'}^{C} - Q^A(S,A)]$.

5. Interval-valued analysis

The efficiency of the proposed work is analyzed using interval-valued NSS analysis method [18]. Assume that the AD = {AD$^{A^*}$, AD$^{B^*}$, AD$^{C^*}$} be the generated provisioning decisions under consideration in a real SARSA learning agent and let E be the set of parameters describing the quality of $\text{AD}_i^* \in$ AD and $E = \{e_1 = \text{low}, e_2 = \text{medium, and } e_3 = \text{high}\}$. The analysis is carried out in following steps.

- Input the job and resource parameters.
- Construct n interval valued NSS, i.e., INSS_k consisting of three components, i.e., NSS truth membership function T_k, NSS indeterminacy function I_k and NSS falsity membership function F_k, which are populated as follows:

$$\text{INSS}_1 = [T_k, I_k, F_k] \quad \dots \quad \text{INSS}_n = [T_k, I_k, F_k]$$
$$\dots \qquad \dots \qquad \dots$$

$\mathrm{INSS}_m = [T_k, I_k, F_k\ ]$ ... $\mathrm{INSS}_n = [T_k, I_k, F_k\ ]$

- Input the threshold of $\mathrm{INSS}_k\langle\alpha,\beta,\gamma\rangle$ using average decision rules
- Compute average of INSS_k, i.e., $\mathrm{INSS}_k^{\mathrm{avg}}\langle\alpha,\beta,\gamma\rangle$

$$\mathrm{INSS}_k^{\mathrm{avg}}\langle\alpha,\beta,\gamma\rangle = \langle\frac{[T_k, I_k, F_k\ ]}{\mathrm{AD}_1^*},\dots,\frac{[T_k, I_k, F_k\ ]}{\mathrm{AD}_i^*}\rangle$$

- Output the average of $\mathrm{INSS}_k\langle\alpha,\beta,\gamma\rangle$, i.e., $\mathrm{INSS}_k^{\mathrm{avg}}$

$$\begin{array}{ccc} \mathrm{INSS}_1^{\mathrm{avg}}\langle\alpha,\beta,\gamma\rangle & \cdots & \mathrm{INSS}_n^{\mathrm{avg}}\langle\alpha,\beta,\gamma\rangle \\ \dots & \dots & \dots \\ \mathrm{INSS}_m^{\mathrm{avg}}\langle\alpha,\beta,\gamma\rangle & \cdots & \mathrm{INSS}_n^{\mathrm{avg}}\langle\alpha,\beta,\gamma\rangle \end{array}$$

- Compute the optimal choice C_i=$\max_{C_i \in C}\{C_i\}$

Example

A. Triple SARSA learning

Input a sample POMDP(J_i)rd = {$J_1(80)$,$J_2(60)$, $J_3(40)$,$J_4(34)$, $J_5(55)$, $J_6(41)$, $J_7(62)$,$J_8(63)$,$J_9(99)$} and the HMM(R_i)rd ={$R_1(32)$, $R_2(59)$, $R_3(55)$, $R_4(61)$, $R_5(83)$, $R_6(99)$, $R_7(67)$, $R_8(76)$, $R_9(80)$}

- Construct 3 $\mathrm{INSS}_k\langle\alpha,\beta,\gamma\rangle$

[0.5, 0.4, 0.1]	[0.5, 0.3, 0.2]	[0.1, 0.1, 0.8]
[0.4, 0.2, 0.4]	[0.3, 0.1, 0.6]	[0.1, 0.8, 0.1]
[0.5, 0.4, 0.1]	[0.3, 0.6, 0.1]	[0.3, 0.2, 0.5]

- Compute the threshold of $\mathrm{INSS}_k\langle\alpha,\beta,\gamma\rangle$+={[0.8, 0.2, 0.0], [0.4, 0.6, 0.0]}
- Compute the $\mathrm{INSS}_k^{\mathrm{avg}}\langle\alpha,\beta,\gamma\rangle$={$\mathrm{AD}_1^* = 0.5, \mathrm{AD}_2^* = 0.8,\ \mathrm{AD}_3^* = 0.2$ }
- Summarize the computed $\mathrm{INSS}_k^{\mathrm{avg}}$

0.8	0.5	0.2
0.1	0.3	0.4
0.5	0.1	0.9

Output the optimal choice C_i=0.9

B. Double SARSA learning

Input a sample POMDP(J_i)$^{\mathrm{rd}}$ = {$J_1(80)$, $J_2(60)$, $J_3(40)$, $J_4(34)$, $J_5(55)$, $J_6(41)$, $J_7(62)$,$J_8(63)$, $J_9(99)$} and the HMM(R_i)$^{\mathrm{rd}}$ ={$R_1(32)$, $R_2(59)$, $R_3(55)$, $R_4(61)$, $R_5(83)$, $R_6(99)$, $R_7(67)$, $R_8(76)$, $R_9(80)$}

- Construct 3-$\mathrm{INSS}_k\langle\alpha,\beta,\gamma\rangle$

[0.5, 0.3, 0.2]	[0.6, 0.3, 0.1]	[0.0, 0.1, 0.9]
[0.3, 0.4, 0.3]	[0.3, 0.4, 0.3]	[0.3, 0.3, 0.4]
[0.4, 0.2, 0.4]	[0.1, 0.5, 0.4]	[0.6, 0.0, 0.4]

- Compute the threshold of $\mathrm{INSS}_k\ \langle\alpha,\beta,\gamma\rangle^+$={[0.4,0.2,0.4], [0.5,0.3,0.3]}
- Compute the $\mathrm{INSS}_k^{\mathrm{avg}}\langle\alpha,\beta,\gamma\rangle$={$\mathrm{AD}_1^* = 0.1, \mathrm{AD}_2^* = 0.2,$ and $\mathrm{AD}_3^* = 0.6$ }
- Summarize the computed $\mathrm{INSS}_k^{\mathrm{avg}}$

0.1	0.5	0.3
0.8	0.6	0.2
0.4	0.3	0.2

Output the optimal choice C_i=0.52

C. Single SARSA learning

Input a sample POMDP$(J_i)^{\mathrm{rd}}$ = $\{J_1(80), J_2(60), J_3(40), J_4(34), J_5(55), J_6(41), J_7(62), J_8(63), J_9(99)\}$ and the HMM$(R_i)^{\mathrm{rd}}$ $=\{R_1(32), R_2(59), R_3(55), R_4(61), R_5(83), R_6(99), R_7(67), R_8(76), R_9(80)\}$

- Construct $\mathrm{INSS}_k\langle\alpha,\beta,\gamma\rangle$

[0.3, 0.2, 0.4]	[0.2, 0.2, 0.6]	[0.4, 0.2, 0.3]
[0.4, 0.3, 0.3]	[0.3, 0.4, 0.3]	[0.3, 0.5, 0.2]
[0.3, 0.4, 0.3]	[0.3, 0.4, 0.2]	[0.6, 0.2, 0.2]

- Compute the threshold of $\mathrm{INSS}_k\langle\alpha,\beta,\gamma\rangle^{+}$={[0.3,0.3,0.4],[0.5,0.4,0.1]}
- Compute the $\mathrm{INSS}_k^{\mathrm{avg}}\langle\alpha,\beta,\gamma\rangle$={$\mathrm{AD}_1^* = 0.1, \mathrm{AD}_2^* = 0.6,$ and $\mathrm{AD}_3^* = 0.3$ }
- Summarize the computed $\mathrm{INSS}_k^{\mathrm{avg}}$

0.5	0.5	0.3
0.0	0.2	0.5
0.6	0.4	0.8

Output the optimal choice C_i =0.2

The C_i of proposed expected 3-SARSA is 0.9, C_i of the double SARSA is 0.5, and the C_i of the single SARSA is 0.2. Hence the C_i of the expected 3-SARSA is higher compared to the C_i of the double and single SARSA.

6. Results and discussion

The performance of the expected 3-SARSA learning in the Proposed Work (PW) is compared with the fuzzy SARSA learning in the Existing Work (EW) with respect to throughput achieved and rate of learning [24]. The default parameters for the SARSA Algorithm are determined by measuring the performance of the jobs running on Virtual Machines (VMs) versus resources offered by the VMs. The system-wide performance of the jobs running on VMs is evaluated using interactive benchmark workloads with varying workload scenarios.

6.1. Experimental setup

For experimentation purpose, we used the Xen Hypervisor based paravirtualization model, over which more than 100 instances of VM's have been created. Each of the benchmark workloads is deployed on clusters of VM's, which are enabled with Hypertext Preprocessor (PHP) and MySQL accessibility services. To support memory-intensive behavior the connections timeout is set to 10 s and to prevent bottleneck situations memory consumption limit is not enforced on the applications running on VM's [24].

6.2. Benchmark applications

The typical workloads considered for performance evaluation are RUBiS, RUBBoS, and Olio. The RUBiS is a dynamic workload, modeled after the application of eBay.com, which consists of the emulator to create client jobs of varying load. The RUBBoS is modeled after the application of slashdot on-line news form, which provides both regular and moderate level of access to clients. Olio is a social events calendar application used to support Web 2.0 applications with networking functions like commenting on posts, posting the reviews, sharing the post, and tagging friends in the posts. To verify the efficiency of the proposed work with respect to throughput and learning rate, two types of experiments are carried out one is with the homogeneous workload and other with the heterogeneous workload.

6.3. Experiment-1: Homogeneous workload

For Experiment-1 the following workloads are being considered, i.e., (RUBiS; 21,000 browsing clients; time 50 s), and (RUBiS; 30,000 bidding clients, time 50 s). Table 1 shows the performance comparison of proposed work with existing work on homogeneous workload.

A graph of the number of iterations versus throughput (number of requests successfully completed per iteration) is shown in Fig. 5. The successful job completion rate of the proposed work considering RUBiS workload increases with the increase in the number of iterations for both browsing and bidding clients as the expected 3-SARSA Algorithm learns quickly with minimum exploration rate. But with respect to existing work, the successful job completion rate is moderate for RUBiS workload with bidding clients and it is low for RUBiS workload with browsing clients as the exploration rate of Fuzzy SARSA Algorithm is high.

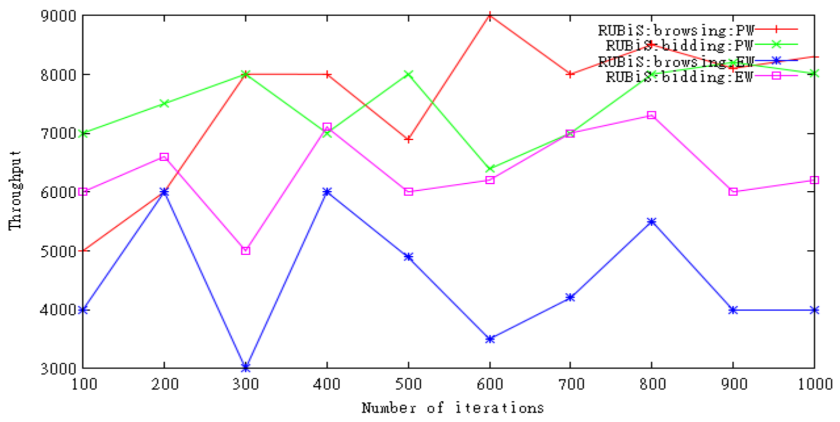


Fig. 5. Number of iterations versus throughput

A graph of time versus learning rate is shown in Fig. 6. The learning rate of the proposed work for RUBiS workload with browsing clients is found to be high between 0.7 and 0.8 and for the RUBiS workload with bidding clients the learning rate is moderate between 0.5 and 0.6 as the expected 3-SARSA Algorithm collects maximum possible rewards. Whereas the existing work learning rate for RUBiS workload with bidding clients is found to be moderate between 0.5 and 0.6 and RUBiS workload with browsing clients the learning rate is lower, i.e., between 0.1 and 0.2 because the Fuzzy SARSA Algorithm collects minimum possible rewards.

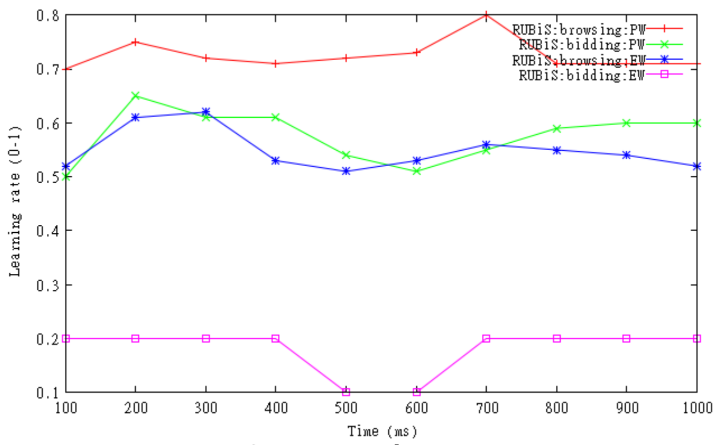


Fig. 6. Time versus learning rate

Table 1. Performance comparison of proposed work with existing work on homogeneous workload

Works considered for analysis	Workload type	Performance metric		
		Throughput (3000-9000 jobs)		
		Number of iterations (100-1000 iterations)		
		Fewer iterations (100-400)	Moderate iterations (400-700)	Higher iterations (700-1000)
Proposed work	RUBiS: Browsing	5000-8000	8000-9000	8000-9000
	RUBiS: Bidding	7000-8000	6000-8000	7000-8000
Existing work	RUBiS: Browsing	5000-6000	4000-6000	4000-5000
	RUBiS: Bidding	6000-7000	6000-7000	6000-7000
Works considered for analysis	Workload type	Learning rate (0-1)		
		Time interval (100-1000 ms)		
		Lower time interval (100-400)	Moderate time interval (400-700)	Higher time interval (700-1000)
Proposed work	RUBiS: Browsing	0.7-0.75	0.7-0.8	0.7-0.8
	RUBiS: Bidding	0.5-0.6	0.5-0.6	0.5-0.56
Existing work	RUBiS: Browsing	0.5-0.6	0.5-0.55	0.5-0.55
	RUBiS: Bidding	0.2-0.22	0.1-0.2	0.2-0.22

Table 1 compares the performance of the proposed work with the existing work concerning performance metrics like throughput and learning rate under the homogeneous workload. Concerning RUBiS workload the performance of the proposed work is very high towards throughput and is moderate towards learning rate whereas the performance of the existing work is moderate towards throughput but is weak towards learning rate.

6.4. Experiment 2: heterogeneous workload

For Experiment 2 the following workloads are being considered, i.e., (RUBiS; 3,000 browsing clients, 13,000 selling clients; time 50 s), (RUBBoS; 30,000 bidding clients, 12,000 concurrent clients; time 50 s), and (Olio; 30,000 concurrent clients; time 50 s). Table 2 shows the performance comparison of proposed work with existing work on heterogeneous workload.

Table 2. Performance comparison of proposed work with existing work on heterogeneous workload

Works considered for analysis	Workload type	Performance metric		
		Throughput (3000-9000 jobs)		
		Number of iterations (100-1000 iterations)		
		Fewer iterations (100-400)	Moderate iterations (400-700)	Higher iterations (700-1000)
Proposed work	RUBiS: Browsing	6000-9000	7000-9000	7000-8000
	RUBiS: Selling	7000-8000	7000-7500	7200-7500
Existing work	RUBiS: Browsing	5000-6000	5000-6000	4000-5000
	RUBiS: Selling	2000-3000	2000-5000	3000-4500
Works considered for analysis	Workload type	Learning rate (0-1)		
		Time interval (100-1000 ms)		
		Lower time interval (100-400)	Moderate time interval (400-700)	Higher time interval (700-1000)
Proposed work	RUBiS: Browsing	0.7-0.75	0.7-0.8	0.75-0.80
	RUBiS: Selling	0.5-0.7	0.55-0.65	0.65-0.75
Existing work	RUBiS: Browsing	0.2-0.4	0.2-0.4	0.2-0.4
	RUBiS: Selling	0.2-0.3	0.1-0.3	0.1-0.2
Works considered for analysis	Workload type	Throughput (3000-9000 Jobs)		
		Number of iterations (100-1000 iterations)		
		Fewer iterations (100-400)	Moderate iterations (400-700)	Higher iterations (700-1000)
Proposed work	RUBBoS: bidding	6000-9000	7000-9000	7000-7300
	RUBBoS: concurrent	4000-7000	4000-7000	6500-7000
Existing work	RUBBoS: bidding	5000-8000	5000-7000	6000-7000
	RUBBoS: concurrent	3000-3500	3500-4000	3000-3200
Works considered for analysis	Workload type	Learning rate (0-1)		
		Time interval (100-1000 ms)		
		Lower time interval (100-400)	Moderate time interval (400-700)	Higher time interval (700-1000)
Proposed work	RUBBoS: bidding	0.7-0.9	0.7-0.9	0.7-0.9
	RUBBoS: concurrent	0.7-0.9	0.5-0.9	0.7-0.72
Existing work	RUBBoS: bidding	0.3-0.5	0.5-0.51	0.3-0.4
	RUBBoS: concurrent	0.1-0.5	0.1-0.5	0.3-0.5
Works considered for analysis	Workload type	Throughput (3000-9000 jobs)		
		Number of iterations (100-1000 iterations)		
		Fewer iterations (100-400)	Moderate iterations (400-700)	Higher iterations (700-1000)
Proposed work	Olio: concurrent	6000-7000	6000-7500	7000-8000
Existing work	Olio: concurrent	2000-5000	2500-3500	3000-5000
Works considered for analysis	Workload type	Learning rate (0-1)		
		Time interval (100-1000 ms)		
		Lower time interval (100-400)	Moderate time interval (400-700)	Higher time interval (700-1000)
Proposed work	Olio: concurrent	0.1-0.6	0.6-0.62	0.6-0.9
Existing work	Olio: concurrent	0.1-0.6	0.2-0.6	0.2-0.4

RUBiS workload

The performance of the proposed and existing work is evaluated with respect to throughput and learning rate by considering browsing and selling clients of RUBiS workload.

A graph of the number of iterations versus throughput with respect to RUBiS workload with browsing and selling clients is shown in Fig. 7. The successful job completion rate is found to be high for the proposed work with both browsing and selling clients as the dynamic nature of the RUBiS workload is handled smoothly using NSS and FNSS enabled 3-SARSA Algorithm which is capable of handling different uncertainties in the input parameters. Whereas the successful job completion rate of the existing work is found to be lower for selling clients and moderate for browsing clients as the dynamic nature of the RUBiS workload is not handled properly in Fuzzy SARSA Algorithm because of the use of non-differentiable polygon membership function.

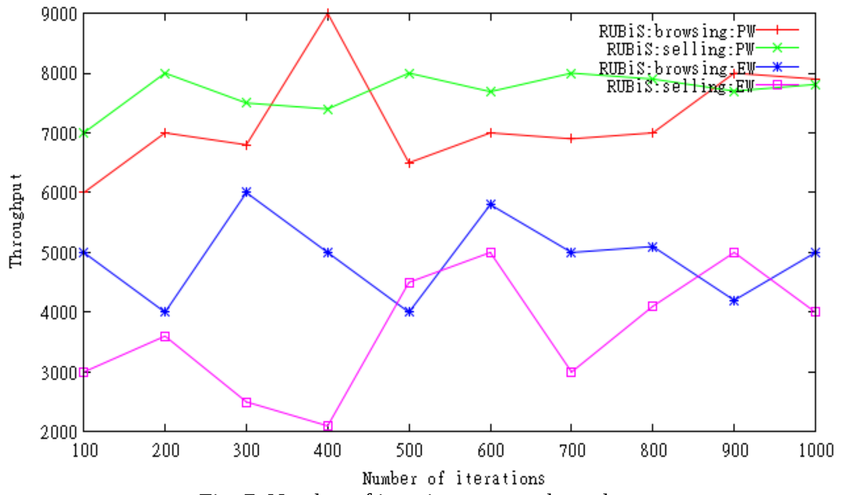


Fig. 7. Number of iterations versus throughput

A graph of time versus learning rate with respect to RUBiS workload with browsing and selling clients is shown in Fig. 8. The learning rate is high for the proposed work with browsing clients as it falls in the range of 0.7 to 0.8 and for selling clients it is in the moderate range, i.e., between 0.5 to 0.8 owing to the approximate and easily adaptable nature of 3-SARSA Algorithm. But the existing work learning rate is lower for both browsing and selling clients due to the individual specific nature of the polygon membership function used in the Fuzzy SARSA Algorithm.

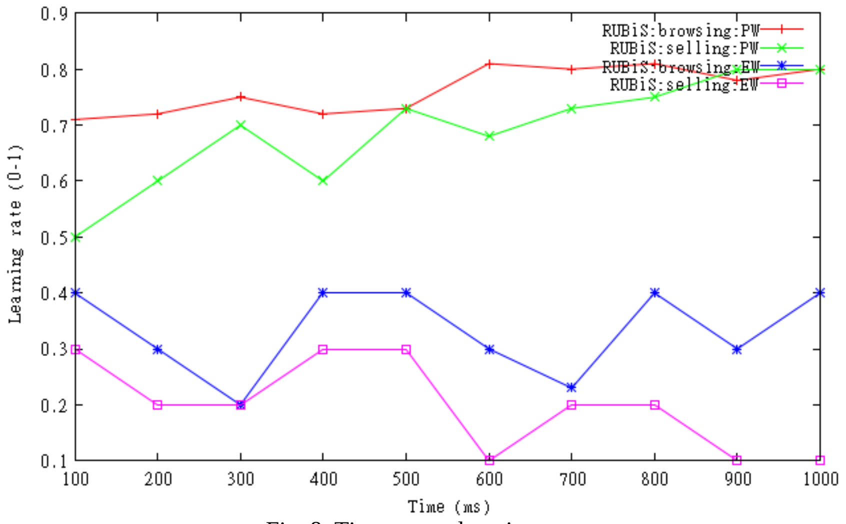


Fig. 8. Time versus learning rate

RUBBoS workload

The performance of the proposed and existing work is evaluated with respect to throughput and learning rate by considering bidding and concurrent clients of RUBBoS workload.

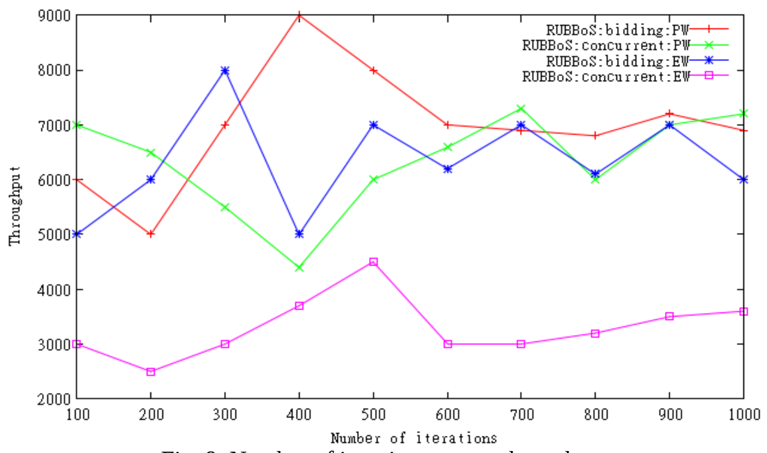


Fig. 9. Number of iterations versus throughput

A graph of the number of iterations versus throughput with respect to RUBBoS workload with bidding and concurrent clients is shown in Fig. 9. The successful job completion rate of the proposed work is high for bidding clients and remains moderate for concurrent clients as the 3-SARSA Algorithm easily handles the stochastically unstable phenomena in the workload using NSS and FNSS theory. Whereas the successful job completion rate of the existing work is found to be high for the bidding clients and low for concurrent clients as the Fuzzy SARSA Algorithm cannot easily handle the stochastically unstable phenomena in the workload because of the tedious procedure involved in the calculation of fuzzy membership function.

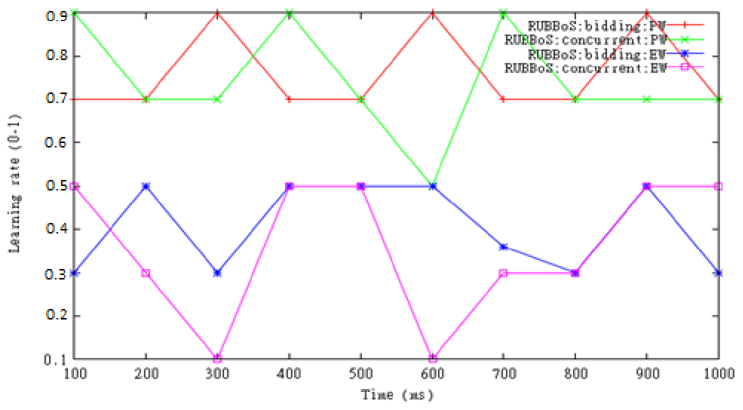


Fig. 10. Time versus learning rate

A graph of time versus learning rate with respect to RUBBoS workload with bidding and concurrent clients is shown in Fig. 10. The learning rate of the proposed work remained constant between 0.7 and 0.9 for the proposed work with both bidding and concurrent clients owing to the exploratory learning policy of 3-SARSA Algorithm. Whereas, the learning rate of the existing work is found to be lower between 0.1 and 0.5 for concurrent clients and moderate for bidding clients owing to the non-exploratory learning policy of Fuzzy SARSA Algorithm.

Olio workload

The performance of the proposed and existing work is evaluated with respect to throughput and learning rate by considering concurrent clients of Olio workload.

A graph of the number of iterations versus throughput with respect to Olio workload made up of concurrent clients is shown in Fig. 11. The successful job completion rate of the proposed work is found to be moderate as the 3-SARSA Algorithm can capture maximum possible uncertainties in the incoming workload using NSS and FNSS theory. But there is a huge drop in the successful job completion rate for the existing work as the Fuzzy SARSA Algorithm fails to capture all possible uncertainties in the incoming workload using not so continuously differentiable polygon fuzzy membership function.

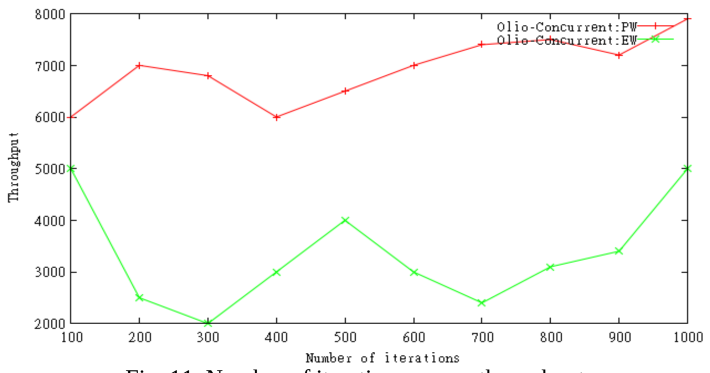


Fig. 11. Number of iterations versus throughput

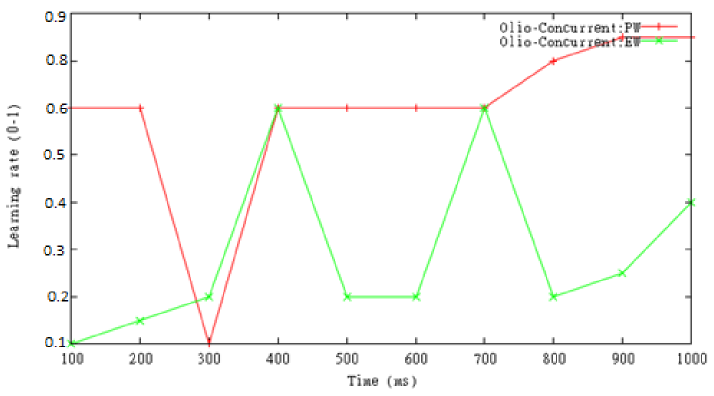


Fig. 12. Time versus learning rate

A graph of time versus learning rate with respect to Olio workload with concurrent clients is shown in Fig. 12. The learning rate of the proposed work considering concurrent clients is moderate between 0.6 and 0.8 because of the superior resource provisioning ability of the 3-SARSA Algorithm as it considers expected three states while forming the resource provisioning policies. Whereas, the learning rate of the existing work with concurrent clients is found to be fluctuating between 0.1 and 0.6 in a scale of 0 to 1 owing to not so superior resource provisioning ability of the Fuzzy SARSA Algorithm because it does not considers adjacent states while forming resource provisioning policies.

Table 2 compares the performance of the proposed work with the existing work concerning performance metrics like throughput and learning rate under the heterogeneous workload. Concerning RUBiS workload; the performance of the proposed work is high towards throughput and is moderate towards learning rate whereas the performance of the existing work is weak towards both throughput and learning rate. Concerning RUBBoS workload; the performance of the proposed work is moderate towards both throughput and learning rate whereas the performance of the existing work is moderate towards both throughput and learning rate. Concerning Olio workload the performance of the proposed work is high towards throughput and is moderate towards learning rate whereas the performance of the existing work is weak towards throughput but is moderate towards learning rate.

7. Conclusion

The paper presents a new NSS and FNSS based expected 3-SARSA learning framework for resource provisioning in the cloud environment. Here the irrelevant parameters or outliers of the jobs and resources are reduced, this influences on the quality of the resource provisioning decision taken. The proposed agent compares the current state with the expected other three states to form optimal decision pertaining to resource provisioning, which increases the number of rewards collected by the agent and stabilizes the learning. Its performance is found to be good with respect to successful job completion rate and learning rate. In future work, the expected 3-SARSA learning framework is improvised to be self-adaptable and capable enough of doing both resource scheduling and resource provisioning at runtime with minimum SLA violation, and the cost incurred.

References

1. Al-Dhuraibi, Y., F. Paraiso, N. Djarallah, P. Merle. Elasticity in Cloud Computing: State of the Art and Research Challenges. – IEEE Transactions on Services Computing, Vol. **11**, 2018, pp. 430-447.
2. Ullah, A., J. Li., Y. Shen, A. Hussain. A Control Theoretical View of Cloud Elasticity: Taxonomy, Survey and Challenges. – Cluster Computing, Vol. **21**, 2018, pp. 1735-1764.
3. Dar, A. R., D. Ravindran. A Comprehensive Study on Cloud Computing. – International Journal of Advance Research in Science and Engineering, Vol. **7**, 2018, pp. 235-242.
4. Babu, A. A., V. M. A. Rajam. Resource Scheduling Algorithms in Cloud Environment – A Survey. – In: Proc. of 2nd International Conference on Recent Trends and Challenges in Computational Models (ICRTCCM), 2017, pp. 25-30.
5. Parikh, S. M., N. M. Patel, H. B. Prajapati. Resource Management in Cloud Computing: Classification and Taxonomy. – Distributed, Parallel, and Cluster Computing, 2017, pp. 1-10.

6. E l k h a l i k, W. A., A. S a l a h, I. E l-H e n a w y. A Survey on Cloud Computing Scheduling Algorithms. – International Journal of Engineering Trends and Technology (IJETT), Vol. **60**, pp. 65-70.
7. P h a m, N. M. N., V. S. L e, H. H. C. N g u y e n. Energy Efficient Resource Allocation for Virtual Services Based on Heterogeneous Shared Hosting Platforms in Cloud Computing. – Cybernetics and Information Technologies, Vol. **17**, 2017, pp. 47-58.
8. S e n t h i l k u m a r, M. Energy-AwareTask Scheduling Using Hybrid Firefly-BAT (FFABAT) in Big Data. – Cybernetics and Information Technologies, Vol. **18**, 2018, pp. 98-111.
9. G i l l, S. S., R. B u y y a. Resource Provisioning based Scheduling Framework for Execution of Heterogeneous and Clustered Workloads in Clouds: From Fundamental to Autonomic Offering. – Journal of Grid Computing, 2018, pp.1-33.
10. P h a m, N. M. N., H. H. C. N g u y e n. Energy Efficient Resource Allocation for Virtual Services Based on Heterogeneous Shared Hosting Platforms in Cloud Computing. – Cybernetics and Information Technologies, Vol. **17**, 2017, pp. 47-58.
11. M e z n i, H., A. H a d j a l i, S. A r i d h i. The Uncertain Cloud: State of the Art and Research Challenges. – International Journal of Approximate Reasoning, Vol. **103**, 2018, pp. 139-151.
12. C a y i r c i, E., A. S. D. O l i v e i r a. Modelling Trust and Risk for Cloud Services. – Journal of Cloud Computing Advances, Systems and Applications, Vol. **7**, 2018, pp. 1-14.
13. O u a m m o u, A., B. T. A b d e l g h a n i, M. H a n i n i. Analytical Approach to Evaluate the Impact of Uncertainty in Virtual Machine Placement in a Cloud Computing Environment. 1st Winter School on Complex Systems, Modeling & Simulation, 2018, p. 1.
14. L i u, Y., K. Q i n, L. M a r t i n e z. Improving Decision Making Approaches Based on Fuzzy Soft Sets and Rough Soft Sets. – Applied Soft Computing, Vol. **65**, 2018, pp. 320-332.
15. D a n j u m a , S., T. H e r a w a n, M. A. I s m a i l, H. C h i r o m a, A. I. A b u b a k a r, A. M. Z e k i. A Review on Soft Set-Based Parameter Reduction and Decision Making. – IEEE Access, Vol. **5**, 2017, pp. 4671-4689.
16. N a s e f, A. A., M. K. E l-S a y e d. Molodtsov’s Soft Set Theory and Its Applications in Decision Making. – International Journal of Engineering Science Invention, Vol. **6**, 2017, pp. 86-90.
17. R i a z, M., M. R. H a s h m i. Fixed Points of Fuzzy Neutrosophic Soft Mapping with Decision-Making. – Fixed Point Theory and Applications, Vol. **1**, 2018, p. 7.
18. D e l i, I. Interval-Valued Neutrosophic Soft Sets and Its Decision Making. – International Journal of Machine Learning and Cybernetics, Vol. **8**, 2017, pp. 665-676.
19. B e n i f a, J. B., D. D e j e y. RLPAS: Reinforcement Learning-Based Proactive Auto-Scaler for Resource Provisioning in Cloud Environment. – Mobile Networks and Applications, 2018, pp. 1-16.
20. C h e n g, M., J. L i, S. N a z a r i a n. DRL-Cloud: Deep Reinforcement Learning-Based Resource Provisioning and Task Scheduling for Cloud Service Providers. – In: Proc. of 23rd Asia and South Pacific Design Automation Conference, 2018, pp. 129-134.
21. G o n g, Z., X. G u, J. W i l k e s, PRESS: PRedictive Elastic ReSource Scaling for Cloud Systems. – In: 6th IEEE/IFIP International Conference on Network and Service Management (CNSM), 2010, pp. 9-16.
22. R a m i r e z-V e l a r d e, R., A. T c h e r n y k h, C. B a r b a-J i m e n e z, A. H i r a l e s-C a r b a-j a l, J. N o l a z c o-F l o r e s. Adaptive Resource Allocation with Job Runtime Uncertainty. – Journal of Grid Computing, Vol. **15**, 2017, pp. 415-434.
23. G a n d h i, A., P. D u b e, A. K a r v e, A. K o c h u t, L. Z h a n g. Model-Driven Optimal Resource Scaling in Cloud. – Software & Systems Modeling, Vol. **17**, 2018, pp. 509-526.
24. A r a b n e j a d, H., C. P a h l, P. J a m s h i d i, G. E s t r a d a. A Comparison of Reinforcement Learning Techniques for Fuzzy Cloud Auto-Scaling. – In: Proc. of 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2017, pp. 64-73.
25. S o t i r i a d i s, S., N. B e s s i s, R. B u y y a. Self Managed Virtual Machine Scheduling in Cloud Systems. – Information Sciences, Vol. **433**, 2018, pp. 381-400.
26. G a w a l i, M. B., S. K. S h i n d e. Task Scheduling and Resource Allocation in Cloud Computing Using a Heuristic Approach. – Journal of Cloud Computing, Vol. **7**, 2018, pp. 1-16.
27. V o z m e d i a n o, R. M., R. S. M o n t e r o, E. H u e d o, I. M. L l o r e n t e. Efficient Resource Provisioning for Elastic Cloud Services Based on Machine Learning Techniques. – Journal of Cloud Computing: Advances, Systems and Applications, Vol. **8**, 2019, pp. 1-18.

28. B i t s a k o s, C., I. K o n s t a n t i n o u, N. K o z i r i s. A Deep Reinforcement Learning CloudSystem for Elastic Resource Provisioning. – In: Proc. of IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2018, pp. 21-29.
29. K u m a r, K. D., E. U m a m a h e s w a r i. Resource Provisioning in Cloud Computing Using Prediction Models: A Survey. – International Journal of Pure and Applied Mathematics, Vol. **119**, 2018, pp. 333-342.
30. T h e i n, T., M. M. M y o, S. P a r v i n, A. G a w a n m e h. Reinforcement Learning Based Methodology for Energy-Efficient Resource Allocation in Cloud Data Centers. – Journal of King Saud University – Computer and Information Sciences, 2018.
31. N a I k, K. B., G. M. G a n d h i, S. H. P a t i l. Pareto Based Virtual Machine Selection with Load Balancing in Cloud Data Centre. – Cybernetics and Information Technologies, Vol. **18**, 2018, pp. 23-36.
32. P e r u m a l, B., Ra. K. S a r a v a n a g u r u, A. M u r u g a i y a n. Fuzzy Bio-Inspired Hybrid Techniques for Server Consolidation and Virtual Machine Placement in Cloud Environment. – Cybernetics and Information Technologies, Vol. **17**, 2017, pp. 52-68.
33. M i r e s l a m i. S., M. W a n g, L. R a k a i, B. H. F a r. Dynamic Cloud Resource Allocation Considering Demand Uncertainty. – IEEE Transactions on Cloud Computing, 2019, pp. 1-14.

Received: 30.12.2018; Second Version: 18.05.2019; Accepted: 30.05.2019