

Received June 30, 2021, accepted July 27, 2021, date of publication July 30, 2021, date of current version August 13, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3101454

# Optimizing Workflow Task Clustering Using Reinforcement Learning

CHIN POH LEONG<sup>1</sup>, CHEE SUN LIEW<sup>1</sup>, CHEE SENG CHAN<sup>2</sup>, (Senior Member, IEEE), AND MUHAMMAD HABIB UR REHMAN<sup>3</sup>, (Senior Member, IEEE)

<sup>1</sup>Department of Computer System and Technology, Faculty of Computer Science and Information Technology, Universiti Malaya, Kuala Lumpur 50603, Malaysia

<sup>2</sup>Department of Artificial Intelligence, Faculty of Computer Science and Information Technology, Universiti Malaya, Kuala Lumpur 50603, Malaysia

<sup>3</sup>Center for Cyber-Physical Systems, Khalifa University of Science and Technology, Abu Dhabi, United Arab Emirates

Corresponding authors: Chin Poh Leong (rayson1223@siswa.um.edu.my) and Chee Sun Liew (csliew@um.edu.my)

This work was supported in part by the Ministry of Higher Education Malaysia under Grant LRGS-LR002-2020.

**ABSTRACT** Scientific workflows are composed of many fine-grained computational tasks. Generally, a large number of small tasks will slow down the workflow performance due to the scheduling overhead incurs during the execution time. Task clustering is an optimization technique that aggregates multiple small tasks into a large task to reduce the scheduling overhead, and thus it will reduce the overall workflow makespan, *i.e.* the total execution time taken by the resources to complete the execution of all of the tasks. However, finding an optimal clustering number is a big challenge as it usually requires manual intervention of experienced researchers to define the clustering parameter. In this paper, we proposed the use of reinforcement learning to tackle this problem by automating the discovery of the optimal clustering number for the submitted workflow. First, we model the workflow environment that allows the reinforcement learning agent to interact by determining the cluster number for every round of workflow execution. Then, based on the provenance records after the execution, the workflow environment will analyze the performance data and then determine either a reward or a punishment as the feedback to the reinforcement learning agent. The evaluation experiments are performed using real-world scientific workflow (*e.g.* Montage in this research), to demonstrate our model capability to identify the optimal cluster number and thus lay the groundwork for the adoption of reinforcement learning in workflow task clustering.

**INDEX TERMS** Clustering method, machine learning, workflow management software.

## I. INTRODUCTION

In general, large-scale scientific experiments produce a massive amount of complex and heterogeneous data streams [1]. Although the convergence of modern computing systems (such as clusters and clouds) with big data technologies enables data-driven scientific applications, there is still a need for software systems to enable dynamic coordination and resource optimization. For example, Workflow management systems (WMS) enables a mapping of various components of the scientific applications and effectively utilize their underlying computational resources. The basic element of WMS is a workflow which is commonly represented by direct acyclic graph (DAG) where the computational tasks are mapped on the nodes while both the data and control dependencies are represented on the edges. A few common WMS include

Pegasus [2], Wings [3], KNIME [4], Airavata [5], Hermes [6], Cloudbus [7] and Galaxy [8]. These WMS systems primarily handle multiple workflow submissions by researchers and perform scheduling of tasks across various computational resources, handle node/system failure, enable recovery, execution outcome, and monitor the workflow execution lifecycles. Thus, an efficient identification and utilization of the available computational resource are critical to achieve the optimal performance.

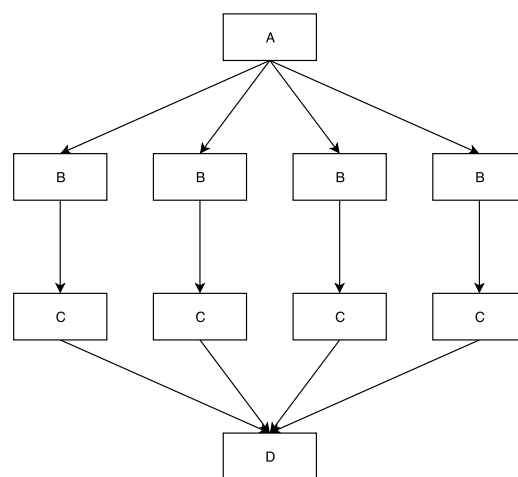
Scientific workflow usually comprises many granular tasks where the lifespan of these tasks is varied accordingly. The worst-case scenarios of these tasks arise when the scheduling overhead becomes longer than the execution time of the tasks. The situation may become even worst when the scheduling is performed in the large-scale computing systems where both the scheduling and coordination time introduce latency when compared with the execution time. Numerous research studies [9]–[11] proposed a task clustering approach to cluster

The associate editor coordinating the review of this manuscript and approving it for publication was Mostafa M. Fouda<sup>1</sup>.

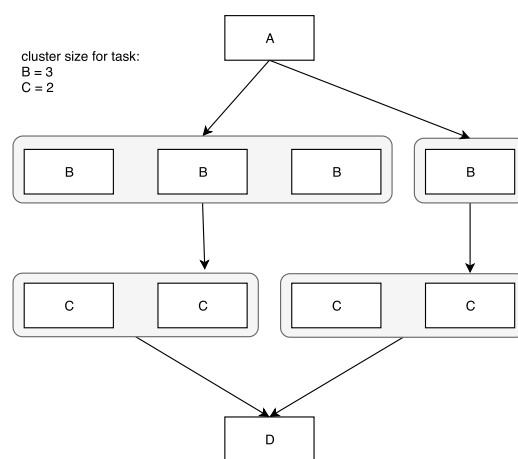
the fine-grained tasks into the coarse-grained task in order to reduce the number of tasks, *i.e.* reduces the overhead and optimizes the performance of the workflow execution. Generally, workflows are executed in four different phases which are (i) composition, (ii) mapping, (iii) execution, and finally (iv) provenance [12]. The configurations of each phase vary accordingly to the problem domain and the optimization objectives, while task clustering optimization is commonly applied in both the mapping and execution phase in the WMS environment.

Task clustering consists of two variants, namely horizontal and vertical clustering. Horizontal clustering merges multiple tasks within the same horizontal level of the workflow. In contrast, vertical clustering merges tasks at the same pipeline vertically. Generally, the granularity of the task is based on the *cluster size* and *cluster number*. *Cluster size* controls the number of tasks in a group and *cluster number* controls the total number of the cluster for the specific task. Fig. 1(a) shows an original workflow that undergoes before the clustering process. Fig. 1(b) demonstrates the effect of horizontal clustering with *cluster size* for task B = 3 and *cluster size* for C = 2, respectively. However, it is visualized that there are only four tasks B in the same layer. Even it is an insufficient task for the second cluster of task B to fulfill the size of 3, it still groups task B into two clusters. In contrast, if the configuration is *cluster number*, task B in the same horizontal layer will be enforced by the WMS engine to group accordingly as requested, as shown in Fig. 1(c). From here, it can be seen that determining an optimal *cluster number* to deliver the optimal performance of the workflow execution remains a hurdle in WMS as it is commonly specified by the scientists based on their understanding of the workflow structures. As a result of this, it is impractical that having a scientist that familiar with different domains of workflow and optimize it all day long manually. Thus, the main motivation of this paper is to automate the optimal cluster number process and improve the workflow performance over time without human intervention.

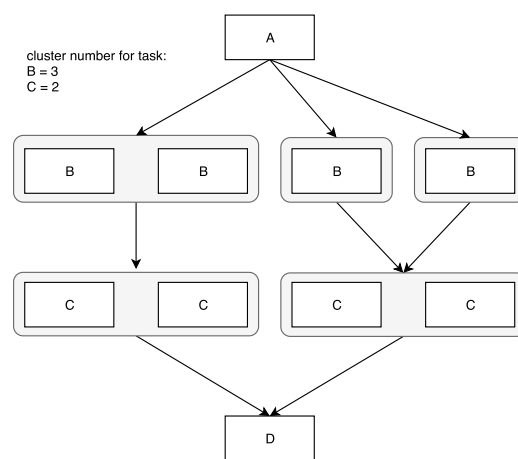
In the previous study [13], we had explored a Hybrid Genetic Algorithm (HGA) to optimize the scheduling plan in the scheduler. Yet, due to the nature of scientific workflow where the workflow is repeatedly executed over time with slight variations on the parameters, inputs, or workflow structures, the evolutionary algorithm shows limitations on adapting the change and the lack of the recognition ability. On the contrary, machine learning methods have the potential to discover underlying patterns with minimal intervention. Thus, in this study, we hypothesize that a WMS with the ability to collect data and leverage machine learning to continuously analyze and improve its execution performance is an interesting endeavor. Instead of focusing on the scheduler to optimize the scheduling plan over the assigned fine-grained tasks in the execution phase of the workflow lifecycle, we focus on optimizing the workflow tasks clustering in the mapping phase whereby the workflow tasks are not assigned to any physical resource for execution. By clustering the fine-grained



(a) Original Workflow



(b) Horizontal clustering with cluster size



(c) Horizontal clustering with cluster number

FIGURE 1. Horizontal Clustering: Cluster Size and Cluster Number.

tasks during the mapping phase, lower down the number of tasks for the scheduler in the execution phase and improve the makespan of the workflow execution. Specifically, our objective is to closely monitor the performance of workflow

execution and determine the optimal number of clusters to gain adequate performance.

Nevertheless, determining which machine learning approach toward task clustering optimization is pivotal. With the absence of high-quality labeled data that are prohibitively expensive and almost impossible to generate due to the computational complexity of the scientific workflows (Scheduling optimization is an NP-hard problem). It limits the approaches of machine learning on identifying cluster numbers for the workflow. Another consideration is that the machine learning method learning from examples for NP-hard problems is a suboptimal approach. As the model's performance largely depends on the quality of the labeled data and it would not be of interest to train a model that mimics the results of a heuristic solver algorithm. Previous study [14] has empirically demonstrated on the Traveling Salesman and Knapsack combinatorial optimization problems that even optimal solutions are used as labeled data in a supervised learning approach, the generalization can be rather poor compared to a reinforcement learning (RL) agent that can discover different solutions and observe based on the corresponding rewards. Given the inability to obtain optimal labels data nor open repositories, supervised learning does not apply to the scheduling problem as well as to most combinatorial optimization problems. Utilizing a reinforcement learning approach, it is possible to compare the quality of a set of solutions and provide reward feedback to course a learning algorithm.

In this paper, we examined the existing machine learning methods and determine the suitable approach, *i.e.* reinforcement learning, toward identifying optimal cluster numbers for the task clustering optimization. Two experiments are conducted to examine the proposed approach with different environments and WMS such as Pegasus and WorkflowSim. The result of the experiments shows that the RL agent can improvise the task clustering decision throughout the workflow execution over several executions and obtain optimal performance. The proposed model is extendable to provide a baseline of the reinforcement learning adoption in scientific workflow task clustering optimization. The rest of this paper is organized as follows. Section II gives an overview of the related work. Our proposed approach and execution environments model is presented in Section III. Section IV reports the experiments results in discussion. We conclude and discuss the further research in Section V.

## II. RELATED WORK

Workflow scheduling has been a challenging issue in workflow execution. The high quantities of the fine-grained tasks is a common problem for the performance in distributing platforms where the scheduling overhead and queuing times at resources are high. Task clustering is a technique that combined fine-grained tasks in the workflow into coarse-grained tasks to reduce the scheduling overhead by simplifying the workflow structure and improve the workflow execution performance. For instance, Singh *et al.* [15] proposed level and

label-based clustering whereby an user is able to specify either the number of clusters or tasks per cluster within the same level of workflow or labels manually by the user to group the tasks respectively. Even though their work considered data dependencies between workflow levels, it is still not flexible enough for everyone as not every user has the knowledge over workflow optimization. Nonetheless, this method is easy prone to errors and unscalable. Chen *et al.* [9] proposed an algorithm that clustering the task based on the balance between the task run-time and dependency in order to improve the overall makespan of the workflow execution. Based on Chen *et al.* balance task clustering approach, Dong *et al.* [10] further enhance the algorithm toward cost-effective awareness by clustering the tasks vertically and horizontally with a greedy allocation of the resources in the cloud environment without missing the deadline constraint (part of the QoS in a cloud environment) [16]. Avinash *et al.* [17] assessed the task-dependency of the workflow by calculating the impact factors on the workflow structure tasks that focus on single parent single child relationships and available resources to make the clustering decision. However, applying the task clustering technique in different phases of the workflow lifecycle yields different behavior and outcome due to the information disparity and availability different across phases. Wang and Sinnen [11] explained that task clustering scheduling involves three simple procedures: clustering, cluster merging, and task order. The clustering process is a simple step where it groups similar or labeled task together. Clustering merging is combining the task groups if the number of clusters is more than the number of processors available. At last, map each cluster to the processor, and task ordering happen at the machine execution layer. As compared to the mapping phase, only the clustering merging process is applied. Task clustering in the mapping phase is focused on high-level abstraction of workflow processes to perform clustering optimization. In contrast with the execution & mapping phase, the task clustering algorithm is focused on identifying workflow structure and clustering tasks. Our proposed approach is to apply reinforcement learning on identifying the optimal cluster number for the workflow in the mapping phase. We believe that providing a feedback from the workflow execution performance data (which only available in the execution and monitoring phase) to the mapping phase, will help the WMS to make a better decision on identifying the cluster number towards the submitted workflow and thus improve over the number of executions.

Recently, researchers have shown an increased interest in adopting reinforcement learning into the scientific workflow optimization domain. For instance, Tong *et al.* [18] applied the Q-learning method [19] to assist the HEFT scheduling algorithm to improvise the task orders in an optimal order to improve the scheduler performance. Barret *et al.* [20] applied reinforcement learning with the Q-learning method toward user request demands to determine the resource allocation decision. Wu *et al.* [21] apply REINFORCE, policy

gradient approach to obtain an optimal scheduling plan to reduce the makespan of the execution. Orhean *et al.* [22] proposed a framework that used a reinforcement learning algorithm to learn and determine the optimum scheduling algorithm for workflows within a heterogeneous and distributed resources environment to reduce overall execution time. Kintsakis *et al.* [23] extended Hermes [6] WMS capabilities that leverage machine learning to predict workflow task runtime and the probability of failure on task assignment to execution sites. As the task failure decrease over the executions, thus workflow runtime has been reduced. The policy network model for task prediction is trained offline by using the reinforcement learning approach. We argue that heuristic scheduling approaches have the difficulty to handle the versatile of workflow design and diversities of task execution in modern scientific discovery experiments. We adopt the reinforcement learning method that will grant the WMS to identify the optimal cluster number of tasks during the execution on computational resources, with different workflow computational requirements, to obtain the shortest workflow makespan.

### III. MODEL AND DESIGN

Reinforcement learning has been applied successfully across a range of domains supporting automated control and allocation of resources [20]. The main notations adopted in this study are listed in Table 1.

**TABLE 1. Main notations used in this study.**

Symbol	Description
$R_t$	The current cycle reward by the reward function
$A$	The action space of the workflow environment
$A_t$	The current cycle action taken by the RL agent
$S$	The state space of the workflow environment
$S_t$	The current cycle state of the workflow environment
$\alpha$	The learning rate of the Q-learning algorithm
$\gamma$	The discount factor of the Q-learning algorithm
$\epsilon$	The epsilon-greedy policy probabilities
$t_{\text{current makespan}}$	The current cycle of workflow execution makespan
$t_{10_{th}, \text{percentile}}$	The current cycle of 10 <sub>th</sub> percentile of the makespan
$job_{\text{makespan}}$	The makespan of execution task
$t_{\text{pre-scheduling}}$	The total time taken to assign and deliver the execution task
$t_{\text{execution}}$	The total time taken to complete the execution task
$t_{\text{post-scheduling}}$	The total time taken to return the execution results back to scheduler
$R_{\text{makespan}}$	The records of each cycle workflow execution makespan
$R_{10_{th}, \text{percentile}}$	The 10 <sub>th</sub> percentile records of each workflow execution makespan
$R_{\text{percentile}}$	The mean of the $R_{10_{th}, \text{percentile}}$ records

#### A. TASK CLUSTERING OPTIMIZATION MDP FORMULATION

Reinforcement Learning problems can be modeled using MDP. Reinforcement learning methods facilitate solutions to

MDPs in the absence of a complete environment model. This is particularly useful when dealing with real-world problems as the model can often be unknown or difficult to approximate. To implement task clustering workflow optimization, firstly required to model the optimization problem into MDP. MDPs are a particular mathematical framework suited to modeling decision-making under uncertainty.

It constructs by four elements,  $M = \langle S, A, P, R \rangle$ , where  $S$ , represents the sets of all possible states of that RL agent's observable world.  $A$  represents the total action space,  $P$  defines a probability distribution governing state transitions.  $R$  defines the probability distribution governing the rewards received on the actions perform,  $a_t$  during  $s_t$ .

In the context of workflow task clustering optimization, the ideal MDP description will be using workflow makespan as the state space,  $S_{\text{makespan}}$ , clustering parameter as the action space,  $A_{\text{cluster number}}$ , probability of state shift will be based on the clustering parameter chosen that affect the workflow makespan,  $P_{\text{cluster number}}$ , and along with the reward mechanism for indicating the clustering parameter yield better performance,  $R_{\text{makespan}}$  as shown in 1(a).

$$M = \{S_{\text{makespan}}, A_{\text{cluster number}}, P_{\text{makespan}}, R_{\text{makespan}}\} \quad (1a)$$

$$M = \{S_{\text{cluster number}}, A_{\text{cluster number}}, P_{\text{cluster number}}, R_{\text{makespan}}\} \quad (1b)$$

There are several challenges to adopt the MDP description in 1(a). Firstly, if workflow makespan,  $t_{\text{makespan}}$  is used as the state of the MDP description, then the state space of the MDP will be from the range of 0 to infinity,  $0 < t_{\text{makespan}} < \infty$ , as the workflow makespan is time-series data. With the endless possibility of state space, the probability of state transition based on the action goes infinite as well, which the agent deemed to be challenging in identifying the optimal clustering parameter which yields the best reward. Apart from infinite state and possibility, the workflow makespan also varies from each execution with the same configuration given as the resources and scheduling time is inconsistent. Based on these assessments, using workflow makespan as the state of the MDP description in 1(a) is invalid.

The state of the MDP is required to be repeatable, predictable, and controllable. Thus, the cluster number is chosen for the proposed method. In contrast with workflow makespan, cluster number ranges from 0 to the number of total CPU in the resource pool,  $n_{\text{cpu}}$ . As the number of cluster numbers more than the total number of CPU in the resource pool, the performance of the workflow will be worse. The probability of the state transition on each state will be model as  $\frac{1}{n_{\text{cpu}}}$ . It is a more predictable and repeatable state identified, the MDP description based on the cluster number is used for the proposed method as shown in Equation 1(b).

#### B. REINFORCEMENT LEARNING WORKFLOW ENVIRONMENT PROPERTIES

The concept of developing a *workflow environment* for reinforcement learning (RL) is inspired by Arcade Learning

Environment (ALE) that was proposed by Bellmare et. al [24] as a software framework for interfacing with emulated 500 games that originally developed for the Atari 2600. Before designing the environment, it is crucial to first understand that the environment's predictability and complexity. *Workflow environment* is a model based on the nature of WMS execution over workflow submitted on fixed resource pools that are allocated. Typically, workflow submitted to WMS for execution will schedule by the scheduler and each workflow execution will obtain a portion of necessary resources from the pool to execute. When there are multiple workflows able to schedule and execute concurrently, the availability of the resources becomes unpredictable which leads to the unpredictable workflow makespan. The objective of the *workflow environment* is to provide a learning environment for the RL agent to learn about the optimal clustering which yields the best makespan for particular workflow execution. Russell and Norvig [25] suggested seven properties of the environment to determine the difficulty, which are determinism, static, observable, agency, knowledge, episodic, and discreteness. Based on the observation of WMS executing workflow, the properties of environment can be inference as stochastic (*i.e.* workflow makespan rely on the resource availability), dynamic (*i.e.* computation resources availability), fully observable (*i.e.* provenance data), multi-agent (*i.e.* number of users of WMS), unknown (*i.e.* the outcome of the clustering parameter toward makespan), episodic (*i.e.* clustering parameter is independent to workflow makespan on every execution), and continuous (*i.e.* WMS is constantly executing submitted workflow without turning off) by nature.

However, the inconsistency of resource availability and makespan of the workflow enactment is dynamic and unpredictable, which is anticipated as a challenge for the RL agent that leads to the high error rates on learning the outcome of the experiment. Hence, certain properties of the *workflow environment* required modification for the goal of the experiment design. First, the observable of the WMS environment from full to partial observable, as the RL agent does not require to obtain all performance provenance data for decision making but solely the workflow makespan. The workflow makespan is a time-series data, whereby it is a continuous variable and difficult to form an internal representation model within the RL agent under two conditions, the workflow makespan can be identical regardless of different clustering parameter is provided, and the makespan of the workflow can be different with an interval of time, for instance, 100 and 105 seconds respectively, by using an identical clustering parameter provided in different rounds of execution. Therefore, instead of feeding back RL agents with the workflow makespan, clustering parameter as the state of the environment is more preferable. Next, the static of the WMS environment is altered from dynamic to static by controlling the number of workflow enactment to ensure the availability of resources available and the consistency of workflow makespan. Lastly, the discreteness of the environment is the change from continuous to discrete for the workflow environment design to control the

range of clustering parameters and the number of workflow enactments per episode.

In practice, workflow submission and enactments are ongoing tasks, and the optimal clustering parameter for each workflow is unique without constraint. Yet, enforcing a constraint on both workflow submission and clustering parameters is necessary for the validation of this research as a control variable in the experiment. By setting a limited number of workflow enactments per episode, the workflow makespan is more consistent and predictable for the RL agent to figure out the optimal clustering parameter for a workflow. Therefore, the properties and behavior of the workflow environment is model as stochastic (previous workflow execution is independent to the future execution), static (computation resource availability), partial observable (clustering parameter), single-agent (number of users using the WMS), unknown (unpredictable makespan), episodic (workflow makespan is independent on previous execution), discrete (number of workflow execution per cycle).

### C. PROPOSED SOLUTION

At any discrete time step  $t = 0, 1, \dots, T - 1$  where  $T$  is the final step, the RL agent receives a representation of the environment's state,  $s_t \in S$ , where  $S$  is the set of all possible states, and on that basis the RL agent selects an action,  $a_t \in A$ , where  $A$  is the set of all available actions. One time step later, at  $t + 1$ , as the results of the agent's action, the RL agent receives a reward  $r_{t+1} \in R$  and an update of on the new state,  $s_{t+1}$  from the environment. Each cycle of the interaction is called a transition experience,  $e_{t+1} = (s_t, a_t, r_{t+1}, s_{t+1})$ . Fig. 2 shows a general interaction between an RL agent and the environment.

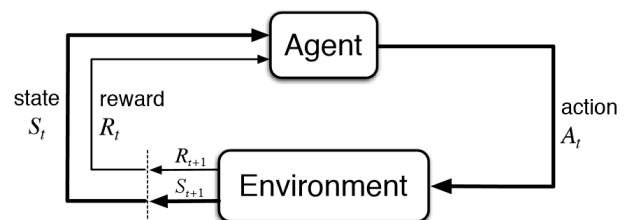


FIGURE 2. The RL agent and environment interaction in reinforcement learning.

At each time step, the RL agent implements a mapping from states to all possible actions,  $\pi : S \rightarrow A$ , called policy. The objective of the RL problem is to obtain the highest expected value of return,  $G_t = \sum_{t'=t}^T \gamma^{t'-t} r_{t'+1}$  for all  $t = 0, 1, \dots, T - 1$ , where  $T$  is the final step and  $\gamma \in [0, 1]$  is a discount factor. The *optimal policy*, is the policy that maximizes the expected return, i.e.  $\pi_* = \operatorname{argmax}_{\pi} E_{\pi}[G_t]$ , where  $E_{\pi}[\cdot]$  denotes the expected value of random variable given that the RL agents follows policy  $\pi$ . In order to solve this optimization problem, it is often useful to define a *value function*  $q : S \times A \rightarrow R$ . In order to seek the optimal policy, the RL agent can search for the optimal state-action value

function,  $q_* = \max_{\pi} q_{\pi}(s, a)$  for all  $s$  and  $a$ . In this paper, the RL agent has adopted the Q-learning policy to identify the optimal cluster number for the submitted workflow.

In our implementation, as shown in Fig. 3, the agent-environment interactions begin with the initialization of the workflow environment by generating an initial state,  $S_0$  by the *state function*, where  $0 < S_0 < \infty$  represent the cluster number that the environment was chosen and pass to the RL agent to determine the suitable action to act upon. Based on the observation state,  $S_0$ , the RL agent employs Q-learning as the optimal policy to determine the action,  $A_0$ , the optimal clustering number for the workflow execution. When the *workflow environment* received the action,  $A_0$  from the RL agent, the embed WMS will responsible to execute the experiment. The interaction between the RL agent and the environment will be on hold until the workflow is completely executed. The environment is always ensuring resource availability and only one workflow is executing at a time. After the workflow is completed, the interaction is resumed with reward evaluation by querying the provenance records of the workflow whereby extended from existing WMS features. The pseudo-code for the RL agent that interacts with the workflow environment is given in Algorithm 1.

$$f(\text{reward}) = R\left(\sum_n \text{job}_{\text{makespan}}\right); \quad (2a)$$

$$\text{job}_{\text{makespan}} = t_{\text{pre-scheduling}} + t_{\text{execution}} + t_{\text{post-scheduling}}; \quad (2b)$$

---

**Algorithm 1:** Reinforcement Learning Agent Policy: Q-Learning
 

---

**Input** :  $A_t$   
**Parameters:**  $\alpha = 0.5, \gamma = 0.7, \epsilon = 0.3$   
 Initialize  $Q(S, A)$ ;  
**foreach** *episode* **do**  
   Initialize  $S_0$ ;  
   **while** *terminate condition does not match* **do**  
     Choose  $A_t$  from  $S$  using policy derived from  $Q$   
     (e.g.  $\epsilon$ -greedy);  
     Take action  $A_t$  and execute experiment;  
     Observe  $R_t$  (return by Reward Function 2) and  
      $S_{t+1}$ ;  
      $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_t +$   
      $\gamma \max Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$ ;  
      $S \leftarrow S_{t+1}$ ;  
   **end**  
**end**

---

The rewarding function,  $f(\text{reward})$  evaluation shown in 2a is based on the makespan of the workflow execution.  $\text{job}_{\text{makespan}}$  represents the sum of all workflow jobs pre-scheduling time  $t_{\text{pre-scheduling}}$ , execution time  $t_{\text{execution}}$ , and post-scheduling time  $t_{\text{post-scheduling}}$ , as shown in 2b. Every cycle of the workflow execution will be stored within

an array of execution records,  $R_{\text{makespan}}$ , and the 10<sup>th</sup> percentile of the  $R_{\text{makespan}}$  is evaluated before stored in an array,  $R_{10^{\text{th}} \text{ percentile}}$ , which is used as the benchmark of RL agent rewards. If the current cycle of the makespan,  $t_{\text{current makespan}}$ , is performed better than the mean of the records of 10<sup>th</sup> percentile,  $\bar{R}_{\text{percentile}}$ , then a positive reward, +1 will be returned to the RL agent as a positive action, else -1 will be rewarded as a punishment of the chosen action. To prevent the data distribution bias of the  $\bar{R}_{\text{percentile}}$  to one end of the records, the 10<sup>th</sup> percentile of the  $R_{\text{makespan}}$  of each execution is evaluated by improvement of 10% before added into the list of  $R_{10^{\text{th}} \text{ percentile}}$ , as shown in the Reward Function 2.

---

**Reward Function 2:** Compare With the Mean of Percentile Records
 

---

Variable:

$R_{\text{makespan}} = \{\text{records of each cycle makespan}\}$

$R_{10^{\text{th}} \text{ percentile}} =$   
 $\{10^{\text{th}} \text{ percentile records of each cycle makespan}\}$

$\bar{R}_{\text{percentile}} = \text{mean of the } R_{10^{\text{th}} \text{ percentile}}$

$t_{\text{current makespan}} = \text{current cycle of makespan}$

$t_{10^{\text{th}} \text{ percentile}} =$   
 the 10<sup>th</sup> percentile of the makespan from the records

**Function** Reward ( $t_{\text{makespan}}$ ):

```

if  $\frac{|t_{10^{\text{th}} \text{ percentile}} - \bar{R}_{\text{percentile}}|}{t_{10^{\text{th}} \text{ percentile}}} * 100 > 10$  then
  | add  $t_{10^{\text{th}} \text{ percentile}}$  to  $R_{10^{\text{th}} \text{ percentile}}$ ;
if  $t_{\text{current makespan}} \leq \bar{R}_{\text{percentile}}$  then
  |  $R_t = 1$ ;
else
  |  $R_t = -1$ ;
return  $R_t$ ;
  
```

---

At the same time, the environment will generate the next state,  $S_1$  along with the reward,  $R_0$  to be returned to RL agent, and counted as a complete single cycle of the episode during the experiment. The reward function represents both the *unknown* and *stochastic* of the workflow environment characteristic, whereby the makespan of the workflow is unpredictable for each execution and every execution is independent of the previous execution respectively. The experience and interaction of the RL agent and workflow environment are represented as the *episodic* and *discrete* of the environment. The RL agent will continuously interact with the environment by acting for a maximum of 100 cycles as an episode until the terminal state or objective of the environment is achieved.

## IV. EXPERIMENTS AND DISCUSSION

### A. EXPERIMENT SETUP

Two experiments were conducted to validate our proposed model. The first experiment was carried out using Pegasus [2], deployed in UM Data-Intensive Computing Centre. The master node server is equipped with a 2.40 GHz Intel

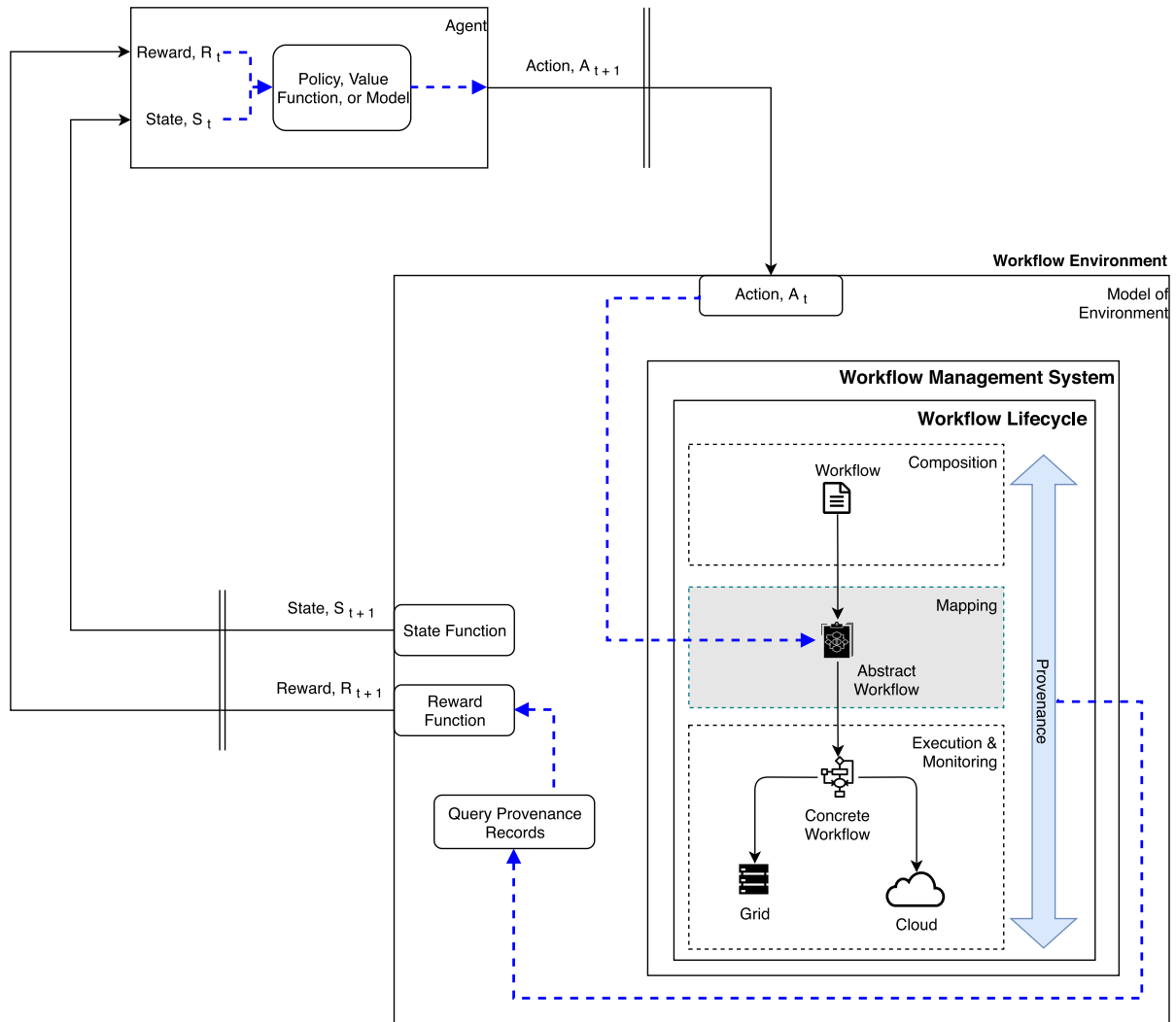


FIGURE 3. The proposed workflow reinforcement learning environment.

E5645 processor, 8 GB memory, and 100 GB of storage, together with two slave nodes with the same Intel E5645 processor, each with 12 CPUs running at 2.40 Ghz, 16 GB memory, and 1TB storage to evaluate real-time performances. Due to the resource limitation in DICC, we only managed to set up a testbed with 24 slave nodes. Thus, we conducted a scale-up experiment in WorkflowSim [26], a cloud workflow simulation tool. The environment comprises 100 slave nodes, and each node consists of 1 CPU, 512 MB memory, and 10 GB of storage.

All of the machines were installed with Java (version 8) and Python (version 3.6) to support the experiment framework that comprises: a) the Pegasus (version 4.9.1) as the workflow manager to manage the workflow submission, provenance query, and enactment monitoring, b) HTCCondor [27] as the workflow execution engine that manages the task scheduling and execution, and c) OpenAI Gym [28], a toolkit for developing and comparing reinforcement learning algorithm. Open AI Gym is used to model the *workflow environment* and apply the reinforcement learning algorithm for learning the

clustering parameter optimization. The Montage (version 5.0) was used as the use case to generate the test workload as the input for the experiments.

Montage is a science-grade mosaic generating engine that is commonly used in astronomy. It was initially funded by NASA’s Earth Sciences Technology Office and is currently maintained by the Infrared Processing and Analysis Centre (IPAC). Montage is used to combine multiple satellite images within the same region into a mosaic. The input of the Montage workflow contains the region of the sky that is desired, the size of the mosaic in terms of square degrees, and other parameters such as which image archive to be used. The images for the mosaics are taken from the images archives, such as the Two Micron All Sky Survey (2MASS) [29], Sloan Digital Sky Survey (SDSS) [30], and the Digitised Sky Surveys at the Space Telescope Science Institute (STScI). The input images are first reprojected to the coordinated space of the output mosaic. The background of the reprojected images will then be rectified and added to become the final output mosaics according to degrees specified, as shown in Fig. 4.

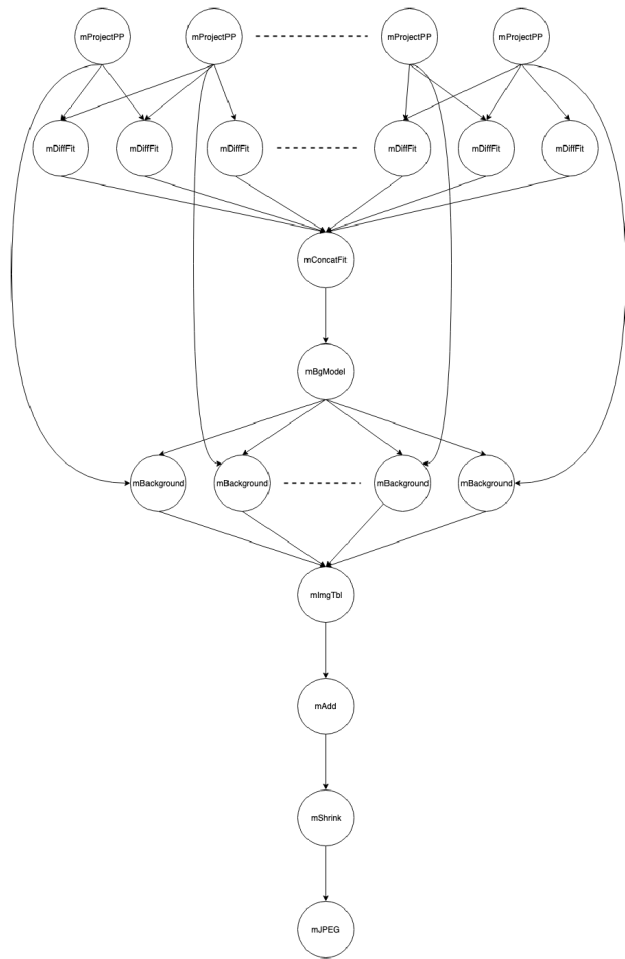


FIGURE 4. Overall structure of the Montage workflow.

Montage is a highly scalable and complex data-intensive application [31]. It comprises a large number of tasks with small runtime of at most of a few minutes. Bharathi *et al.* have developed a workflow generator called Workflow Generator [32], to create workflow models of any size that resemble real workflow applications. The workflow model generated by the Workflow Generator uses the XML file format to represent the DAG workflow model (DAX), *i.e.* the workflow representation in the Pegasus. One of the generated workflow models of Montage with 1000 nodes was used in WorkflowSim to examine the effectiveness of the proposed approach in the second experiment. The complete source code and experiment setup can be found at our Github [33].

### B. EXPERIMENT 1: PROPOSED MODEL EVALUATION USING PEGASUS WMS

The first experiment aimed to validate our proposed reinforcement-learning model for workflow task-clustering described in Section III-C. We had chosen the Montage workflow with a single band (red) of color with 0.5 degrees over the 2MASS dataset. This workflow was used as the test workload for the execution on the Pegasus WMS testbed set up in DICC, which consists of 24 slave nodes.

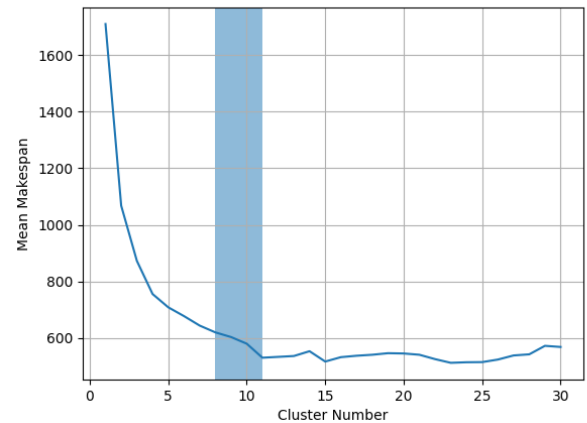


FIGURE 5. Pegasus makespan over cluster number.

In Pegasus, the task clustering process happens during the workflow mapping phase [2]. The workflow manager uses the *transformation catalog* to map the abstract workflow to a concrete workflow with executable tasks, which are ready to be enacted on the execution platform. In our experiment testbed, the workflow environment received the action from the RL agent, *i.e.* the desired cluster number, and changed the transformation catalog accordingly (refer to Fig. 3). Based on Chen *et al.* [34], we have identified four tasks to be clustered in the Montage workflow, *i.e.* *mProject*, *mDiff*, *mDiffFit*, and *mBackground*. The performance data for each execution is captured by the Pegasus provenance module. Thus, the makespan of each run can be obtained by querying the Pegasus provenance record to update the reward function. To validate our proposed model, we performed a preliminary run of 30 execution over each cluster number within the range of  $0 < clusternumber < 30$  to collect the workflow makespan. The optimum cluster number found is 9 and 10, and will be used as the ideal cluster number range to benchmark the performance of the RL agent, as shown in Fig. 5.

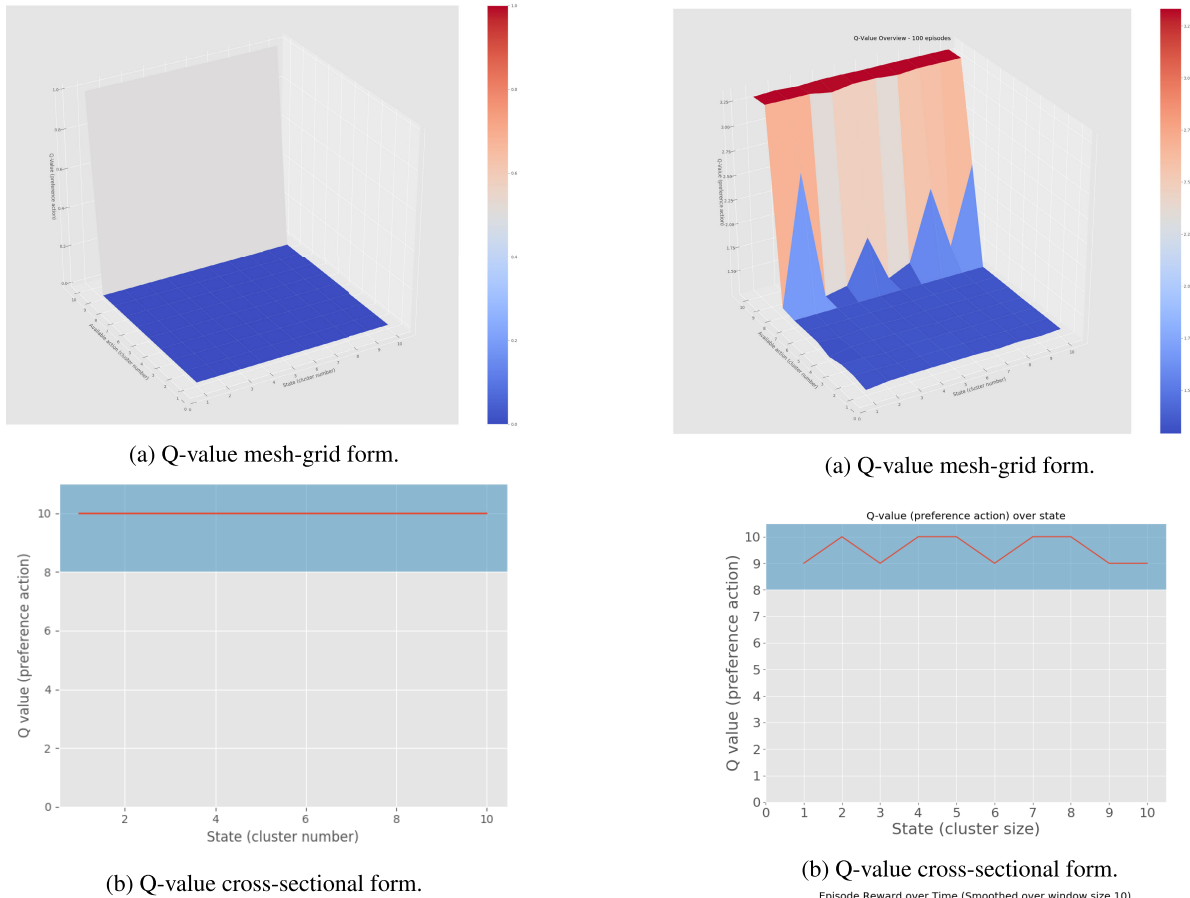
#### 1) EXPERIMENT SETUP

- Action range,  $A$ :  $1 \leq A \leq 10$
- Ideal action range:  $9 \leq A \leq 10$  as shown in Fig. 5
- Environment State,  $S$ : Uniform random between 1 to 10
- Episode Setting: 100 cycles of execution
- Episode Terminate Condition:  $R_{episode} > 20$  or  $R_{episode} < -20$  or reach 100 cycles of execution
- Total episodes of the experiment: 100 episodes

#### 2) EXPERIMENT RESULTS

The results of the first experiment are shown in Fig. 7. The Q-value in Fig. 7a and Fig. 7b shows that the RL agent managed to determine the right pattern shown in Fig. 6, where the positive rewards are given for cluster number 9 and 10. Fig. 7c and Fig. 7d show that the RL agent able to consistently terminate the episode earlier, with an average of 38 cycles per episode and a positive reward of more than +20. After 5 episodes of execution, the RL agent had found the optimal





(a) Q-value mesh-grid form.

(a) Q-value mesh-grid form.

(b) Q-value cross-sectional form.

(b) Q-value cross-sectional form.

FIGURE 6. Ideal Q-value for experiment 1.

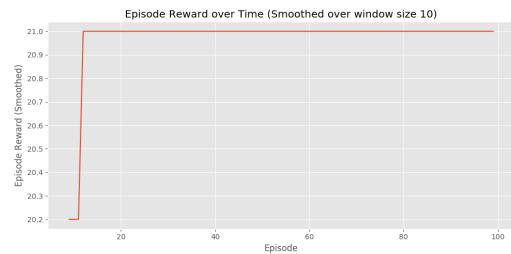
TABLE 2. Overhead parameters configuration in WorkflowSim.

Overhead Parameter	Distribution	Scale	Shape
Cluster delay	Weibull	2.0	1.0
Postscript delay	Weibull	2.0	1.0
Queue delay	Weibull	1.0	1.0
Workflow engine delay	Weibull	1.0	1.0

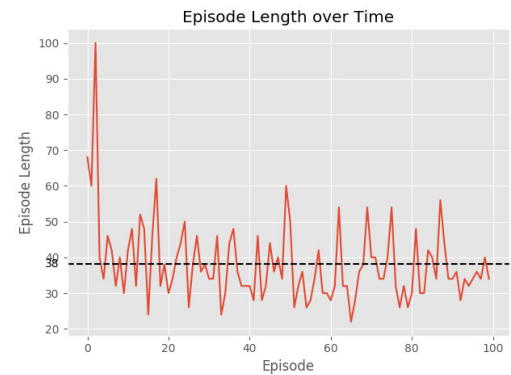
cluster number. However, for the rest of the 95 episodes, we can see a fluctuation range between 25 to 65 cycles per episode. This is due to the RL agent constantly exploring other potential states that yield positive rewards while maintaining learning objectives.

C. EXPERIMENT 2: SCALE-UP EVALUATION USING WorkflowSim

The second experiment aimed to perform a scale-up evaluation of the proposed reinforcement-learning model for workflow task clustering using the WorkflowSim [26], a workflow simulator developed by the Pegasus team as a flexible workflow engine simulator. It is a common tool that has been widely adopted within the scientific workflow community to simulate large-scale scientific workflow experiments. WorkflowSim offers various configurations (e.g. hardware specifications, network speed, and latency) to make the workflow simulation execution to be more realistic. There are four types of overhead parameters provided



(c) Episode rewards.



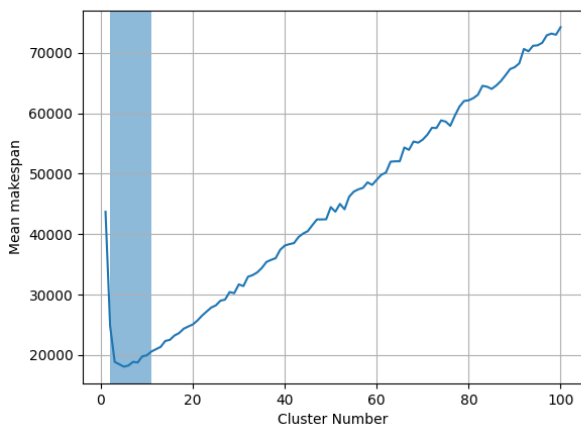
(d) Episode length.

FIGURE 7. Results of experiment 1 over 100 episodes.

in the WorkflowSim for the configuration of task clustering optimization problems: cluster delay, postscript delay, queue delay, and workflow engine delay. The cluster delay

**TABLE 3. Qualitative comparison of related work.**

Study	Workflow Lifecycle Phase	Optimisation Approach
Chen <i>et al.</i> [9]	Execution phase	Optimizing task clustering load balance problem with heuristic approach toward runtime and dependency imbalance
Dong <i>et al.</i> [10]	Execution phase	Improve task clustering with cost and time awareness capabilities
Avinash <i>et al.</i> [17]	Execution phase	Optimizing task clustering by balancing tasks impact factor with hybrid balance task clustering
Tong <i>et al.</i> [18]	Execution phase	Optimizing HEFT tasks sorting and processor allocation with RL Q-learning
Barret <i>et al.</i> [20]	Execution phase	Optimizing scaling policy with RL Q-learning
Wu <i>et al.</i> [21]	Execution phase	Optimizing scheduling plan with RL REINFORCE
Orhean <i>et al.</i> [22]	Execution phase	Offer scheduling solutions as a services based on RL Q-learning and SARSA
Kintsakis <i>et al.</i> [23]	Execution phase	Optimizing scheduling on predicting the workflow task runtime and the probability of task failure by using Recurrent Neural Networks (RNNs) and training with RL Policy Gradient
Proposed model	Mapping phase	Optimizing task clustering parameter with RL Q-learning

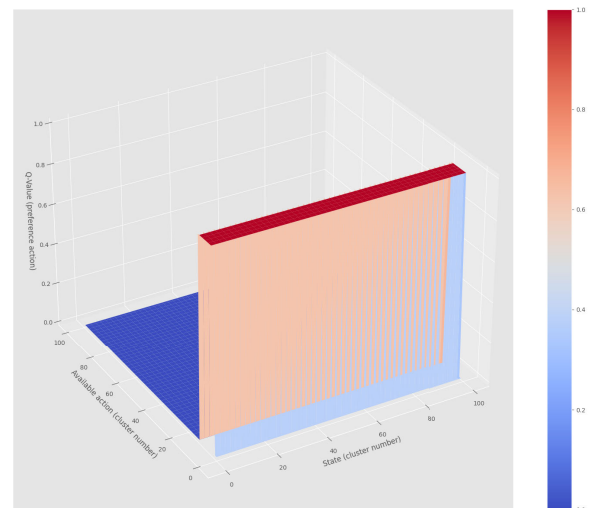


**FIGURE 8. Mean makespan for preliminary run on WorkflowSim.**

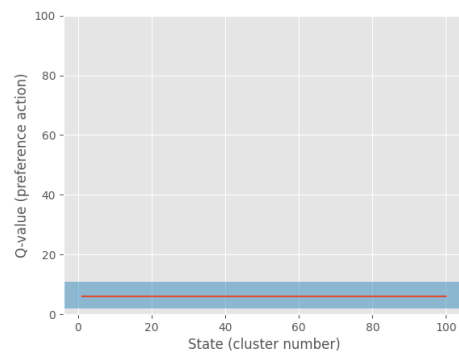
is the time required for the WMS to cluster the jobs. The postscript delay is the time required for the WMS to clean up the jobs after the job finishes the execution. The queue delay is the time for the job queue in the scheduler. Lastly, the workflow engine delay is the time required for WMS to schedule the workflow for execution. In the experiment, we had configured the overhead parameter based on the finding of Chen *et al.* [35] on the overhead analysis over scientific workflow execution, as showed in Table 2. The Montage template with 1000 nodes of tasks used in this experiment is adopted from Bharathi *et al.* [32].

1) EXPERIMENT SETUP

- Action range,  $A$ :  $1 \leq \text{cluster number} \leq 100$
- Ideal cluster number range:  $2 \leq \text{cluster number} \leq 11$  as shown in Fig. 8
- Environment State,  $S$ : Uniform random between 1 to 100
- Episode Setting: 100 cycles of execution
- Episode Terminate Condition:  $R_{\text{episode}} > 20$  or  $R_{\text{episode}} < -20$  or reach 100 cycles of execution
- Total episodes of the experiment: 100 episodes



(a) Mesh-grid form.

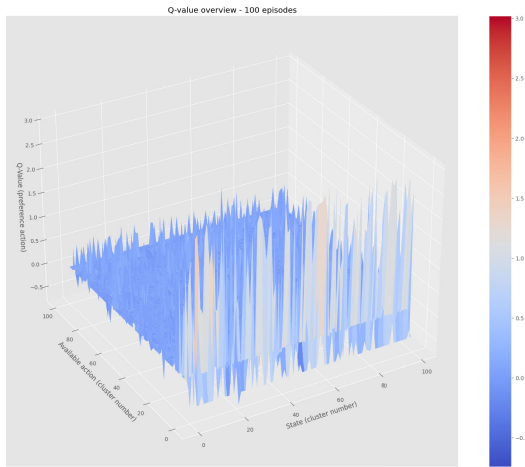


(b) Cross-sectional form.

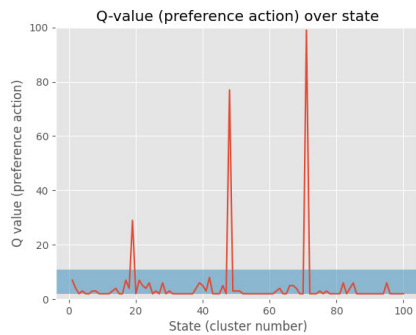
**FIGURE 9. Ideal Q-value for experiment 2.**

2) EXPERIMENT RESULTS

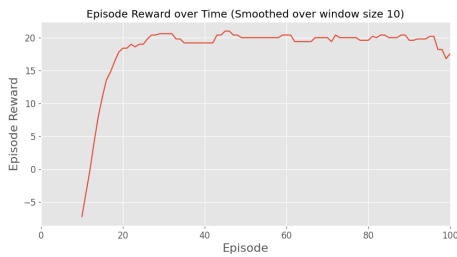
The result of the second experiment is shown in Fig. 10. The Q-value function in Fig. 10a and Fig. 10b shows that the RL agent managed to determine the right pattern, where positive



(a) Q-value mesh-grid form.



(b) Q-value cross-sectional form.



(c) Episode rewards.



(d) Episode length.

**FIGURE 10. Results of experiment 2 over 100 episodes.**

rewards are given to lower cluster number from the range between 2 to 11, as shown in Fig. 9. The RL agent demonstrates that over 97% of the states prefer the cluster number within the optimal cluster range. Fig. 10c also shows that

the RL agent can consistently terminate the episode earlier with an average of 78 cycles per episode in Fig. 10d and a positive reward of more than +20. However, we observed that many states remain unexplored by the RL agent from Fig. 10a. This phenomenon is due to insufficient episodes for the RL agent to explore and sample all of the states, *i.e.* 10000 cluster numbers in the state space.

**D. DISCUSSION**

The finding of both experiment shows that the RL agent can identify and maintain the optimal cluster number over the course of the experiment. Due to the absence of the similar works optimising task clustering during mapping phase that can be used for the performance comparison and analysis, we have performed a qualitative comparison with the relevant approaches, as presented in Table 3. The comparison was made based on the optimization phase, target, and machine learning approach. From the optimization target perspective, the existing task clustering optimization approaches [9], [10], [17] focus on identifying the imbalance distribution of runtime and dependency of the tasks while maintaining the timeline and budget of the experiment through heuristic approaches, while the other machine-learning optimization approaches [20]–[23] are generally improving the scheduler performance by applying machine learning over the scheduling plan and resource allocations. All of the existing studies on task clustering and machine learning optimization approaches are focusing on the execution phase of the workflow lifecycle. As compared to our study, we applied task clustering optimization over the mapping phase during the workflow enactment. Task clustering in the mapping phase normalizes the workflow structures and indirectly reduces the number of tasks submitted to the scheduler to improve the overall workflow performance. This paper is the first attempt that applied the machine-learning approach toward mapping phase optimization in the context of the scientific workflows.

**V. CONCLUSION**

Scientific workflows consist of many fine-grained computational tasks. The execution time for each of the tasks may vary from milliseconds to hours. For workflows with a large number of small tasks, such as image re-projection and difference in the Montage workflow, we can optimise its execution by grouping these fine-grained tasks into a larger cluster to reduce the execution overhead. In the current practice, most of the clustering process required domain expert expertise and experience to decide on the clustering parameter, *i.e.* cluster size and cluster number. However, it is impracticable to have a human intervention to optimize workflows submitted at all the time. In this paper, we propose to address this problem by identifying the optimal clustering number using a machine learning approach. We have defined a new approach that incorporates the use of RL agent in the WMS. The workflow environment queries the provenance records of previous execution to update the reward function for the RL agent so that the RL agent can determine the best parameter

to cluster the workflow tasks in the upcoming execution. This approach allows the continuous improvement of the workflow makespan over time.

We have conducted two experiments to evaluate the proposed approach. We validated the proposed RL model using a real-world scientific application, namely Montage and was enacted using Pegasus WMS. The experiment results showed that the RL agent was able to learn and identify the positive rewarded actions, *i.e.* optimal cluster number, for the selected workflow. We then performed a scale-up experiment on WorkflowSim to simulate the workflow environment with a larger number of computing resources. The RL agent managed to converge as planned. However, we also observed the limitation of the Q-learning algorithm chosen for our RL model. After an average of 7,400 execution cycles, there is still a large number of unexplored states in the RL agent state space. Thus, we foresee that it will be difficult to scale into millions of CPU in a real-world HPC environment which involves the production of an excessively large Q-table and update.

In the future work, we will explore alternative RL algorithms to overcome the large-state-space convergence efficiency using deep reinforcement learning (DQN) [36], gradient boost [37], and actor-critic [38]. We also consider to further improve the performance of our proposed RL model using a neural network to discover the underlying workflow patterns before the execution. This information is useful in identifying which tasks are suitable for task clustering and further explore other clustering dimensions.

## ACKNOWLEDGMENT

This work was supported in part by the Ministry of Higher Education Malaysia under Grant LRGS-LR002-2020. The authors would like to thank the reviewers and the associate editor for their comments which improved this manuscript. This work was completed in part with resources provided by the Universiti Malaya DICC.

## REFERENCES

- [1] C. S. Liew, M. P. Atkinson, M. Galea, T. F. Ang, P. Martin, and J. I. van Hemert, "Scientific workflows: Moving across paradigms," *ACM Comput. Surv.*, vol. 49, no. 4, pp. 1–39, 2016.
- [2] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger, "Pegasus, a workflow management system for science automation," *Future Gener. Comput. Syst.*, vol. 46, pp. 17–35, May 2015.
- [3] Y. Gil, V. Ratnakar, J. Kim, P. Gonzalez-Calero, P. Groth, J. Moody, and E. Deelman, "Wings: Intelligent workflow-based design of computational experiments," *IEEE Intell. Syst.*, vol. 26, no. 1, pp. 62–72, Jan. 2011.
- [4] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel, "KNIME—the Konstanz information miner: Version 2.0 and beyond," *ACM SIGKDD Explor. Newslett.*, vol. 11, no. 1, pp. 26–31, 2009.
- [5] S. Marru, L. Gunathilake, C. Herath, P. Tangchaisin, M. Pierce, C. Mattmann, and R. Singh, "Apache airavata: A framework for distributed applications and computational workflows," in *Proc. ACM Workshop Gateway Comput. Environ.*, 2011, pp. 21–28.
- [6] A. M. Kintsakis, F. E. Psomopoulos, A. L. Symeonidis, and P. A. Mitkas, "Hermes: Seamless delivery of containerized bioinformatics workflows in hybrid cloud (HTC) environments," *SoftwareX*, vol. 6, pp. 217–224, Mar. 2017.
- [7] M. A. Rodriguez and R. Buyya, "Scientific workflow management system for clouds," in *Software Architecture for Big Data and the Cloud*, I. Mistrik, R. Bahsoon, N. Ali, M. Heisel, and B. Maxim, Eds. Boston, MA, USA: Morgan Kaufmann, 2017, ch. 18, pp. 367–387.
- [8] E. Afgan, D. Baker, B. Batut, M. van den Beek, D. Bouvier, M. Cech, J. Chilton, D. Clements, N. Coraor, B. A. Grüning, A. Guerler, J. Hillman-Jackson, S. D. Hiltmann, V. Jalili, H. Rasche, N. Soranzo, J. Goecks, J. Taylor, A. Nekrutenko, and D. J. Blankenberg, "The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update," *Nucleic Acids Res.*, vol. 46, no. W1, pp. 537–544, 2018.
- [9] W. Chen, R. Ferreira da Silva, E. Deelman, and R. Sakellariou, "Using imbalance metrics to optimize task clustering in scientific workflow executions," *Future Gener. Comput. Syst.*, vol. 46, pp. 69–84, May 2015.
- [10] M. Dong, L. Fan, and C. Jing, "ECOS: An efficient task-clustering based cost-effective aware scheduling algorithm for scientific workflows execution on heterogeneous cloud systems," *J. Syst. Softw.*, vol. 158, Dec. 2019, Art. no. 110405.
- [11] H. Wang and O. Sinnens, "List-scheduling versus cluster-scheduling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 3, pp. 1736–1749, Aug. 2018.
- [12] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-science: An overview of workflow system features and capabilities," *Future Gener. Comput. Syst.*, vol. 25, no. 5, pp. 528–540, May 2009.
- [13] S. G. Ahmad, C. S. Liew, E. U. Munir, T. F. Ang, and S. U. Khan, "A hybrid genetic algorithm for optimization of scheduling workflow applications in heterogeneous computing systems," *J. Parallel Distrib. Comput.*, vol. 87, pp. 80–90, Jan. 2016.
- [14] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," 2016, *arXiv:1611.09940*. [Online]. Available: <http://arxiv.org/abs/1611.09940>
- [15] G. Singh, M.-H. Su, K. Vahi, E. Deelman, B. Berriman, J. Good, D. S. Katz, and G. Mehta, "Workflow task clustering for best effort systems with pegasus," in *Proc. 15th ACM Mardi Gras Conf. Lightweight Mash-Ups Lambda Grids: Understand. Spectr. Distrib. Comput. Requirements, Appl., Tools, Infrastructures, Interoperability, Incremental Adoption Key Capabilities (MG)*, New York, NY, USA, 2008, p. 9.
- [16] E. N. Alkhanak, S. P. Lee, R. Rezaei, and R. M. Parizi, "Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: A review, classifications, and open issues," *J. Syst. Softw.*, vol. 113, pp. 1–26, Mar. 2016.
- [17] A. Kaur, P. Gupta, and M. Singh, "Hybrid balanced task clustering algorithm for scientific workflows in cloud computing," *Scalable Comput., Pract. Exper.*, vol. 20, no. 2, pp. 237–258, May 2019.
- [18] Z. Tong, X. Deng, H. Chen, J. Mei, and H. Liu, "QL-HEFT: A novel machine learning scheduling scheme based on cloud computing environment," *Neural Comput. Appl.*, vol. 32, no. 10, pp. 5553–5570, May 2020.
- [19] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.
- [20] E. Barrett, E. Howley, and J. Duggan, "Applying reinforcement learning towards automating resource allocation and application scalability in the cloud," *Concurrency Comput., Pract. Exper.*, vol. 25, no. 12, pp. 1656–1674, Aug. 2013.
- [21] Q. Wu, Z. Wu, Y. Zhuang, and Y. Cheng, "Adaptive DAG tasks scheduling with deep reinforcement learning," in *Algorithms Architectures for Parallel Process.*, J. Vaidya and J. Li, Eds. Cham, Switzerland: Springer, 2018, pp. 477–490.
- [22] 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, Jul. 2018.
- [23] A. M. Kintsakis, F. E. Psomopoulos, and P. A. Mitkas, "Reinforcement learning based scheduling in a workflow management system," *Eng. Appl. Artif. Intell.*, vol. 81, pp. 94–106, May 2019.
- [24] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, Jun. 2013.
- [25] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2009.
- [26] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," in *Proc. IEEE 8th Int. Conf. E-Sci.*, Oct. 2012, pp. 1–8.
- [27] P. Couvares, T. Kosar, A. Roy, J. Weber, and K. Wenger, "Workflow management in condor," in *Workflows for e-Science*. Springer, 2007, pp. 357–375.

- [28] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," 2016, *arXiv:1606.01540*. [Online]. Available: <http://arxiv.org/abs/1606.01540>
- [29] (2006). *The 2Mass Project*. [Online]. Available: <http://www.ipac.caltech.edu/2mass>
- [30] R. Ahumada et al., "The 16th data release of the Sloan digital sky surveys: First release from the APOGEE-2 southern survey and full release of eBOSS spectra," *Astrophys. J. Suppl.*, vol. 249, no. 1, p. 3, 2020.
- [31] G. B. Berriman, E. Deelman, J. C. Good, J. C. Jacob, D. S. Katz, C. Kesselman, A. C. Laity, T. A. Prince, G. Singh, and M.-H. Su, "Montage: A grid-enabled engine for delivering custom science-grade mosaics on demand," in *Optimizing Scientific Return for Astronomy through Information Technologies*, vol. 5493, P. J. Quinn and A. Bridger, Eds. Bellingham, WA, USA: SPIE, 2004, pp. 221–232.
- [32] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *Proc. 3rd Workshop Workflows Support Large-Scale Sci.*, Nov. 2008, pp. 1–10.
- [33] C. P. Leong. *Rayson1223/Gym-Workflow*. Accessed: Jun. 23, 2021. [Online]. Available: <https://github.com/rayson1223/gym-workflow>
- [34] W. Chen, R. F. D. Silva, E. Deelman, and R. Sakellariou, "Balanced task clustering in scientific workflows," in *Proc. IEEE 9th Int. Conf. e-Science*, Oct. 2013, pp. 188–195.
- [35] W. Chen and E. Deelman, "Workflow overhead analysis and optimizations," in *Proc. 6th Workshop Workflows Support Large-Scale Sci. (WORKS)*, 2011, pp. 11–20.
- [36] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: <http://arxiv.org/abs/1312.5602>
- [37] J. H. Friedman, "Stochastic gradient boosting," *Comput. Statist. Data Anal.*, vol. 38, no. 4, pp. 367–378, 2002.
- [38] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1008–1014.



**CHEE SUN LIEW** received the Ph.D. degree in informatics from The University of Edinburgh, under the supervision of former U.K. eScience Envoy Prof. M. Atkinson, and contributes to a European funded project: Advanced Data Mining and Integration Research for Europe. He is currently the Founder and the Head of UM Data-Intensive Computing Centre that facilitates the use of data-intensive computing and high-performance computing technologies in accelerating the advancement in scientific discovery. He is also chairing the Working Group on Infrastructure of the Malaysia Open Science Platform.



**CHEE SENG CHAN** (Senior Member, IEEE) received the Ph.D. degree from the University of Portsmouth, U.K., in 2008. In general, his research interests include computer vision and machine learning with a focus on scene understanding. He is also interested in the interplay between vision and language—generating sentential descriptions about complex scenes. He was a recipient of the Young Scientist Network-Academy of Sciences Malaysia (YSN-ASM), in 2015 and the Hitachi Research Fellowship, in 2013.



**MUHAMMAD HABIB UR REHMAN** (Senior Member, IEEE) received the bachelor's and master's degrees from COMSATS University Islamabad, Pakistan, and the Ph.D. degree from the Faculty of Computer Science and Information Technology, University of Malaya, Malaysia. He is currently working with the Center for Cyber-Physical Systems, Khalifa University of Science and Technology, United Arab Emirates, as a Postdoctoral Research Fellow. He is also working on trustworthy blockchain technologies for intelligent cyber-physical systems. His research interests include research and development of trust models for decentralized and trustworthy artificial intelligence applications for cyber-physical systems. He has been an alumnae of DAAD's Postdoctoral Network, since September 2019. He has authored or coauthored 40 international publications, including journal articles, conference proceedings, book chapters, and magazine articles, whereby his four articles are categorized as highly cited publications by Web-of-Science. His research interests include blockchain technologies, cyber-physical systems, secure key management, big data, edge computing, and the industrial IoT. He is also a Bright Spark Fellow. He received gold medals and 100% fee-waiver scholarships from COMSATS University Islamabad.

...



**CHIN POH LEONG** received the bachelor's degree from the Faculty of Computer Science and Information Technology, University of Malaya, Malaysia. His research interests include workflow technologies, reinforcement learning, and artificial intelligence applications for scientific workflow. He had held research positions in the Data Intensive Computing Center (DICC), University of Malaya.