modelling

MDPI

Article

Adaptive Multi-Objective Resource Allocation for Edge-Cloud Workflow Optimization Using Deep Reinforcement Learning

Husam Lahza [1], Sreenivasa B R [2,*], Hassan Fareed M. Lahza [3] and Shreyas J [4]

[1] Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 23589, Saudi Arabia; hlahza@kau.edu.sa
[2] Department of Computer Science & Design, Bapuji Institute of Engineering & Technology, Davanagere 577004, Karnataka, India
[3] Cybersecurity Department, Faculty of Computing, Umm Al-Qura University, Makkah 24382, Saudi Arabia
[4] Department of Information Technology, Manipal Institute of Technology Bengaluru, Manipal Academy of Higher Education, Manipal 576104, Karnataka, India; shreyas.j@manipal.edu
* Correspondence: sreenivasabr@bietdvg.edu

Abstract: This study investigates the transformative impact of smart intelligence, leveraging the Internet of Things and edge-cloud platforms in smart urban development. Smart urban development, by integrating diverse digital technologies, generates substantial data crucial for informed decision-making in disaster management and effective urban well-being. The edge-cloud platform, with its dynamic resource allocation, plays a crucial role in prioritizing tasks, reducing service delivery latency, and ensuring critical operations receive timely computational power, thereby improving urban services. However, the current method has struggled to meet the strict quality of service (QoS) requirements of complex workflow applications. In this study, these shortcomings in edge-cloud are addressed by introducing a multi-objective resource optimization (MORO) scheduler for diverse urban setups. This scheduler, with its emphasis on granular task prioritization and consideration of diverse makespans, costs, and energy constraints, underscores the complexity of the task and the need for a sophisticated solution. The multi-objective makespan–energy optimization is achieved by employing a deep reinforcement learning (DRL) model. The simulation results indicate consistent improvements with average makespan enhancements of 31.6% and 70.09%, average cost reductions of 62.64% and 73.24%, and average energy consumption reductions of 25.02% and 17.77%, respectively, by MORO over-reliability enhancement strategies for workflow scheduling (RESWS) and multi-objective priority workflow scheduling (MOPWS) for SIPHT workflow. Similarly, consistent improvements with average makespan enhancements of 37.98% and 74.44%, average cost reductions of 65.53% and 74.89%, and average energy consumption reductions of 29.52% and 24.73%, respectively, by MORO over RESWS and MOPWS for CyberShake workflow, highlighting the proposed model's efficiency gains. These findings substantiate the model's potential to enhance computational efficiency, reduce costs, and improve energy conservation in real-world smart urban scenarios.

Keywords: cloud; cost; energy; optimization; task execution; time; workflows

check for updates

Citation: Lahza, H.; B R, S.; Lahza, H.F.M.; J, S. Adaptive Multi-Objective Resource Allocation for Edge-Cloud Workflow Optimization Using Deep Reinforcement Learning. *Modelling* **2024**, *5*, 1298–1313. https://doi.org/10.3390/modelling5030067

Academic Editor: Paolo Bocciarelli

Received: 7 August 2024
Revised: 10 September 2024
Accepted: 12 September 2024
Published: 18 September 2024

1. Introduction

In today's urban settings, AI is swiftly becoming a major changer, revolutionizing how technology is utilized to improve city inhabitants' lives. These urban landscapes extensively use digital technology to improve infrastructure, utilities, and services. The ability to gather and evaluate enormous amounts of data produced by many networked systems and devices forms the basis of smart cities [1]. Traffic sensors, environmental monitoring systems, and security cameras produce data that help municipal administration and decision-making [2]. Advanced technologies must be used to process the data received in an urban setting to enable responsive actions and deliver insightful information [3]. Figure 1 demonstrates the use of cloud computing in creating smart city applications by showing how big workflow datasets may be managed and reviewed in real time [4]. This can be attributed to the computation's scalable and adaptive capabilities. The authors [5,6] provide a hybrid RNN and FFNN prediction model that learns both short- and long-term client behavior to solve research concerns about ignoring long-term data from multiple sessions and concentrating primarily on short-term communication in a single session, which helps to prioritize the tasks.

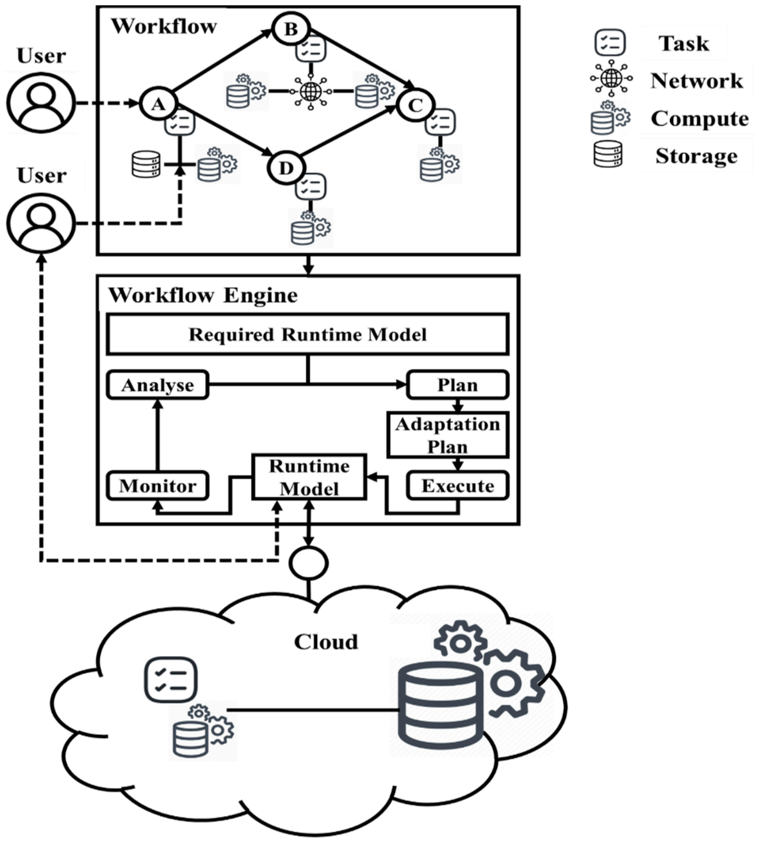

Figure 1. Traditional workflow scheduling approach in the cloud.

The effective operation of smart city surroundings depends on the given task priorities. The effective application of smart city technologies—including traffic management and quick response to environmental changes—depends on establishing priorities. Dynamic resource allocation made possible by cloud computing guarantees that the most important jobs are finished on time and with enough processing capability, therefore simplifying the work prioritizing process [7]. In smart urban environments, peripheral clouds are planned to maximize the operation of low-energy consumption, and low-cost and delay-free [8] processes. Time-sensitive procedures such as traffic optimization, emergency response systems, and catastrophe recovery must be carried out precisely and following the set schedule [9] if they are to be effectively completed. Given the significant financial commitments needed for building and maintaining a smart city, it is necessary to prioritize cost-effectiveness [10]. Since it aligns with general environmental goals [11], energy efficiency is of great relevance.

The duties that are prioritized will affect the effectiveness of smart city environments. Setting priorities is critical for ensuring that smart city technology is properly deployed, including rapid responses to environmental changes and efficient traffic management. Prioritizing jobs is facilitated by cloud computing's dynamic resource distribution, which ensures that critical tasks are completed on time and with adequate processing power [7]. Perimeter clouds are strategically deployed in smart cities to ensure that low-cost, delay-free [8] procedures are carried out as quickly as feasible. To be successful, time-sensitive procedures such as disaster recovery, traffic optimization, and emergency response systems must be completed precisely and on schedule [9]. Cost-effectiveness must be prioritized because developing and maintaining a smart city demands significant financial investments [10]. Energy efficiency is critical because it coincides with long-term environmental goals [11].

Despite the potential advantages, existing edge-cloud task scheduling offloading optimization models [12,13] often fall short in addressing these multifaceted challenges. Many models lack the granularity required for prioritizing tasks effectively, leading to suboptimal resource allocation [14,15]. Moreover, the failure to consider the varying time, cost, and energy constraints of diverse smart urban applications hinders the overall efficiency of edge-cloud-based systems. To bridge these gaps, this work proposes a model that reduces the time, cost, and energy associated with processing the tasks within a smart urban workload. By considering the unique characteristics of smart urban data and the diverse nature of tasks, the model aims to optimize edge-cloud task scheduling offloading optimization for enhanced efficiency.

The contribution of this work is as follows. The introduction of a model designed to minimize the time, cost, and energy associated with processing tasks within workloads. The model introduces a novel multi-objective task resource optimization technique to analyze data-intensive workloads with greater energy efficiency with processing time and cost. The multi-objective optimization uses a deep reinforcement learning technique where a cost-efficient approach for workflow execution in both edge servers and the cloud is performed. The proposed task resource optimization between edge and cloud servers efficiently reduces costs and provides better energy efficiency when evaluated using scientific workflows.

The manuscript is structured as follows: Section 2 provides an overview of existing works related to workflow execution, with a specific emphasis on reducing time, cost, and energy. Section 3 introduces the proposed model. Section 4 presents and evaluates the proposed model's results, highlighting its significant impact on processing time, cost, and energy. Comparative analyses with existing models emphasize the proposed model's effectiveness. Section 5 concludes the work, summarizing the findings and discussing avenues for future research.

2. Literature Survey

Our research emphasizes reducing task performance in terms of energy, money, and time. Workflow scheduling with deep reinforcement learning (DRL) [16] was developed to optimize workflow distribution, increase operational effectiveness, and ensure assignment success. Furthermore, the Deep-Q-Network (DQN) was used to solve the complicated and multivariate job scheduling problem. According to the simulations, the proposed solution outperformed the state-of-the-art in terms of service time, failed job completion, and virtual machine utilization. This DRL-based solution effectively addressed the hard problem of edge computing duty scheduling.

The authors present a hybrid metaheuristic for cloud process scheduling that minimizes makespan while accounting for virtual resource heterogeneity [17]. Hybrid TLBO combines HEFT, TLBO, OBL, and genetic modifications. One method uses HEFT to generate a diverse, high-quality beginning population. Second, mixed OBL (MOBL) models frequently use border and population history search data. Final genetic operations during the learner stage help the algorithm avoid local optima. HTLBO's performance is scientifically evaluated. The average makespan, running time, and nonparametric statistics are compared to HEFT and advanced hybrid metaheuristics. HTLBO enhances the population variance while maintaining scheduling effectiveness and efficiency, as demonstrated by significant schedule quality improvements. The authors propose policy-based reinforcement learning for hyper-heuristics [18]. Hyper-heuristic agents choose the best generalized constructive low-level heuristics for combinatorial optimization. The architecture was assessed for travelling salespersons, capacitated vehicle routing, and bin packing. This strategy outperforms meta- and hyper-heuristic-based algorithms on all significant challenges in all areas. The framework was tested for a tight deadline for hybrid cloud process scheduling cost optimization. Eight agents were trained on medium-sized processes with two deadlines and compared to meta- and hyper-heuristic techniques for smaller and larger workflows with unexpected deadlines. This study includes four workflow apps, three sizes, and three deadlines.

Jobs with unknown requirements were assigned the required resources via intelligent fuzzy scheduling [19]. Using a unique salp swarm application, the model discovered and optimized fuzzy task-resource allocation norms. The current strategy enabled for the speedier and more efficient creation of an intelligent salp-swarm-scheduler using fitness-based-quasi-reflection (ISSS-FQR). According to the trial results, ISSS-FQR is clearly more effective than other conventional approaches. They proposed a cyber-physical framework for the Internet of Things (IoT) in [20] to optimize resource utilization in fog and cloud settings while decreasing execution costs. This framework's development was aimed at the medical community. They introduced the multi-objective heuristic approach ant colony optimization (MOHACO-TS) work scheduling approach. This was accomplished in conjunction with excellent cloud work scheduling and resource optimization. With this form of job scheduling, in this study, we planned to accomplish as much as possible in the allowed time while making the most use of our resources. The researchers evaluated the IoT-HCPS idea with five datasets and cutting-edge task scheduling and categorization approaches. The IoT-HCPS study's results outperformed those of other tactics. The bi-objective workflow scheduling problem (Bi-OWSP) was first introduced by [21]. The invention of Bi-OWSP was inspired by the desire to improve workflow planning while maintaining a coherent balance. Researchers developed a strategy to increase dependability while taking energy usage into account.

This strategy presented in [22] improves dependability while reducing energy use. It chooses the best task–server combinations on purpose. The efficiency of Bi-OWSP was shown through simulation testing on both actual and simulated workflow scenarios. Based on the findings of these experiments, Bi-OWSP outperforms the current strategies for regulating energy reliability. Using a task-planning system with several objectives could save production time and energy usage [22]. A novel technique called IETIF was also proposed in [23], which combined NSGA-III with simulation-based annealing. IETIF's simulations of energy consumption and makespan outperform the current status quo. Using cloud and fog-based Internet of Things (IoT) systems, they investigated the topic of maximizing the longevity of data-intensive job scheduling [23]. Data location aware, they devised job-scheduling, an integer-linear programming heuristic. Similar to the mean, the test results indicated the usefulness of the proposed strategy. We pioneered a cloud computing workflow planning technique to keep costs low and IoT application deadlines met in [24]. Ant colony optimization (ACO) and cost-driven prediction were the foundations of the F-ACO integrated intelligence approach. They used their F-ACO approach in real-world situations. Then, they compared their strategy to other innovative approaches. Compared to other cloud computing IoT scheduling systems, their empirical results showed that F-ACO worked better. The authors of [25] described a new approach to process organization based on DRL and listed several objectives. The initial step was to use linkage to identify the most critical jobs for each workflow. Virtual machine prioritization was determined by evaluating data center power expenses. Processes were correctly assigned to virtual machines. The task scheduler uses deep Q-network to prioritize tasks and virtual machines. They compared their method to cat swarm optimization, heterogeneous earliest first deadline, and ACO. Compared to cutting-edge algorithms, theirs performed better in makespan and power consumption. Unreliable workflow execution resulted from the current method's incapacity to resolve energy minimization-makespan reduction trade-offs [26].

Table 1 provides a comparative study of various existing methods with proposed approaches. The study in the table shows that testing the model considering complex scientific applications is important. The current method leveraging deep reinforcement learning has attained good results; however, the current methods have been studied considering a homogenous cloud platform. Conversely, the proposed approach employs the DRL for resource optimization in a heterogenous platform adopting an edge-cloud computation paradigm. The following section of this paper proposes a novel strategy for workflow execution to address research difficulties.

Table 1. Comparative study.

	Computation Model	Optimization Metrics	Optimization Model	Workflow Type	Workflow Used	Result Studied
QMTSF [13]	Cloud computing	Resource allocation and efficiency	Reinforcement learning	Simple	Simulated workflow	Makespan and average task processing time
EASA [14]	5G and Cloud	Resource utilization and energy consumption	Optimal	Simple 5G traffic dataset	Smart city traffic	False discovery rate, false omission rate, prevalence threshold, and critical success index.
F-ACO [24]	Cloud computing	Execution cost and idle time	Ant colony optimization with cost-driven heuristic	Simple workflow	IoT workflow application	cost
MOPWS [25]	Cloud computing	Makespan and energy	Deep Q-Network	Complex Scientific workflow	CyberShake, Epigenomics, LIGO, and Montage	Makespan, and energy consumption
RESWS [26]	Cloud computing	Energy and Reliability	Heuristic	Scientific workflow	Three realistic workflow	Energy, deadline, and reliability
MOROWS [Proposed]	Edge-cloud	Energy and makespan	Deep reinforcement learning	Complex scientific workflow	SIPHT and CyberShake	Makespan, processing cost, and energy consumption

3. Proposed Methodology

This section introduces a novel design for a multi-objective resource optimization scheduler optimized through a deep reinforcement learning model for effectively executing complex workflow tasks into the edge-cloud server. The offloading optimization is achieved through a minimization function to reduce overall energy, time, and cost.

3.1. Architecture

In this section, we will discuss the architecture of the task offloading model in the edge-cloud for the effective execution of the smart urban scientific workflow. In Figure 2, the user initiates the smart urban workflow like SIPHT and CyberShake required for execution inside this architecture. The smart urban workflow covers a variety of actions and requires storage. In some cases, a smart urban workflow may also require network connectivity. Furthermore, smart urban workflows might be cyclic or acyclic graphs. Our study focuses on directed acyclic graphs (DAGs). In this design, the smart urban workflow is processed within the workflow engine by the cloud resource manager, which separates it for a more streamlined and efficient execution process. This partitioning consists of two main phases: the planning phase, which offers the partitioning strategy, and the execution phase, which implements the proposed partitioning approach. A thorough analysis of the workflow partitioning strategy is offered according to the QoS and SLA manager. Furthermore, the workflow engine of the cloud resource manager continuously monitors and assesses the process's progress. The workflow is then offloaded to an edge server. When an edge server cannot complete a task, it is forwarded to the cloud for processing. A thorough investigation of the entire process is conducted.

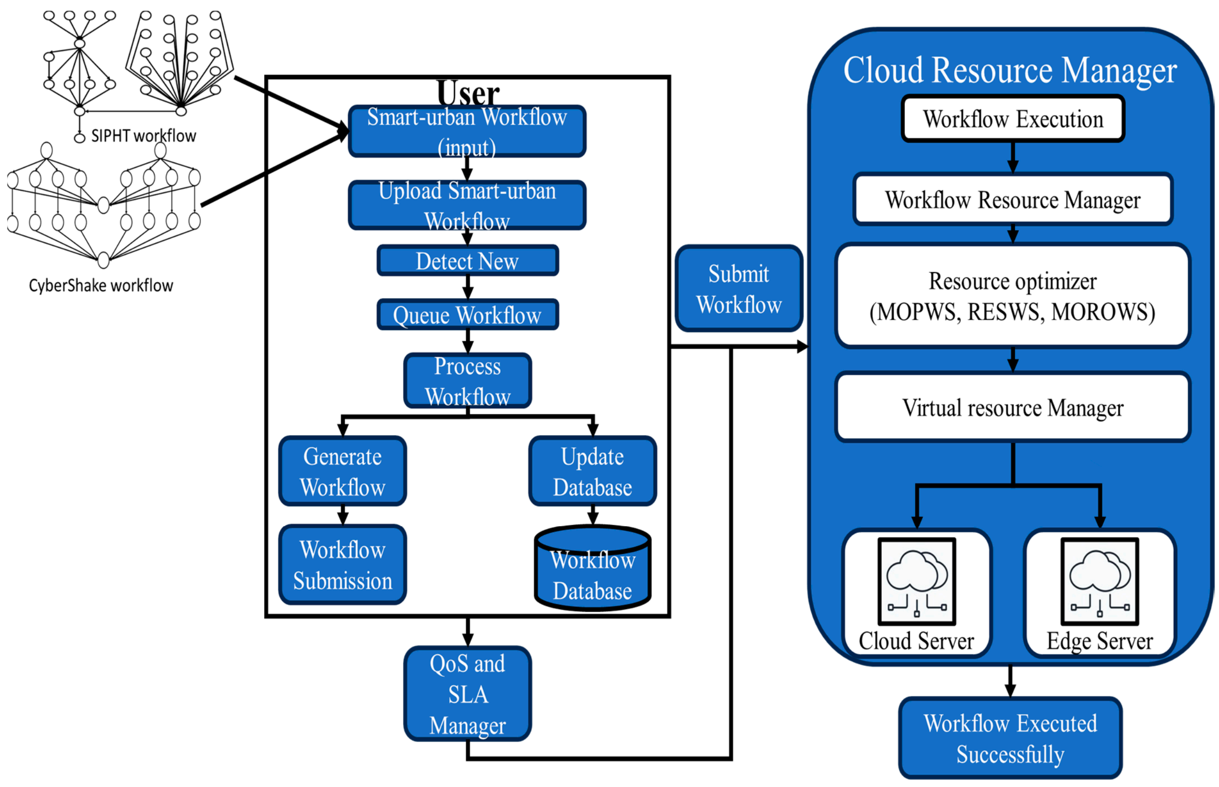

Figure 2. Effective task offloading model in edge-cloud architecture.

3.2. Multi-Objective Resource Optimization in Edge-Cloud

In this section, this work explains how to use cloud resources efficiently on an edge-cloud platform to analyze highly data-intensive workflows. To avoid bottlenecks in an edge-cloud environment while still satisfying task deadlines and execution requirements, the proposed approach is designed to schedule tasks with the least cost. In this work, an efficient task-queuing mechanism is modeled to achieve reliability with effective load-optimization. Each server during the task-queuing mechanism is heterogeneous, having its own unique set of capabilities. Consider o servers represented as $T_1, T_2, T_3, \ldots T_o$ which are heterogenous in nature. Also, consider each server having a size of $n_1, n_2, n_3, \ldots,\ n_o$ and executional capabilities of $t_1, t_2, t_3, \ldots t_o$. Further, assume a server T_j in a heterogenous environment having n_j multiple servers which have executional capabilities of t_j. Taking into account the Poisson distribution method based on an $M/M/m$ queuing mechanism, the arrival load α follows an exponential distribution having a standard deviation (s) and mean average of s as $(\bar{s})\ 1/\alpha$. The proposed approach divides the workflow tasks into o subsets, with each subset communicating its arrival load α_j towards the T_j, where j ranges from 1 to o and α represents the sum of $\alpha_1 + \alpha_2 + \alpha_3 \cdots + \alpha_o$. Moreover, T_j keeps an infinite queue of tasks in a line for execution, even when n_j is occupied. Priority is given to those who can be served first, using exponential randomization s along with the mean average $\bar{s}$ for scheduling purposes. For each T_j, the n_j is kept the same for all t_j. Hence, the duration for execution having exponential randomization s can be evaluated using Equation (1), as follows:

$$y_j = \frac{s}{t_j} \tag{1}$$

The mean average of Equation (1) is evaluated using Equation (2), as follows:

$$\bar{y}_j = \frac{\bar{s}}{t_j}. \tag{2}$$

The computation of the mean average aids in obtaining an ideal estimation in meeting workflow execution in the edge-cloud platform. In T_j, the average task execution rate can be evaluated using the following equation:

$$\beta_j = \frac{1}{\overline{y}_j}. \tag{3}$$

Further, the resources consumed by T_j whenever busy (time utilized) is evaluated using Equation (4), as follows:

$$\gamma_j = \frac{\alpha_j}{n_j\beta_j} = \frac{\alpha_j\overline{y}_j}{n_j} = \frac{\alpha_j\overline{s}}{n_j t_j}, \tag{4}$$

The estimation of resource usage during the survey aids the model in having an assumption task in the queue. The parameter n_j defines the server which is busy in T_j, $p_{j,l}$ denotes the likelihood for task l which is waiting in a queue in a given T_j, then the likelihood is evaluated using Equation (5), as follows:

$$p_{j,l} = \begin{cases} p_{j,0}\frac{(n_j\gamma_j)^l}{l!}, & l < n_j; \\ p_{j,0}\frac{n_j^{n_j}\gamma_j^l}{l!}, & l \geq n_j; \end{cases} \tag{5}$$

When $l = 0$, the following equation is obtained:

$$p_{j,0} = \left(\sum_{l=0}^{n_j-1}\frac{(n_j\gamma_j)^l}{l!} + \frac{(n_j\gamma_j)^{n_j}}{n_j!}\cdot\frac{1}{1-\gamma_j}\right)^{-1}. \tag{6}$$

Equation (6) provides the likelihood of a minimum waiting period in an edge-cloud platform considering the presence of no arrival of new workflow task l. The likelihood for each workflow task arriving one after the other in T_j whenever T_j remains busy is evaluated in Equation (7):

$$P_{r,j} = \frac{p_j, n_j}{1-\gamma_j} = p_{j,0}\frac{n_j^{n_j}}{n_j!}\cdot\frac{\gamma_j^{n_j}}{1-\gamma_j}. \tag{7}$$

Moreover, in T_j the evaluation is performed using Equation (8) for the mean workflow tasks that are waiting in a queue or being executed, as follows:

$$\overline{O}_j = \sum_{l=0}^{\infty} l p_{j,l} = n_j\gamma_j + \frac{\gamma_j}{1-\gamma_j}P_{r,j}. \tag{8}$$

This work only considers mean waiting to attain optimal measurement. Also, the mean workflow task completion duration by each T_j is evaluated using Equation (9):

$$U_j = \frac{\overline{O}_j}{\alpha_j} = \overline{y}_j + \frac{P_{r,j}}{n_j(1-\gamma_j)}\overline{y}_j = \overline{y}_j\left(1 + \frac{P_{r,j}}{n_j(1-\gamma_j)}\right) \tag{9}$$

Equation (9) is simplified by substituting Equations (2), (4), and (7). Hence, from this, Equation (10) is obtained:

$$U_j = \frac{\overline{s}}{t_j}\left(1 + p_{j,0}\frac{n_j^{n_j-1}}{n_j!}\cdot\frac{\gamma_j^{n_j}}{(1-\gamma_j)^2}\right). \tag{10}$$

Further, the required energy for executing workflow tasks is evaluated using Equation (11):

$$Q = a\mathcal{C}\mathcal{V}^2\mathcal{F} = \delta t^{\mu} \tag{11}$$

In Equation (11), a denotes the characteristic of workflow-task, $\mathcal{V}$ denotes voltage, $\mathcal{C}$ denotes load capacitance, $\mathcal{F}$ represents clock frequencies, and t represents speed of T_j processor. Further, δ in Equation (11) is evaluated using Equation (12):

$$\delta = \frac{ab^2\mathcal{C}}{c^{2\rho+1}} \tag{12}$$

In Equation (12), b and ρ are variables used for defining constants (i.e., $b, \rho \neq 0$). Also, the μ in Equation (11) is evaluated using Equation (13):

$$\mu = 2\rho + 1. \tag{13}$$

The current approaches consider both δ along with μ throughout T_j. Nonetheless, in this work, the situation varies due to the heterogeneous environment used, resulting in varying values of δ and μ. The two main energy categories in a heterogeneous environment are dynamic and static energy. When the T_j does not execute any workflow tasks, then the T_j is in a static state. Hence, energy consumption for the static state is evaluated using Equation (14):

$$Q_j = n_j\left(\gamma_j \delta_j t_j^{\mu_j} + Q_j^*\right) = \alpha_j \bar{t} \delta_j t_j^{\mu_j - 1} + n_j Q_j^*. \tag{14}$$

Further, when the T_j executes any workflow tasks, then the T_j is in a dynamic state. Hence, energy consumption for a dynamic state is evaluated using Equation (15):

$$Q_j = n_j\left(\delta_j t_j^{\mu_j} + Q_j^*\right). \tag{15}$$

The goal of this study was to allocate resources efficiently by reducing energy and computation duration when carrying out workflow tasks in a heterogeneous environment that varies computational speed and energy consumption for executing different tasks. Consider, T_j having a size of $n_1, n_2, n_3, \ldots,\ n_o$ and the execution taking place in a dynamic state having requirement $\bar{s}$ with arrival load α and load distribution defined as $\alpha_1, \alpha_2, \ldots, \alpha_o$. To achieve the best efficiency, the minimization function is defined as follows:

$$\min U(\alpha_1, \alpha_2, \ldots, \alpha_o) \tag{16}$$

A constraint is imposed on Equation (16) as follows:

$$G(\alpha_1, \alpha_2, \ldots, \alpha_o) = \alpha, \tag{17}$$

where

$$G(\alpha_1, \alpha_2, \ldots, \alpha_o) = \alpha_1 + \alpha_2 + \ldots + \alpha_o, \tag{18}$$

and $\gamma_j < 1, \forall\, 1 \leq j \leq o$. Consider T_j having a size of $n_1, n_2, n_3, \ldots,\ n_o$ and the execution taking place in a dynamic state having requirement $\bar{s}$ with arrival load α and load distribution defined as $\alpha_1, \alpha_2, \ldots, \alpha_o$. To reduce energy consumption, the minimization function is defined as follows:

$$\min Q(\alpha_1, \alpha_2, \ldots, \alpha_o) \tag{19}$$

A constraint is imposed on Equation (19) as follows:

$$G(\alpha_1, \alpha_2, \ldots, \alpha_o) = \alpha, \tag{20}$$

where

$$G(\alpha_1, \alpha_2, \ldots, \alpha_o) = \alpha_1 + \alpha_2 + \ldots + \alpha_o, \tag{21}$$

and $\gamma_j < 1, \forall\, 1 \leq j \leq o$. The cost of executing workflow tasks on T_j can be evaluated by inversing Equation (9):

$$C = \frac{1}{U_j} \tag{22}$$

Nevertheless, this work considers two states, i.e., dynamic and static, hence the energy factor Q_j has to be considered. Hence, the cost function can be redefined as Equation (23):

$$S_j = Q_j U_j. \tag{23}$$

The average cost S for different T_j, i.e., $T_1, T_2, \ldots, T_o$ can be evaluated using Equation (24):

$$S(\alpha_1, \alpha_2, \ldots, \alpha_o) = \frac{\alpha_1}{\alpha} S_1 + \frac{\alpha_2}{\alpha} S_2 + \ldots + \frac{\alpha_o}{\alpha} S_o \tag{24}$$

Equation (24) can be reformulated by substituting Equation (23) in Equation (24); from this, Equation (25) is achieved:

$$S(\alpha_1, \alpha_2, \ldots, \alpha_o) = \frac{\alpha_1}{\alpha} Q_1 U_1 + \frac{\alpha_2}{\alpha} Q_2 U_2 + \ldots + \frac{\alpha_o}{\alpha} Q_o U_o \tag{25}$$

In the proposed work, the constraints in Equations (16) and (19) were minimized using Equations (17), (18), (20), and (21) through the usage deep reinforcement learning model [25] for the efficient scheduling of workflow tasks and achieving better performance and reducing cost. Further, if the edge server fails to execute the task or more tasks are in the edge server, in this situation, the workflow will be executed at the edge/cloud computational platform and vice versa.

3.3. Algorithm

The step-by-step working of proposed multi-objective workflow scheduler is given in Algorithm 1. The line 2, the queue is initialized to null; the line 3, the smart urban workflow arrival load is initialized; in line 4, following the arrival of the new workflow, the non-executed scheduling decision is stopped. Lines 5 and 6 define how the server information is collected, and the smart urban workflow's start time is computed. In lines 7 to 9, the incoming workflow load is added to the queue. In lines 9 and 10, the task that is ready to be executed is obtained and arranged in increasing order according to its priority level with respect to its start time. In line 11, the complete workflow in $\mathcal{D}$ will be looking for resource to be scheduled. In line 12, the server is set to null and in line 13, using the deep reinforcement learning model [25], the computational node that meets the makespan and energy constraint is added into the server as in line 15. In lines 16 and 21, if the selected computational node is free, the task is executed, and if not, it is offloaded and scheduled in the cloud platform till all the tasks in $\mathcal{D}$ are executed. In line 22, finally, compute the overall makespan, energy, and cost of processing the smart urban workflow in the edge-cloud platform.

In reliability enhancement strategies for workflow scheduling (RESWS) algorithm, the model is scheduled employing a heuristic approach to optimize energy reliability constraint; thus, the optimal performance is not guaranteed considering multi-level dependencies, thus, resulting in higher makespan and energy consumption and the other existing methods namely multi-objective priority workflow scheduling (MOPWS) algorithm employs DRL to perform scheduling considering multi-level heterogeneous dependencies. The parameters, such as energy and makespan, have been optimized employing DRL; however, the model extended to a heterogeneous platform yields poor results. However, in the proposed model, the same energy makespan metrics are designed considering the heterogenous platform, and DRL is employed to select the computational node to execute between the edge and the cloud; the optimization process significantly aided in reducing energy and cost-leveraging edge-cloud paradigm; Finally, the work introduced a novel cost metrics considering both energy makespan with multi-level scheduling constraint. This shows why the proposed MORO algorithm attains much improved results than the current MOPWS and RESWS algorithms considering smart urban scientific workflows in the edge-cloud platform.

Algorithm 1. Multi-objective resource optimization workflow scheduler

```
1. Start
2. $Queue \leftarrow \varnothing$;
3. $\forall$ smart urban workflow load arrival $\alpha_j$ do
4.     Stop the non-executed scheduling decisions;
5.     Collect the information and update to available computational servers;
6.     Compute start time for every task of smart-urban workflow;
7.     Add complete task of workflow load $\alpha_j$ into $Queue$;
8.     While $Queue$ composed of unscheduled tasks do
9.         $\mathcal{D} \leftarrow$ obtain task from $Queue$ that are ready;
10.        Arrange $\mathcal{D}$ in increasing order with starting time;
11.        $\forall$ workflow task $\in \mathcal{D}$ do
12.            $\mathcal{C}Server \leftarrow \varnothing$;
13.            $\mathcal{C}Servers \leftarrow$ all servers satisfying makespan constraint of Equation (16) and energy
constraint of Equation (19) is obtained by employing deep reinforcement learning model;
14.            $\forall$ $Computaional\ node \in \mathcal{C}Servers$ do
15.                If $Computaional\ node == free$
16.                    Schedule the task in edge-platform;
17.                Else
18.                    Schedule the task in cloud-platform;
19.            End $\forall$
20.        End $\forall$
21. End $\forall$
22. Compute the overall makespan using Equation (10), the processing energy

consumption using Equation (15), and the cost using Equation (23);
23. Stop.
```

4. Results and Discussion

In the execution of this work, both the existing model MOPWS [25], RESWS [26], and the proposed model were run on an identical system configuration. The proposed work considered two latest models, namely MOPWS and RESWS, for comparison, because the MOPWS optimize both energy and makespan using a deep reinforcement learning model, and performance is studied in terms of energy and overall makespan and on the other side, the work RESWS optimized energy reliability constrained. Finally, both models considered complex DAG scientific workflows for validating the model. The proposed model, similar to [24], has additionally considered cost parameters to validate the performance of all three models. The system utilized an Intel Core i7 processor, 16 GB of RAM, a Windows 11 Operating System, and a 500 GB SSD. The edge-cloud environment was established with a CloudSimSDN simulator [27]. Both models were developed using an object-oriented programming language. Both tools play crucial roles in the simulation process, enabling the evaluation and analysis of system performance, resource utilization, and other parameters in their respective domains. The results were evaluated regarding processing time, cost, and energy. The SIPHT workflow [28,29], derived from the bioinformatics project at Harvard, serves the purpose of automating the exploration for untranslated ribonucleic acids (RNAs), small RNAs (sRNAs) associated with bacterial replicons within the NCBI database. A sample graphical representation of the SIPHT workflow is given in Figure 3. This workflow is pivotal in genomics and molecular biology, contributing to understanding genetic regulatory mechanisms in bacterial systems. An effective analysis, i.e., faster execution with minimal cost, plays an important role in effective epidemic management in urban environments. Further, the experiment is conducted on CyberShake workflow [28,30]. The CyberShake workflow is specifically designed for seismic risk assessment, employed by the Southern California Earthquake Centre, and serves as a crucial tool for characterizing earthquake hazards in each region. A sample graphical representation of the CyberShake workflow is given in Figure 4.

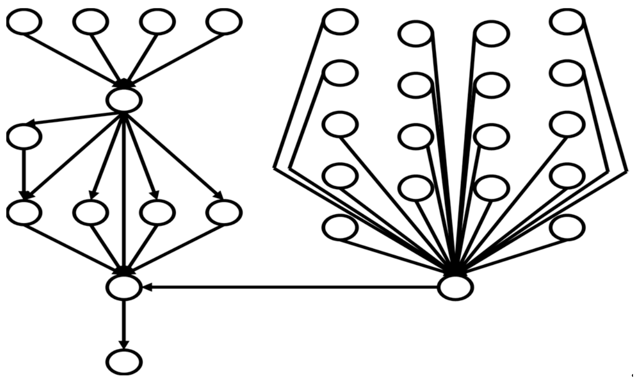

Figure 3. SIPHT workflow.

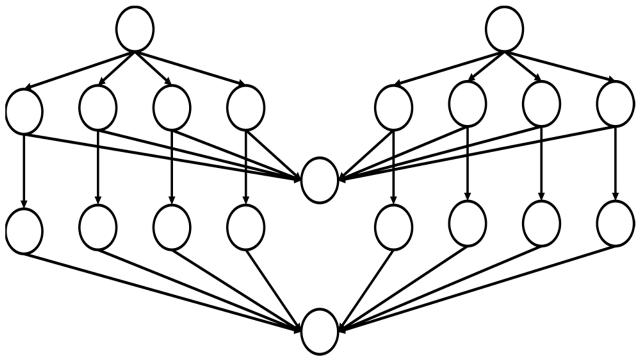

Figure 4. CyberShake workflow.

4.1. Makespan Performance

Figure 5 presents the makespan, measured in seconds, for SIPHT workflows with sizes of 30, 60, and 100 tasks for RESWS, MOPWS, and MOROWS models. The makespan of all three models is measured using Equation (10). The existing model demonstrates more makespan for different workflows than the proposed MOROWS model.

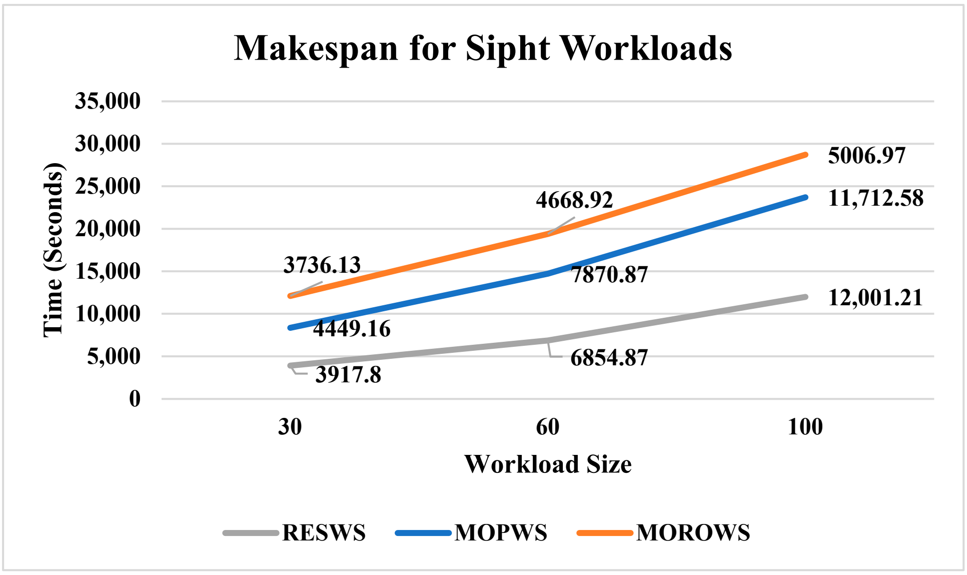

Figure 5. Makespan is required for executing SIPHT workflows using existing and proposed models.

For the SIPHT workflow, the proposed MOROWS model reduced the average makespan by 31.6% and 37.98% over RESWS and MOPWS, respectively. This reduction in makespan indicates improved efficiency and performance in the proposed model, which may have practical implications for applications requiring timely data processing within the SIPHT framework. Figure 6 presents the makespan, measured in seconds, for CyberShake workflows, with sizes of 30, 60, and 100 tasks for RESWS, MOPWS, and MOROWS models. The existing model demonstrates more makespan for different workflows than the proposed MOROWS model. For the CyberShake workflow, the proposed MOROWS model reduced the average makespan by 70.09% and 74.44% over RESWS and MOPWS, respectively. This reduction in makespan indicates improved efficiency and performance in the proposed model, which may have practical implications for applications requiring timely data processing within the CyberShake framework. The DRL-based optimization presented in MOPWS attained a slightly poorer makespan result than RESWS. However, the energy-makespan model presented in MOPWS failed to work for a heterogeneous computational platform. On the other side, the significant makespan reduction result attained by MOROWS for both SIPHT and CyberShake workflow is due to optimization performed to minimize Equation (16) keeping the constraints of Equations (17) and (18) through deep reinforcement learning model.

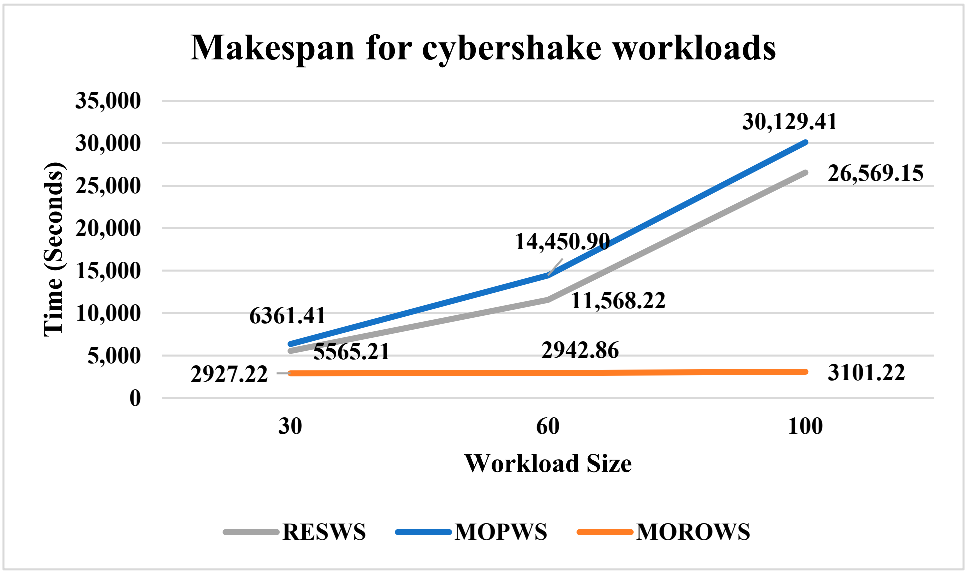

Figure 6. Makespan is required for executing CyberShake workflows using existing and proposed models.

4.2. Processing Cost

Figure 7 presents the makespan, measured in dollars (USD) using Equation (23), for SIPHT workflows, with sizes of 30, 60, and 100 tasks for RESWS, MOPWS, and MOROWS models. The cost of all three models is measured using Equation (23). The existing model demonstrates more cost for different workflows than the proposed MOROWS model. For the SIPHT workflow, the proposed MOROWS model reduced average cost by 62.64% and 65.53% over RESWS and MOPWS, respectively. Figure 8 presents the cost, measured in dollars (USD), for CyberShake workflows, with sizes of 30, 60, and 100 tasks for RESWS, MOPWS, and MOROWS models. The existing model demonstrates more cost for different workflows than the proposed MOROWS model. For the CyberShake workflow, the proposed MOROWS model reduced average cost by 73.24% and 74.89% over RESWS and MOPWS, respectively. This cost reduction indicates improved efficiency and wide applicability in realistic deployments. The DRL-based optimization presented in MOPWS attained slightly poorer cost results than RESWS. However, the energy-makespan model presented in MOPWS failed to work for a heterogeneous computational platform. On the other side, the significant makespan reduction result attained by MOROWS for both SIPHT and CyberShake workflow is due to optimization performed to minimize Equations (16) and (19) together, keeping constraints of Equations (17), (18), (20), and (21) through the deep reinforcement learning model.

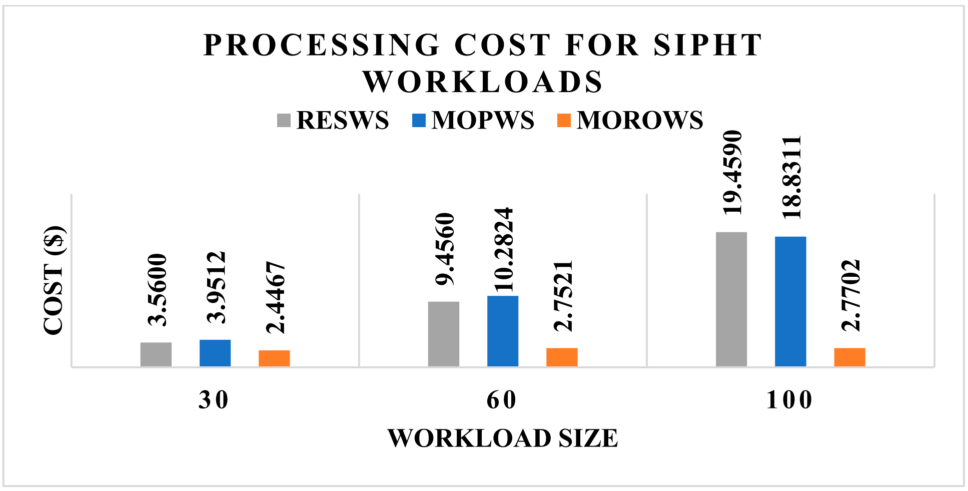

Figure 7. Processing cost required for executing SIPHT workflows using existing and proposed models.

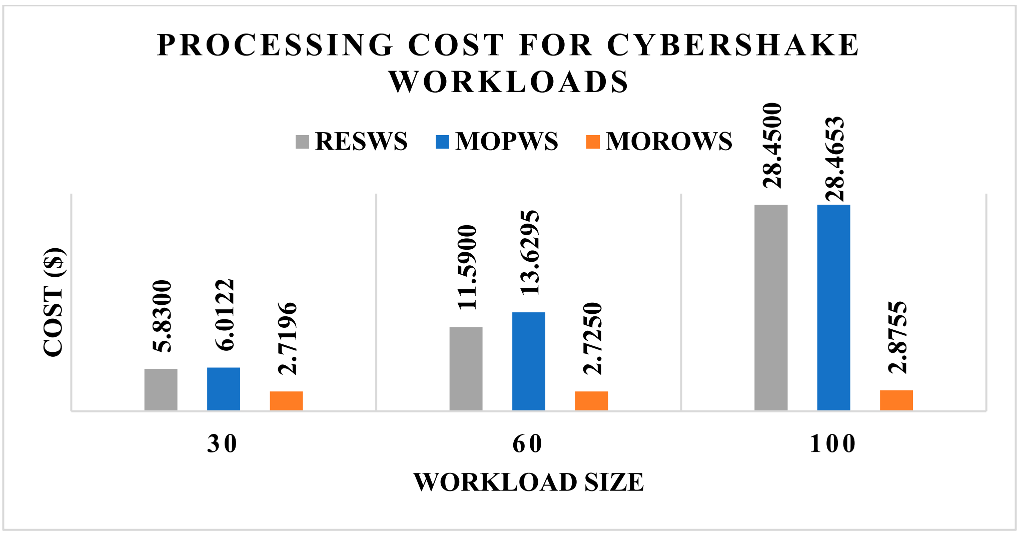

Figure 8. Processing cost required for executing CyberShake workflows using existing and proposed models.

4.3. Processing Energy Consumption

Figure 9 presents the energy consumption, measured in watt hours, for SIPHT workflows, with sizes of 30, 60, and 100 tasks for RESWS, MOPWS, and MOROWS models. The energy consumption of all three models is measured using Equation (15). The existing model demonstrates energy consumption for different workflows compared to the proposed MOROWS model. For the SIPHT workflow, the proposed MOROWS model reduced average energy consumption by 25.02% and 29.52% over RESWS and MOPWS, respectively. This reduction in makespan indicates improved efficiency and performance in the proposed model, which may have practical implications for applications requiring timely data processing within the SIPHT framework. The provided Figure 10 presents the makespan, measured in seconds, for CyberShake workflows, with sizes of 30, 60, and 100 tasks for RESWS, MOPWS, and MOROWS models. The existing model demonstrates more makespan for different workflows compared to the proposed MOROWS model. For the CyberShake workflow, the proposed MOROWS model reduced the average makespan by 17.77% and 24.73% over RESWS and MOPWS, respectively. This energy reduction indicates improved efficiency and performance in the proposed model, which may have practical implications for applications requiring timely data processing within the CyberShake framework. The DRL-based optimization presented in MOPWS attained slightly poorer energy results than RESWS. However, the energy makespan model presented in MOPWS failed to work for a heterogeneous computational platform. On the other side, the significant makespan reduction result attained by MOROWS for both SIPHT and Cyber-Shake workflow is due to an optimization performed to minimize Equation (19), keeping the constraints of Equation (20), Equation (21) through the deep reinforcement learning model.

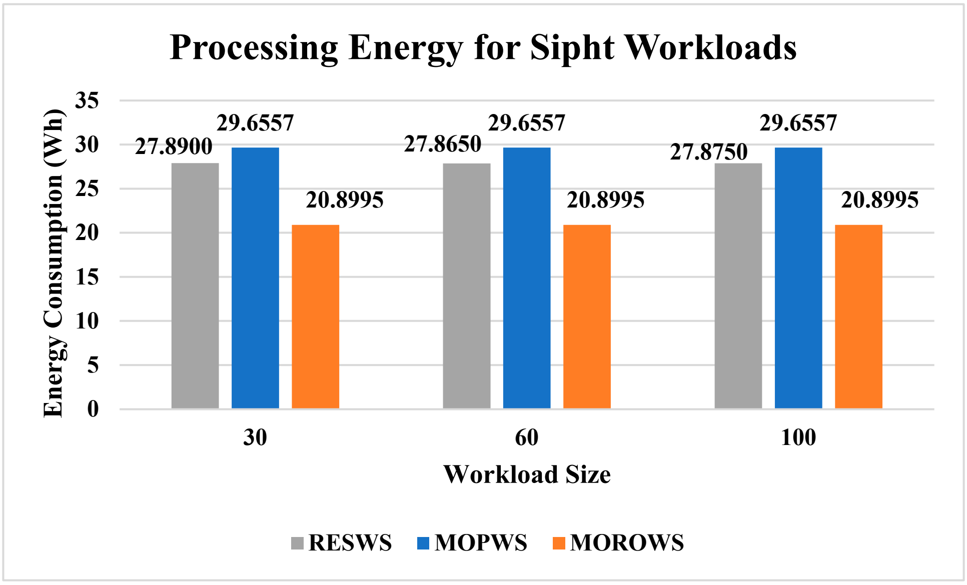

Figure 9. Processing energy required for executing SIPHT workflows using existing and proposed models.

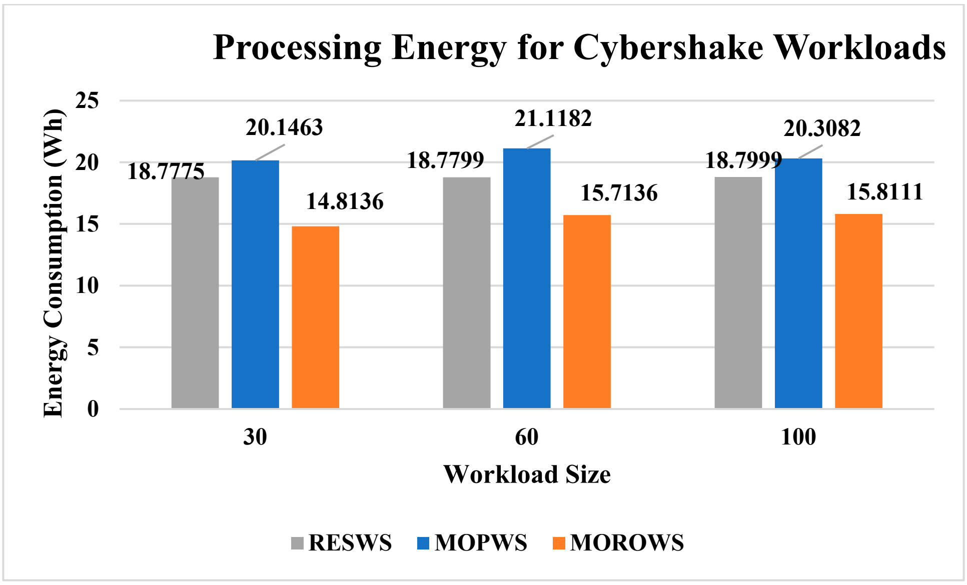

Figure 10. Processing energy required for executing CyberShake workflows using existing and proposed models.

5. Conclusions

Existing cloud scheduling methods often fail to address smart city concerns. Many models lack granularity for job prioritization, resulting in inefficient resource allocation. Additionally, neglecting to handle time, cost, and energy restrictions affects cloud-based system efficiency. A technique to reduce processing time, cost, and energy in smart urban operations is proposed in this research to close these gaps. The findings are compared to existing models for processing time, cost, and energy. SIPHT and CyberShake, two significant smart urban workflows, are used to test the approach. The suggested model consistently outperforms the present model across workflow sizes. The proposed paradigm improves processing time, cost, and energy use. Simulation results show an average makespan enhancements of 31.6% and 70.09%, average cost reductions of 62.64% and 73.24%, and average energy consumption reductions of 25.02% and 17.77%, respectively, by MOROWS over reliability enhancement strategies for workflow scheduling (RESWS) and multi-objective priority workflow scheduling (MOPWS) for SIPHT workflow. Similarly, consistent improvements with average makespan enhancements of 37.98% and 74.44%, average cost reductions of 65.53% and 74.89%, and average energy consumption reductions of 29.52% and 24.73%, respectively, by MOROWS over RESWS and MOPWS for CyberShake workflows. The simulation results show that using the approach in smart urban scenarios can improve computing efficiency, cut costs, and conserve energy. Testing on more workflows and optimizing the job offloading mechanism would improve model performance. Thus, the future work will consider validating the proposed job offloading mechanism considering other scientific workflows like I/O and memory intensive Inspiral workflow, which is used for analyzing blackholes and neutron stars; CPU-intensive Montage workflow, which is used for performing a task like an image mosaicking of millions of images collected from space. Alongside these, we would consider optimizing the offloading task using parameters like SLA violation and reliability.

Author Contributions: Conceptualization, S.B.R.; data curation, H.F.M.L. and S.J.; formal analysis, H.F.M.L.; investigation, H.L. and S.B.R.; methodology, H.L. and S.J.; project administration, H.L.; software, H.F.M.L.; supervision, S.B.R.; visualization, H.F.M.L.; writing—original draft, H.L., S.B.R., and S.J.; writing—review and editing, H.L. and S.B.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Rani, S.; Mishra, R.K.; Usman, M.; Kataria, A.; Kumar, P.; Bhambri, P.; Mishra, A.K. Amalgamation of Advanced Technologies for Sustainable Development of Smart City Environment: A Review. *IEEE Access* **2021**, *9*, 150060–150087. [CrossRef]
2. Saleem, M.; Abbas, S.; Ghazal, T.M.; Khan, M.A.; Sahawneh, N.; Ahmad, M. Smart cities: Fusion-based intelligent traffic congestion control system for vehicular networks using machine learning techniques. *Egypt. Inform. J.* **2022**, *23*, 417–426. [CrossRef]
3. Syed, S.; Sierra-Sosa, D.; Kumar, A.; Elmaghraby, A. IoT in Smart Cities: A Survey of Technologies, Practices and Challenges. *Smart Cities* **2021**, *4*, 429–475. [CrossRef]
4. Alam, T. Cloud-Based IoT Applications and Their Roles in Smart Cities. *Smart Cities* **2021**, *4*, 1196–1219. [CrossRef]
5. Sreenivasa, B.R.; Nirmala, C.R. Hybrid Time Centric Recommendation Model for E-Commerce Applications Using Behavioral Traits of User. *Inf. Technol. Manag.* **2022**, *24*, 133–146. [CrossRef]
6. Sreenivasa, B.R.; Nirmala, C.R. Hybrid Location-Centric e-Commerce Recommendation Model Using Dynamic Behavioral Traits of Customer. *Iran J. Comput. Sci.* **2019**, *2*, 179–188. [CrossRef]
7. Banerjee, P. MTD-DHJS: Makespan-Optimized Task Scheduling Algorithm for Cloud Computing with Dynamic Computational Time Prediction. *IEEE Access* **2023**, *11*, 105578–105618. [CrossRef]
8. Kumar, M.S.; Karri, G.R. EEOA: Cost and Energy Efficient Task Scheduling in a Cloud-Fog Framework. *Sensors* **2023**, *23*, 2445. [CrossRef] [PubMed]
9. Costa, D.G. A Survey of Emergencies Management Systems in Smart Cities. *IEEE Access* **2022**, *10*, 61843–61872. [CrossRef]
10. Febrer, N.; Folkvord, F.; Lupiañez-Villanueva, F. Cost-Effectiveness Assessment of Internet of Things in Smart Cities. *Front. Digit. Health* **2021**, *3*, 662874. [CrossRef]
11. Zhou, Q.; Sun, Y.; Lu, H.; Wang, K. Learning-based Green Workload Placement for Energy Internet in Smart Cities. *J. Mod. Power Syst. Clean Energy* **2022**, *10*, 91–99. [CrossRef]

12. Jia, L.; Li, K.; Shi, X. Cloud Computing Task Scheduling Model Based on Improved Whale Optimization Algorithm. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 1–13. [CrossRef]
13. Wang, Y.; Dong, S.; Fan, W. Task Scheduling Mechanism Based on Reinforcement Learning in Cloud Computing. *Mathematics* **2023**, *11*, 3364. [CrossRef]
14. Sangeetha, S.; Logeshwaran, J.; Faheem, M.; Kannadasan, R.; Sundararaju, S.; Vijayaraja, L. Smart performance optimization of energy-aware scheduling model for resource sharing in 5G green communication systems. *J. Eng.* **2024**, *2024*, e12358. [CrossRef]
15. Murad, S.A.; Muzahid, A.J.M.; Azmi, Z.R.M.; Hoque, M.I.; Kowsher, M. A review on job scheduling technique in cloud computing and priority rule based intelligent framework. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 2309–2331. [CrossRef]
16. Zheng, T.; Wan, J.; Zhang, J.; Jiang, C. Deep Reinforcement Learning-Based Workload Scheduling for Edge Computing. *J. Cloud Comput.* **2022**, *11*, 3. [CrossRef]
17. He, J.; Liu, X. Hybrid Teaching–Learning-Based Optimization for Workflow Scheduling in Cloud Environment. *IEEE Access* **2023**, *11*, 100755–100768. [CrossRef]
18. Udomkasemsub, O.; Sirinaovakul, B.; Achalakul, T. PHH: Policy-Based Hyper-Heuristic With Reinforcement Learning. *IEEE Access* **2023**, *11*, 52026–52049. [CrossRef]
19. Rizvi, N.; Ramesh, D.; Rao, P.C.S.; Mondal, K. Intelligent Salp Swarm Scheduler With Fitness Based Quasi-Reflection Method for Scientific Workflows in Hybrid Cloud-Fog Environment. *IEEE Trans. Autom. Sci. Eng.* **2023**, *20*, 862–877. [CrossRef]
20. Nagarajan, S.M.; Devarajan, G.G.; Mohammed, A.S.; Ramana, T.V.; Ghosh, U. Intelligent Task Scheduling Approach for IoT Integrated Healthcare Cyber Physical Systems. *IEEE Trans. Netw. Sci. Eng.* **2023**, *10*, 2429–2438. [CrossRef]
21. Khaleel, M.I. A fault tolerance aware green IoT workflow scheduling algorithm for multi-dimensional resource utilization in sustainable cloud computing. *Internet Things* **2023**, *23*, 100909. [CrossRef]
22. Nazari, A.; Sohrabi, S.; Mohammadi, R.; Nassiri, M.; Mansoorizadeh, M. IETIF: Intelligent Energy-Aware Task Scheduling Technique in IoT/Fog Networks. *J. Sens.* **2023**, *2023*, 1–13. [CrossRef]
23. Khezri, E.; Yahya, R.O.; Hassanzadeh, H.; Mohaidat, M.; Ahmadi, S.; Trik, M. DLJSF: Data-Locality Aware Job Scheduling IoT tasks in fog-cloud computing environments. *Results Eng.* **2024**, *21*, 101780. [CrossRef]
24. Ye, L.; Yang, L.; Xia, Y.; Zhao, X. A Cost-Driven Intelligence Scheduling Approach for Deadline-Constrained IoT Workflow Applications in Cloud Computing. *IEEE Internet Things J.* **2024**, *11*, 16033–16047. [CrossRef]
25. Mangalampalli, S. Multi Objective Prioritized Workflow Scheduling Using Deep Reinforcement Based Learning in Cloud Computing. *IEEE Access* **2024**, *12*, 5373–5392. [CrossRef]
26. Zhang, L.; Ai, M.; Liu, K.; Chen, J.; Li, K. Reliability Enhancement Strategies for Workflow Scheduling Under Energy Consumption Constraints in Clouds. *IEEE Trans. Sustain. Comput.* **2024**, *9*, 155–169. [CrossRef]
27. Son, J.; He, T.; Buyya, R. CloudSimSDN-NFV: Modeling and simulation of network function virtualization and service function chaining in edge computing environments. *Softw. Pract. Exp.* **2019**, *49*, 1748–1764. [CrossRef]
28. Workflow Dataset from Pegasus WMS. Available online: https://pegasus.isi.edu/workflow_gallery/ (accessed on 11 March 2024).
29. SIPHT Dataset from Pegasus WMS. Available online: https://pegasus.isi.edu/workflow_gallery/gallery/sipht/index.php (accessed on 11 March 2024).
30. CyberShake Dataset from Pegasus WMS. Available online: https://pegasus.isi.edu/workflow_gallery/gallery/cybershake/index.php (accessed on 11 March 2024).