Multi-Agent Deep Reinforcement Learning Framework for Renewable Energy-Aware Workflow Scheduling on Distributed Cloud Data Centers

Amanda Jayanetti, Saman Halgamuge, *Fellow, IEEE*, and Rajkumar Buyya, *Fellow, IEEE*

***Abstract*—The ever-increasing demand for the cloud computing paradigm has resulted in the widespread deployment of multiple datacenters, the operations of which consume very high levels of energy. The carbon footprint resulting from these operations threatens environmental sustainability while the increased energy costs have a direct impact on the profitability of cloud providers. Using renewable energy sources to satisfy the energy demands of datacenters has emerged as a viable approach to overcome the aforementioned issues. The problem of scheduling workflows across multi-cloud environments powered through a combination of brown and green energy sources includes multiple levels of complexities. First, the general case of workflow scheduling in a distributed system itself is NP-hard. The need to schedule workflows across geo-distributed cloud datacenters adds a further layer of complexity atop the general problem. The problem becomes further challenging when the datacenters are powered through renewable sources which are inherently intermittent in nature. Consequently, traditional workflow scheduling algorithms and single-agent reinforcement learning algorithms are incapable of efficiently meeting the decentralized and adaptive control required for addressing these challenges. To this end, we have leveraged the recent advancements in the paradigm of MARL (Multi-Agent Reinforcement Learning) for designing and developing a multi-agent RL framework for optimizing the green energy utilization of workflow executions across multi-cloud environments. The results of extensive simulations demonstrate that the proposed approach outperforms the comparison algorithms with respect to minimizing energy consumption of workflow executions by 47% while also keeping the makespan of workflows in par with comparison algorithms. Furthermore, with the proposed optimizations, the multi-agent technique learnt 5 times faster than a generic multi-agent algorithm.**

***Index Terms*—Deep reinforcement learning, cloud computing, workflow scheduling, green computing.**

I. Introduction

In RECENT times, cloud computing has become a frequently used platform for running resource-intensive workloads of commercial as well as scientific applications owing to its inherent ability of provisioning compute and network resources in an on-demand manner. Cloud computing services are offered by service providers via multiple geographically distributed datacenters for improving the Quality of Service (QoS) experienced by geographically dispersed consumers as well as for other requirements such as disaster recovery and high availability.

The excessive demand for cloud computing services has given rise to an exponential rise in the power consumption of cloud datacenters [1], hence adversely impacting the sustainability of the computing paradigm. Furthermore, high power consumption levels also increase the operational costs of datacenters which in turn reduces the profitability of service providers. As a step towards mitigating the adverse impacts of high power consumption and associated effects such as CO2 emissions, increasingly more renewable energy sources (such as solar power and wind power) are leveraged by cloud providers for satisfying the power requirements of cloud datacenters.

Geographically dispersed datacenters can be powered through locally available renewable energy sources thereby minimizing the use of brown energy. However, due to the intermittent nature of renewable energy sources, their utilization for powering cloud datacenters gives rise to several challenges. Variations in weather conditions and other location-dependent factors could drastically impact the levels of power generated by renewable energy sources. Distribution of workloads among geographically distributed datacenters while considering the renewable energy generation levels as well as diverse QoS requirements of heterogeneous workloads makes it possible to handle the intermittent nature of renewable energy sources in an efficient manner.

Workflow is a popular application model which can be used to represent a wide variety of workloads ranging from scientific to commercial applications. Cloud computing environments are widely used for the execution of resource-intensive workflows. As opposed to scheduling individual workloads, workflow scheduling in cloud computing environments is more complex due to the presence of precedence relations between workflow tasks. The problem becomes even more challenging when workflows are to be scheduled across multiple geographically distributed datacenters some of which are operating on renewable energy sources. Only very few studies in existing literature have simultaneously attempted to tackle the

Manuscript received 12 April 2023; revised 17 January 2024; accepted 19 January 2024. Date of publication 31 January 2024; date of current version 4 March 2024. The work of Amanda Jayanetti was supported by the Australian Research Council under Grant DP210101135. Recommended for acceptance by M. Li. *(Corresponding author: Amanda Jayanetti.)*

The authors are with the Cloud Computing and Distributed Systems (CLOUDS) Lab, School of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3010, Australia (e-mail: amjayanetti@student.unimelb.edu.au; saman@unimelb.edu.au; rbuyya@unimelb.edu.au).

Digital Object Identifier 10.1109/TPDS.2024.3360448

aforementioned challenges. The scheduling frameworks proposed in these studies have primarily used heuristic, metaheuristic or contemporary techniques for achieving the desired objectives. These approaches generally suffer from problems such as high computational complexity, low adaptability to dynamically changing conditions, and so on.

Reinforcement Learning (RL) has emerged as a promising solution for dealing with highly dynamic and unpredictable problems. In particular, the more recent combination of Deep Neural Networks (DNN) with Reinforcement Learning which gave rise to the Deep Reinforcement Learning (DRL) paradigm, is proven to have the potential of solving complex problems in highly stochastic environments [2]. DRL agents operate with no prior knowledge of the environment, and they learn by interacting with the environment and gathering rewards for actions performed. The rewards enable the agents to learn desirable and undesirable behaviour in different situations, thus enabling them to identify the actions that result in the maximization of overall rewards.

DRL techniques have been used in several studies for handling resource management problems in cloud computing environments [3], [4]. However, a vast majority of these works have proposed straightforward adaptations of popular RL algorithms such as Deep Q Learning for designing single-agent scheduling frameworks that are more suited for satisfying the centralized scheduling requirements of the traditional cloud computing paradigm. Such approaches are not suitable when workloads are to be scheduled across more complex distributed infrastructures such as federated clouds and emerging fog computing environments [5]. Accordingly, Multi-Agent Systems (MAS) in which interactions between multiple agents are leveraged, are proven to be a better fit for problem-solving in highly distributed and stochastic environments compared to its single-agent counterpart [6].

The direct implementation of single agent RL algorithms in multiple agents leads to a non-stationary environment which prevents such methods from converging to an optimal solution [7]. Therefore, the design of efficient multi-agent RL (MARL) algorithms requires overcoming multiple challenges including nonstationarity and partial observability of the environment, scalability issues that arise as the joint action space grows with the increasing number of agents, and communication between agents which is particularly challenging in partially observable environments [7]. As the name implies, in partially observable environments, the agent does not perceive the complete state of the environment, instead only a partial observation is visible based on which the agent takes an action. The existing works that have proposed DRL-based scheduling techniques in multi-cloud environments have assumed full observability. However, realistically, most multi-cloud environments tend to be partially observable since it is unlikely that local information about all the clouds will be globally available at all times.

To overcome the aforementioned challenges and design an RL framework that can efficiently schedule workflows across partially observable and highly distributed multi-cloud environments, we leveraged the multi-agent coordination technique proposed by R. Lowe et al. [8]. In this technique, the traditional actor-critic method is extended to a multi-agent setting by providing the critic with extra information about the actions and observations of other agents during the training process. During execution, the agents operate based on local observations. Accordingly, the proposed technique conforms to the paradigm of centralized training and distributed execution which is proven to be highly effective in multi-agent systems [9]. A drawback of the aforementioned multi-agent coordination technique is that the Q function grows linearly with the number of agents, and this in turn could adversely impact the learning process of the agents. To overcome the potential impediment to agent learning, we leveraged domain-specific characteristics of the problem to design multi-agent coordination in a hierarchical manner. This in turn leads to the achievement of better training efficacy as evidenced by the empirical results from extensive simulations.

The following are the main contributions of this work:

- A hierarchical design for the multi-cloud workflow scheduling problem in which a global RL agent assigns tasks to datacenters and local RL agents assign tasks to nodes. Coupled with this, we present a novel formulation of the agent environment comprising state space, action space, and reward as a Partially Observable Markov Decision Process (POMDP).
- Design of a MARL framework capable of scheduling workflows across the partially observable and highly distributed multi-cloud environments in an efficient manner by sharing extra information during training, and operating solely based on local information during execution. Furthermore, we propose a shared reward structure that motivates agents to act cooperatively for achieving a common goal.
- Design of a novel approach to handling the curse of dimensionality and thereby improving training efficiency by leveraging the hierarchical nature of multi-cloud scheduler for limiting the observations shared by agents to a local neighborhood.

The rest of the paper is organized as follows: In Section II we review relevant literature, and in Section III we present the formulation of the workflow scheduling problem in multi-cloud environments. The proposed RL framework is presented in Section IV and its performance evaluation results are discussed in Section V. Finally, the conclusion of this work is presented in Section VI along with future work.

II. Related Work

In this section, we review workflow scheduling algorithms that are related to the problem considered in this work. Several different techniques are proposed in academia for enhancing the energy-efficiency of workflow executions in cloud computing environments. Amongst the solutions that have been identified, techniques such as Dynamic Power Management (DPM) in which workloads are consolidated into a minimum of servers, and unused servers and networking elements are hibernated to achieve energy efficiency [10], Dynamic Voltage and Frequency Scaling (DVFS) based methods [11] which operate by dynamically adjusting the supply voltage or operating frequencies of computing elements (CPU, memory) to reduce energy

consumption, and powering datacenters with renewable energy sources [12] are predominantly used for minimizing datacenter power consumption.

A number of studies have focused on the problem of scheduling workflows across federated clouds [13], [14], [15]. In [13], a heuristic-based technique is proposed for scheduling workflows across multiple datacenters with the objective of minimizing electricity costs. They propose strategies for sorting workflows and sequencing workflow tasks, and also for allocating resources to tasks such that the resulting framework can achieve the desired objectives. [14], [15] also proposed algorithms for scheduling workflows across federated cloud datacenters with the objectives of minimizing cost and improving reliability.

Several works in existing literature have focused on the problem of scheduling workflows across geo-distributed datacenters powered through renewable energy sources [12], [16]. In [12], a hierarchical scheduling framework is proposed for scheduling workflows across multiple geo-distributed datacenters powered via renewable energy partially. High-level scheduler allocates workflows to datacenters while a low-level scheduler assigns tasks of workflows to compute resources for achieving the overall objectives of improved energy efficiency, makespan, and deadline hits. [16] used a genetic algorithm to design a workflow scheduling framework that maximizes the use of renewable energy while minimizing the cost of electricity.

Renewable energy aware scheduling of independent tasks/jobs across multi-cloud environments is also studied in a number of works [17], [18], [19]. In [17] Integer Linear Programming (ILP) is used for designing a task scheduling framework for minimizing the use of brown energy in datacenters while satisfying user-defined time constraints and electricity budgets. An obvious drawback of the proposed approach is the high time complexity associated with ILP based problem formulation. [18] used simulated annealing coupled with bees algorithm for scheduling tasks across distributed cloud computing environments with the objective of minimizing energy consumption. In [19] a job allocation algorithm is proposed to assign transactional and batch jobs to datacenters such that the power consumption of datacenters can be altered to suit variable conditions such as green energy availability.

While considerable research efforts have been focused on designing RL-based algorithms for scheduling workloads within a single datacenter [20], [21], [22], not much work has been done with RL in multi-cloud scheduling scenarios. Only a few works have leveraged the advanced capabilities of RL for designing workload scheduling algorithms across distributed cloud computing environments [4], [23]. In [23], Proximal Policy Optimization (PPO) [24] was used for designing a scheduling policy capable of determining if the job should be executed in a particular server of a private datacenter or offloaded to a VM of a certain type in the public cloud depending on multiple factors including predicted renewable energy availability, deadline constraints, and cost. C. Xu et al. [4] proposed an RL-based algorithm for migrating jobs across multiple datacenters with the objective of minimizing energy cost.

A majority of existing studies have used single agent RL in workflow scheduling algorithms [25], [26] and these methods are designed to operate within a single cloud datacenter rather than across multiple datacenters. However, multi-agent RL is likely to be a better candidate for multi-cloud environments due to the need for decentralized decision-making. Owing to the inherently distributed nature of edge and fog computing environments, multi-agent RL is likely to be an efficient candidate for schedulers that operate in such environments as well. However, a majority of existing works have proposed single agent RL methods for scheduling problems in edge and fog computing environments [27], [28]. Only few works have designed decentralized execution and/or training based techniques for these environments [5], [29], [30]. In [5], IMPALA framework is used for training distributed brokers which are responsible for making placement decisions of applications in a fog computing environment. In [30], multiple actor networks are trained with a common critic network. [29] uses a value decomposition network coupled with centralized training and distributed execution method for handling the non-stationarity of environment arising due to the presence of multiple agents. Table I presents a high-level comparison of proposed method with relevant literature.

The design of multi-agent systems is more complicated since the presence of multiple agents could make the environment nonstationary [7]. Despite the complexities, multi-agent RL frameworks are likely to be more effective in such environments since they can be used for developing decentralized scheduling policies which are more suited for environments benefiting from decentralized control. However, none of the existing works have efficiently leveraged the power of multi-agent RL for scheduling workflows across multi-cloud environments.

III. System Model

Directed Acyclic Graphs (DAG) are used for modeling the workflows that are scheduled by the proposed hierarchical scheduling framework. The tasks of a workflow are represented by the set of nodes, $V = \{v_0, v_1..v_n\}$ and the precedence constraints between tasks are represented by the set of edges, $E = \{(v_i, v_j)|v_i, v_j \in V\}$ of a DAG, $G = (V, E)$. It is also assumed that workflows are containerized (each task is a container).

Workflows are to be scheduled across a federation of geo-distributed datacenters $DC = \{dc_1, dc_2, ..dc_n\}$ which are powered through a combination of green energy from renewable energy sources and brown energy from the grid. Therefore, at any instance, the total power consumption, P_{total} of the datacenter federation is the sum of green power, P_{green} and brown power, P_{brown} consumed by the underlying cloud infrastructures and operations. The datacenter, dc_i comprises of a set of λ_i heterogeneous severs, $\{m_1, m_2, ..m_{\lambda_i}\}$. Power consumed by a server, m_i is calculated using the CPU utilization-based power model presented in [31] as follows:

$$P_{m_i} = \begin{cases} P_{m_i}^{idle} + \left(P_{m_i}^{dynamic} - P_{m_i}^{idle}\right).u_{m_i}, & \text{if } u_{m_i} > 0 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where $P_{m_i}^{idle}$ is the idle power consumption of the server which is a constant regardless of its current utilization and $P_{m_i}^{dynamic}$ is the dynamic power consumption which is dependent on the current server utilization, u_{m_i}. Accordingly, the total power consumed by the datacenter, dc_i during the kth time interval is computed

as follows:

$$P_i^{total}(k) = \sum_{i=1}^{\lambda_i} P_{m_i}(k) \tag{2}$$

The objective of the proposed hierarchical scheduling framework is to minimize total brown energy utilization of the cloud datacenter federation while also optimizing workflow execution time. The total brown energy consumption at the datacenter dc_i during the kth time interval can be represented as:

$$P_i^{brown}(k) = P_i^{total}(k) - P_i^{green}(k) \tag{3}$$

where $P_i^{green}(k)$ is the total green energy available at the datacenter dc_i during the kth time interval. Accordingly, the primary objective of the global scheduler which mainly focuses on minimizing the brown energy usage during the kth time interval can be represented as follows:

$$\text{Min: } P_{total}^{brown}(k) = \sum_{i=1}^{N} P_i^{brown}(k) \tag{4}$$

Once a task is assigned to a datacenter dc_i, the local scheduler allocates the task to a server that jointly minimizes the total energy consumption and execution time of the task. Hence, the objective of the local scheduler during the kth interval can be represented as follows:

$$\text{Minimize: } \sum_{j=1}^{N} \alpha T_{t_j} + (1-\alpha) E_{t_j} \tag{5}$$

where N is the total number of tasks executed during the kth time interval. T_{t_j} and E_{t_j} denotes the total execution time and total energy consumption associated with the execution of task t_j, respectively. The execution time T_{t_j} includes the maximum data transfer time from predecessor nodes, computation time of the task as well as the waiting time of the task at the node (WT_{t_j}) before the task is actually executed. It can be computed as follows:

$$T_{t_j} = \frac{L_{t_j}}{F} + WT_{t_j} + \max_{t_i \in pred(t_j)} DT_{t_i,t_j} \tag{6}$$

where, L_{t_j} is the size of the task t_j, and F is the processing rate of the node to which task is assigned. The ratio $\frac{L_{t_j}}{F}$ is the computation time of the task. DT_{t_i,t_j} is the data transfer time from the node in which predecessor t_i executed to the execution node of task t_j. If tasks are in the same datacenter, then DT_{t_i,t_j} can be expressed as the ratio $\frac{D_{t_i,t_j}}{B_{in}}$ where D_{t_i,t_j} is the size of data to be transferred from t_i to t_j and B_{in} is the network bandwidth within the same datacenter. If data transfer is between nodes in different datacenters, DT_{t_i,t_j} can be represented as $\frac{D_{t_i,t_j}}{B_{out}}$, where B_{out} represents the network bandwidth between datacenters. Total energy consumed during the execution of task t_j is computed as follows:

$$EE_{t_j} = T_{t_j} \times [U \times P_{active} + P_{idle}] \tag{7}$$

where U is the current CPU utilization level of the execution node, and the rates of power consumption at active and idle states of the processors are denoted by P_{active} and P_{idle}, respectively. Considering the power consumption associated with the transmission of data to be P_{comm}, the energy consumed during the transfer of data from predecessor nodes is computed as follows:

$$ET(t_j) = \sum_{t_i \in pred(t_j)} \frac{D_{t_i,t_j}}{B} \times P_{comm} \tag{8}$$

The total energy consumed E_{t_j} is the sum of computation and communication energy, and is computed as:

$$E_{t_j} = EE_{t_j} + ET(t_j) \tag{9}$$

IV. Reinforcement Learning

A. Background

Reinforcement Learning (RL) is a branch of the broader machine learning paradigm that operates by training an intelligent agent to learn a desired behavior in a given environment by learning through its interactions with the environment. The learning process is governed by the rewards which are received by the agent in return for the actions that it chooses to perform in the environment. Rewards serve as an indication of the degree of desirability of the action taken under the prevalent conditions toward achieving a pre-defined goal. RL problems are commonly modelled using the mathematical framework, Markov Decision Process (MDP).

At each decision epoch, the immediate situation the agent encounters, which is referred to as the current state (s_t) of the environment is taken into account by the agent for taking an action (a_t), which then results in a state transition from the current state (s_t) to the next state (s_{t+1}). Depending on the impact of the action on the environment, a reward (r_t) is given to the agent. Reward serves as a measure of the success of the agent's action in the given situation. As the agent progresses through the learning process, it learns to produce actions that result in the maximization of cumulative rewards over time. The strategy the agent employs to determine actions in this manner is called a policy ($\pi(a_t|s_t)$).

Despite the advanced capabilities of RL, the traditional RL paradigm suffers from the problem of dimensionality curse which makes its application to complex problems with very large state spaces practically infeasible. The combination of RL with deep learning which is referred to as Deep Reinforcement Learning (DRL) has successfully proven to overcome the aforementioned issue through function approximation, thereby eliminating the need for agents to visit all states during the training process and for storing state transition data in space-consuming tabular formats. Accordingly, a neural network is used for representing the policy ($\pi(a_t|s_t)$) as a parameterized function with respect to an adjustable parameter θ. The resulting parameterized policy can be denoted as ($\pi_\theta(a_t|s_t)$).

Policy Gradients is a family of RL algorithms that operate by directly updating policy parameters for maximizing a performance objective, $J(\theta)$ that is defined as the expected cumulative discounted reward as shown in (10). This is achieved by repeatedly updating the policy parameters in the direction of the gradient of performance objective, $\nabla_\theta J(\theta)$. Gradient of

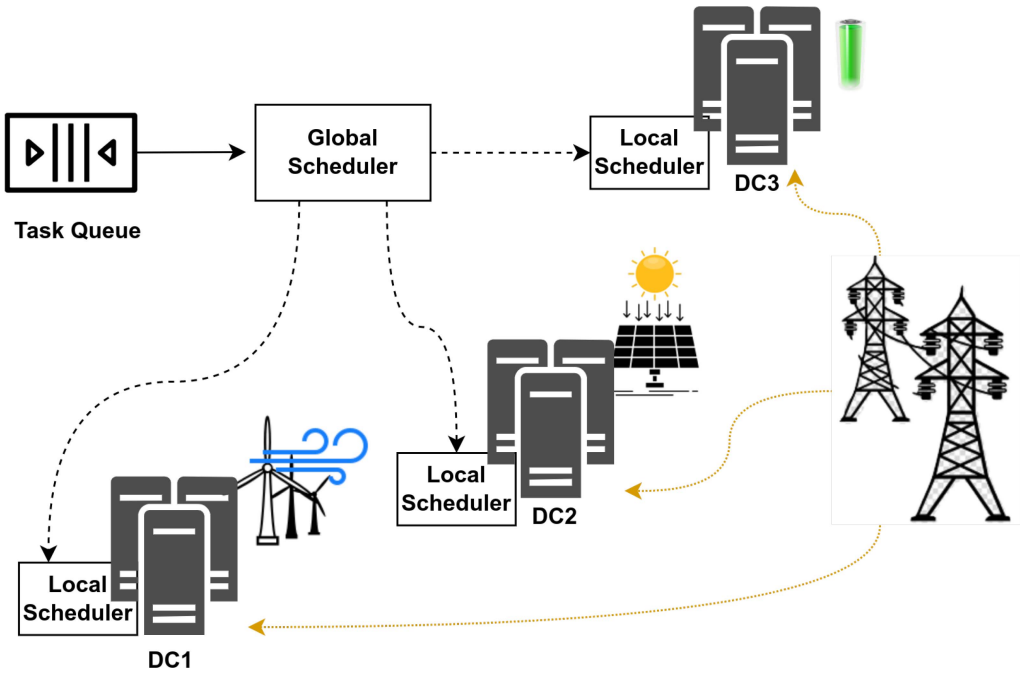


Fig. 1. High-level overview of hierarchical workflow scheduling in multi-cloud environments.

performance objective, $J(\theta)$ can be expressed as in (11).

$$J(\theta) = E_{\pi_\theta}\left[\sum_{t=1}^{\infty} \gamma^t r_t\right] \tag{10}$$

where γ is a discounting factor that is used for discounting future rewards, and $\gamma \in (0, 1)$.

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)\right] \tag{11}$$

where $Q^{\pi_\theta}(s, a)$ is the state-action value function (Q function) which indicates the desirability of an action, a in a state, s with a policy π_θ. Different policy gradient algorithms use different techniques for estimating the Q function. For the scheduling problem addressed in this work, we use the Actor-Critic technique in a multi-agent scenario as described in the next section.

B. Proposed Multi-Agent Actor-Critic Scheduling Framework

In actor-critic methods, the actor takes the current state of the environment as input and outputs a probability distribution over the actions that can be taken from this state. It does so by directly learning the policy function $\pi_\theta(a_t|s_t)$. Critic estimates the Q value $Q^{\pi_\theta}(s, a)$ based on the reward received for the action by the actor and the next state of the system. It then computes the Temporal Difference (TD) error which is used for updating the policy parameters of the actor-network in the direction of improvements, and for updating the network parameters of the critic so it can predict the Q function more accurately.

We propose a hierarchical scheduling framework for the workflow scheduling problem in which a global scheduler assigns workflow tasks to datacenters and the local scheduler in each datacenter assigns the tasks to the physical machines. Fig. 1 shows a high-level overview of the proposed framework. A single-agent DRL framework is inappropriate for the aforementioned scenario owing to its inherently distributed nature. Therefore, in this work, we propose a multi-agent DRL framework in which one global agent acts as the global scheduler and multiple local agents act as local schedulers.

In comparison to single-agent problems, multi-agent problems are much more complex since the actions of other agents cause the environment to be non-stationary. Therefore, the changes in the environment observed by an agent are not solely due to its own actions, but also due to the actions of other agents on the environment. Furthermore, due to the distributed nature of the multi-cloud environment, it is impractical to assume each agent has complete information about the real-time status of the entire environment. Partially Observable Markov Decision Process (POMDP) provides the flexibility of modelling the RL environment without requiring the agents to directly observe the actual state of the environment. Rather, what the agent receives is an observation which is in fact a belief over the environment's actual state. In order to model the multi-agent DRL framework proposed in this work, we use Partially Observable Markov games [32].

A Markov game can be defined by a 7-tuple (N, S, ϕ, $\{A_i\}$, P, $\{O_i\}$, $\{R_i\}$):

- N: A finite set of agents
- S: A finite set of states
- ϕ: Initial state distribution
- A_i: A finite set of actions available to agent i
- P: State transition function which determines the probability of joint action $A_1 \times A_2 \times \cdots \times A_i$ in state S_t leading to a transition to state S_{t+1}. The actions of each agent are governed by a policy π_{θ_i}
- O_i: A finite set of observations of agent i
- R_i: S $\times$ $A_i \rightarrow \mathbb{R}$ is the reward function of agent i

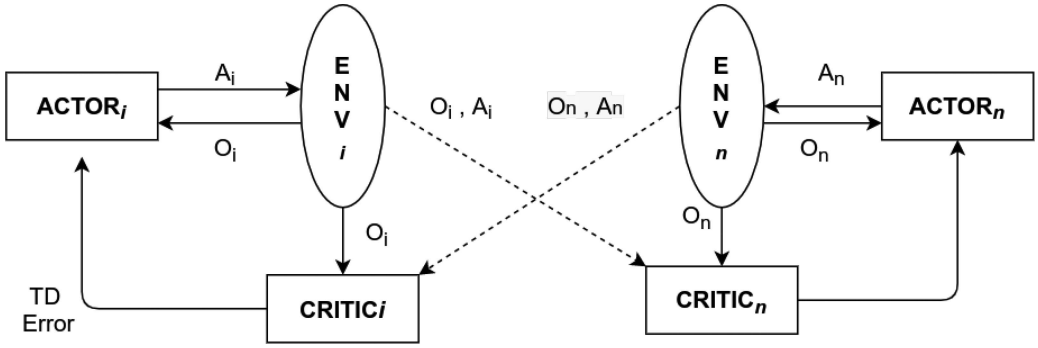


Fig. 2. Multi-agent actor-critic architecture in which every critic is augmented with actions and observations of other agents.

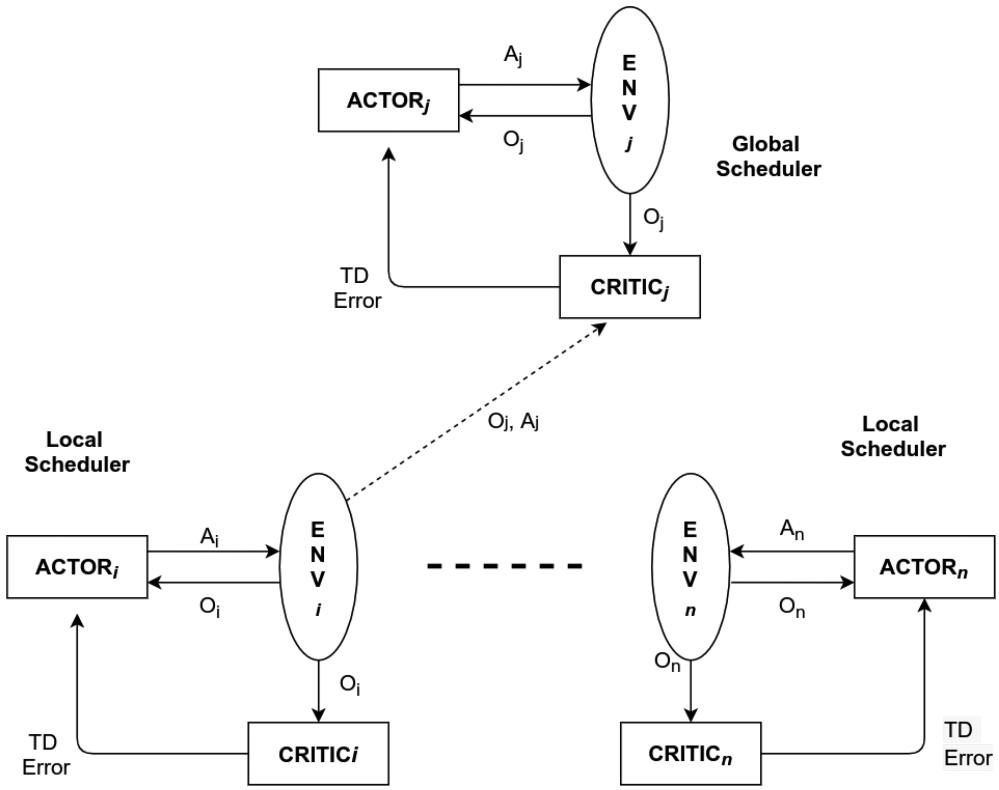


Fig. 3. Proposed multi-agent actor-critic architecture where shared actions and observations are limited to a local neighborhood.

High variance in gradient estimates is an inevitable weakness of naive policy gradients. This effect is intensified in multi-agent scenarios since the reward received by an agent may not be solely due to its own actions but also the actions of other agents [8]. Therefore, in such a setting, ignoring the impact of the actions of other agents, and conditioning the rewards only on agents' own actions inevitably leads to high variability which in turn results in gradient updates with high variance. Since naive policy gradients are not capable of handling multi-agent problems efficiently, we adapt the decentralized actor and centralized critic technique proposed in [8] for the multi-agent setting in our problem. This method focuses on providing the critic with extra information, at training time, about the policies of other agents. Accordingly, in a Markov game with N agents, the gradient of the performance objective of agent i, $J(\theta)$ in (11) can be expressed as follows:

$$\nabla_{\theta_i} J(\theta_i) = E_{\pi_{\theta_i}} \left[\nabla_{\theta_i} \log \pi_{\theta_i}(a_i|O_i) Q^{\pi_{\theta_i}}(x, a_1, a_2, \ldots, a_N) \right] \tag{12}$$

In the above equation, $Q^{\pi_{\theta_i}}(x, a_1, a_2, \ldots, a_N)$ is the Q function estimated by the critic. Note that different from the Q function of the naive actor-critic technique which takes as inputs the state of the agent and the action, in this case along with the state of the environment (x) the critic takes as input the actions $(a_1, a_2, \ldots, a_N)$ of all agents as well. The state of the environment, x may include the observations of all the agents and any additional state information. With this approach, each agent could have different reward structures since Q functions are learned separately.

Algorithm 1: Actor-Critic Based Training Process of Global Scheduler.

Initialize actor network $\pi_\theta(a|s)$ and critic network $Q_\omega(s,a)$ with random weights
for episode $=1$ to N **do**
  Reset the environment
  Input the initial state of the environment to actor network $\pi_\theta(a|s)$
  for step $=1$ to T **do**
    Select action a_{global} from the actor-network based on the current policy $\pi_\theta(a|s)$, and observe the corresponding reward R_{global}
    $R_{local}, a_{local} =$ SendTaskToSelectedLocalScheduler(t_j)
    $R_t = R_{global} + R_{local}$
    $a_t = a_{global} + a_{local}$
    Update network parameters of critic
    Update network parameters of actor
return

Algorithm 2: Actor-Critic Based Training Process of Local Schedulers.

Initialize actor network $\pi_\theta(a|s)$ and critic network $Q_\omega(s,a)$ with random weights
Reset the environment
Input the initial state of the environment to actor network $\pi_\theta(a|s)$
for every t_j assigned by global scheduler **do**
  Select action a_t from the actor-network based on the current policy $\pi_\theta(a|s)$
  Execute action a_t and observe the corresponding reward R_t and next state of the system s_{t+1}
  Update network parameters of critic
  Update network parameters of actor
  SendRewardAndActionToGlobalScheduler(a_t, R_t)
return

Fig. 2 shows the aforementioned multi-agent coordination technique. One obvious drawback of this method is the input space of the Q function increases with the increasing number of agents. To overcome this problem, we leverage the characteristics of the hierarchical scheduler design proposed in this work. Accordingly, we provide the Q function of each local agent only its observation and action since the actions and observations of other local agents have no impact on the reward it receives or the next observation. To the Q function of the global agent, at each scheduling step, we only provide the action of the local agent to which it assigns the current task in addition to its own action and observation. Fig. 3 shows the proposed multi-agent coordination technique that restricts communications to a local neighborhood. Limiting the shared experiences to a local neighborhood in this manner allows us to prevent the expansion of the input space of all agents substantially. Where the dimensions of state and action spaces of the local agents are different, a state encoding technique [33] can be used to extract a representation with a fixed dimensionality for providing as an input to the state information of the global agent.

Global Scheduler. A workflow consists of multiple containerized tasks the execution of which is constrained by precedence relations. Therefore, when a workflow is submitted, all the tasks in it cannot be scheduled for execution at once. The global scheduler identifies the tasks that can be executed directly, and the rest of the tasks will be pending execution until the tasks that they have precedence relations with complete execution. Upon the receipt of a task completion notification, the global scheduler executes the DRL agent multiple times with each of the tasks that can now be scheduled for execution due to their precedence relations being satisfied. The state of the global agent comprises of the following:

- An array $(P_{dc_1}, P_{dc_2}, ..P_{dc_i})$ where each element P_{dc_i} represents green energy surplus or deficit levels of the ith datacenter.
- An array $(F_{dc_1}, F_{dc_2}, ..F_{dc_i})$ where each element F_{dc_i} represents the average processing speed of a server in the ith datacenter.
- An array $(U_{dc_1}, U_{dc_2}, ..U_{dc_i})$ where each element U_{dc_i} represents the current utilization level of the ith datacenter.
- CPU requirement of jth task, t_j^{cpu}
- Memory requirement of jth task, t_j^{mem}

Action corresponds to the selection of a datacenter (hence a local scheduler) to which the task will be submitted for execution. It can be represented as follows:

$$A = \{dc_i | dc_i \in \{dc_1, dc_2, .., dc_n\}\} \tag{13}$$

The reward, R_{global} comprises of two components; The first component corresponds to the current green energy deficit or surplus of the selected datacenter and the second component corresponds to the reward received by the local scheduler that allocated the task to a node for execution. Incorporation of a component that reflects the desirability of the local agent's action in the global agent's reward in this manner enhances the learning process of the global agent. Reward can be represented as follows:

$$R_{global} = \left[P_{dc_i}^{green} - P_{dc_i}^{total}\right] + R_{local} \tag{14}$$

Algorithm 1 summarizes the steps included in the training process of the global scheduler.

Local Scheduler. At each cloud datacenter, the tasks that are submitted to it by the global scheduler are immediately scheduled for execution. If the DC has no free capacity for task execution, then the global scheduler is notified. The allocation of a task to a node is an action performed by the local DRL agent, based on the characteristics of the task and the status of servers in the datacenter provided to it through state information. Upon the allocation of a task to a server, an immediate reward is received by the local agent which reflects the success of the allocation with respect to the objective, which in this case is energy efficiency and time minimization. The global scheduler is notified upon the completion of task execution. Since the reward received by the local agent also contributes to the global agent's reward, it is communicated along with the results of execution

TABLE I
COMPARISON OF RELEVANT LITERATURE WITH PROPOSED WORK

Work	Application Model		Multi-Cloud	Algorithm	Renewable Energy Aware	Objectives
	Workflow	BoT/Job				
[16] Z. Wen et al. (2020)	✓		✓	Genetic Algorithm	✓	energy, cost
[12] S. Iturriaga et al. (2016)	✓		✓	Genetic Algorithm	✓	energy, makespan, deadline
[13] L. Xiaoping et al. (2020)	✓		✓	Heuristic		electricity cost
[14] W. Zhenyu et al. (2016)	✓		✓	Heuristic		cost, reliability
[16] W. Zhenyu et al. (2016)	✓		✓	Genetic Algorithm		cost, reliability
[17] C. Gu et al. (2015)		✓	✓	Integer Linear Programming	✓	carbon emissions
[18] Y. Haitao et al. (2020)		✓	✓	Bees Algorithm	✓	energy
[15] Z. Wen et al. (2016)		✓	✓	Genetic Algorithm, Heuristic	✓	cost, failure
[19] D. Cheng et al. (2020)		✓	✓	Nonlinear programming, Q-Learning	✓	system goodput
[23] J. Zhao et. al (2021)		✓	✓	PPO	✓	brown energy, deadline
[4] C. Xu et. al (2018)		✓	✓	Deep Q Learning	✓	energy
[20] N. Liu et. al (2017)		✓		Deep Q Learning		energy, latency
[21] D. Ding et. al (2020)		✓		Q Learning		energy
[22] D. Cui et. al (2017)		✓		Q Learning		makespan
[25] A. Nascimento et. al (2019)	✓			Q Learning		makespan
[26] Z. Tong et. al (2020)	✓			Deep Q Learning		makespan
Proposed	✓		✓	Actor-Critic	✓	energy, makespan

to the global scheduler. The state of the local agent comprises of the processing power and utilization levels of the servers and task size.

- An array $(F_1, F_2, ..F_{\lambda_i})$ where each element F_i represents the processing speed of λ_ith server in datacenter, dc_i.
- An array $(U_1, U_2, ..U_{\lambda_i})$ where each element U_{λ_i} represents the current utilization level of the λ_ith server in datacenter, dc_i.
- CPU requirement of jth task, t_j^{cpu}
- Memory requirement of jth task, t_j^{mem}

Action is the selection of a server, m_i in which the task will be executed, and is represented as follows.

$$A = \{m_i | m_i \in \{m_1, m_2, .., m_{\lambda_i}\}\} \tag{15}$$

The reward, R_{local} indicated below is a weighted function of the execution time of the task and corresponding energy consumption as indicated in (6) and (9), respectively.

$$R_{local} = \alpha T_{t_j} + (1 - \alpha) ET_{t_j} \tag{16}$$

where, α is a co-efficient that can be adapted to suit system preferences. Algorithm 2 summarizes the steps included in the training process of the local schedulers. Different from traditional algorithms, reinforcement learning based algorithms have two phases: an online execution phase and an offline training phase. During execution it only takes few milliseconds for the trained model to produce an action, and the impact of problem size (no. of data centers, no. of nodes) on computation time is insignificant. However, the size of the problem has a direct impact on training process of DRL agents (summarised in Algorithms 1 and 2) and large state and action spaces can lead to the problem of 'curse of dimensionality'. As the number of datacenters and servers increase, the size of state and action spaces of the DRL model increase and along with that the training time increase. This is because, with increasing problem size and complexity, more layers and more neurons per layer are needed in the underlying neural networks. Also, more episodes of training are required for the agent to learn the problem. The time complexity of Algorithms 1 and 2 can be approximated to the time complexity of training the underlying neural networks [34]. If the neural networks have L layers and u_l neurons in layer l, then the time complexity of Algorithms 1 and 2 can be specified as $O(NT\Sigma_{l=0}^{L-1} u_l u_{l+1})$, where N is the total number of episodes and T is the number of steps in each episode [34].

V. PERFORMANCE EVALUATION

A. Experimental Setup

The proposed multi-agent deep reinforcement learning-oriented workflow scheduling framework was tested using an extension of the CloudSim simulation toolkit [36]. For deep learning-related implementation, Keras library was integrated with the simulation environment [37]. For the simulation environment, we used 10 datacenters each with 25 heterogeneous servers. The resource specifications of the servers including processing power, memory, power consumption etc. are derived from the popular SPECpower_ssjÂ 2008 benchmark suite [35]. Table II indicates the specifications of the servers used in the experiments. Renewable energy data for the simulations were obtained from the actual solar energy generated by multiple photovoltaic (PV) sites installed at the Gatton campus of the University of Queensland [38].

B. Dataset

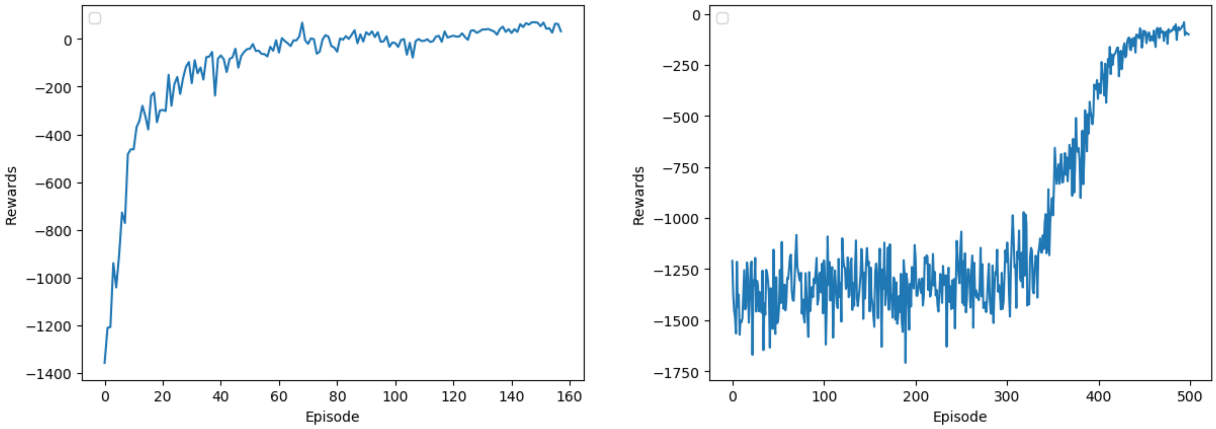
For evaluation of the proposed algorithm, a dataset comprising of 1000 workflows was derived from the synthetic workflow structures provided by the popular Peagusus workflow framework [39]. The size of data dependencies among tasks and the sizes of tasks were randomly selected from the ranges 0.1 k to 10 k and 0.5 k to 1000 k respectively. A Poisson distribution was used for modeling workflow arrival times since it is shown to be an effective way of modeling datacenter job arrivals [40].

C. Comparison Algorithms

For evaluating the performance of the proposed multi-agent framework, three comparison algorithms were used. *Random* algorithm is a baseline that allocates workflow tasks to datacenters in a random manner. In the datacenters, the selection of hosts for task execution is also performed randomly without any consideration on the impact of such allocations on execution time or energy consumption. This algorithm is used for comparisons since it provides a baseline to compare the

TABLE II
HOST CONFIGURATIONS DERIVED FROM SPEC BENCHMARK [35] FOR EXPERIMENTAL SETUP

Server Name	Processor	Cores	MIPS	RAM (GB)	Bandwidth (GB/s)	Power (Watts)	
						Idle	Active
Dell Inc. PowerEdge R740	Intel Xeon Platinum 8280 2.70 GHz	56	604.8k	64	1.5	50	432
PowerEdge C6100	Intel Xeon X5675 3.06 GHz	48	587.52k	64	1.5	227	895
Dell PowerEdge R820	Intel Xeon E5-4650L 2.60 GHz	32	332.8k	64	1	110	446
IBM System x iDataPlex dx360 M2	Intel Xeon X5570 2.933 GHz	16	187.712k	48	1	116	475
Dell PowerEdge R710	Intel Xenon 5675 3.06 GHz	12	146.88k	64	1	62	227



(a) Reward convergence with proposed DRL framework

(b) Reward convergence with a generic multi-agent DRL technique

Fig. 4. Comparison of learning efficiency of the proposed framework and a generic algorithm (as evidenced through the number of episodes required for reward convergence).

improvements achievable with the DRL method proposed in this work. *Green-Opt* is an algorithm that operates with the objective of minimizing brown energy usage by allocating workflow tasks to the datacenters with the highest accumulated green energy. The algorithm is designed to favour task allocations to 'active' hosts with sufficient capacity for executing new tasks. The selection of active hosts rather than idle ones leads to higher energy savings which are reflected in the results of the experiments in the next section. This algorithm is used for comparisons since it provides a means to evaluate the improvements achievable with the proposed DRL method in comparison to a heuristic that is specifically aimed at brown energy minimization. The third comparison algorithm *Common-Actor* is a DRL-based multi-agent actor-critic algorithm. As the algorithm name implies, multiple actor networks are guided by a common critic. This algorithm provides a way of evaluating the manner in which the novel DRL method proposed in this work outperforms a generic multi-agent DRL technique with respect to relevant objectives.

D. Experimental Results

The number of episodes required for a reinforcement learning algorithm to converge directly impacts the training time of the model. In order to ensure that the learned model remains up to date with the highly dynamic conditions in multi-cloud environments, it may be required to train and re-train the reinforcement learning agents incorporated in the resource schedulers. Therefore, convergence speed is an important factor that should be taken into account when designing reinforcement learning-oriented cloud resource schedulers. As evidenced through Fig. 4, the local neighborhood-based multi-agent method proposed in this work is capable of achieving a significantly improved convergence speed in comparison to generic reinforcement algorithms. The generic multi-agent DRL technique has required 500 episodes of training for convergence, whereas the proposed model has converged in less than 100 episodes, this proves that the learning efficiency of the local neighborhood-based approach is more than five times better than that of the generic approach.

The performance of the algorithms was evaluated in three different experimental workload settings with respect to total energy consumption incurred during workflow executions, the total energy surplus of all the datacenters post to the execution of workflows and the total time taken for workflow executions. We normalised results prior to performing the numerical comparison based performance evaluation, and the comparisons are presented in Figs. 5, 6, and 7.

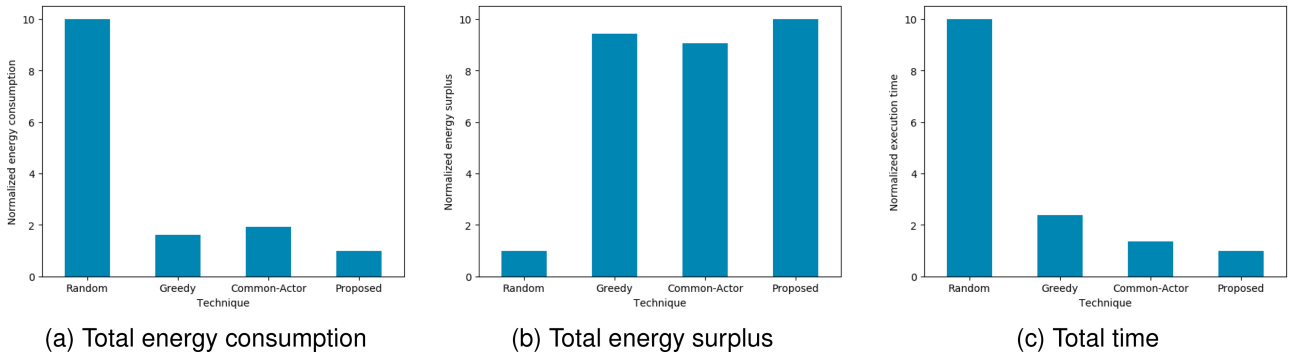


Fig. 5. Comparison of performance of scheduling algorithms on an experimental dataset derived from the synthetic workflow structures provided by the popular Peagusus workflow framework [39].

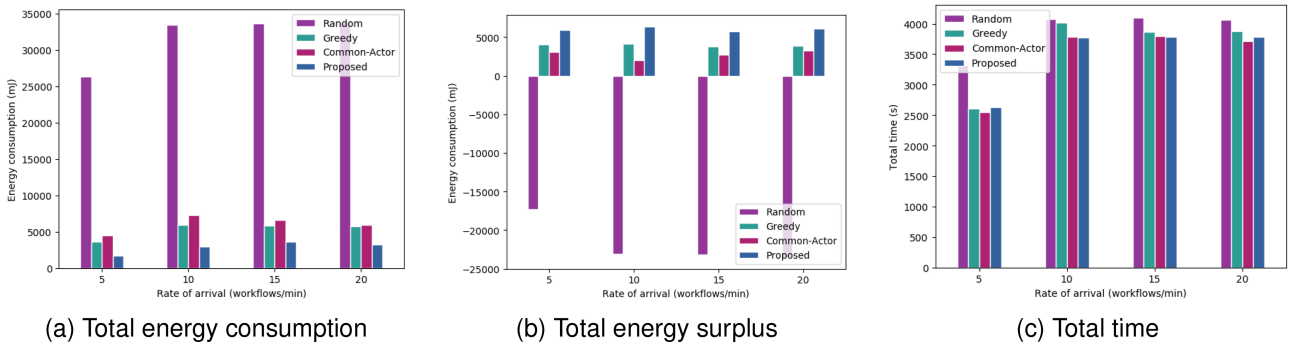


Fig. 6. Comparison of performance of scheduling algorithms as workflow arrival rate varies.

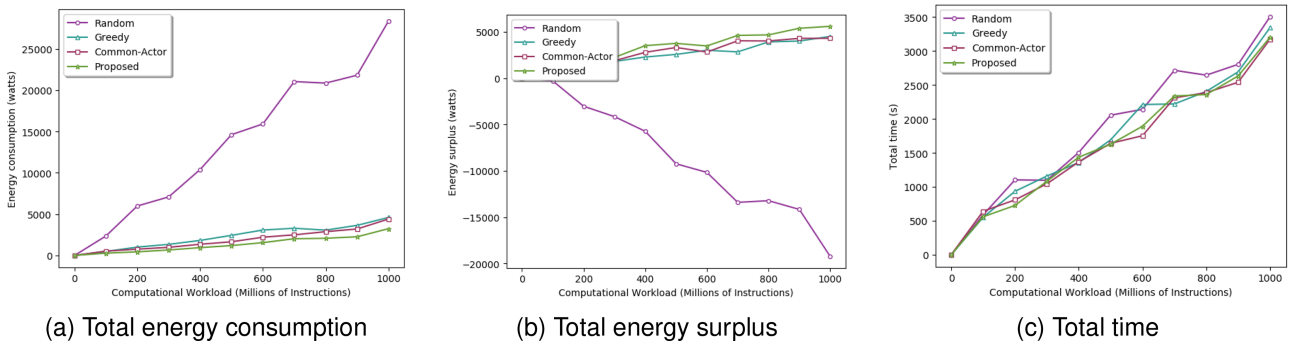


Fig. 7. Comparison of performance of scheduling algorithms as the size of computational workload varies.

Graphs in Fig. 5 depict the performance of the algorithms on the experimental dataset. The Random algorithm has consumed the highest amount of energy since it completely disregards the impact of allocations on energy-efficiency as well as time. Accordingly, as demonstrated in Fig. 5(b) and (c), workflows scheduled with Random algorithm results in the lowest energy surplus and consume the longest execution duration, respectively. In comparison to the Random algorithm, the Green-Opt algorithm has produced much better results in energy consumption and surplus. This is the expected behavior since it operates with the greedy objective of minimizing brown energy consumption. And also, the selection of active hosts over idle ones further contributes to improving energy-efficiency. Common Actor and Proposed Algorithms both use the same reward structures and as previously mentioned the difference between the two is that in the common actor method, one critic network is used to guide multiple actor networks. As evident through the results, the performance of the proposed algorithm is better than all comparison algorithms with respect to all three metrics. This is because the proposed multi-agent reinforcement algorithm is capable of finding the most efficient balance between energy consumption and execution time during the training process.

Graphs in Fig. 6 demonstrate the performance of the algorithms as the workflow arrival rate varies. The Random algorithm has again failed to deliver favorable results signalling the importance of more fine-tuned algorithms for scheduling workflows in cloud computing environments. It is clearly evident that the proposed algorithm significantly outperforms all the other algorithms at all arrival rates with respect to energy-efficiency while also maintaining comparable performance with respect to total execution time. This indicates the fact that the learned model is capable of adapting the scheduling decisions to perform equally well under highly dynamic conditions. The difference in performance between the common actor and the proposed method clearly highlights the fact that the multi-agent coordination achieved through the proposed local neighborhood-based technique leads to better learning which in turn leads to better performance.

Fig. 7 depicts the performance of the algorithms as the computational workload varies. In the Random algorithm, very significant degradation in energy-efficiency is observable at high workloads. All the other algorithms are better capable of maintaining energy consumption levels at considerably moderate levels despite the increasing workload. The proposed algorithm has yet again managed to outperform all the other algorithms with respect to energy consumption as well as energy surplus, while performing equally well with respect to time total execution time.

VI. Conclusions and Future Work

We proposed a hierarchical multi-agent scheduling framework for scheduling workflows across geo-distributed cloud datacenters with the objectives of minimizing brown energy usage, while also keeping execution times in par with comparison algorithms. The agent environment is modelled as a POMDP, and the paradigm of centralized training and distributed execution is adopted by allowing the agents to share extra information during training and operating solely based on local information during execution. Furthermore, a novel approach for limiting observation sharing to a local neighborhood is presented for overcoming the curse of dimensionality and thereby improving training efficiency. As evidenced through the empirical results, the incorporation of domain-specific characteristics for designing the multi-agent coordination in a local neighborhood-oriented manner reduced training time by 5 times compared to a generic multi-agent technique. The results also clearly demonstrated that the proposed algorithm outperformed the generic DRL algorithm with respect to minimizing total energy consumption by 47%, while outperforming the baseline algorithms with even larger margins.

As part of future work, we intend to integrate the proposed scheduling algorithm into an open-source workflow engine and thereafter apply the proposed algorithm to a real multi-cloud environment. Furthermore, the characteristics of the proposed method such as decentralized execution and ability to efficiently operate in partially observable environments makes it a viable candidate for handling the complexities associated with edge computing environments. Therefore, it is worth exploring how the proposed approach can be adapted in workflow schedulers designed to operate across edge computing infrastructures.

References

[1] M. Avgerinou, P. Bertoldi, and L. Castellazzi, "Trends in data centre energy consumption under the european code of conduct for data centre energy efficiency," *Energies*, vol. 10, no. 10, 2017, Art. no. 1470.

[2] V. Mnih et al., "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[3] W. Guo, W. Tian, Y. Ye, L. Xu, and K. Wu, "Cloud resource scheduling with deep reinforcement learning and imitation learning," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3576–3586, Mar. 2021.

[4] C. Xu, K. Wang, P. Li, R. Xia, S. Guo, and M. Guo, "Renewable energy-aware big data analytics in geo-distributed data centers with reinforcement learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 205–215, First Quarter, 2020.

[5] M. Goudarzi, M. Palaniswami, and R. Buyya, "A distributed deep reinforcement learning technique for application placement in edge and fog computing environments," *IEEE Trans. Mobile Comput.*, vol. 22, no. 5, pp. 2491–2505, May 2023.

[6] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," in *Innovations in Multi-Agent Systems and Applications-1*. Berlin, Germany: Springer, 2010, pp. 183–221.

[7] L. Canese et al., "Multi-agent reinforcement learning: A review of challenges and applications," *Appl. Sci.*, vol. 11, no. 11, 2021, Art. no. 4948.

[8] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6382–6393.

[9] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 2145–2153.

[10] A. Jayanetti and R. Buyya, "J-OPT: A joint host and network optimization algorithm for energy-efficient workflow scheduling in cloud data centers," in *Proc. IEEE/ACM 12th Int. Conf. Utility Cloud Comput.*, 2019, pp. 199–208.

[11] G. L. Stavrinides and H. D. Karatza, "An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFs and approximate computations," *Future Gener. Comput. Syst.*, vol. 96, pp. 216–226, 2019.

[12] S. Iturriaga, S. Nesmachnow, A. Tchernykh, and B. Dorronsoro, "Multiobjective workflow scheduling in a federation of heterogeneous green-powered data centers," in *Proc. IEEE/ACM 16th Int. Symp. Cluster Cloud Grid Comput.*, 2016, pp. 596–599.

[13] X. Li, W. Yu, R. Ruiz, and J. Zhu, "Energy-aware cloud workflow applications scheduling with geo-distributed data," *IEEE Trans. Services Comput.*, vol. 15, no. 2, pp. 891–903, Mar./Apr. 2022.

[14] Z. Wen, J. Cała, P. Watson, and A. Romanovsky, "Cost effective, reliable and secure workflow deployment over federated clouds," *IEEE Trans. Services Comput.*, vol. 10, no. 6, pp. 929–941, Nov./Dec. 2017.

[15] Z. Wen, R. Qasha, Z. Li, R. Ranjan, P. Watson, and A. Romanovsky, "Dynamically partitioning workflow over federated clouds for optimising the monetary cost and handling run-time failures," *IEEE Trans. Cloud Comput.*, vol. 8, no. 4, pp. 1093–1107, Fourth Quarter 2020.

[16] Z. Wen et al., "Running industrial workflow applications in a software-defined multicloud environment using green energy aware scheduling algorithm," *IEEE Trans. Ind. Inform.*, vol. 17, no. 8, pp. 5645–5656, Aug. 2021.

[17] C. Gu, C. Liu, J. Zhang, H. Huang, and X. Jia, "Green scheduling for cloud data centers using renewable resources," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2015, pp. 354–359.

[18] H. Yuan, M. Zhou, Q. Liu, and A. Abusorrah, "Fine-grained resource provisioning and task scheduling for heterogeneous applications in distributed green clouds," *IEEE/CAA J. Automatica Sinica*, vol. 7, no. 5, pp. 1380–1393, Sep. 2020.

[19] D. Cheng, X. Zhou, Z. Ding, Y. Wang, and M. Ji, "Heterogeneity aware workload management in distributed sustainable datacenters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 2, pp. 375–387, Feb. 2019.

[20] N. Liu et al., "A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 372–382.

[21] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng, "Q-learning based dynamic task scheduling for energy-efficient cloud computing," *Future Gener. Comput. Syst.*, vol. 108, pp. 361–371, 2020.

[22] D. Cui, Z. Peng, J. Xiong, B. Xu, and W. Lin, "A reinforcement learning-based mixed job scheduler scheme for grid or IaaS cloud," *IEEE Trans. Cloud Comput.*, vol. 8, no. 4, pp. 1030–1039, Fourth Quarter 2020.
[23] J. Zhao, M. A. Rodríguez, and R. Buyya, "A deep reinforcement learning approach to resource management in hybrid clouds harnessing renewable energy and task scheduling," in *Proc. IEEE 14th Int. Conf. Cloud Comput.*, 2021, pp. 240–249.
[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv: 1707.06347*.
[25] A. Nascimento, V. Olimpio, V. Silva, A. Paes, and D. de Oliveira, "A reinforcement learning scheduling strategy for parallel cloud-based workflows," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops*, 2019, pp. 817–824.
[26] 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.*, vol. 512, pp. 1170–1191, 2020.
[27] F. Xue, Q. Hai, T. Dong, Z. Cui, and Y. Gong, "A deep reinforcement learning based hybrid algorithm for efficient resource scheduling in edge computing environment," *Inf. Sci.*, vol. 608, pp. 362–374, 2022.
[28] Y. Zhang, Z. Zhou, Z. Shi, L. Meng, and Z. Zhang, "Online scheduling optimization for DAG-based requests through reinforcement learning in collaboration edge networks," *IEEE Access*, vol. 8, pp. 72985–72996, 2020.
[29] Y. Zhang, R. Li, Y. Zhao, R. Li, Y. Wang, and Z. Zhou, "Multi-agent deep reinforcement learning for online request scheduling in edge cooperation networks," *Future Gener. Comput. Syst.*, vol. 141, pp. 258–268, 2023.
[30] A. Jayanetti, S. Halgamuge, and R. Buyya, "Deep reinforcement learning for energy and time optimized scheduling of precedence-constrained tasks in edge–cloud computing environments," *Future Gener. Comput. Syst.*, vol. 137, pp. 14–30, 2022.
[31] S. Pelley, D. Meisner, T. F. Wenisch, and J. W. VanGilder, "Understanding and abstracting total data center power," in *Proc. Workshop Energy-Efficient Des.*, 2009, pp. 1–6.
[32] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proc. 11th Int. Conf. Mach. Learn.*, 1994, pp. 157–163.
[33] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016.
[34] J. Tan and W. Guan, "Resource allocation of fog radio access network based on deep reinforcement learning," *Eng. Rep.*, vol. 4, no. 5, 2022, Art. no. e12497.
[35] Standard Performance Evaluation Corporation, "Spec power," 2023. Accessed: Jan. 10, 2023. [Online]. Available: https://www.spec.org/power_ssj2008/
[36] J. Son, A. V. Dastjerdi, R. N. Calheiros, X. Ji, Y. Yoon, and R. Buyya, "CloudSimSDN: Modeling and simulation of software-defined cloud data centers," in *Proc. IEEE/ACM 15th Int. Symp. Cluster Cloud Grid Comput.*, 2015, pp. 475–484.
[37] F. Chollet et al., "Keras," 2015. [Online]. Available: https://keras.io
[38] The University of Queensland, "UQ Solar Photovoltaic Data," Accessed: Apr. 27, 2021. [Online]. Available: http://solar.uq.edu.au/user/reportPower.php
[39] L. Ramakrishnan and D. Gannon, "A survey of distributed workflow characteristics and resource requirements," Tech. Rep., Indiana University, 2008.
[40] S. Di, D. Kondo, and F. Cappello, "Characterizing and modeling cloud applications/jobs on a Google data center," *J. Supercomputing*, vol. 69, no. 1, pp. 139–160, 2014.

Amanda Jayanetti is currently working toward the PhD degree with the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Department of Computing and Information Systems, University of Melbourne, Australia. Her research interests include artificial intelligence (AI), cloud computing, and edge computing. Her current research focuses on harnessing the capabilities of Artificial Intelligence (AI) techniques for enhancing the performance of cloud and edge computing environments.

Saman Halgamuge (Fellow, IEEE) received the BSc engineering degree in electronics and telecommunication from the University of Moratuwa, Sri Lanka, and the Dipl.-Ing and PhD degrees in data engineering from the Technical University of Darmstadt, Germany. He is currently a professor with the Department of Mechanical Engineering, School of Electrical Mechanical and Infrastructure Engineering, University of Melbourne. He is listed as a top 2% most cited researcher for AI and Image Processing in the Stanford database. He was a distinguished lecturer of IEEE Computational Intelligence Society (2018-21). He supervised 50 PhD students and 16 postdocs in Australia to completion. His research is funded by Australian Research Council, National Health and Medical Research Council, US DoD Biomedical Research program and International industry. His previous leadership roles include head, School of Engineering, Australian National University and associate dean of the Engineering and IT Faculty, University of Melbourne.

Rajkumar Buyya (Fellow, IEEE) is a Redmond Barry distinguished professor and director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne, Australia. He has authored more than 625 publications and seven text books including "Mastering Cloud Computing" published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and international markets respectively. He is one of the highly cited authors in computer science and software engineering worldwide (h-index=149, g-index=322, 116,700+ citations).