IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 9, Issue 6, Dec - Jan 2022
ISSN: 2320 – 8791 (Impact Factor: 2.317)
www.ijreat.org

Deep Reinforcement Learning for Performance Evaluation of Cloud Workflow Scheduling

K. Basavaraj[1], S. Sharanappa[2]
[1,2]*Assistant Professor, STJIT, Ranebennur Karnataka*

ABSTRACT

Cloud computing is a platform that provides refined services to a large number of users over a network. Scheduling is one of the fundamental solutions to enhance the efficiency of all cloud-based services. Cloud scheduling assigns accessible cloud resources to tasks and optimizes numerous performance metrics. The massive scale of workflow as well as the elasticity and heterogeneity of cloud resources make cloud workflow scheduling difficult. In such a case, machine learning based scheduling models using neural networks can be leveraged to solve this challenging problem. The makespan and execution cost are the two critical performance metrics in workflow scheduling. In this study, a scheduling strategy for workflow is proposed that uses a deep neural network model in a reinforcement learning setting. The proponed Deep Reinforcement Learning based Workflow Scheduling (DRLWS) model minimizes the makespan and total execution cost. Simulated experiments show that the DRLWS model can find better results.

Keywords: Deep Reinforcement learning, Cloud Computing, Workflow Scheduling, Deep Neural network.

I.INTRODUCTION

Cloud computing is a pay-as-you-go service-oriented model used to provide services to users as per their respective demands. As the rapid increase in demand for these services, there is an underlying need of improving this platform to improve its quality of service (QoS). Scheduling plays an important role in improving all cloud-based services and optimizing overall system performance [1]. Scheduling workflows in cloud is referred to as matching workflow tasks onto respective acquired virtual machines (VMs), which is aimed at complete execution of workflows by considering their QoS requirements.

A workflow is a model of a complex computation, representing it as a group of specific smaller tasks and dependencies. Workflows may be simple or scientific. Scientific workflows are defined with the help of Directed Acyclic Graphs (DAG) as they frequently describe the precedence constraints of tasks in a workflow application [5]. The scientific workflows can be memory, CPU, or I/O intensive based on the nature of the user application. The CPU intensive workflows spend most of the time executing the tasks on the processors. However, the memory-intensive workflows require more physical memory to store the data on a server. Finally, the I/O-intensive workflows spend most of the time performing an input-output operation on the server [2]. Many scientific applications like astronomy, physics and bioinformatics are based upon these workflows.

Scientific workflows contain many distinct tasks and complex structures. The large number of tasks and inter-dependencies between different tasks make it difficult to efficiently schedule cloud resources to scientific workflows. The scheduler must consider these dependencies while scheduling workflows [3]. The workflow scheduling problem in heterogeneous computing systems like cloud is an NP-hard optimization problem, because the amount of computation required for finding optimum

solutions increases exponentially as the problem size increases [4]. Numerous state of the art workflow scheduling schemes has been put forwarded for scheduling scientific workflows in clouds. The existing algorithms offered solutions from numerous aspects. These algorithms are mainly divided into two types: heuristics [6] and meta-heuristics [7]. Single heuristic or a combination of heuristics and meta-heuristics [8] called hybrid schemes has been designed in the existing works, there remains a need for a workflow scheduling scheme that can quickly and efficiently solve a scheduling problem thereby optimizing the QoS constraints like makespan, cost, response time, resource utilization etc.,

Machine learning (ML) algorithms are the most popular methods for solving workflow scheduling problems. ML is a vast domain of artificial intelligence that give programs the capability to learn patterns, behavior, models and functions, and use these informations to make better decisions. Reinforcement Learning (RL) is an important branch of ML that does not learn from a labeled training set, but learns from the feedback information of the environment, which is vital for scheduling problems because high quality labeled data is impossible to generate [9]. Deep learning is a collection of techniques for using neural networks to solve ML tasks [10]. Deep reinforcement learning combines the RL and deep learning, which can solve more complex problems [11].

Based on above observations, in this work, the workflow scheduling problem is formulated into a Deep Reinforcement Learning Algorithm for multi-objective workflow scheduling aiming at optimizing both makespan and cost.

II. RELATED WORK

Scientific workflow applications are collections of several structured activities and fine-grained computational tasks related to data and control flow dependencies. Efficient scheduling is very important to scientific workflows. Scheduling deals with the allocation of VMs to workflow tasks. Due to the diverse set of workflow applications, the particular challenges and opportunities for workflow scheduling need to be developed. Hence, several works have been proposed in the field of workflow scheduling in the last two decades with the aim of optimizing one or more objectives such as makespan, mean flow time, mean tardiness, resource utilization, total execution cost, etc.. This section briefly reviews various workflow scheduling algorithms that has been proposed in different literatures.

A heuristic is a technique designed for finding an approximate solution to a problem with complex data more quickly when classic methods fail to find any exact solution. The traditional methods are mainly based on heuristic algorithms. Farzaneh Abazari et al. [12] designed a heuristic algorithm Multi Objective Workflow Scheduling (MOWS) based on the task's completion time and security requirements. A new attack response approach was presented in their work that reduces certain security threats providing a reliable scheduling of workflows. Cropper et al. [13], a multi-objective list scheduling approach for workflow applications is proposed. Based on a set of objectives constraints and weights defined by user, the algorithm attempts to find an appropriate Pareto solution in the region of interest for the users. The algorithm is customized and analyzed for four objectives: makespan, cost, reliability, and energy.

A metaheuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines to develop heuristic optimization algorithms. Metaheuristic algorithms are usually designed for global optimization. Shahram Jamali et al.[14] introduced a new hybrid metaheuristic algorithm based on particle swarm optimization (PSO) and gravitation search algorithms. The proposed algorithm, in addition to processing cost and transfer cost, takes deadline limitations into account. The

proposed workflow scheduling approach can be used by both end-users and utility providers. Shirvani et al. [15] presented a hybrid discrete particle swarm optimization (HDPSO) algorithm that has three main phases. At the first phase a random algorithm following by novel theorems is applied to produce swarm members; it is as input of presented new discrete particle swarm optimization (DPSO) algorithm in the second phase. To avoid getting stuck in sub-optimal trap and to balance between exploration and exploitation, local search improvement is randomly combined in DPSO by calling Hill Climbing technique at the third phase to enhance overall performance. Second and third phases are iterated till the termination criterion is met.

RL is one of the three basic machine learning paradigms together with supervised learning and unsupervised learning. RL has been developed as a promising approach to solve the sequential decision-making problems where the agent makes sequential decisions by continually interacting with the environment [16, 17]. Model-free deep reinforcement learning is a combination of the deep neural network (DNN) with RL which is capable of making intelligent sequential decisions in sophisticated environments. Orhean et al. [18] solved the workflow scheduling problem for heterogeneous distributed resources using reinforcement learning (Q-learning and SARSA) to reduce the task execution time by implementing a Machine Learning Box (MBox). The Machine Learning Box offers scheduling services through the perspective of reinforcement learning algorithm. Huifang Li, et al.[19] proposed an improved Deep Q Network (DQN)-based RL algorithm for workflow scheduling to optimize dual objectives like makespan and cost simultaneously. The performance of DQN and Actor-critic (AC) based RL algorithm in scheduling workflows was tested respectively, and then the reward function for the DQN algorithm was modified to improve its convergence.

Mikhail Melnik et al.[20] proposed a scheduling scheme based on Artificial Neural Networks and the principles of Reinforcement Learning for scheduling workflows. Experimental results showed that the Neural Network Scheduling (NNS) algorithm is able to learn how to provide qualitative schedules in terms of workflows' makespan. Wei et al. [21] proposed a QoS-aware job scheduling algorithm for applications in a cloud deployment. They used DQN with target network and experience replay to improve the stability of the algorithm. The main objective was to improve the average job response time while maximizing VM resource utilization. Wang et al. [22] solved workflow scheduling problem aiming at minimizing completion time and cost using a DQN model. The authors applied a deep-Q-network model in a multi-agent reinforcement learning setting to guide the scheduling of multi-workflows over infrastructure-as-a-service clouds. To optimize multi-workflow completion time and user's cost, they considered a Markov game model, which takes the number of workflow applications and heterogeneous virtual machines as state input and the maximum completion time and cost as rewards. To the best of authors' knowledge, few works can be found using DRL method in the literature. In this work, a novel scheduling strategy DRLWS is established for scheduling workflows in cloud.

III. PROBLEM DESCRIPTION

3.1 Workflow Application Model

The workflow scheduling problem is presented as a DAG W f =< T, E > where T = $\{t_i\}$ is a set of tasks and E = $\{e_{j,k}\}$ is a set of edges. Each task $\mathbf{t_i}$ represents a computational model or application that should be executed. An edge $e_{j,k}$ between tasks t_j and t_k corresponds to data dependencies among them. In this case, task $\mathbf{t_k}$ is a child task and it could not begin its execution before it receives all required input data from parent task t_j .Two functions are characterized : ***suc(t)*** is a set of children of task t and

prd(t) returns its parent tasks. Tasks without parents are called initial tasks and tasks without children are called exit tasks. An example of a workflow model is given in Figure 1**.** The labels *T1, T2….T11* represent the tasks and the nodes of the DAG, while the edges show the dependencies between the tasks. These dependencies show that child tasks cannot start before all parent tasks finish.

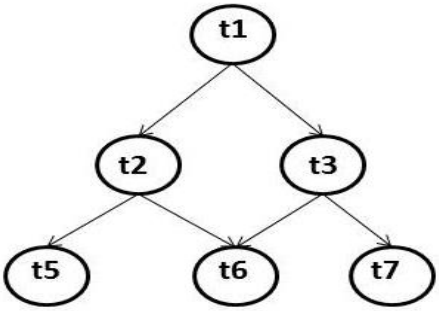


Figure 1. Sample DAG with 7 tasks

The cloud provides different types of VMs to its clients. These VMs are connected through a network and can communicate with each other, can be represented as $V=\{V_1,V_2,\ldots V_n\}$ where n is the number of VMs.

3.2 Workflow Scheduling Formulation

. Workflows are commonly used in distributed computing environments like clouds for their powerful capabilities in modeling a wide range of applications, including scientific computing, multiprocessors system and big data processing applications [18]. Therefore, the workflow scheduling problem is the mapping of workflow tasks to the virtual machines $(T \rightarrow V)$.For scheduling workflows deep reinforcement learning is used in this study and the objectives of the proposed work are to minimize the makespan and cost. The scheduling problem can be formulated as a multi-objective problem given by

$$\text{Minimize } F_w(x) = (f(x), c(x)) \tag{1}$$

$$\text{Where } f(x) = MS \tag{2}$$

$$c(x) = \sum_{i=1}^{r} p_{V_k} * Et(t_i) \tag{3}$$

The cost is the total rental cost of all VMs for the whole workflow execution and calculated as in (3). p_{V_k} represents cost per the interval unit of virtual machine V_k. The makespan *MS* is calculated as follows:

$$MS = \max\left(\sum_{i=1}^{n} Et(t_i)\right)$$

(4)

The execution time Et of task t_i on resource V_k, can be calculated as follows:

$$Et(t_i) = FS_i / PC_{V_k}$$

(5)

FS_i is the length of the task t_i and PC_{V_k} is the processing capacity of the virtual machine V_k.

The processing capacity PC of Virtual Machine V_k is given as follows:

$$PC_{V_k} = MIPS(V_k) * pe(V_k)$$

(6)

where $MIPS(V_k)$ - Processing speed of V_k measured in million instructions per second.

$pe(V_k)$ - Processing elements of V_k.

If task t_i and t_r are scheduled to different virtual machines V_h and V_k, the data transmitting time tr between the two VMs can be calculated by (7). Otherwise, tr is negligible.

$$tr(p,i) = \frac{ds(p,i)}{bw(h,k)}$$

(7)

where $ds(p,i)$ represents size of the communication data between task t_p and task t_i. $bw(h,k)$ represents the bandwidth between the VMs V_h and V_k.

3.3 Reinforcement Learning

Reinforcement learning combines the fields of dynamic programming and supervised learning to yield powerful machine-learning systems. RL is the branch of machine learning that deals with training agents to take an action *a*, as a response to the state s of the environment to get a notion of reward, *r* as shown in Figure 2. The ideas involved in RL were originally developed by Sutton and Barto [24].

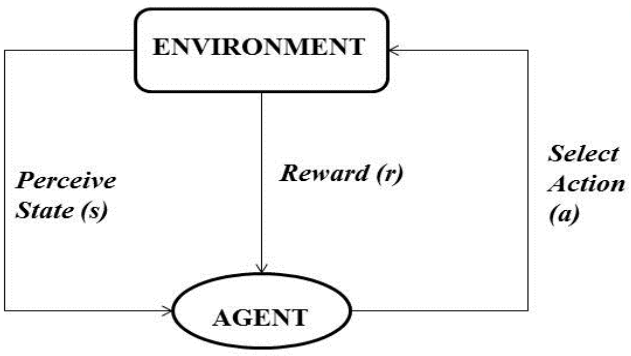


Figure 2. The basic reinforcement learning model

The task of the agent is to learn a policy for choosing actions in each state to receive the maximal long-run cumulative rewards. RL methods explore the environment over time to come up with a desired policy [25].

3.4. Deep Reinforcement Learning (DRL)

DRL is a subfield of machine learning that combines RL and deep learning. Deep learning (DL) is a form of machine learning that utilizes a neural network to transform a set of inputs into a set of outputs via an artificial neural network. DL is a collection of techniques and methods for using neural networks to solve ML tasks, Supervised Learning, Unsupervised Learning, or Reinforcement Learning. DRL is based on training deep neural networks to approximate the value functions. The key components of the DRL are described below.

Agent: The agent is the scheduler which is responsible for scheduling workflow tasks to VMs. At each time step, it observes the system state and takes an action. Based on the action, it receives a reward and the next observable state from the environment. The agent's sole objective is to maximize the total reward it receives in the long run.

Environment: The environment gives the agent a state. The agent receives the state and chooses an action. The action is applied to the environment and the environment returns a reward and a new state.

Action: The agent performs an action on the environment based on the state .The action is the selection of the appropriate VM to assign the task.

Reward: The reward represents the feedback value after the action was performed.

Episode: The sequence of actions from the start to the terminal state is an episode, or a trial. An episode is the time interval from when the agent schedules the first task and the state to when it finishes scheduling all the workflow tasks.

Value Function: Value functions are state-action pair functions that predict how good a certain action will be in a given state, or what the expected return will be. This function outputs an estimate of the reward the agent will receive until the end of the episode.

Deep-Q-Network (DQN): In DQN, a neural network is used to approximate a value function in a Q-Learning framework. The state is supplied as an input, and the output is the Q-value of all potential actions. The Q-learning algorithm learns how much long-term reward the agent will get for each state-action pair (s,a). This algorithm represents it as the function Q(s,a). The procedure for Q-learning algorithm is given below:

1. Reset the Q-values table, Q(s, a).
2. Observe the current state, s.
3. Choose an action, a, for that state s.
4. Take the action, and observe the reward, **r**, as well as the new state, s'.
5. Update the Q-value for the state using the observed reward and the highest reward achievable for the following state according to the formula.

$$Q(s,a) = Q(s,a) + \alpha\left[r + \gamma \max_{\alpha}, Q(s',a') - Q(s,a)\right]$$

6. Set the state to the new state, and repeat the process until a terminal state is reached.

State-Action-Reward-State-Action (SARSA): It is an RL algorithm and an on policy technique and uses the action performed by the current policy to learn the Q-value ie., the action value

[17]. The major difference between SARSA and Q-learning is that the maximum reward for the next state is not necessarily used for updating the Q-values. Instead, a new action, and therefore reward, is selected using the same policy that determined the original action. The name SARSA actually comes from the fact that the updates are done using the quintuple Q (s_t, a_t, r_t, s_{t+1}, a_{t+1}). Here, s_t, a_t are the original state and action, r_t is the reward observed in the following state and s_{t+1}, a_{t+1} are the new state-action pair. The procedure for SARSA algorithm is given below:

1. Initialize the Q-values table, $Q(s_t, a_t)$.
2. Observe the current state, s_t.
3. Choose an action, a_t, for that state s_t.
4. Take the action, and observe the reward, r_t, as well as the new state, s_{t+1}.
5. Choose an action, a_{t+1}, for that state s_{t+1}.
6. Update the Q-value for the state using the observed reward and the highest reward achievable for the following state according to the formula.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\left[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$$

7. Set the state to the new state and action to the new action.
8. Repeat the process until a terminal state is reached.

3.5. Proposed DRLWS scheduling algorithm

In order to solve the workflow scheduling problem given in (1) with makespan and cost minimization, a DRLWS algorithm is put forwarded. The learning agent in this algorithm is a scheduler which tries to schedule the workflow tasks to appropriate VMs. The reward it gets from the environment is directly associated with the key scheduling objectives such as makespan and cost. Therefore, by maximizing the reward, the agent learns the policy which can optimize the target objectives. When interacting with the environment and taking an action, the agent enters a new state and receives a reward from the environment.

The DRLWS system consisted of two components: environment and scheduling agent. As shown in Figure 3, the environment contained task queue, virtual machine cluster, resource and task managing module. The task queue was the pool to collect all the unimplemented tasks based on priority. The virtual machine cluster was the container of virtual machines. The resource and task managing module provides information about the available VMs and the tasks. The environment provides the reward for the actions taken by the agent.

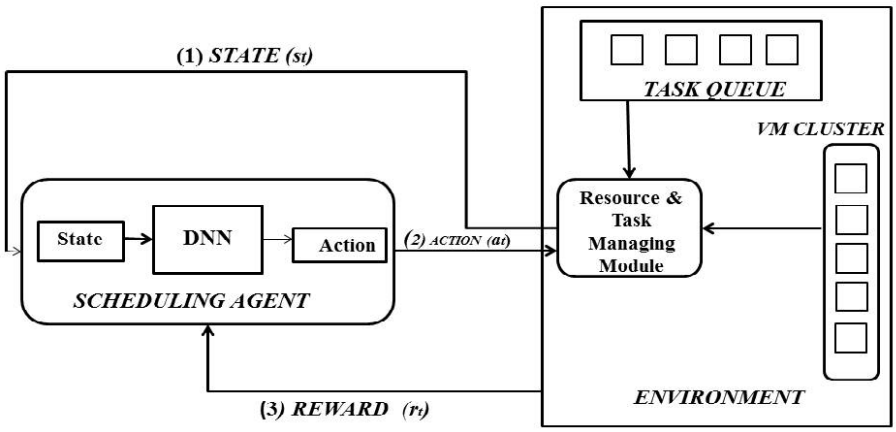


Figure 3. The Deep Reinforcement Learning Workflow Scheduling model

There are many workflow tasks waiting in the task queue to be processed. The DRLWS model selected tasks according to the priority calculated by $\Pr(t_i)$, the task priority function defined in equation (8).

$$\Pr(t_i) = FS_i + \max_{t_i \in succ(t_j)} \left(tr(h,k) + \Pr(t_j)\right) \quad (8)$$

In each time-step tm, the scheduling agent gets an observation (state s_t) from the environment, which includes (t_i, $Et(t_i)$,, $\Pr(t_i)$, N), where $Et(t_i)$ is the execution time of task t_i to be scheduled, , $\Pr(t_i)$ is the priority of the task t_i and N is the available VMs in the VM cluster.

Then it outputs an action a_t. An action is the selection of a specific VM to execute the current task. The rules for how to choose actions are called policy, a probability distribution in which a state is mapped to an action. The action at time moment tm is selected on the basis of the constructed utility function Q (s_t, a_t). The evaluation function is updated according to the SARSA principle :

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\left[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \quad (9)$$

where α is a learning rate and set to 0.1 to 0.7. It initially takes the value 0.1 and gradually increases with the training times. γ is a future actions importance factor and set to 0.5.

The environment transforms the action a_t taken in the current state s_t into the next state s_{t+1} and a reward r_t. The SARSA scheduling agent transforms the new state s_{t+1} and reward r_t into the next action a_{t+1}. The scheme of the neural network used in the scheduling agent is given in Figure 4.

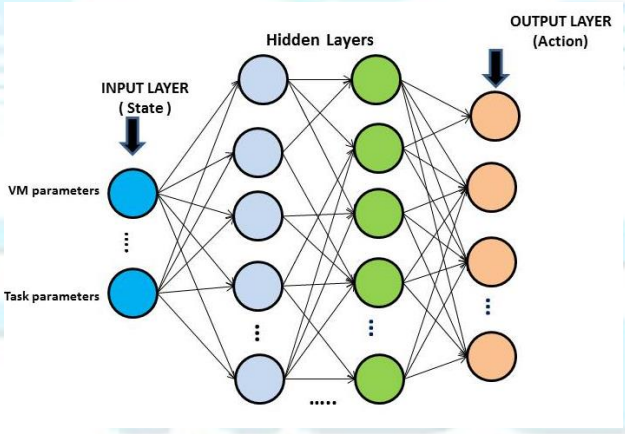


Figure 4. Neural Network Structure for DRLWS model

In the process of scheduling workflows, the makespan and cost can be obtained only when a complete scheduling terminates. Therefore, in the process of incomplete scheduling, we define the corresponding reward as 0. After a workflow completes its scheduling, the total reward r can be calculated such as

$$r = 1 - (1-\omega) * rt_m - \omega * rt_c \quad (10)$$

where rt_m is calculated using (4) and rt_c is calculated using (3) respectively. $\omega \in [0,1]$ is a coefficient which is used to set the target objectives.

The main steps involved in the training process of the DRLWS model are given below:

***Step* 1:** Initialize the agent and environment.

Step 2: Reset the Environment state.

Step 3: After observing the environmental state, the scheduling agent chooses the action using equation (9).

Step 4: Then the environment changes its state and generates a reward in return according to the action. The model stores these state-action pairs generated from this interaction into the memory pool.

Step 5: After completing an interaction, the model checks whether all workflow tasks complete its execution. If not so, go to ***Step* 3**.

Step 6: When all the tasks are executed, the total reward is calculated using (10).

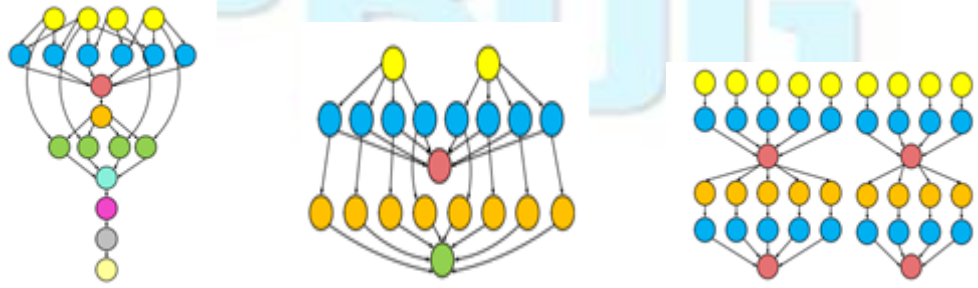
Step 7: The model extracts a batch of optimal sequences from the memory pool to train the DRL model for workflow scheduling;

Step 8: If the number of iterations meets its maximum value, the training of the DRL model terminates; otherwise, go to ***Step* 2**.

IV. RESULTS AND DISCUSSION

Cloud computing enhances its performance and throughput by using an efficient scheduling algorithm that executes a task on selected VM based on performance metrics. The metrics include execution time, deadline, cost, bandwidth of communication; makespan, reliability, scalability and many others. The proposed DRLWS technique intends to attain the scheduling of workflow tasks with minimum makespan and minimal execution cost. Makespan represents the completion time consumed by the last finished task while the cost represents the total execution cost incurred in scheduling all the workflow tasks. To illustrate the feasibility and efficiency of the DRLWS method, the performance of the algorithm is analyzed. The simulation was conducted in Workflowsim toolkit [27].

The DRLWS is evaluated on a real set of scientific workflow applications. Montage, Cybershake and Ligo Inspiral workflows provided by the Pegasus workflow management system are considered [26]. The structures of these workflows are given in Figure 5. The three workflows have different structures, data and computational requirements.



a) Montage b) Cybershake c) Ligo Inspiral

Figure 5. The structure of the scientific workflows

To determine the significance of the DRLWS scheduling model concerning makespan and cost, these scientific workflows are used. Four different categories of these workflows are chosen, small (30 tasks), medium (50 tasks), large (100 tasks) and extra-large (1000 tasks) for simulation. The

performance results of workflows such as makespan and cost reduction during the DRLWS learning process is illustrated in Table 1.

When a workflow is submitted for execution, the priority for all the tasks in the workflow is calculated based on its task dependencies. Then the available VMs with its specified parameters are identified. The scheduling agent gets all these information as state s_t from the environment.The agent then takes an action a_t which is the selection of VMs to execute the tasks.

The number of training episodes is set to 500. The average improvement of makespan and cost for Montage workflows as 48.8% and 5.75% respectively. For Ligo Inspiral workflows, the average improvement of makespan and cost are 45.05% and 8.98% respectively. Results of Cybershake workflows demonstrate the average improvement of makespan is 49.47% and the cost is 8.51%. According to results, DRLWS algorithm learning to create effective schedules across all workflows in comparison to schedules which were performed at initial steps.

Table 1. Makespan and Cost reduction during DRLWS learning process for all workflows

Data set	No. of Nodes	Makespan		Improvement	Cost		Improvement
		Initial	Final		Initial	Final	
Montage	**25**	116.10	54.25	53%	813.4	680.7	16.31%
	50	282.30	140.28	50.31%	1823.86	1532.7	15.93%
	100	421.22	220.32	47.69%	3683.51	3120.59	15.28%
	1000	3845.10	2134.20	44.49%	34845.71	32145.13	7.75%
Ligo Inspiral	**30**	2597.36	1232.15	52%	19982.28	18752.27	6.15%
	50	3971.55	2123.19	46.54%	34187.59	30643.76	10.37%
	100	6997.70	3916.8	44.02%	64140.20	59254.18	7.61%
	1000	66296.90	41432.11	37.5%	676211.41	596425.11	11.79%
Cybershake	**30**	371.42	149.428	59.7%	19923.45	18435.12	7.47%
	50	576.57	274.34	52.4%	39675.32	36532.67	7.92%
	100	840.03	465.04	44.64%	79586.12	73296.63	7.90%
	1000	5573.90	3280.82	41.15%	136081.7	121408.71	10.78%

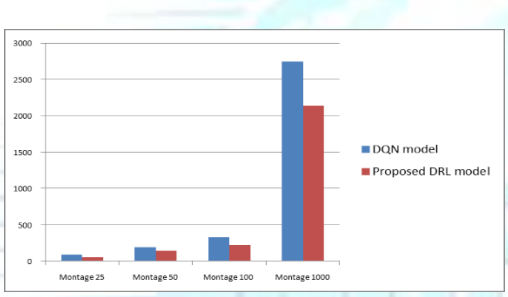
. A set of experiments are carried out with existing DQN technique to compare the performance of DRLWS. The total cost and makespan required for scheduling the scientific workflows using DRLWS and DQN are given in Table 2.

Table 2. Comparative results of Makespan and Cost for Montage, Ligo Inspiral and Cybershake

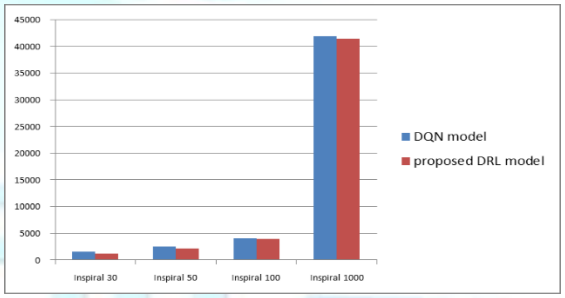
Data set	No. of Nodes	Makespan		Cost	
		DQN model	DRLWS model	DQN model	DRLWS model
Montage	**25**	90.4784	54.25	713.4	680.7
	50	187.8	140.28	1623.86	1532.7
	100	329.839	220.32	3180.51	3120.59
	1000	2742.11	2134.20	32845.71	32145.13

Ligo Inspiral	**30**	1497.04	1232.15	19982.28	18752.27
	50	2464.82	2123.19	31187.59	30643.76
	100	4102.19	3916.8	60140.20	59254.18
	1000	41923.28	41432.11	606211.41	596425.11
Cybershake	**30**	174.25	149.428	19123.45	18435.12
	50	298.23	274.34	37675.32	36532.67
	100	498.20	465.04	75186.12	73296.63
	1000	3415.20	3280.82	122081.7	121408.71

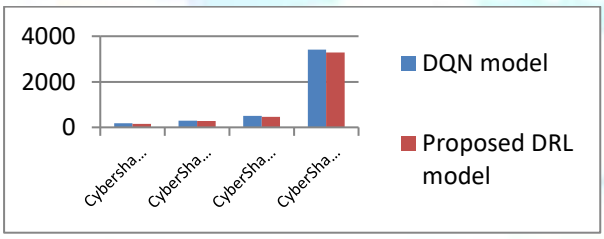
The makespan results of the proposed DRLWS model and existing DQN model for Montage, Ligoinspiral and Cybershake are shown in Figure 6. The horizontal axis represents the different set of nodes of the three scientific workflow applications considered for the experiments. The vertical axis gives the actual makespan taken by the DRLWS and DQN scheduling methods.



a) Makespan Analysis –Montage



b)Makespan Analysis – Ligo Inspiral



c) Makespan Analysis – Cybershake

Figure 6. Visual representation of makespan analysis

The results show that DRLWS significantly decreases the makespan compared with DQN for all three workflows. The makespan and the total cost of DRLWS against the DQN are statistically better in each case as shown in Table 2. The comparison analysis of makespan and cost evidently depicts that the DRLWS performs much better than DQN. In comparison to the DQN, the results exhibit that the DRLWS algorithm provides better optimality for minimizing makespan and cost using the Montage, Cybershake and Ligo Inspiral scientific workflow applications.

V. CONCLUSION

The problem of workflow scheduling in cloud has become a crucial research topic and it is a broader class of combinatorial optimization problem. The purpose is to search a most ideal approach to allocate tasks to available VMs thereby optimizing the performance metrics. The proposed DRLWS algorithm for cloud workflow scheduling problem is relies on Deep Reinforcement Learning. In order to expedite evaluation of DRLWS, the real application workflows Montage, Cybershake and Ligo

Inspiral utilised in diverse scientific areas were applied in this work. Conducted simulation experiments indicated that the DRLWS algorithm functioned significantly well.

REFERENCES

[1] K.Nithyanandakumari, S.Sivakumar, "Simulation of a Scheduling Strategy for Dependent Tasks in Cloud Computing", International Journal of Computational and Applied Mathematics. ISSN 1819-4966 Volume 12, Number 1 (2017)© Research India Publications http://www.ripublication.com.

[2] J.Sahni , DP.Vidyarthi, "A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment", IEEE Transactions in Cloud Computing Volume.6, Issue 1,pp.:2–18.,2018.

[3] X Kong, C Lin, Y Jiang, et al., " Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction", Journal of Network Computing Applications , Volume.34(4), pp: 1068–1077,2011.

[4] C. Jianfang, C. Junjie, Z. Qingshan, "An optimized scheduling algorithm on a cloud workflow using a discrete particle swarm," Cybernetics and Information Technologies, vol. 14, pp. 25-39, 2014.

[5] M. Masdari, S. ValiKardan, Z. Shahi, S. I. Azar, "Towards workflow scheduling in cloud computing: a comprehensive analysis", Journal of Network and Computer Applications, vol. 66, pp. 64-82, 2016.

[6] K. Nithyanandakumari , S.Sivakumar , "Performance Evaluation of Enhanced heterogeneous Earliest Finish Time Algorithm for DAG Task Scheduling in Cloud Computing", International Journal of Advanced Science and Technology, Vol. 28, No. 17, pp. 178-191, ISSN: 2005-4238 IJAST , Copyright ⓒ 2019 SERSC.

[7] K. Nithyanandakumari , S.Sivakumar, " Assessment of Ant Colony Optimization Algorithm for DAG Task Scheduling in Cloud Computing", International Journal of Advanced Trends in Computer Science and Engineering, Vol.9,No.4,July-Aug 2020, ISSN 2278-3091.

[8] Shengjun Xue, Mengying Li, Xiaolong Xu, Jingyi Chen, " An ACO-LB Algorithm for Task Scheduling in the Cloud Environment", Journal of Software, Vol. 9, no. 2, pp.466-473,February 2014.

[9] Wang X, Zhu M, Cheng Y. Reinforcement Learning Principle and Its Application. Beijing, China: Science Press; 2014.

[10] Lin CC, Deng DJ, Chih YL, Chiu HT, "Smart manufacturing scheduling with edge computing using multi-class deep Q network", IEEE Transactions Industrial Informatics, Volume 15(7), pp:4276-4284. 27. 2019.

[11] Dai H, Khalil EB, Zhang Y, Dilkina B, Le S, "Learning combinatorial optimization algorithms over graphs" In Proceedings of Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, 2017.