An Efficient tasks scheduling using DAG for Cloud Computing

[1]Zainab KashalanYeeser; [2] Prof. Khaldun I. Arif

[1,2] Department of Computer Science, College of Education for Pure Science, Thi_Qar University, Iraq;
; khaldun.i.a.2014@gmailedpcompzainab@utq.edu.iq
Article History: Received: 28 April 2021; Accepted: 07 May 2021; Published online: 12 July 2021

Abstract
Cloud Computing is a common model for storing, accessing, processing, and retrieving data from a distance. It combines high-performance computing with a massive resource pool.Cloud services are available on-demand to meet a variety of consumer QoS specifications. One of the main challenges in the field of the cloud computing is the task scheduling problem.Task Scheduling is known as NP-complete problem. The main objective of task scheduling is to assign tasks on to available processors with the aim of producing minimum Makespan and without violating the precedence constraints.in this paper modified the QL_HEFT algorithm, which efficiently distributes work among the processors and reduces the time it takes for applications to complete. The Proposed algorithm performs better than the current QL_HEFT algorithm, according to computational results.
Keywords: Cloud computing , Directed acyclic graph , Makespan , Reinforcement learning , Task scheduling .

1.INTRODUCTION

The Cloud is a huge, interconnected system of Powerful servers that provides businesses and individuals with services [1] The concept (Cloud Computing) refers to the ability for online users to share resources offered by the service provider. Without needing to buy expensive hardware, to leverage the high-service provider's capabilities[2]. The main goal of the cloud computing model is to allow users to share resources and data, Software as a service (SaaS), application as a service (PaaS), and infrastructure as a service (IaaS). As the number of cloud users has grown in recent years, the number of tasks that must be managed propositionally has increased, necessitating task scheduling[3]. methodology is based on Reinforcement learning Algorithms to use the Q-learning self-learning method to solve static task scheduling problems in cloud computing environments.Objects in RL model include agent, state, action, and reward. An agent is in charge of deciding what action to take in order to alter the state of the environment. The world rewards an agent in one of two ways: positively or negatively. An agent chooses an active transition state from the current to the next state in accordance with the concept of increasing the reward value.This process is replicated until the learning has completed the course[3]. We proposed algorithm to more effectively distribute workload across the processors and shorten the time it takes for applications to run Several benchmarks are used to evaluate the algorithm's accuracy. (including CyberShake, Montage and Epigenomics) on WorkflowSim, where the evaluation indicators include the makespan, ART, speedup and efficiency. in particular, the algorithm aims to optimize the incentive by using previous experience and potential knowledge and finding the most appropriate solution for task scheduling.Disadvantages of Q-learning Algorithm excessively large Q-table producing an excessive update time[7].The rest of this paper is laid out as follows. The related work on task scheduling algorithms in cloud computing environments is presented in Section 2. Section 3 explains the model of the system. The methodology of the research is

explained in Section 4. The results obtained and experiments performed are explained in Section 5. Section 6 concludes the research.

2 . RELATED WORK

The task scheduling algorithm's main goal is to ensure that tasks are completed as efficiently as possible. List scheduling algorithms are used in the task scheduling process. In list scheduling algorithms, there are two distinct phases. The first phase entails determining the tasks' priority, and the second phase entails assigning tasks to the processor in the order determined[3], They will be discussed as follow.In 2017 (Wei *et al.*)[4] t has been proposed a task scheduling algorithm based on Q-learning and the mutual value function (QS). To solve the problem of routine information sharing in current cooperative learning algorithms, the QS developed the task model and Q-learning model, as well as the constraint condition, in order to reduce the switching frequency of cooperative data while maintaining good learning performance.in 2017 (Akbar *et al.*) [5]this a new task scheduling algorithm called Median Deviation Dependent Task Scheduling (MDTS), which uses the Estimated Time to Compute (ETC) Median Absolute Deviation (MAD) of a task as a major attribute for calculating the ranks of the tasks. He uses a technique based on coefficient-of-variation (COV) that considers task and system heterogeneity to calculate the ETC of a particular DAG.in 2018 ((Dubey, Kumar)[6] The HEFT algorithm has been modified to construct a group of tasks based on their rating and map the task with the processor. After that, delegate the tasks to the heterogeneous processor so that the time of makepan can be decreased. Compared to the current HEFT and CPOP algorithms, decrease makepan. Increase the load balance compared to the current HEFT and CPOP algorithm Sleek time, efficient load balancing, energy consumption.in 2018 (Orhean, Pop and Raicu)[7]To solve the scheduling problem in distributed systems, the reinforcement learning algorithm proposed the algorithm used the reinforcement learning SARSA to consider the heterogeneity of the task nodes and the arrangement of tasks in a dependent DAG, finally obtaining a scheduling strategy with a shorter execution period ,The machine learning approach takes the heterogeneity of the nodes and their grid structure into account and the arrangement of tasks in a directed acyclic dependency graph, effectively deciding a scheduling strategy for a better execution period,Load balance is improved over the current HEFT and CPOP algorithms, Fast turnaround time , Inefficient load balancing, Use of energy .In 2019(Arif*et al.*)[8] PPEFT algorithm is to schedule dependent tasks in a heterogeneous environment. PPEFT algorithm has two phases.Tasks prioritization phase is a crucial stage of task planning. This phase assigns a priority to each task based on the ranks determined by parental prioritization. The ranks of all tasks are sorted to create a task list. Starting with the initial task, ranks are computed top-down in a DAG, Processor Assigning Phase In this phase,The processor assigns each task to it based on the tasks that were scheduled during the tasks prioritization phase,Overall efficiency has improved, Time and cost of execution are reduced,Suffer from unbalanced load. in 2020 (Jiang, Wang and Ye) [9] PVBTS algorithm dynamically the mission execution order based on the penalty value that is determined on the basis of the heterogeneity of the completion period of the execution on a given set of processors. PVBTS algorithm maintains a ready list of all the independent tasks in each stage, then selects the task with the highest penalty value and maps it to a processor that gives the task the minimum completion time for execution.in 2020 (Rajak and Shukla)[10]A new DAG task scheduling technique which is based on two well-known critical path and static level attributes. It created new CPS attributes, which are the number of critical path and static level, by using these attributes. The new approach operates in two stages, such as the importance of the mission and the choice of resources.

3 .SYSTEM MODELS

In this section, we'll go through the DAG task application model we used. as well asin cloud computing environments, the job scheduling model

3.1 Workflow model

A directed acyclic graph, G=(V,E), represents an application, with V representing the set of v tasks and E representing the set of e edges between the tasks. Each edge (i,j) E represents a precedence constraint, requiring task n_i to finish before task n_j can begin.Data is a v×v matrix of communication data, with $data_{i,k}$ indicating the amount of data to be transmitted from task n_i to task n_k. DAG scheduling object: node tasks are assigned object resources that must satisfy a chronological order constraint in order to reduce the total time to completion.

3.2 Scheduling model[11]

The work graph layer contains all of the functions shown in Fig(1). The resource graph layer represents the network of virtual machines that must be allocated. The cloud infrastructure layer, the last layer, is made up of network connected data centers.The aim of scheduling is to achieve optimal scheduling of tasks requested by users while also increasing the overall performance of the cloud computing system. The cloud computing platform can be used by users to apply their work. Each job is a DAG containing a certain number of tasks with dependencies, which are then queued.The Datacenter Broker assigns tasks to cloud nodes according to a scheduling plan, and the processing unit of the cloud node performs tasks and completes calculation requests submitted by users.

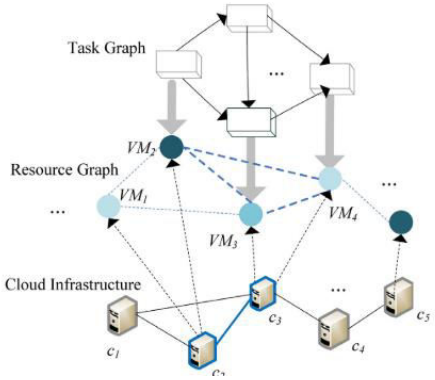


Fig (1) system model of DAG scheduling service on clouds [12].

4. RESEARCH METHODOLOGY

In this section, workflow scheduling problem is formulated.

4.1 Problem formulation

The main goal of this paper is to find a schedule for executing a workflow on cloud computing services while keeping the overall time of execution low.the background of the QL_HEFT algorithm is represented, The QL-HEFT algorithm is one of the most well-known static DAG task scheduling algorithms, and it has a good makespan efficiency. The QL-HEFT algorithm for solving task scheduling problems is first defined in terms of its design theory. There are two main phases to the QL-HEFT algorithm: Phase Optimal order of execution.We use a random selection strategy in state s and move to state s' in the first step, and Q-learning is used to sort the original task order into an optimal one. We also ensure that the agent's action selection on each state is legal to reduce the search space and algorithm complexity. It is founded on the task dependency relationship of the DAG, which means it follows the task dependency relationship of the DAG. We use equation (1) to change the corresponding Q-value Q(s,a) in the Q-table after the agent selects action an in state s. All of the states are visited in each episode, and a new Q-table is established.We obtain a Q-table that no longer changes in the iterative method after a finite number of iterations, indicating that the algorithm has converged. The final converged Q-table shows that (by ensuring that the task selection is legal, we use the maximum Q-value approach to achieve an optimal task order.We choose the highest possible Q-value each time). Setting the immediate reward r is critical for updating the Q-value so that the agent can improve its learning efficiency. The ranku is used as a form of immediate remuneration. Allocation of processors in

phases The best task order was decided in the first process The allocation strategy is used in the next step, as shown in equations (2), to assign the best processor to a task. In role is assigned to a processor in turn. Each task is assigned a processor in turn, based on the best task order, with the allocation concept being based on which processor can complete the task the fastest.

$$EST(T_i, P_k) = \max\left(\max_{T_j \in pred(T_i)}\left(FT\left(T_q\right) + c_{q,i}\right), AT(P_k)\right) \quad (1)$$

where pred(T_i) is the collection of task Ti's immediate predecessor tasks, FT(T_q) is the finish time, and AT(P_k) is the processor P_k earliest available time. Furthermore, an entry task is EST is 0. Similarly, on processor P_k, the earliest finish time (EFT) of task T_i is described as [13],

$$EFT(T_i, P_k) = w_{i,k} + EST(T_i, P_k) \quad (2)$$

4.2 Components of proposed algorithm

This section provides some context information on RL, MDP, and the Q-learning algorithm.

4.2.1 Reinforcement learning

Reinforcement learning (RL) is a general class of machine learning algorithms aimed at allowing an agent to learn how to act in a situation where the only feedback is a scalar reward signal[14]. RL is the method of learning to communicate with an unknown external environment without being directed in order to arrive at the best solution [13].The agent represents the entity that must decide at a certain time to assign a mission to whom. The agent will have several episodes to practice and out what arrangement of tasks on the nodes is best and obtains the lowest overall execution time. But this will only work if the assignments are not based on them.Each node must signal at one time in order to provide the agent with the understanding of the DAG structure. If it has in its queue a task representing the parent of this current task, and if it has tasks of the same parent, it may be performed in parallel.The last details the agent requires is the number of tasks assigned to each node at a time when it is appropriate to decide where the current task should be allocated.

1.State space :The remaining configuration capacity of the VM, denoted by s_i, can be defined as a state. The state space for the cloud computing platform can be expressed by a vector of the form: $s_i = (s_1, s_2,..., s_n)$[15].

2.Action space :The action set can be defined as (0/1) for each user job, which means the current (reject/receive) user job for a certain VM[16].

3.Immediate reward :The immediate reward is used for The correct running state and the performance of the task represent scheduling[15].

4.2.2 Markov decision process[17]

The issue of decision-making under uncertainty is formulated as an MDP, with the goal of maximising a cumulative reward value. In more formal terms, an MDP is a tuple with the following properties. More formally, an MDP is a tuple < S, A, T, R >characterised by:

1. A set of states S = {s1, s2,. . ., sN} where st is a state in S.
2. A set of actions A = {a1, a2, . . . , aM} available to the agent in each state s
3. A transition distribution T(s' \ s , a) maps a pair composed of a state s and an action a to a probability distribution of state s'.
4. A reward function R : S × A × S →R gives the expected reward when the agent makes the transition from state s to state s' using action a.

The immediate scalar reward received at time t is denoted by rt, where

rt = R(st+1 = s' \ st = s, at = a) = E{ rt \ st+1 = s', st = s, at = a}

Because the distribution of the next states and incentives is independent of the past through the current state and action, this process is Markovian. T(st+1 \ st, at) = T(s t+1 \ st, at, . . . , s1, a1)

The action selection mechanism in an MDP is described by a policy π : S × A →[0, 1] that specifies a probability of selecting an action a in a specific state s. The quality of a policy in state s is determined by the predicted discounted sum of future rewards.

$$V^{\pi} = E_{\pi}[R_t \backslash s_t = s] = E_{\pi}[\sum_{t=0}^{\infty} \gamma^t . r_t \backslash s_t = s] \quad (3)$$

where Rt is the return and is a real value denoted by the discount factor to weight the immediate reward more heavily than the reward received in the future, 0≤ γ < 1.

V^{π} is the expected return of an agent following policy π. Alternatively, the action value function for policy π, Q^{π}(s, a), is defined as the expected return when taking action a in state s under policy π. Thus

$$Q^{\pi}(s, a) = E_{\pi}[Rt \backslash st = s, at = a] \quad (4)$$

The goal of any MDP is to find the best policy π^* that maximises the expected return. The optimal state value function, for any state s, is

$$V^*(s) = max_{\pi} V^{\pi}(s) \quad (5)$$

The strategy used to evaluate optimum policies is a classification criterion for RL methods.

4.2.3 Q-learning

The Q-learning algorithm is a reinforcement learning technique in which the agent attempts to learn an optimal state-action strategy from a series of state action-rewards that reflect the agent's experiences with the environment. This approach does not involve knowledge of the world model in order to optimize the reward by calculating the utility of state-actions.the optimal strategy is achieved by selecting the best state-actions based on the learned utility values. The Q-learning technique is made up of an entity, a set of world states S, a set of actions A, and a description of how actions affect the world T : S × A → S, a set of rewards,also known as transition dynamicsR : S × A → R atable of utilities is given for each action Q : S × A → R and a policy π : S → A.The agent's aim is to optimize the reward, and in order to do so, it must learn which behavior from each state is the best, with the highest long term reward [18]. In order for such a solution to be successful, an agent must run several training episodes in order to explore and find the best policy.in a deterministic and finite universe, the term algorithm refers to the Q-learning algorithm. [7].

$$Q[s, a] = Q[s, a] + \alpha (r + d \cdot max_{a'} Q[s', a'] - Q[s, a]) \quad (6)$$

Algorithm Q-Learning

Input:Random state process.

Output:Q_Table.

1: ***function Q-Learning ($s_{initial}$, $s_{terminal}$, α, d)***

2: initialize Q[S, A]

3: s $s_{initial}$

4: While s! =$s_{terminal}$ do

5: select α

6: r =R (s , a)

7: s' =T (s , a)

8:Update Q(s,a) with Eg(6).

9: s s'

4.3 Proposed algorithm

In this section, The proposed algorithm is defined as a solution to the problem of workflow task scheduling in a cloud computing setting,The main objective is to reduce makespan.The proposed algorithm has two stages: obtaining the task's order and allocating the task to the processor in that order.

Proposed Scheduling Algorithm

Input: DAG all Tasks.
Output: The makespan.
Procedure:
1: Create DAG for all tasks.
2: Set gamma parameter, environment rewards in matrix R.
3: Initialize materix Q to zero.
4: Repeat for each episode.
5: Select an initial state.
6: While the goal state not reached Do.
7: Select possible actions for the current state. Go
8: to the next state.
9: Get maximum Q value with Eg (6).
10: Set next state as a current state.
11: Update Q(state, action) with Eg (6).
12: Obtian tasks order according to updated Q-table.
13: map task to the processor which have the minimum execution time.
14: Calculate the makespan.
15: Until no longer changes in makespan.

In the first step, Q-learning is used to sort the original task order into an optimal one. In the first state, we use a selection technique and then move on to the next state to ensure sufficient planning. After the agent selects action an in state s, we use Equation (6) to modify the corresponding Q-value (Q(s,a)) in the Q-table. Each episode traverses all of the states, resulting in a changed Q-table.After a finite number of iterations, we obtain a Q-table that no longer changes in the iterative process, implying that the algorithm has converged. Based on the final converged Q-table, we use the highest Q-value strategy (by ensuring that the task selection is legal, etc.). We select the highest Q-value each time to get the best task order.In order for the agent to have a better understanding of the situation. The job is then allocated to the processor who can complete it in the shortest amount of time. We assign a processor to each task based on the optimal task order, which is based on which processor can complete the task the fastest, and then we assign the task to that processor for execution.

4.4Simple Example

We'll show you a basic example of static task scheduling in this segment.

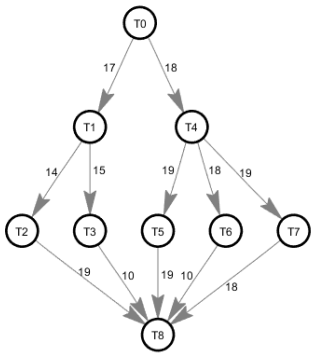


Fig 2 An application DAG example

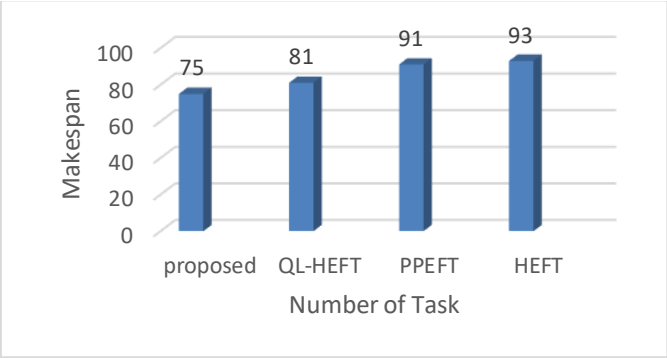
Table 1 Computational costs

Task	***P1***	***P2***	***P3***
T1	9	16	14
T2	10	19	18
T3	11	13	19
T4	13	8	17
T5	12	7	13
T6	13	9	16
T7	11	15	7
T8	14	5	11
T9	18	20	9

Table 2 Schedule produced by the Proposed algorithm

Task	*FT(P1)*	*FT(P2)*	*FT(P3)*	*Processor assigned*
T0	9	16	14	P1
T1	19	28	27	P1
T2	30	32	38	P1
T3	43	38	47	P2
T4	45	50	51	P1
T5	58	54	61	P2
T6	65	69	61	P3
T7	75	66	72	P2
T8	84	86	75	P3

Started with the first task in list that is T1 . T1 is execution time is compared to P1, P2, and P3, and it is determined that resource P1 is the best, so map the task to this resource .



Fig(3)Execution time of Proposed, HEFT,PPEFT and QL_HEFT algorithms

Figure 3 shows the comparison of the overall Makespan of Proposed algorithm with HEFT , PPEFT and QL_HEFT algorithms. This comparison is made by taking three processors and nine tasks.

5. RESULTS AND DISCUSSION

In this section, the results of proposed Algorithm are compared with HEFT, PPEFT and QL-HEFT.

5.1 Experimental evaluation

The workflowsim extended version of the cloudsim simulator is used for the experimental evaluation. According to the DAG tasks graph, the tasks in the sequence are provided by workflow tasks.Three scientific workflow applications are used to test the experiment (Montage,Cybershake and Epigenomics workflow).

5.2 Performance metrics

The following metrics are used to compare the performance of scheduling algorithms:

- Makespan

The schedule length (makespan)The completion time of the last task when all tasks have been completed is a key metric for calculating the efficiency of a task scheduling algorithm. The gap between two points is referred to as the markspan.

Makespan=max(AT(Ti)) , iϵ{1,2,…,n} (8)

- Speedup

The ratio of sequential execution time (SET) to parallel execution time (PET) is the speedup value for a DAG (PET). The SET is determined by allocating each task to a single processor, while the PET is computed by allocating tasks to multiple processors.

$$\text{Speedup}= \frac{\min Pj\in Q\{\sum_{ni}\in VWi,j\}}{makespan} \quad (9)$$

3. Efficiency

Efficiency is defined as the ratio of the speedup value to the total resources used in the task scheduling graph.

$$\text{Efficiency}= \frac{\text{speedup}}{\text{total resources}} \quad (10)$$

- ART

The average response time (ART) is the time it takes for a task to be completed from the time it can be started. Smaller ART values indicate that users would have a better time completing the mission. Eq(11) may be used to describe the ART:

$$\text{ART}=\frac{1}{n}\sum_{i=1}^{n}\left(EFT(T_i,P_k)-\max_{T_q\in pred(T_i)}(FT(T_q)+c_{q,i})\right) \quad (11)$$

The proposed algorithm's most important parameters are efficiency and convergence speed. The first parameter is the discount factor, which indicates how much weight is given to future rewards. The greater

the commitment to immediate compensation, the smaller the discount factor, and vice versa.The second parameter represents an agent's learning rate and shows how much attention is paid to forward benefit during the learning process. The findings of the experiment are discussed. The discount factor is set to 0.8, and the learning rate is set to 1.0, according to the experiments.
The curve fluctuation is less and the convergence speed is faster at this value. Table (3) lists the parameters used in the proposed algorithm.

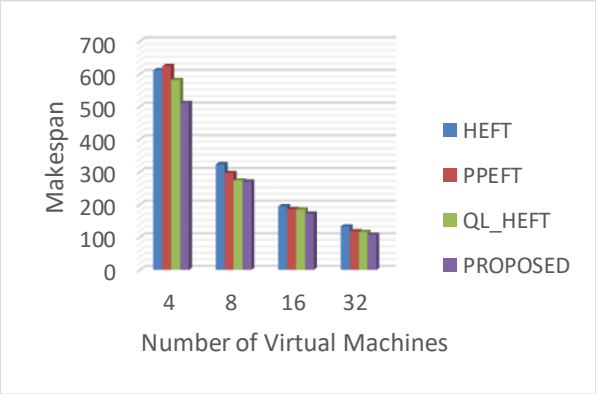
Table(3)Parameters of Proposed Algorithm

Parameters	***Value***
Discount factor	**0.8**
Learning rate	**1.0**
Numberof Tasks	**25,50,100,1000**
Number of VMs	**4,8,16, 32**
Benchmark types	**3**

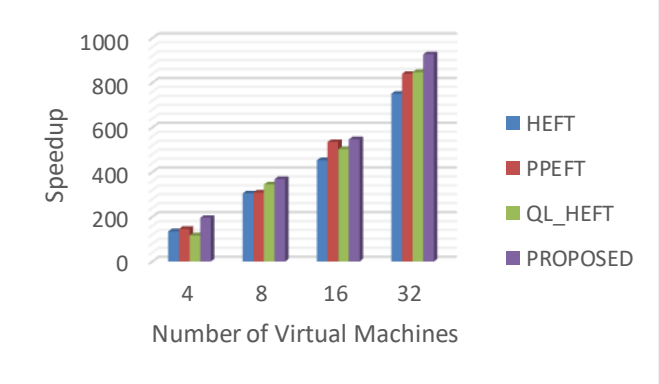
5.3 Experimental results

Our proposed algorithm has been checked with scientific workflow applications to equate its output to that of other current algorithms. Following are the comprehensive findings and discussions for each workflow.

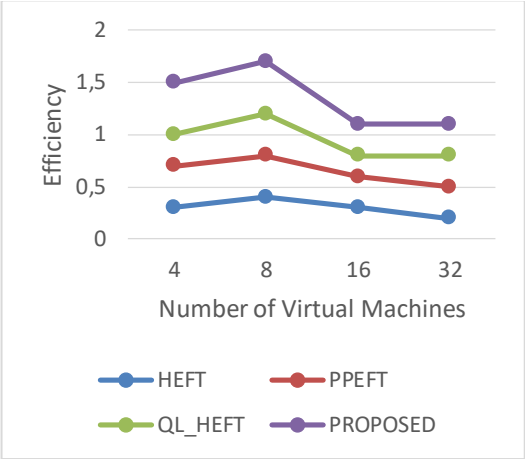
- Experiments with Montage benchmark



Fig(4)Makespan comparison of proposed and existing algorithms for Montage workflow application



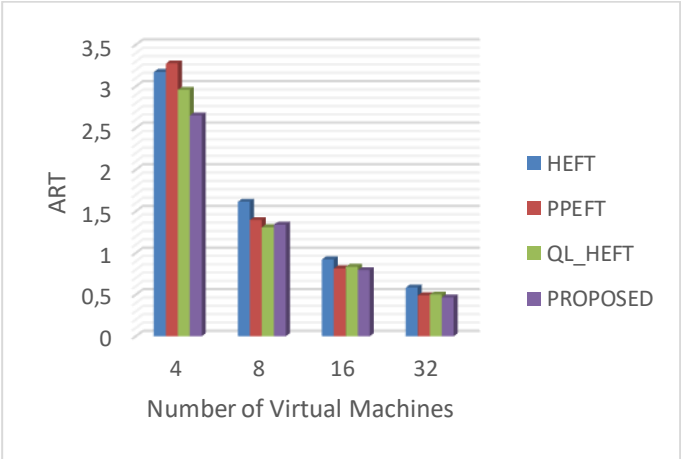
Figure(5)Speedup comparison of proposed and existing algorithms for Montage workflow application



Figure(6)Efficiency comparison of proposed and existing algorithms for Montage workflow application

Table(4)Response Time for Montage Scientific Workflow

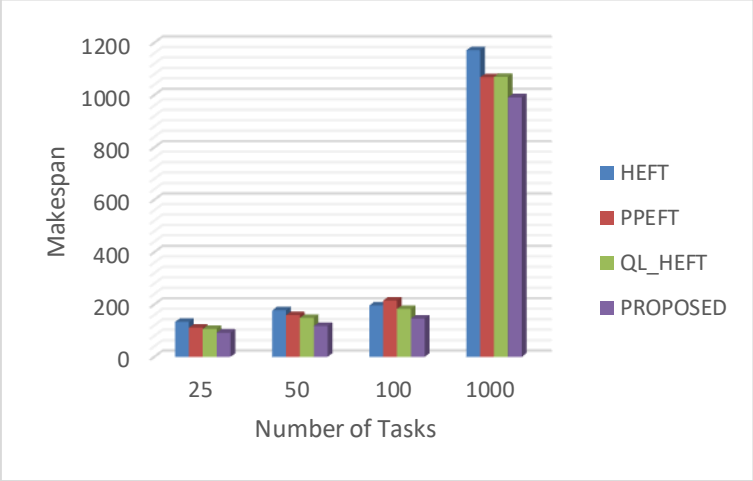
	Average response time			
Algorithm	**4 VMs**	**8 VMs**	**16 VMs**	**32 VMs**
HEFT	317.17	161.42	92.133	58.419
PPEFT	327.44	139.32	81.314	48.912
QL_HEFT	295.84	130.73	83.227	50.002
PROPOSED	265.01	133.92	79.283	46.503



Fig(7)Response time for Montage workflow application

The four algorithm performance is compared according to the definedmetric. The number of virtual machines is set to 4, 8, 16, 32 and the number of tasks is set to 100. The results Makespan of different algorithms on Montage workflow are depicted in Figure (4),It can be seen that the proposed algorithm performs better than the other comparative algorithms Proposed in sorting achieves a better order of task execution and is found to be the most balanced algorithm.The results of the speedup of all the algorithms using the Montage workflow application is depicted in Figure (5). It has been observed that the greater the number of virtual machines, the faster the algorithms become. In contrast to all other algorithms, the

proposed Algorithm achieves the highest speedup for the same number of virtual machines,Figures 6 show that average efficiency of Proposed algorithm is far better
than of HEFT,PPEFT and QL_HEFT algorithm. With increasing the number of processors, roposed algorithm gives more improved results. Using Montage and 100 tasks, we compared the outputs of the four algorithms with the ART. Table 4 shows the average response times of all algorithms for various virtual machine counts. Table 4 displays the results. As a result, the efficiency of the Proposed Algorithm is the highest in general. This is because, during the sorting process, the Proposed algorithm obtains a better task execution order. As a result, the time interval between two adjacent tasks running on the same virtual machine is shorter, resulting in a shorter ART with the shortest makespan. The Proposed algorithm, on the other hand, is the most balanced algorithm in terms of makespan and ART. It is capable of completing tasks quickly and provides a superior user experience.



Fig(8)Montage simulations of four algorithms with varying numbers of Tasks.

Figure 8 shows the average makespan of Montage workflow. In this experiment, four different sizes of Montage workflow are used. Montage_25 is the smallest size containing 25 tasks, Montage_50 is the medium size containing 50 tasks, Montage_100 is the large size containing 100 tasks, and Montage_1000 is the extra large size containing 1000 tasks. Experimental results show that Proposed algorithm works better on Montage workflowas compared to HEFT,PPEFT and QL_HEFT algorithms.

- Experiments with CyberShake benchmark

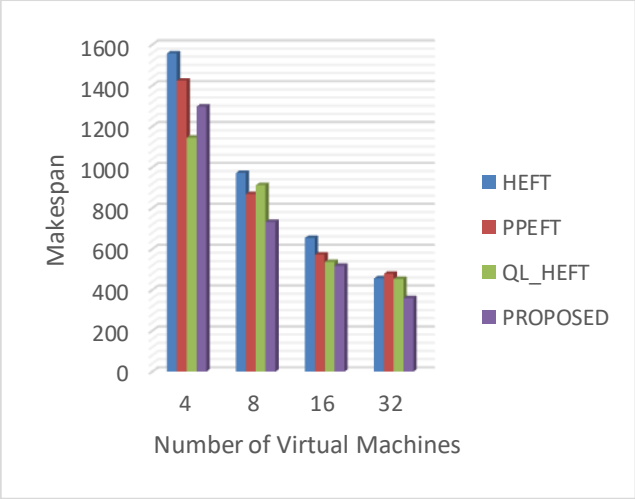
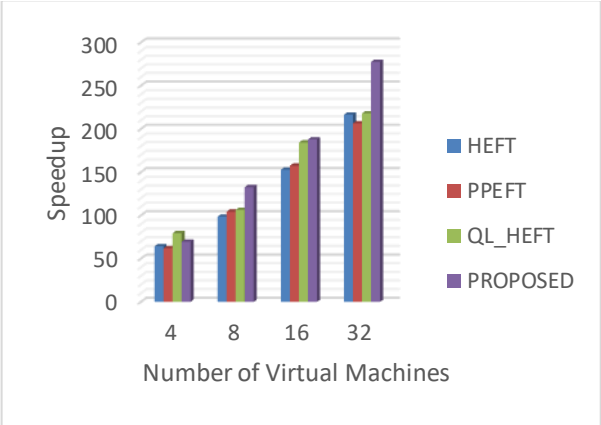
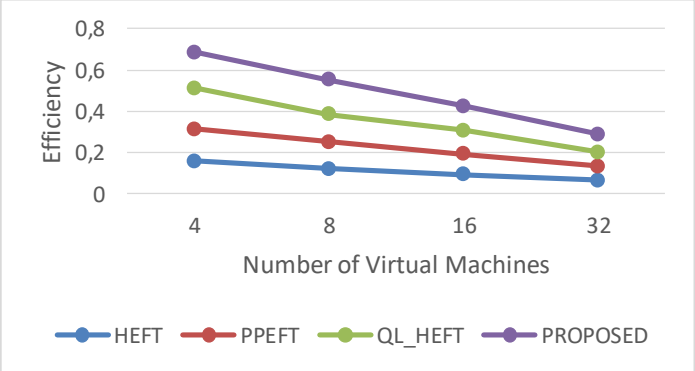


Fig (9)Makespan comparison of proposed and existing algorithms for CyberShake workflow application



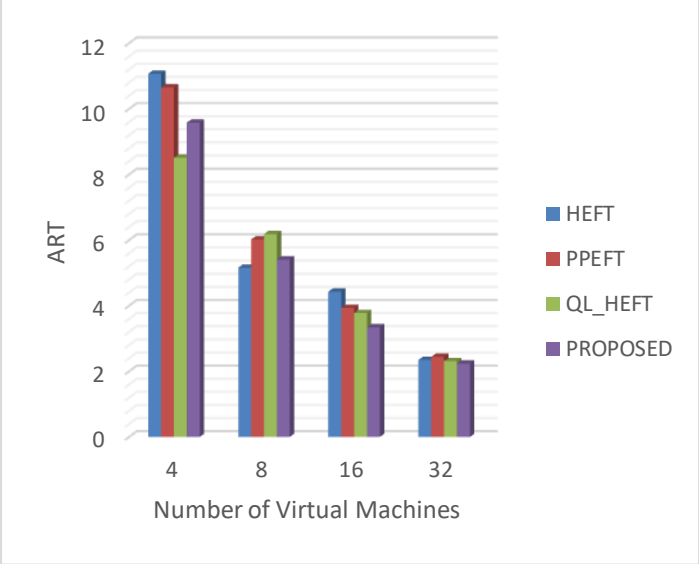
Fig(10)Speedup comparison of proposed and existing algorithms for CyberShake workflow application



Fig(11)Efficiency comparison of proposed and existing algorithms for CyberShake workflow application

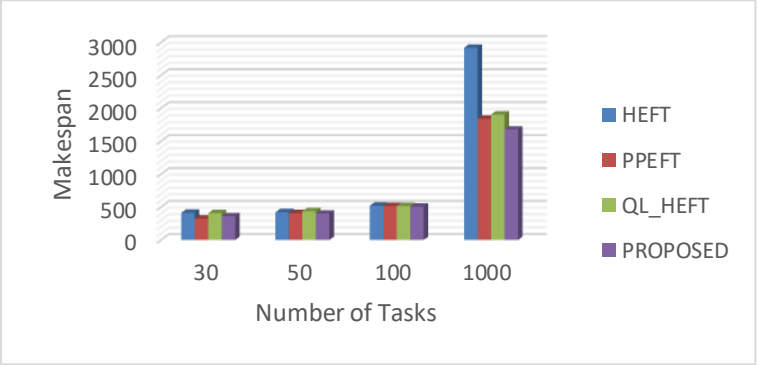
Table(5)Response Time for Montage Scientific Workflow

Average response time				
Algorithm	**4 VMs**	**8 VMs**	**16 VMs**	**32 VMs**
HEFT	1107.2	516.14	443.15	234.92
PPEFT	1065.6	602.1	393.57	244.4
QL_HEFT	851.12	618.36	377.69	231.36
PROPOSED	958.1	541.06	334.36	223.92



Fig(12)Response time for CyberShake workflow application

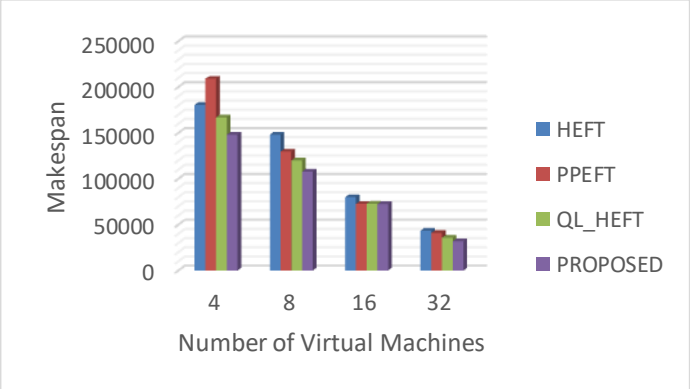
The results are depicted in Figure 9 and in Table 5 , The makespan of proposed Algorithm is least in comparison with other algorithms. The makespan is calculated with The number of virtual machines is set to 4, 8, 16, 32 and the number of tasks is set to 100. From the results, it can be analyzed that execution is faster in the proposed Algorithm for cybershake workflow application. The speedup comparison of the proposed algorithmwith other algorithms using the cybershake workflow application is depicted in Figure (10) and efficiency comparison is depicted in Figure (11),we used Proposed Algorithm to achieve better performance than other algorithms.



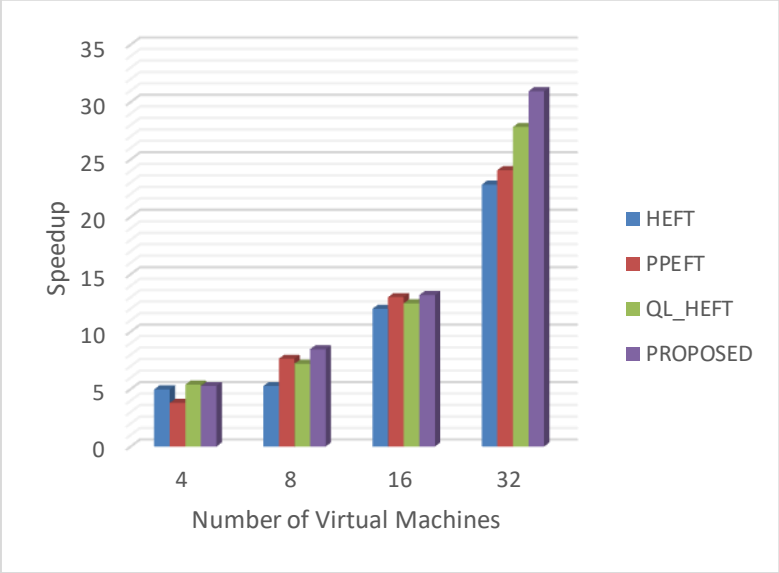
Figure(13)CyberShake simulations of four algorithms with varying numbers of Tasks.

Figure 13 shows the CyberShake workflow used for experimentation. Four different sizes are used (small, medium, large and extra large) containing 30, 50, 100 and 1000 tasks, respectively. Experimental results show that Proposed algorithm outperforms HEFT,PPEFT and QL_HEFT algorithm on CyberShake workflow.

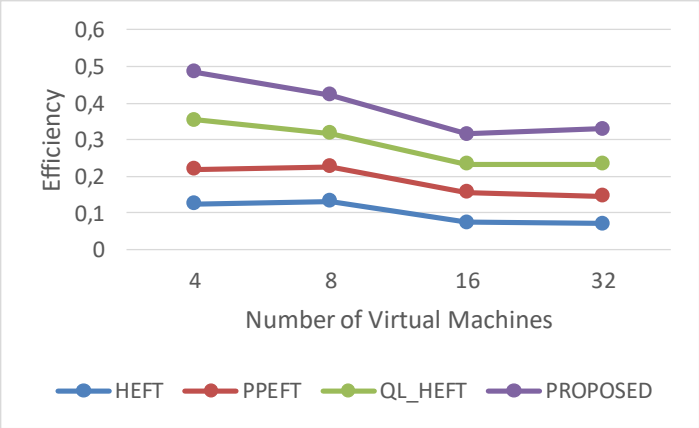
- Experiments with Epigenomics benchmark



Fig(14)Makespan comparison of proposed and existing algorithms for Epigenomics workflow application



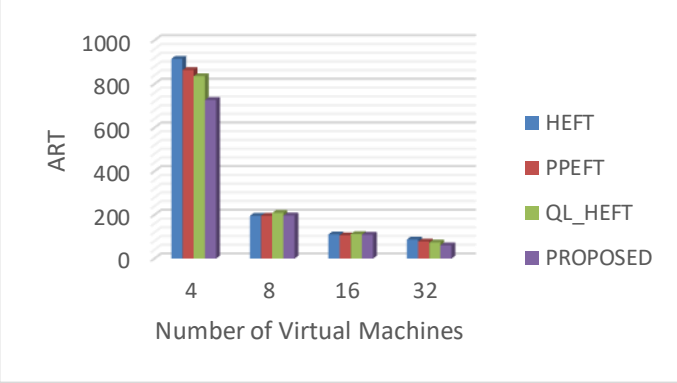
Fig(15)Speedup comparison of proposed and existingalgorithms for Epigenomics workflow application



Fig(16)Efficiency comparison of proposed and existing algorithms for Epigenomics workflow application

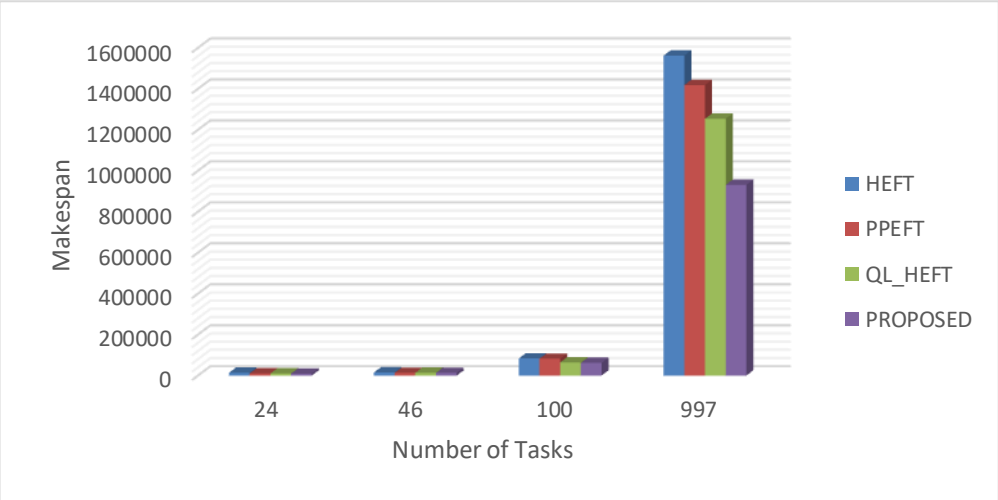
Table(6)Response Time for Epigenomics Scientific Workflow

Average response time				
Algorithm	4 VMs	8 VMs	16 VMs	32 VMs
HEFT	91556	19584	10968	8643.3
PPEFT	86353	19518	10547	7708.8
QL_HEFT	83523	20915	11183	7277
PROPOSED	72661	19650	11794	6008



Fig(17)Response time for Epigenomics workflow application

The results of an epigenomics workflow program with the same number of virtual machines as previous experiments are shown in Table 6 and Figure 14. Figures 15 and 16 display the efficiency and speedup results for the epigenomics workflow.



Fig(18)Epigenomics simulations of four algorithms with varying numbers of Tasks.

Figure 18 shows the Epigenomics workflow used for experimentation. Four different sizes are used (small, medium, large and extra large) containing 24, 46, 100 and 997 tasks, respectively. Experimental results show that Proposed algorithm outperforms HEFT , PPEFT ana QL_HEFT algorithm on Epigenomics workflow.

6 CONCLUSION

Existing scheduling algorithms focused on the time. The main goal of these schedulers is to reduce the overall Makespan of the workflow. Gaps in current workflow scheduling strategies in the cloud environments were studied in this thesis, and an effective scheduling method for workflow management in the cloud setting was proposed based on the gap analysis. It has been determined that the current scheme is effective enough to make the best use of the available resources.There are two stages to the algorithm design theory. The Q-Learning algorithm first sorts the original task order using the converged Q-table to find the best order. Second, based on the best order, we use the MET allocation strategy to assign the best processor to a task.The proposed approach has been implemented in in the simulation environment by using the WorkflowSim simulator. The experimental results have shown that proposed scheduling approach minimizes the overall execution time of workflow.in the future we plan to extend our the algorithm .We'll look at multi-objective optimization and dynamic task scheduling. to solve the workflow scheduling problem while taking into account the monetary costs of both communication and storage .

REFERENCES

[1] M. Y. Thanoun, H. Luqman, and A. L. Sawaf, "Design and Implementation of Task Scheduling Model for Cloud Computing Using Simevents," vol. 13, no. 4, pp. 18–28, 2019, doi: 10.22587/ajbas.2019.13.4.4.

[2] N. Almezeini and A. Hafez, "Review on Scheduling in Cloud Computing," *Int. J. Comput. Sci. Netw. Secur.*, vol. 18, no. 2, pp. 108–111, 2018.

[3] A. Kaur, P. Singh, R. Singh Batth, and C. Peng Lim, "Deep-Q learning-based heterogeneous earliest finish time scheduling algorithm for scientific workflows in cloud," *Softw. - Pract. Exp.*, no. October 2019, pp. 1–21, 2020, doi: 10.1002/spe.2802.

[4] Z. Wei, Y. Zhang, X. Xu, L. Shi, and L. Feng, "A task scheduling algorithm based on Q-learning and shared value function for WSNs," *Comput. Networks*, vol. 126, pp. 141–149, 2017, doi: 10.1016/j.comnet.2017.06.005.

[5] M. F. Akbar, E. U. Munir, M. M. Rafique, Z. Malik, S. U. Khan, and L. T. Yang, "List-Based Task Scheduling for Cloud Computing," *Proc. - 2016 IEEE Int. Conf. Internet Things; IEEE Green Comput. Commun. IEEE Cyber, Phys. Soc. Comput. IEEE Smart Data, iThings-GreenCom-CPSCom-Smart Data 2016*, pp. 652–659, 2017, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.143.

[6] K. Dubey, M. Kumar, and S. C. Sharma, "Modified HEFT Algorithm for Task Scheduling in Cloud Environment," *Procedia Comput. Sci.*, vol. 125, pp. 725–732, 2018, doi: 10.1016/j.procs.2017.12.093.

[7] A. I. Orhean, F. Pop, and I. Raicu, "New scheduling approach using reinforcement learning for heterogeneous distributed systems," *J. Parallel Distrib. Comput.*, vol. 117, pp. 292–302, 2018, doi: 10.1016/j.jpdc.2017.05.001.

[8] M. S. Arif, Z. Iqbal, R. Tariq, F. Aadil, and M. Awais, "Parental Prioritization-Based Task Scheduling in Heterogeneous Systems," *Arab. J. Sci. Eng.*, vol. 44, no. 4, pp. 3943–3952, 2019, doi: 10.1007/s13369-018-03698-2.

[9] C. Jiang, J. Wang, and X. Ye, "PVBTS: A novel task scheduling algorithm for heterogeneous computing platforms," *Int. J. Innov. Comput. Inf. Control*, vol. 16, no. 2, pp. 701–713, 2020, doi: 10.24507/ijicic.16.02.701.

[10] N. Rajak and D. Shukla, *An Efficient Task Scheduling Strategy for DAG in Cloud Computing Environment*, vol. 1097. Springer Singapore, 2020.

[11] D. Cui, W. Ke, Z. Peng, and J. Zuo, "Multiple DAGs workflow scheduling algorithm based on reinforcement learning in cloud computing," *Commun. Comput. Inf. Sci.*, vol. 575, pp. 305–311, 2016, doi: 10.1007/978-981-10-0356-1_31.

[12] W. Chen, G. Xie, R. Li, Y. Bai, C. Fan, and K. Li, "Efficient task scheduling for budget constrained parallel applications on heterogeneous cloud computing systems," *Futur. Gener. Comput. Syst.*, vol. 74, pp. 1–11, 2017, doi: 10.1016/j.future.2017.03.008.

[13] Z. Tong, X. Deng, H. Chen, J. Mei, and H. Liu, "QL-HEFT: a novel machine learning scheduling scheme base on cloud computing environment," *Neural Comput. Appl.*, vol. 32, no. 10, pp. 5553–5570, 2020, doi: 10.1007/s00521-019-04118-8.

[14] "5 key reinforcement learning principles." https://hub.packtpub.com/5-key-reinforcement-learning-principles-explained-by-ai-expert/ (accessed Apr. 04, 2021).

[15] Z. Peng, D. Cui, J. Zuo, Q. Li, B. Xu, and W. Lin, "Random task scheduling scheme based on reinforcement learning in cloud computing," *Cluster Comput.*, vol. 18, no. 4, pp. 1595–1607, 2015, doi: 10.1007/s10586-015-0484-2.

[16] D. Cui, Z. Peng, xiong jianbin, bo xu, and W. Lin, "A Reinforcement Learning-based Mixed Job

Scheduler Scheme for Grid or IaaS Cloud," *IEEE Trans. Cloud Comput.*, vol. 7161, no. c, pp. 1–1, 2017, doi: 10.1109/tcc.2017.2773078.

[17] M. Drugan, "Reinforcement learning versus evolutionary computation : a survey on hybrid algorithms."

[18] Z. Tong, H. Chen, X. Deng, K. Li, and K. Li, "A scheduling scheme in the cloud computing environment using deep Q-learning," *Inf. Sci. (Ny).*, vol. 512, no. xxxx, pp. 1170–1191, 2020, doi: 10.1016/j.ins.2019.10.035.

[19] Montage: An astronomical image engine. http://montage.ipac. caltech.edu. Accessed 10 Aug 2018

[20] USC Epigenome Center. http://epigenome.usc.edu. Accessed 10 Aug 2018