Journal of Grid Computing (2021) 19:34
https://doi.org/10.1007/s10723-021-09574-y

Offloading Coalition Formation for Scheduling Scientific Workflow Ensembles in Fog Environments

Hajar Siar · Mohammad Izadi

Received: 3 November 2020 / Accepted: 21 June 2021

Abstract Fog computing provides a distributed computing paradigm that executes interactive and distributed applications, such as the Internet of Things (IoT) applications. Large-scale scientific applications, often in the form of workflow ensembles, have a distributed and interactive nature that demands a dispersed execution environment like fog computing. However, handling a large-scale application in heterogeneous environment of fog computing requires harmonizing heterologous resources over the continuum from the IoT to the cloud. This paper investigates offloading and task allocation problems for orchestrating the resources in a fog computing environment where the IoT application is considered in the form of workflow ensembles. We called *Offload-Location* a mechanism which has been designed to find offloading coalition structure alongside a matching algorithm for allocating the offloaded tasks to fog/cloud resources. The introduced solution attempts to minimize the execution time and minimize the price paid to servers for executing the tasks provided that Quality of Service (QoS) requirements of the ensemble's deadline and budget are retaining. These objectives lead to maximizing the number of completed workflows of the ensemble as an ultimate goal. The appropriate performance of this mechanism is studied under different workflow applications and circumstances.

Keywords Fog computing · Workflow ensembles · Offloading · Mechanism design · Coalition games · Matching theory

1 Introduction

IoT refers to a diverse ecosystem of connected devices to the internet. This technology, which has a distributed and interactive environment, enables executing large-scale distributed and scientific applications. A scientific application with large-scale computations and data analysis includes a sequence of steps to complete. An essential characteristic of these applications is that computations and data may distribute in the execution environment. Also, users need dynamic interaction with the applications to see preliminary results on time and construct the application's continuation. Workflow considers an efficient way to model and manage the steps of complex, distributed, interactive, and scientific applications. It is an important method to provide a formal and declarative representation of these applications [11, 22].

IoT services are commonly provide using cloud computing. However, this computing paradigm has a centralized model, in which all data are transmitted to the cloud datacenters for processing and storing.

H. Siar · M. Izadi (✉)
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
e-mail: izadi@sharif.edu

H. Siar
e-mail: siar@ce.sharif.edu

Published online: 21 July 2021

Cloud computing is tackling network bandwidth and data transmission delay challenges, especially for delay-sensitive and distributed applications like scientific applications [23, 24]. An IoT environment following the architecture of cloud computing is known as cloud-centric IoT [8] which faces challenges of BLURS: bandwidth, latency, uninterrupted, resource-constraint, and security. However, the distributed and interactive nature of large-scale workflow-based scientific applications has several challenges with the limitations of current executing environments. For example, many large-scale computational problems need interactions with distributed professors around the world to verify the results of each step of the workflow [11].

Many different approaches are studied for extending cloud computing to a more geographically distributed model. Among the various concepts, fog computing gained more attention [8, 23]. Although some studies use fog and edge computing interchangeably, these concepts have several differentiates. Edge computing employs the edge resources, while fog computing has a hierarchical-based distributed computing model, harnesses resources from the edge to the cloud continuum. The closer computation and storage resources to the end-user are, the greater their geo-distribution and the lower their computational power [3, 23, 53]. Fog nodes closing to end-users can reduce latency noticeably compared to cloud computing [9, 24]. For characterizing fog and cloud computing, we can say that fog computing support latency-sensitive and interactive applications, while cloud computing supports CPU-intensive and delay-tolerant applications. Although there are some differences between these two technologies, not only fog and cloud computing are distinct, but also they complement each other. Using fog nodes can address most of the challenges of cloud computing [23, 47, 53]. This distributed network's heterogeneous resources must cooperate effectively to achieve the capabilities of the environment. The orchestrator is a vital component in harmonizing the resources from the IoT to the cloud continuum. Scheduling and assigning the application's tasks to resources is an essential function of this component. The fog and cloud nodes can lease their resources to end-users to efficiently execute their applications. A fog broker, which is often a fog node, decides IoT applications' tasks are execute locally or offload to fog/cloud resources. Hence, task offloading and resource allocation are essential challenges in scheduling the applications in a fog environment [41, 43, 48].

Although considering the applications in the form of workflow can model and manage them effectively, there are few studies on workflow scheduling in fog environments. The literature considered different types of workflow-based applications. Some large-scale applications consist of multiple workflows in which each workflow has a specified QoS. Also, the large-scale application may be represented as a workflow ensemble, where the combined execution of workflows generates the desired output. There are some differences between applications in the form of workflow ensembles and multiple workflows. Firstly, the QoS requirements are determined for the ensemble as a whole, not each workflow. So, all workflows cannot complete necessarily, and an extra objective compared with multiple workflows is maximizing the number of completed workflows from the ensemble. Also, the ensemble's workflows have a similar structure, and the differences are in the input data and workflows' size. The final difference is the number of workflows in the ensemble, which is predefined in most cases [31, 42]. Providing QoS requirements is an important objective in executing workflow ensembles. The behavior of application is influenced by various parameters such as time, available budget, energy, bandwidth, and other parameters. Most of these criteria are interdependent, such that considering several parameters in the solution can provide the QoS requirements efficiently [26, 35, 49].

We proposed a game-theoretic solution using coalition games to address task offloading and allocation problems for executing workflow ensembles in a fog environment. The considered system model consists of multiple IoT, fog, and cloud devices, such that a set of resources in different tiers are cooperating in executing the application. To the best of our knowledge, it is the first study that addresses multi-objective problems of offloading and task allocation for workflow ensembles in a fog environment consisting of IoT-fog-cloud devices. The considered objectives are minimizing execution time and minimizing monetary cost of executing the tasks, while the considered QoS constraints of the ensemble's deadline and budget are not violating. Reaching these objectives leads to

maximizing the number of completed workflows of the ensemble as an ultimate objective. Since cloud computing's bandwidth-delay is a motivation for emerging the fog computing paradigms, a resource management solution in this environment must consider the communication delay as a vital performance metric. Previous studies on workflow management [7, 27] emphasize the influence of data demands and data communications on the execution time, while most studies did not consider this issue [7]. The current research considers the data demands and communication delays for executing the tasks. We defined an optimization problem for the considered scenario and applied a game-theoretic approach by determining IoT devices' payoff function. A mechanism which we named *Offload-Location* is designed to find a coalition of IoT devices that are offloading their tasks to fog/cloud tiers, and the rest of IoT devices are executing their tasks locally. Also, we used a many-to-one stable matching to determine the appropriate fog or cloud resources for allocating the offloaded tasks. We have shown the stability of the proposed *Offload-Location* mechanism and its time complexity. We analyzed the performance of the proposed solution using extensive experiments on benchmark scientific workflows under different situations.

The structure of this paper is as follows. Section 2 presents a literature review of related works and an analysis of the mentioned studies. The solution's considerations are illustrated in Section 3, consisting of system and application model, problem formulation, and game model. The proposed *Offload-Location* mechanism is explained in Section 4. Sections 5 represent experimental evaluations of the proposed solution. Conclusion and possible directions for future works are present in Section 6.

2 Related Works

Generally, there are numerous studies on scheduling in fog and cloud computing environments. In most of these studies, the target application is considered as independent tasks, and the number of works on workflow scheduling in fog paradigms is limited [22, 49]. The main difference between task scheduling studies of independent and dependent tasks is that scheduling of dependent tasks considers dependencies among tasks in the form of workflows, which results in more efficient and actual solutions [22, 49]. According to [14], research studies on orchestration and scheduling in fog environments can classify into two main groups: network architecture-based schemes and algorithmic schemes. Since the proposed offloading and task allocation solution is an algorithmic scheme, the literature study was conducted on algorithmic ones. Algorithmic schemes categorize into two groups by considering the optimization objectives: single objective and multi-objective. We can also classify the literature into two categories by considering the input application as independent tasks or dependent tasks (i.e., workflows). Furthermore, based on Section 1, the dependent tasks can be assumed as a single workflow, multiple workflows, or workflow ensembles.

A joint computation offloading and prioritized task scheduling in mobile edge computing proposed in [17]. The solution attempts to minimize edge devices' energy consumption, using a task offloading strategy and schedule the tasks using a dynamic priority-based task scheduling. In [46], Tianz and et al. are designed the independent task scheduling problem among mobile devices in mobile edge computing as a potential game. The scheduling solution attempts to improve resource usage by minimizing delay and energy consumption in computation offloading and executing the task. The paper of [25] proposed a game-theoretic model for offloading and task allocation of surveillance applications' periodic tasks in mobile cloud computing. The offloading solution attempts to minimize execution time and energy consumption of mobile devices.

The ability of computation offloading in maximizing the system utility in fog computing is proved [43, 54]. The paper of [14] addresses the offloading problem in mobile edge computing. The considered environment consists of mobile edge computing base stations (MEC-BS) such that each MEC-BS provides services to the mobile terminals under its coverage. The objective is to improve the utility of end devices in terms of execution time and energy consumption while balancing the load of MEC-BS by scheduling the tasks over multiple MEC-BSs. In [1], the NSGA-II algorithm is used for allocating independent tasks in an IoT-Fog-Cloud architecture. This approach attemps to minimize time and energy consumption for executing the tasks. In [21] a recursive algorithm is proposed to optimize offloading and scheduling problems of independent tasks in fog computing. The paper of [4]

addresses offloading and scheduling of IoT tasks using the matching theory. The preferences of fog and IoT devices are defined considering fog nodes attempt to minimize their operation cost (i.e., energy consumption) and minimize traffic cost, while IoT devices are interesting in minimizing the execution time. Offloading and resource allocation problems in a multi-tier fog network are address in [18]. The solution attempts to minimize long-term time-average expectation of power consumption in fog tiers. We consider the power as energy in analyzing the related works.

The resources of a network do not necessarily have an incentive to share their powers. For example, a more powerful device having less load may not have incentive to share its resources with an overloaded device with less power. In this case, some mechanisms are needed to motivate the devices. Liu, Y. and et al. are model the interactions between cloud service operator and edge servers as a Stackelberg game to motivate resources to cooperate in the offloading [30]. Zhou. and et al. [56], are designing an auction mechanism in mobile edge computing. The mechanism is forming a shared resource network and an offloading market to motivate mobile devices to share their resources [56]. The literature study of this research explained that most of works in motivating resources are game theory and auction-based [56]. The paper of [55] is a game-theoretic solution for addressing computation offloading in vehicular edge computing using Stackelberg game. The solution uses a multi-level offloading scheme to determine the optimal offloading workload of vehicles while not violating the delay constraints of tasks. Chen and et al. are introduced a socially trusted collaborative network of edge resources. This work indicates that collaboration among edge devices can reduce system costs by about 40 percent [9]

Although there are limited studies on workflow scheduling in fog computing paradigms, most of these research studies use meta-heuristic approaches. Goudarzi and et al. introduced a multi-objective meta-heuristic technique using a memetic algorithm for workflow scheduling among IoT-edge-cloud devices to minimize energy and time consumption. The considered application is in the form of multiple workflows, and QoS constraints did not consider in the solution [20]. The paper of [52] studied on the video edge analytics by considering the application as a workflow. The introduced model is an edge-first scheme focusing on mobile-edge and edge-edge layers to minimize the response time. Also, it is enabling inter-edge collaboration by introducing three task-placement schemes. The paper of [50] studies the problem of workflow scheduling for multimedia applications. It uses a meta-heuristic method for scheduling the tasks. Also, Blockchain technology is used for data security and integrity during offloading. Since Blockchain mining is a computationally-intensive task, this technique must be used rigorously in resource-limited resources of edge/fog. In [45], a cloud and edge-aware heuristic is introduced to schedule multiple real-time workflows in a three-level architecture with IoT-fog-cloud devices. The scheduling solution attempted to assign CPU-intensive and latency tolerant tasks to the cloud and assign latency-sensitive tasks with short running times to fog devices. In [51] the offloading problem for multiple-workflows is formulated as a multi-objective optimization problem that attempts to minimize time and energy consumption. The NSGA-III is used for addressing the optimization problem.

Since the literature review shows no study on resource management for workflow ensembles in fog environments, studies in cloud computing are presented in this section. Geneaz and et al. have introduced a flexible workflow ensemble scheduling in [19], using the PSO algorithm. The authors defined flexibility as changing the objective function and using the solution in different circumstances. They mentioned that a mixture of the objective functions could not be used in their solution, and it is a single-objective solution. Pietri and et al., in [40] are investigate resource provisioning problem to schedule workflow ensembles while the deadline and energy constraints are satisfied. This study considers energy constraints besides the deadline or available budget for scheduling workflow ensembles for the first time. The paper of [31] is an extension of [40] that addressed the scheduling problem of workflow ensembles under constraints of budget and deadline and aiming to maximize the number of completed workflows.

An analysis of the presented related works shows in Table 1. The table compares the studies regarding inputs, architecture, approach, and objectives in the solution. Although the distributive and interactive nature of scientific workflow ensembles demands

Table 1 Related works analysis

		Gao and Moh [17]	Tianze et al. [46]	Jošilo and Dán [25]	Xie et al. [49]	Yi et al. [52]	Xu et al. [50]	Stavrinides and Karatza [45]	Genez et al. [19]	Pietri et al. [40]	Malawski et al. [31]	Fan et al. [14]	Xu et al. [51]	Liu et al. [30]	Zhou et al. [56]	Zhang et al. [55]	Chen and Xu [9]	Goudarzi et al. [20]	Gao et al. [18]	Guo et al. [21]	Arisda-kessian et al. [4]	Abbasi et al. [1]
Input	Independent-tasks	✓	✓	✓								✓		✓	✓	✓	✓		✓	✓	✓	✓
	Single-workflows				✓	✓																
	Multiple-workflow						✓	✓					✓					✓				
	Workflow-ensemble								✓	✓	✓											
Architecture	IoT		✓	✓		✓							✓		✓	✓		✓			✓	✓
	Fog/Edge	✓			✓	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Cloud			✓	✓			✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓		✓
Approach	Heuristic	✓				✓		✓		✓	✓								✓	✓		
	Meta-heuristic				✓		✓		✓			✓	✓					✓				✓
	Machine-Learning																					
	Game-theory		✓	✓										✓	✓	✓	✓				✓	
Objective	Time	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓		✓		✓	✓	✓
	Energy	✓	✓	✓			✓			✓	✓	✓	✓		✓			✓	✓		✓	✓
	Price		✓		✓		✓	✓	✓	✓	✓			✓		✓	✓					
	Securiy																✓					

the distributed environment of fog computing, Table 1 represents no study on workflow ensemble scheduling in this environment. Another aspect of this analysis is that the number of works considering all resources over the continuum from the IoT to the cloud is limited, while a solution that employs all heterogeneous resources in the environment is more efficient. This analysis confirms that most related studies are considered multiple objectives in their solution because of the problem space's multi-objective nature. Although the meta-heuristic approaches are easy to design and implement, they suffer from overhead in time and memory and convergence to a local optimum.

Game-theoretic approaches apply strategic solutions that enable players to decide the optimal decision by considering the strategy of other players [28, 36]. In this paper, a game-theoretic method has been proposed for computation offloading and task allocation among IoT-fog-cloud devices in a fog computing environment. The applications are in the form of workflow ensembles with the QoS constraints of available deadline and budget. A coalition game has been designed to specify the set of IoT devices offloading their task, and the set of devices execute their tasks locally. The allocation of offloaded tasks to fog and cloud nodes is determined using a many-to-one stable matching. All of the decisions in these procedures are made by considering the ensemble's deadline and budget constraints and minimizing the execution time and the price.

3 System and Game Model

This section represents the underlying fog computing environment, the application model, and problem formulation. Besides, some considerations of the solution algorithm and the game model are described.

3.1 System Model

The considered three-tier system model consisting of IoT-fog-cloud devices represents in Fig. 1. We assumed large-scale scientific applications in the form of workflow ensembles inputting to IoT devices. There are predefined QoS constraints for the ensemble, and workflows that do not violate the constraints are entering to the IoT devices randomly, such that all tasks of each workflow are inserting to a specific IoT device with a specified arrival rate. The abbreviations of m, e, and c indicate IoT, fog, and cloud layers, respectively. $layer(t)$ refers to the layer corresponding to the node assigned to task t. For example, if task t is allocating to a device that is in the fog layer, then $layer(t) = e$. We have assumed n_m IoT devices, n_e fog servers, and n_c cloud servers in the system. The computation power of resource k represents by μ_k where for all μ_k's in the same layer, we have the relations of $\mu_m \leq \mu_e \leq \mu_c$. We have considered the connections among nodes as a virtually connected graph, where each node is connected directly to other nodes. $r_{m,e}$ and $r_{e,c}$ represent data communication rates from IoT to fog layer and from fog to cloud layer, respectively. Also, $r_{e,e}$ and $r_{c,c}$ represent data communication rates among servers in the fog layer and servers in the cloud layer, respectively.

Fog and cloud servers receive a price for processing the tasks, which varies due to the heterogeneity of resources in different layers. We assume that resources are leased and charged per billing period. The monetary cost of using resource k in each billing period is considered as cu_k; also τ_e, and τ_c are billing period of resources in the fog and cloud layers, respectively. Since resources in the fog layer have less power than those in the cloud layer, the price of resources in the fog layer is less than those are in the cloud layer. A resource may be idle, while time remains until its next billing period. This remained time will be considered in calculating the monetary cost of processing the next task. If this time is longer than the next task's response time, no price will receive. If this remained time is RT and the estimated response time of ready task t is $P^p_{k,t}$, then the monetary cost of resource k for processing this task can be calculated as follows:

$$c^k_t = \begin{cases} 0 & if\ RT \geq P^p_{k,t} \\ \left\lceil \frac{(P^p_{k,t} - RT)}{\tau_k} \right\rceil cu_k & if\ RT < P^p_{k,t} \end{cases} \quad (1)$$

In the above equation, $k \in [e \cup c]$. If k is a fog or cloud server, then $\tau_k = \tau_e$ or $\tau_k = \tau_c$, respectively.

3.2 Application Model

The input application is in the form of workflow ensembles representing as $A = \{w_i | 1 < i < n_w\}$. Where n_w is the number of workflows in the ensemble. Each workflow, which is in the form of a directed acyclic graph (DAG) of $w^d_i = (T^d_i, E^d_i)$ enters to a

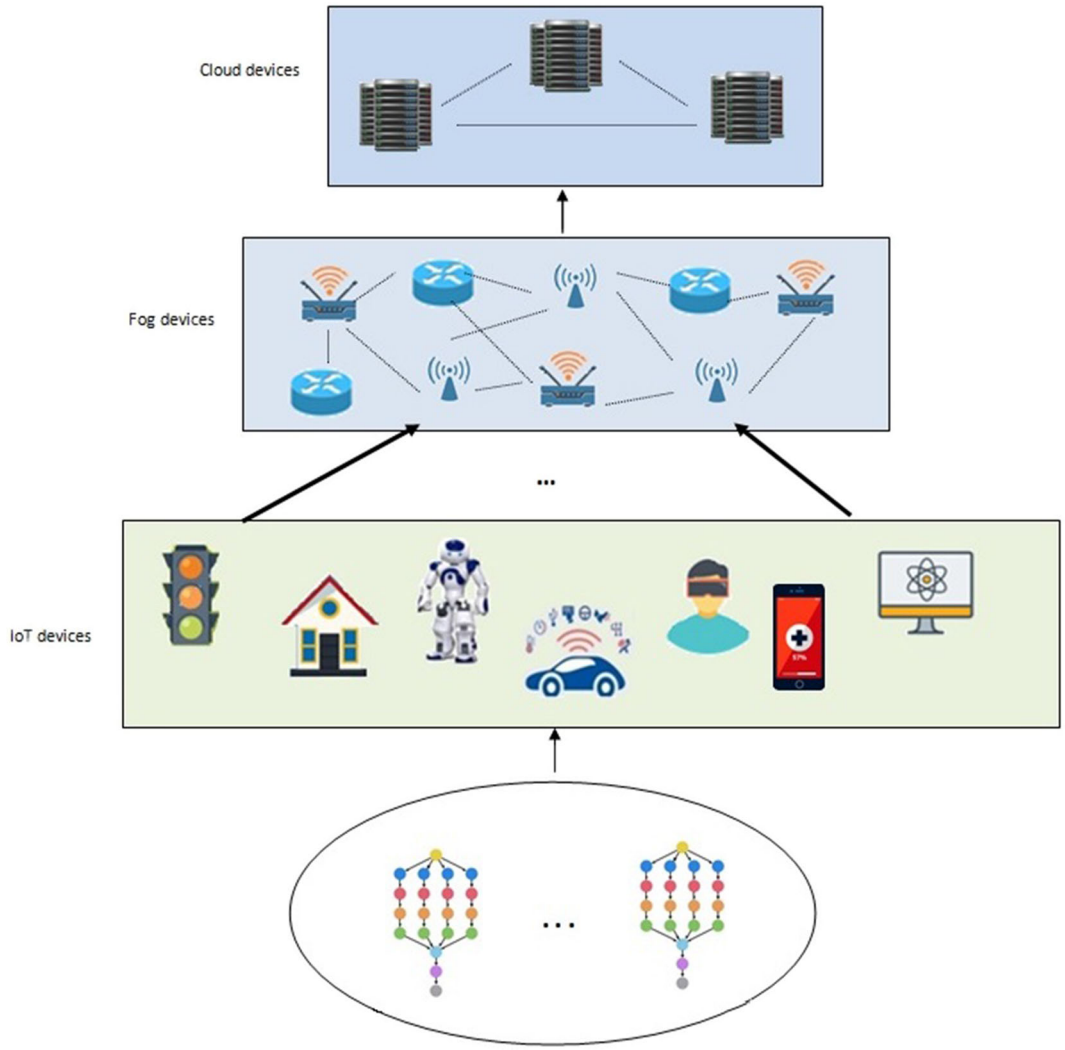


Fig. 1 System model

specific IoT device d, where $T_i^d = \{t_{i,j,d} | 1 < i < n_w, 1 < j < |T_i|, 1 < d < n_m\}$ is the vertex set of w_i^d and $t_{i,j,d}$ is task j of workflow i that enters to IoT device d. The workload size of $t_{i,j,d}$ is represent as $w_{t_{i,j,d}}$. The set of all tasks of all workflows entering IoT-device d are define as $T^d = \{T_i^d | 1 < i < n_w, 1 < d < n_m\}$. E_i^d corresponds to the edge set of w_i^d. An edge of $(u_{i,j,d}, v_{i,j,d})$ in the graph indicates data dependency between tasks $u_{i,j,d}$ and $v_{i,j,d}$. The amount of data must send from the output of $u_{i,j,d}$ to the input of $v_{i,j,d}$ is represent by d_{uv}. In this case, $v_{i,j,d}$ is a child of $u_{i,j,d}$ and execution of $v_{i,j,d}$ starts only after completed execution of its parents and availability of their output data on the resource allocated to this task. If both $u_{i,j,d}$ and $v_{i,j,d}$ are allocate to a same resource, the data communication delay is consider as zero. The set of parents of task t represents as $pred(t)$, and t is said to be ready if $pred(t)$'s output data are available on the resource assigned to t [22].

QoS constraints of scientific workflow ensembles are defining for the ensemble, not each workflow independently. Applying these constraints does not necessarily allow completing all workflows, so executing the maximum number of workflows from the ensemble is desired. Our solution is workflow aware, which evaluates the executability of input workflows according

to the current deadline and budget constraints. For every new task of a workflow, a procedure checks the required computation and monetary cost for executing the workflow, and the new workflow can be considered in the scheduling procedure if these costs are not higher than the current deadline and budget constraints of the ensemble.

3.3 Problem Formulation

This section presents a formal representation of the offloading and task allocation problems for workflow ensemble scheduling in the defined fog environment. Each IoT device can execute its ready task locally or offload it to one of fog or cloud resources. After choosing the offloaded tasks, both fog and cloud devices are participating in the task allocation. Hence, we tackle two problems: which tasks will offload to fog or cloud servers and how to allocate the offloaded tasks among servers.

A list of notations used in this section is present in Table 2. The offloading decision of n_m IoT devices represents using a binary vector of O with the length of n_m such that $O_d = 1$ means IoT device d offloads its task and $O_d = 0$ means local execution of the task. The IoT device d is permitted to offload provided that the sum of computation and monetary cost of executing the task in the fog or cloud layer is not bigger than that of local execution. The allocation decision of offloaded tasks is represented using a matrix S with the size of #offloaded-tasks$\times(n_e + n_c)$, such that the rows indicate the offloaded tasks and the columns correspond to the fog and cloud servers. Each fog or cloud node r must initialize column $\vec{S}_r$ of S, so that for every offloaded task t if $S_{t,r} = 1$ then, task t allocates to resource r, otherwise $S_{t,r} = 0$. Each task allocates to only one resource, so the statement of $\sum_{r=1}^{n_e+n_c} S_{t,r} = 1$ must satisfy. The offloading solution attempts to minimize the computation and monetary cost of executing the tasks provided that the ensemble's deadline and budget constraints are not violated. Achieving these two objectives remains more time and budget, which leads to maximizing the number of completed workflows from the ensemble. The computation cost of resource k in executing task $t_{i,j,d}$ is defined by the sum of response time and the time needed for receiving the dependency data of $t_{i,j,d}$, which is known as communication delay. The response time of executing task $t_{i,j,d}$ on resource k is calculated by formula (2) for both local execution and offloading, respectively.

$$p^{p}_{k,t_{i,j,d}} = \begin{cases} \frac{w_{t_{i,j,d}}}{\mu_d} & , if\ k = d\ \wedge\ O_d = 0 \\ \frac{w_{t_{i,j,d}}}{\mu_k} & , if\ k \neq d\ \wedge O_d = 1 \end{cases} \tag{2}$$

We are using formula (3) for estimating the communication delay of executing task $t_{i,j,d}$ on device k for both cases of local execution and offloading.

$$p^{tr}_{k,t_{i,j,d}} = \begin{cases} \sum_{u \in pred(t_{i,j,d}) \wedge layer(u) \neq m} \frac{d_{u,t_{i,j,d}}}{r_{layer(u),m}}, if\ k = d\ \wedge\ O_d = 0 \\ \sum_{u \in pred(t_{i,j,d}) \wedge layer(u) \neq layer(t_{i,j,d})} \frac{d_{u,t_{i,j,d}}}{r_{layer(u),layer(t_{i,j,d})}}, if\ k \neq d\ \wedge O_d = 1 \end{cases} \tag{3}$$

So, the computation cost of executing task $t_{i,j,d}$ on device k evaluates by formula (4).

$$p^{comput}_{k,t_{i,j,d}} = (p^{p}_{k,t_{i,j,d}} + p^{tr}_{k,t_{i,j,d}}) \tag{4}$$

We assume IoT devices do not receive any price in local execution. However, fog and cloud devices impose a monetary cost for processing, which is explained in Section 3.1. So, we can evaluate the monetary cost of executing $t_{i,j,d}$ on device k as follows.

$$p^{mon}_{k,t_{i,j,d}} = \begin{cases} 0 & , if\ k = d\ \wedge\ O_d = 0 \\ c_k^{t_{i,j,d}} & , if\ k \neq d\ \wedge O_d = 1 \end{cases} \tag{5}$$

Table 2 List of notations

Notation	Definition
D	Deadline constraint of the ensemble (second)
B	Budget constraint of the ensemble
n_m	Number of IoT devices
n_e	Number of fog servers
n_c	Number of cloud servers
n_w	Number of workflows in the ensemble
$t_{i,j,d}$	task j of workflow i entering to IoT device d
$w_{t_{i,j,d}}$	Workload size of $t_{i,j,d}$ (millions of instructions)
μ_k	Computation power of resource k (millions instructions)
$d_{u,v}$	Size of data dependency between tasks u and v (bit)
$r_{layer_m,layer_k}$	Communication rate between two devices such that first device is in a layer that task m allocates to it and second device is in a layer that task k allocates to it (bit/second)
γ^m	The weight attribute of monetary cost
γ^c	The weight attribute of computation cost
τ_k	Billing period of resource k (second)
cu_k	price of using resource k per billing period
c_t^k	Received price of resource k for processing task t
O	Offloading decision vector of IoT devices
S	Task allocation decision matrix of fog and cloud devices
$p^{comput}_{k,t_{i,j,md}}$	Computation cost of executing task $t_{i,j,md}$ on resource k (second)
$p^{mon}_{k,t_{i,j,md}}$	Monetary cost of executing task $t_{i,j,md}$ on resource k

According to the above formulations, we have the following optimization problem.

$$\min_{O_d, S_{t_{i,j,d},k}} \sum_{k=1}^{n_m+n_e+n_c} \sum_{d=1}^{n_m} \sum_{i=1}^{n_w} \sum_{j=1}^{|T_j^d|} O_d S_{t_{i,j,d},k} \times \left[\gamma^c p^{comput}_{k,t_{i,j,d}} + \gamma^m p^{mon}_{k,t_{i,j,d}}\right] + (1 - O_d) p^{comput}_{k,t_{i,j,d}} \quad (6)$$

such that

$$O_d \in [0, 1] \quad , \forall d = 1, ..., n_m \quad (7)$$

$$\sum_{k=1}^{n_e+n_c} S_{t_{i,j,d},k} = 1 \quad , S_{t_{i,j,d},k} \in [0, 1];\ \forall i = 1, ..., n_w;\ j = 1, ..., |T_i^d|;\ d = 1, ..., n_m;\ k = 1, ..., n_m + n_e + n_c \quad (8)$$

$$\sum_{k=1}^{n_m+n_e+n_c} \sum_{d=1}^{n_m} \sum_{i=1}^{n_w} \sum_{j=1}^{|T_j^d|} O_d S_{t_{i,j,d},k} \times p^{comput}_{k,t_{i,j,d}} + (1 - O_d) p^{comput}_{k,t_{i,j,d}} < D \quad (9)$$

$$\sum_{k=1}^{n_m+n_e+n_c} \sum_{d=1}^{n_m} \sum_{i=1}^{n_w} \sum_{j=1}^{|T_j^d|} O_d S_{t_{i,j,d},k} \times p^{mon}_{k,t_{i,j,d}} < B \quad (10)$$

Where γ^c and γ^m are weight attributes to balance the computation cost and monetary cost, respectively. Function (6) is the total cost of executing workflows' tasks provided that QoS constraints are preserved. Constraint (7) denotes O_d is a binary decision parameter. Constraint (8) ensures that each task $t_{i,j,d}$ allocates to exactly one resource of fog or cloud. Constraints (9) and (10) are guarantee that the computation and monetary cost of executing all tasks of all workflows do not violate the deadline and budget constraints of the ensemble, respectively.

It is an integer programming problem attempting to formulate task offloading and allocation of workflow ensembles in a fog environment considering the deadline and budget constraints. The offloading problem is seeking a group of IoT devices offloading their tasks while the rest of IoT devices execute their tasks locally. Function (6) indicates that addressing the task

offloading problem requires determining the resources assigned to the offloaded tasks. Since all IoT devices desire to minimize the computation and monetary cost of executing their tasks with a global incentive to maximize the number of completed workflows of the ensemble, we used a coalition game to address this problem. Also, stable matching is used for handling the task allocation problem.

3.4 Coalition Game Model

This section presents the proposed game-theoretic framework, which uses a coalition game [5, 13, 15, 28, 37] to schedule workflow ensembles in fog environments regards to the illustrated problems of offloading and task allocation. Assuming $MD = \{md_1, md_2, \cdots, md_{n_m}\}$, a coalition game is defined as a tuple of (MD, V) such that MD is the set of players and V is a real-valued characteristic function which defines on $F \subseteq MD$ such that $V \rightarrow \Re^+$ and $V(\emptyset) = 0$. $F \subseteq MD$ is a coalition, and a partitioning of the player set is known as coalition structure, which represents by $\mathcal{F}$. Each player of coalition F receives a payoff according to the characteristic function. A coalition game is looking for a coalition structure that maximizes all players' payoff.

Our solution, named offloading coalition game, considers the IoT devices as players and seeks an optimal coalition of players who desire to offload their tasks as an offloading coalition. Others who do not participate in the offloading coalition will execute their tasks locally in the form of local coalitions with exactly one member. The satisfaction of the deadline and budget constraints are checking in all cases. Since the definition of coalition games attempt to maximize the players' payoff, the defined functions in the offloading optimization problem are used in fraction form to determine the $V(F)$ according to formula (11).

$$V(F) = \begin{cases} \dfrac{1}{\sum_{md \in F}[\gamma^c p^{comput}_{k,t_{i,j,md}} + \gamma^m p^{mon}_{k,t_{i,j,md}}]}, if\ |F| > 0\ and\ F\ is\ a\ feasible\ offloading\ coalition \\ \dfrac{1}{p^{comput}_{md,t_{i,j,md}}}, if\ |F| = 1\ and\ F\ is\ a\ feasible\ local\ coalition\ of\ md \\ 0 \quad , if\ |F| = 0 \\ \alpha \quad if\ F\ is\ not\ feasible \end{cases} \tag{11}$$

A coalition of IoT devices is feasible if the computation cost and monetary cost of executing this coalition's tasks do not violate the available deadline and budget, respectively. The parameter of α in the characteristic function of infeasible coalitions is a negative and small value, which prevents players from choosing these coalitions. Equation 11 indicates that it is necessary to know the resources assigned to tasks to calculate the coalitions' characteristic function.

The players participate in a coalition with maximum profit, and the coalitions define a payoff division rule to distribute the payoff among participated players. We are sharing the profit among players equally. The following division rule is used to evaluate the payoff of player i in an offloading coalition F.

$$payoff_i = V_{F_i} = \frac{V_F}{|F|} \tag{12}$$

A coalition consists of all players of the game is known as the grand coalition. The core of a coalition game is a distribution of payoff among players in the grand coalition such that no player has an incentive to leave the grand coalition [5, 13, 15, 28, 37]. By considering the feasibility constraint in generating the coalitions, a situation may occur that players do not have any incentive to form the grand coalition. Assume there are five IoT devices of $MD = \{md_1, \cdots, md_5\}$ and the game reached to the offloading coalition of $\{md_1, \cdots, md_4\}$. If joining $\{md_5\}$ in the offloading coalition makes it infeasible, the payoff of $\{md_5\}$ will decrease in the grand coalition, and it has no incentive to join the grand coalition. So the core of the offloading coalition game is empty. Under these situations, other techniques are used to find the optimal coalition and solving the game.

4 Offloading and Allocation Mechanism

4.1 Offloading Coalition Game

We used hedonic games [6] to solve the proposed offloading coalition game. A Hedonic game defines as a pair of $(I, \succcurlyeq)$ where I is the set of players, and $\succcurlyeq$ is an irreflexive and transitive preference relation defines over coalition collections. A coalition collection is a set of disjoint coalitions, and two collections are compared, provided that they have different partitioning on the same set of players [2, 6]. In the illustrated offloading coalition game, a collection with maximum payoff is preferred. We define the payoff of a collection as the summation of its coalitions' payoff. If *payoff(X)* is the payoff of coalition collection X, the preference relation over coalition collections defines by formula (13).

$$X \succcurlyeq X' \Leftrightarrow payoff(X) > payoff(X') \tag{13}$$

Hedonic games are employ two operators of *merge* ($\gg_m$) and *split* ($\gg_s$) to exchange the players among coalitions according to the preferences of coalition collections [2, 6, 32, 33]. These operations are defined in formula (14).

$$\begin{aligned}
&merge: \left\{\bigcup_{j=1}^{k} F_j\right\} \gg_m \{F_1, ..., F_k\};\\
&where \left\{\bigcup_{j=1}^{k} F_j\right\} \succcurlyeq \{F_1, ..., F_k\}\\
&split: \{F_1, ..., F_k\} \gg_s \left\{\bigcup_{j=1}^{k} F_j\right\};\\
&where \{F_1, ..., F_k\} \succcurlyeq \left\{\bigcup_{j=1}^{k} F_j\right\}
\end{aligned} \tag{14}$$

In the offloading coalition game, players' preferences specify according to the coalition's characteristic function of *V(F)*. We defined the merge and split operations of our offloading coalition game according to formulas (15) and (16), respectively. These operations generate coalition structures according to the problem's assumptions. So, the merge operation permits merging single-member coalitions of local with the offloading coalition. Also, the split operation enables splitting the offloading coalition into one or more single-member local coalitions.

$$\begin{aligned}
&merge: \left\{F_O \cup \{\bigcup_{j=1}^{k} F_j\}\right\} \gg_m \{F_O, F_1, ..., F_k\}\\
&\Leftrightarrow\ V_{\left\{F_O \cup \{\bigcup_{j=1}^{k} F_j\}\right\}} > V_{F_O} + \sum_{j=1}^{k} V(F_j)
\end{aligned} \tag{15}$$

$$\begin{aligned}
&split: \{F_O, F_1, ..., F_k\} \gg_s \left\{F_O \cup \left\{\bigcup_{j=1}^{k} F_j\right\}\right\}\\
&\Leftrightarrow\ \forall\ j \in \{1, ..., k\},\\
&\forall md \in F_j, md \in F_O;\ V_{F_O} + \sum_{j=1}^{k} V(F_j)\\
&\geq V_{\left\{F_O \cup \{\bigcup_{j=1}^{k} F_j\}\right\}}\\
&\wedge\ \{\exists j \in \{1, ..., k\};\ V_{\left\{F_O \cup \{\bigcup_{j=1}^{k} F_j\}\right\} \cap F_j} + V_{F_j}\\
&> V_{\left\{F_O \cup \{\bigcup_{j=1}^{k} F_j\}\right\}} \vee\ \exists md_i \in F_O;\\
&\times V_{F_O} + \sum_{j=1}^{k} V(F_j) > V_{\left\{F_O \cup \{\bigcup_{j=1}^{k} F_j\}\right\}}\ \}
\end{aligned} \tag{16}$$

Where F_O is the offloading coalition. The merge and split are happen provided that the payoff of no coalition decreases and no player will be unhappy. The merge occurs when there are one or more players in local coalitions that will be more profited if they offload their tasks, or the payoff of the offloading coalition will increase if this/these player(s) join the offloading coalition. Also, the split happens provided that one or more players will gain more payoff when leaving the offloading coalition, or the payoff of the offloading coalition will increase by removing one or more players provided that their utilities are not decreased.

4.2 Task Allocation Matching

The introduced offloading coalition game determines whether the tasks are executed locally or offloaded to fog or cloud tiers. However, there is another question: how offloaded tasks will assign to the fog or

cloud resources. We used a many-to-one stable matching [12, 16] for addressing this problem. The stable matching works on two disjoint sets of players with a predefined preference list to find a pairing of players while no player has any preference in leaving his/her pair. We defined the task allocation matching over two sets of offloaded tasks and available fog or cloud resources. The matching attempts to find a pairing of offloaded tasks to resources according to the preferences.

Since IoT devices prefer minimizing computation and monetary cost of executing their tasks, the preference of offloaded task $t_{i,j,d}$ on assigning to fog/cloud resource k is defined by formula (17).

$$pref_{t_{i,j,d}}(k) = \frac{1}{\left(\gamma^c p^{comput}_{k,t_{i,j,d}} + \gamma^m p^{mon}_{k,t_{i,j,d}}\right)} \tag{17}$$

Since the tasks are demanding to minimize computation and monetary costs, the preference relation of offloaded task $t_{i,j,d}$ over two fog/cloud resources of k_1 and k_2 is defined as follows.

$$k_1 >_{t_{i,j,d}} k_2 \Leftrightarrow pref_{t_{i,j,d}}(k_1) > pref_{t_{i,j,d}}(k_2) \tag{18}$$

The offloaded tasks are assigned to fog or cloud resources according to their preferences. When a resource has more than two requests, it selects a task with minimum computation cost. So, the preference function and preference relation of fog/cloud resource k over two tasks of t_1 and t_2 represent by formula (19).

$$pref_k(t_{i,j,d}) = \frac{1}{p^{comput}_{k,t_{i,j,d}}} \tag{19}$$

$$t_1 >_{t_{i,j,d}} t_2 \Leftrightarrow pref_k(t_1) > pref_k(t_2) \tag{20}$$

According to the above explanations, the task allocation matching algorithm presents in Algorithm 1.

The proposed *Offload-Location* mechanism is shown in Algorithm 2. This mechanism executes by a fog broker in the fog tier. It starts at lines 3-7 by considering each IoT device in a local coalition and calculating their characteristic functions according to (11); also, the offloading coalition of F_O initializes to an empty set. We considered a matrix of visited to check all possible combinations of coalitions during the merge and split operations. The merge process (lines 11-22) is started by selecting a random local coalition F_{L_i} that did not check its merging

Algorithm 1 Task-Allocation Matching.

```
1: Task-Allocation(OT,SE,SC)
2: Inputs: OT:offloaded-tasks; SE:fog servers; SC:cloud servers
3: for t=1:OT do
4:     Calculate preference list of offloaded-tasks and sort lists in decreasing order
5: while there is any unmatched offloaded-task do
6:     for t=1:OT do
7:         if t is unmatched and it's preference list is not empty then
8:             t propose to it's top-ranked choice s in the preference list
9:             remove s from preference list of t
10:    for s=1:SE+SC do
11:        if s has at least two proposals then
12:            Calculate preference list of s for her proposals and sort them in decreasing order
13:            Match s to her top-ranked proposal
```

possibility with the offloading coalition. Then, the Task-Allocation algorithm is invoked to determine the characteristic function of the coalition that attempts to merge. If the statement of $\{F_O \cup F_{L_i}\} \gg_m \{F_O, F_{L_i}\}$ is true, then F_{L_i} merges with F_O. The merge process repeats until all local coalitions are checked, or the offloading coalition has the same size as the number of IoT devices. The split process begins at line 23 by checking all partitioning of F_O to a reduced offloading coalition of F'_O and single-member local coalitions. The Task-Allocation matching and (11) are used to evaluate the characteristic function of the partitioned coalitions. It is important to note that the split process terminates by finding the first split. So if $\{F'_O, F_{L_1}, ..., F_{L_{|\hat{F}|}}\} \gg_s F_O$ is true, then a split takes place, and the process breaks (line 32). The merge and split processes are repeating until no merge and split happen. After allocating tasks to the resources, the scheduling will be done using first in first out (FCFS) method.

The next section presents an analysis on the stability and time complexity of the proposed mechanism.

4.3 Mechanism Analysis

A coalition game's mechanism is stable if it converges to a coalition structure in which no player has an incentive to leave his/her coalition [36]. To prove the stability of the proposed mechanism, we must show the stability of both Task-Allocation matching and

Algorithm 2 *Offload-Location* mechanism.

Offload-Location (T,SE,SC)
Inputs: MD: mobile devices; SE:fog servers; SC:cloud servers; T: ready tasks
$F_O = \emptyset$
$\mathcal{F} = F_O, F_{L_1}, ..., F_{L_{nm}}$
for all F_{L_i} **do**
    Calculate V(F_{L_i}) for ready-tasks in $F_{L_i} \in \mathcal{F}$ according to (11)
repeat
    $Stop \leftarrow True$
    for all $F_{L_i} \in \mathcal{F}$ **do**
        $Visited[F_O][F_{L_i}] \leftarrow False$
    repeat
        $Flag \leftarrow True$
        Randomly select $F_{L_i} \in F$ such that $Visited[F_O][F_{L_i}] = Flase$
        $Visited[F_O][F_{L_i}] \leftarrow True$
        $V(FO \cup F_{L_i}) = Task-Allocation(\{T(F_O) \cup T(F_{L_i})\}, SE, SC)$
        if $\{F_O \cup F_{L_i}\} \gg_m \{F_O, F_{L_i}\}$ **then**
            $F_O \leftarrow \{F_O \cup F_{L_i}\}$
            $F_{L_i} \leftarrow \emptyset$
        for all $F_{L_i} \in \mathcal{F}$ **do**
            if not $Visited[F_O][F_{L_i}]$ **then**
                $Flag \leftarrow Flase$
    until ($|F_O| = |MD|$) or (flag=True)
    for all partitions of F_O to $F_O = \{F'_O, \hat{F}\}$ such that $F'_O \cap \hat{F} = \emptyset \wedge F'_O \cup \hat{F} = F_O$ **do**
        $\{\hat{F}\} \leftarrow \{F_{L_1}, ..., F_{L_{|\hat{F}|}}\}$ such that $F_{L_i} = \{md_i\}$
        for all $F_{L_i} \in \{\hat{F}\}$ **do**
            Calculate $V(F_{L_i})$ for ready-tasks in $F_{L_i} \in (F)$ according to (11)
        $V(F_O{}')$= Task-Allocation($\{T(F_O{}')\}$,SE,SC)
        if $\{F_O{}', F_{L_1}, ..., F_{L_{|\hat{F}|}}\}$ **then** $\gg_s F_O$
            $F_O \leftarrow F_O{}'$
            $\mathcal{F} = \mathcal{F} \cup \{F_{L_1}, ..., F_{L_{\hat{F}}}\}$
            $Stop \leftarrow Flase$
            Break
until $stop = True$
return $\{F_O, F_{L_i} \in \mathcal{F}\}$

Offload-Location mechanism. According to [12, 16], a stable matching exists for every marriage market. Since many-to-one matching is a type of marriage market, a stable matching exists for the Task-Allocation matching. For the stability of the *Offload-Location* mechanism, we have the following theorem:

Theorem 1 *The proposed Offload-Location mechanism results in a stable coalition structure.*

Proof The coalition structures generate using merge and split operations such that at least one player is happier with the obtained coalition structure, and no player will be disappointed by choosing this coalition structure. So, the merge and split operations do not visit any coalition structure twice. Since the number of possible coalitions is finite, the proposed *Offload-Location* mechanism terminates in a stable coalition structure. □

The time complexity of the proposed mechanism is analyzed by evaluating the number of merge and split operations in the worst-case scenario with n IoT devices. In the worst-case, the merge operation results in an offloading coalition with the size of n, which requires the total number of $\frac{n(n-1)}{2}$ attempts in the order of $O(n^2)$. Also, in the worst-case scenario, the split operation search over all subsets of an offloading coalition of size n, which is in the order of $O(2^n)$. However, it is essential to note that the worst-case scenario is improbable in the *Offload-Location* mechanism for both merge and split operations. Since all coalitions are not feasible, subsequent merges of an infeasible coalition do not check in the merge operation. Also, after finding the first split, there is no need to check other subsets of the offloading coalition, and the split process terminates. These conditions decrease the probability of occurring the worst-case scenario for the merge and split operation significantly.

5 Evaluations

5.1 Simulation Environment

A simulation environment was implemented using MATLAB to study and analyze the proposed method, including 10 IoT devices, eight fog servers, and five cloud servers. The experiments are performed on a laptop computer with Intel(R) Core(TM) i5 2.5GHz CPU and 8 GB RAM, and a high-performance computing (HPC) center with up to 80 cores from two CPU types of Intel(R) Xeon(R) 2.4GHz, and Intel(R) Xeon(R) 3.33GHzup. We have considered homogeneous resources with the same processing power in each tier; also, one VM is assumed for each server. Because of the limitation in real benchmarks for fog environment [34], we have used configurations and parameters of previous studies. IoT devices' processing

power is 500 MIPS, and the processing power of fog and cloud servers are respectively 8 and 10 times faster than IoT devices [20]. Each workflow enters to an IoT device, and local execution of this workflow's tasks perform on that particular IoT device. IoT devices do not receive any price for processing the tasks of their input workflows. The fog and cloud servers are also in charge of executing the offloaded tasks received from all IoT devices. However, fog and cloud servers receive a price for processing the tasks. We assumed fog and cloud servers are charging for leasing their resources with a price of 1 and 8 dollars in billing periods of 1 hour, respectively [42]. Since edge/fog servers are closer to the IoT devices, these devices are communicating in a LAN environment. However, the cloud servers are far from IoT devices, so using the WAN protocol. We have considered the bandwidth of LAN as 2000 Mb and the bandwidth of WAN as 500 Mb. The nodes of each fog and cloud tiers have access to each other, and the intra-bandwidth between fog servers is 4000 Mb, and the intra-bandwidth between cloud servers is 1000 Mb [20]. Table 3 represents the parameters of devices in the environment. Also, the bandwidth parameters of the network are represented in Table 4.

5.2 Workflow Ensembles

We used four types of scientific workflows in the form of workflow ensembles from real applications to investigate the behavior of the proposed method [39]. The Cybershake is a computational study in physics that uses a set of scientific workflows to calculate the seismic hazard curves [10]. The Montage is a computational study for generating science-grade mosaics of the sky. The Montage's workflows generate a single image mosaics individually combining to obtain a complete image mosaic [38]. The objective of the Ligo project [29] is to detect and measure gravitational waves using Ligo's workflows. The Sipht [44] is a bioinformatics project that conducts a comprehensive search for small untranslated RNAs (sRNAs) that regulate several processes such as secretion or virulence in bacteria [10]. Readers are referred to [26] for more details on the scientific projects and workflows.

These scientific workflows contain many CPU-intensive and data-intensive tasks with different structures. Based on the simulation environment's parameters, we will study each application's behavior using the proposed method. Our analyzes investigate ensembles that contain workflows with different sizes (i.e., different numbers of tasks). For each scientific application, workflows with 25, 30, 50, 60, 100 tasks were considered in the ensembles. We considered two different types of workflow ensembles for each application's workflows: constant and uniform. A constant ensemble consists of workflows with a constant size, and a uniform ensemble of an application includes a uniform distribution of workflows with different sizes of that application. We considered the workflow size of constant ensembles' workflows is 100. Also, the ensembles' predefined size (i.e., the number of workflows in an ensemble) is 100. We define the budget and deadline constraints of ensembles and the minimum and maximum values of these parameters, follow what has explained in [31]. The ensemble's minimum price is defined according to the minimum price of executing an individual workflow. The maximum price of an ensemble is estimated according to the maximum price of executing all workflows. Also, the minimum deadline computes according to the minimum time needed to execute the tasks in the critical path of an individual workflow, and the maximum deadline evaluates by the time needed to execute the tasks in the critical path of all workflows. After defining the minimum and maximum values of the deadline and budget parameters, the values will divide into five identical intervals to evaluate the effect of changing the budget

Table 3 Setting of devices

	IoT	Fog	Cloud	Ref
Number of devices	10	8	5	Goudarzi et al. [20]
Processing power	500	500*8	500*12	Goudarzi et al. [20]
Price	0	1$	8$	Rodriguez and Buyya [42]
Billing period	–	1 hour	1 hour	Rodriguez and Buyya [42]

Table 4 Bandwidth parameters of the network

parameter	value	Ref.
Bandwidth of LAN	2000 Mb	Goudarzi et al. [20]
Bandwidth of WAN	500 Mb	Goudarzi et al. [20]
Intra-bandwidth of cloud devices	1000 Mb	Goudarzi et al. [20]
Intra-bandwidth of fog devices	4000 Mb	Goudarzi et al. [20]

and deadline parameters on the performance of the workflow scheduling method.

5.3 Analyzed Methods

We studied the scheduling algorithms' performance using three measures: the number of completed workflows from the ensemble, the execution time, and the monetary cost or price paid for executing the tasks considering the ensemble's QoS constraints of deadline and budget. We considered the number of completed workflows in evaluating the last two performance measures because an algorithm with more completed workflows may consume more time and monetary cost. The following algorithms are implemented to analyze the efficiency of the proposed method.

- fog-offloading-coalition (FOC): It is the proposed method in an IoT-fog-cloud environment that uses the coalition game to determine the offloading set.
- random-offloading-coalition (ROC): It is the proposed method in an IoT-fog-cloud environment such that the offloading coalition game is not employing and the offloading set is determined randomly.
- cloud-offloading-coalition (COC): It is the proposed scheme in an IoT-cloud environment without employing fog devices.
- DPDS [31]: It is an algorithm for cloud provisioning and scheduling of workflow ensembles in a cloud environment. In this algorithm, we do not consider the local execution of IoT devices. Also, fog devices are relinquished. The DPDS had ignored the communication delay of transmitting data among tasks, but we have considered this communication delay in our analysis. This algorithm can assume as a reference point to analyze the performance of dynamic ensemble scheduling algorithms.

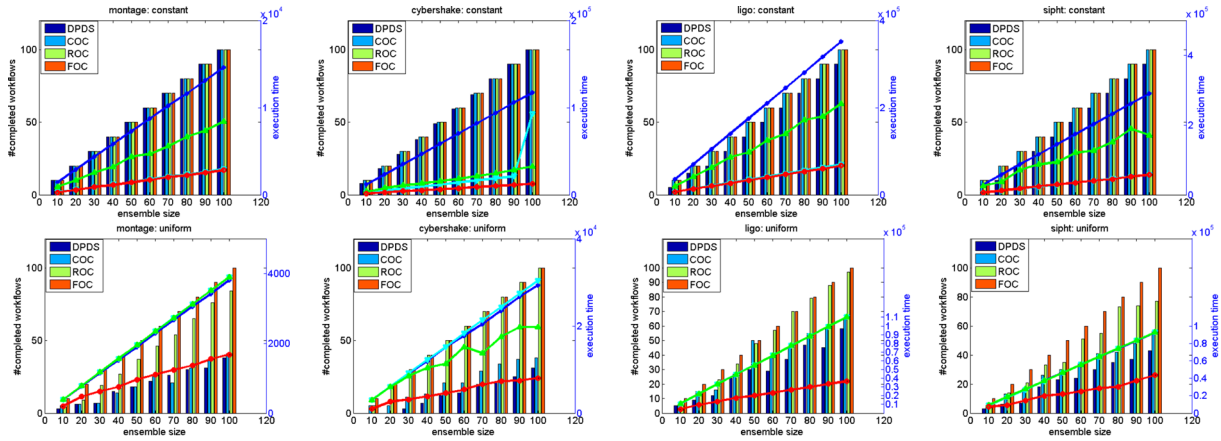
We named the COC and DPDS as cloud-based techniques that rely on cloud servers to execute the tasks. The FOC and ROC are also named as fog-based approaches that employ fog servers besides cloud servers.

5.4 Performance Evaluation

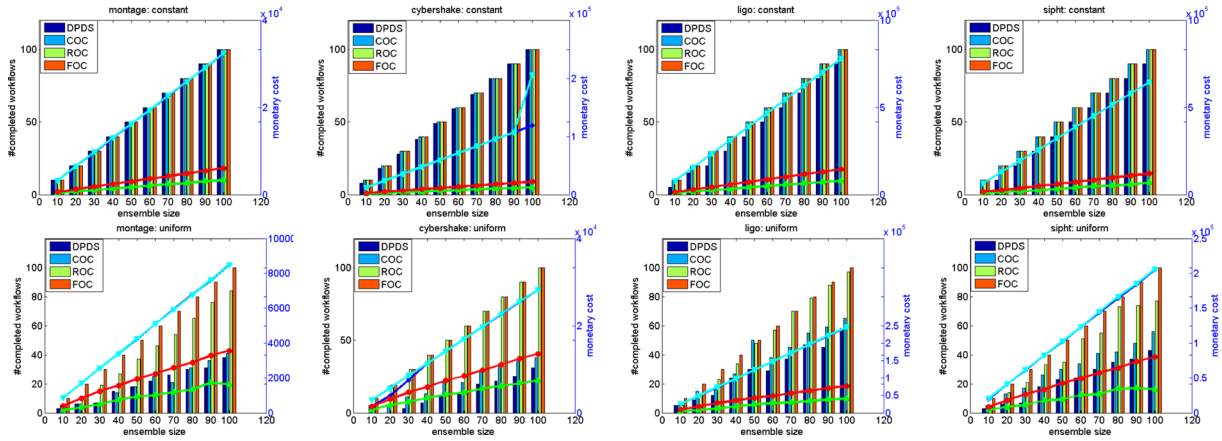
This section compares the algorithms' functionality under different ensembles' sizes. We assume the maximum value of the budget and deadline constraints in the simulations to permit the maximum completion of workflows. Figure 2 represents the results of three performance measures for all algorithms and all workflow ensembles in two cases of constant and uniform, and different sizes for the ensembles (10 through 100 workflows). In all plots, the x-axis indicates ensembles' size. The left y-axis is the number of completed workflows. The right y-axis is the execution time or monetary cost of executing workflows within the ensemble's deadline and budget constraints.

Figure 2 implies that the studied algorithms' execution time and monetary cost increase by incrementing the ensembles' sizes. Violating the deadline and budget constraints yields to failure in completing the workflows of the ensembles. Hence, the number of completed workflows by an algorithm depends on this algorithm's ability to minimize the execution time and monetary cost. The results of constant ensembles in all workflows indicate that the proposed scheme can complete all workflows of the ensemble in both cases of cloud and fog-based techniques. Because of the absence of IoT devices in the DPDS and a policy to overcome the communication delay, offloading all tasks to cloud servers encounters the bandwidth challenge. The DPDS cannot complete all ensembles' workflows even with consuming maximum available time, except for the Montage's constant ensembles with short running time and communication data.

The studied techniques' overall behavior is almost identical in constant and uniform ensembles with different applications. The differences are due to varying workflow structures and the values of running times and communication data. In constant ensembles, the FOC and COC are performing better than the ROC. Simultaneously, the DPDS has the worst performance in case of execution time and the number of completed workflows. However, for the Cybershake ensembles, the COC has a little different behavior. Cybershake workflows have a short critical path consist of a mixture of data-intensive and CPU-intensive tasks with high variances. Hence, the bandwidth challenge is becoming more critical in these ensembles for cloud-based schemes. So, the COC performance is degrading with increasing the size of constant ensembles of Cybershake. Because of offloading the tasks to fog devices, the ROC outperforms the COC in terms of execution time by increasing the Cybershake ensembles' size. Although, it cannot dominate the FOC due to selecting the offloading coalition randomly. This result verifies the necessity of selecting the appropriate set of offloaded tasks in the offloading procedure



(a) Execution time in terms of the number of completed workflows from the ensemble



(b) Monetary cost in terms of the number of completed workflows from the ensemble

Fig. 2 Execution time and monetary cost in terms of the number of completed workflows from the ensemble

to orchestrate the fog computing environment optimally. The results of the ROC scheme's monetary cost indicate that the random selection of offloaded tasks reduces the monetary cost compared to other analyzed schemes. On the other hand, the DPDS and COC are paying the maximum price, while the DPDS cannot complete all workflows in some cases.

In the uniform case, workflows' size has a remarkable effect on analyzed algorithms' behavior so that only FOC can complete all workflows in all cases. Although the COC keeps pace with FOC in terms of execution time in constant ensembles, its performance is degrading in the uniform case, and it is proportional to that of DPDS. These results confirm the significant drawback of variances on communication data of the tasks and bandwidth-delay of WAN on the performance of an ensemble scheduling approach in cloud computing. In this case, the execution time and monetary cost diagrams of the analyzed cloud-based schemes are coinciding, and there are significant differences with the corresponding curves in the FOC, while the cloud-based techniques can not complete all workflows from the ensemble. It is indicated that the execution time and monetary cost of cloud-based schemes in the uniform case are close to the available deadline and budget of the ensembles. The observed behavior of the COC with Cybershake ensembles is apparent in the uniform case. While the DPDS attempts to minimize the provisioned cloud servers, the COC tackle both inter-cloud communication delay and communication delay between cloud and IoT devices. So, the performance of the COC degrades compared to the DPDS in some cases. Nevertheless, our proposed cloud-based scheme can complete more

Table 5 The number of completed workflows

		DPDS	COC	ROC	FOC
Montage	*constant*	100	100	100	100
	uniform	30	41	84	100
Cybershake	*constant*	100	100	100	100
	uniform	31	38	100	100
Ligo	*constant*	90	100	100	100
	uniform	58	65	97	100
Sipht	*constant*	90	100	100	100
	uniform	43	56	77	100

workflows, thanks to IoT devices. The ROC's execution time is the same as that of cloud-based techniques while paying the least price compared to all analyzed schemes, thanks to the random selection of locally executed tasks. Because of Cybershake workflows' properties and using the fog devices in the ROC, this algorithm can complete all workflows of the ensembles and perform better than cloud-based schemes in terms of execution time and the number of completed workflows.

Observations on different workflow ensembles confirm the influence of the structure of input workflows on the workflow scheduling approach's outcome. The results imply that workflows' characterizations, including sizes of tasks' running time and data files, and the critical path's length, affect the algorithm's behavior. This conclusion is even more critical in fog computing's heterogeneous environment, which employs offloading decision to distribute tasks among resources over the IoT to the cloud continuum.

Table 5 represents an analysis of the number of completed workflows using implemented methods. All schemes of the proposed method complete all workflows in constant ensembles, with an average improvement of 5% compared to the DPDS. However, in uniform ensembles, FOC has significantly better performance than analyzed schemes. In this case, the average improvement of FOC from the DPDS and the COC are 59.5% and 50%, respectively. It confirms the remarkable effect of using edge/fog nodes in executing scientific applications. Also, we see an average improvement of 10.5% from ROC. This result verifies that proper use of fog devices significantly affects system utilization and establishing end-user requirements. No appropriate orchestration of homogeneous nodes in different tiers of a fog environment may degrade performance to workflow ensemble scheduling technique in cloud computing.

5.5 Varying Deadline and Budget

Figure 3 represents the number of completed workflows from ensembles in the analyzed algorithms for different budgets. The x-axis shows the considered intervals of the ensemble's available budget, explained in Section 5.2. This figure indicates that the number of completed workflows from ensembles grows by increasing the available budget. The FOC can complete all workflows from constant ensembles, except in the Ligo and Sipht ensembles, which have tasks with long running time and data size. The FOC is the only scheme completing some workflows from ensembles with the minimum budget. The COC can complete less than half of the Montage ensembles in the constant case, with the minimum budget. It confirms the short running time of the Montage workflows. Interfering with the size of workflows in uniform ensembles, the

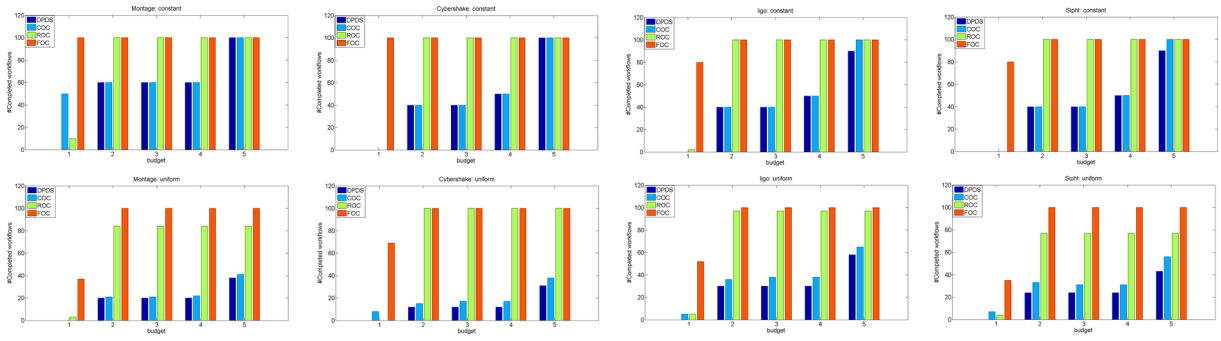


Fig. 3 The number of completed workflows from the ensembles by varying budget constrains

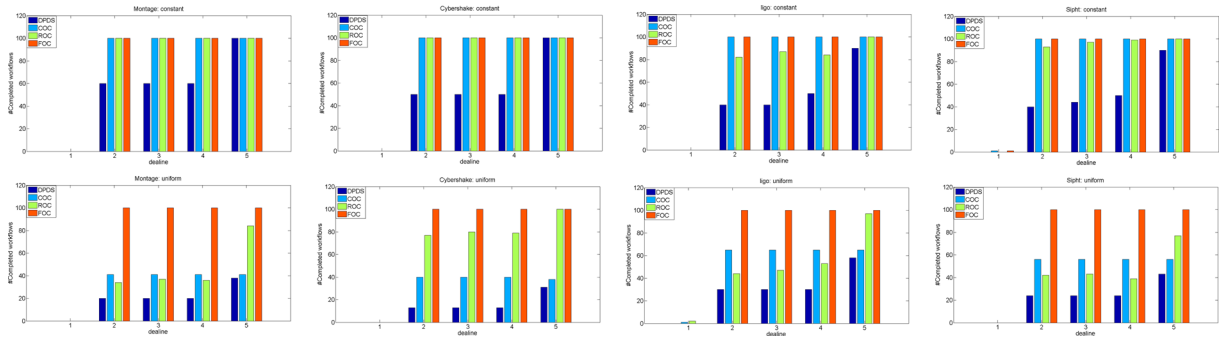


Fig. 4 The number of completed workflows from the ensembles by varying deadline constrains

capability of all schemes in completing workflows is reduced, and it is only the FOC that is robust and has been able to complete all workflows, except where the budget is the minimum value. The ROC can compete with the FOC and execute all workflows in most cases. It indicates that this method consumes low cost and is robust in budget constraints; this result also concludes from Fig. 2.

Figure 4 represents the number of completed workflows from ensembles by changing the deadline constraints. The x-axis shows the considered intervals of the ensemble's deadline, explained in Section 5.2. The COC can complete all work ows in constant ensembles, confirming that this method is less sensitive to the deadline constraint than the DPDS. However, the COC can not complete all workflows in all uniform ensembles, proving the significant effect of varying workflows' size on the workflow ensemble scheduling method's performance. Except for the minimum deadline, the FOC can complete all workflows in constant and uniform ensembles. Although the ROC had acceptable results in the number of completed workflows by varying budgets in random ensembles, this method can not retain its performance when the deadline varies. This behavior proves from the low monetary cost and the high execution time of this method in Fig. 2.

Since cloud servers are powerful, they are less sensitive to deadline constraints. So, we can see COC's better performance when the deadline varies compared to changing the budget. The DPDS algorithm could compete with the COC in terms of the number of completed workflows by varying the budget. However, since this technique does not have any solution in minimizing the execution time, the results degrade when the deadline varies.

6 Conclusion and Future Works

This study focuses on computation offloading and task allocation problems to address workflow ensemble scheduling in a fog environment with multiple IoT, fog, and cloud devices. This work aims to minimize the computation cost and minimize the monetary cost of executing the tasks with an ultimate end of maximizing the number of completed workflows, considering the ensemble's available deadline and budget. We have formulated this problem as an integer programming problem and designed a game-theoretic mechanism to address it. The mechanism is using coalition games for determining the set of offloaded tasks and the tasks executing locally. We are also employing a stable matching algorithm for assigning the offloaded tasks to fog or cloud servers.

We analyzed the proposed scheme using different scientific workflows under variable circumstances. The results confirm that employing edge/fog nodes in the proposed workflow ensemble scheduling method improved performance in terms of the number of completed workflows by about 50%. The proposed mechanism that specifies the offloaded tasks enhances the number of completed workflows, about 10.5%, compared to a task allocation algorithm without determining the optimal set of offloaded tasks. This finding highlights the importance of employing an efficient offloading technique in the performance of the scheduling procedure. The observations showed the effect of the structure of input workflows on the scheduling scheme's execution time and the monetary cost. Hence an interesting aspect of extending the current study is considering the structure of workflows. Also, considering the provisioning problem in [31]

suggests extending our model for investigating this problem.

References

1. Abbasi, M., Pasand, E.M., Khosravi, M.R.: Workload allocation in iot-fog-cloud architecture using a multi-objective genetic algorithm. J. Grid Comput. 1–14 (2020)
2. Apt, K.R., Witzel, A.: A generic approach to coalition formation. Int. Game Theor. Rev. **11**(03), 347–367 (2009)
3. Aral, A., Brandic, I., Uriarte, R.B., De Nicola, R., Scoca, V.: Addressing application latency requirements through edge scheduling. J. Grid Comput. **17**(4), 677–698 (2019)
4. Arisdakessian, S., Wahab, O.A., Mourad, A., Otrok, H., Kara, N.: Fogmatch: An intelligent multi-criteria iot-fog scheduling approach using game theory. IEEE/ACM Trans. Netw. (2020)
5. Bilbao, J.M.: Cooperative games on combinatorial structures, vol. 26. Springer Science & Business Media, Berlin (2012)
6. Bogomolnaia, A., Jackson, M.O., et al.: The stability of hedonic coalition structures. Games Econom. Behav. **38**(2), 201–230 (2002)
7. Bryk, P., Malawski, M., Juve, G., Deelman, E.: Storage-aware algorithms for scheduling of workflow ensembles in clouds. J. Grid Comput. **14**(2), 359–378 (2016)
8. Buyya, R., Srirama, S.N.: Fog and edge computing: principles and paradigms. Wiley, New York (2019)
9. Chen, L., Xu, J.: Socially trusted collaborative edge computing in ultra dense networks. In: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, pp. 1–11 (2017)
10. Cybershake project. https://strike.scec.org/scecpedia/CyberShake_Study_13.4
11. Deelman, E., Gil, Y.: Workshop on the challenges of scientific workflows. Information Sciences Institute (2006)
12. Demange, G., Gale, D.: The strategy structure of two-sided matching markets. Econometrica: J. Econom. Soc. 873–888 (1985)
13. Driessen, T.S.: Cooperative games, solutions and applications, vol. 3. Springer Science & Business Media, Berlin (2013)
14. Fan, W., Liu, Y., Tang, B., Wu, F., Wang, Z.: Computation offloading based on cooperations of mobile edge computing-enabled base stations. IEEE Access **6**, 22622–22633 (2017)
15. Ferguson, T.S.: A course in game theory world scientific (2018)
16. Gale, D., Shapley, L.S.: College admissions and the stability of marriage. Am. Math. Mon. **69**(1), 9–15 (1962)
17. Gao, L., Moh, M.: Joint computation offloading and prioritized scheduling in mobile edge computing. In: 2018 International Conference on High Performance Computing & Simulation (HPCS), pp. 1000–1007. IEEE (2018)
18. Gao, X., Huang, X., Bian, S., Shao, Z., Yang, Y.: Pora: Predictive offloading and resource allocation in dynamic fog computing systems. IEEE Int. Things J. **7**(1), 72–87 (2019)
19. Genez, T.A., Bittencourt, L.F., Sakellariou, R., Madeira, E.R.: A flexible scheduler for workflow ensembles. In: Proceedings of the 9th International Conference on Utility and Cloud Computing, pp. 55–62 (2016)
20. Goudarzi, M., Wu, H., Palaniswami, M.S., Buyya, R.: An application placement technique for concurrent iot applications in edge and fog computing environments. IEEE Trans. Mob. Comput. 1–1 (2020)
21. Guo, K., Sheng, M., Quek, T.Q., Qiu, Z.: Task offloading and scheduling in fog ran: A parallel communication and computation perspective. IEEE Wirel. Commun. Lett. **9**(2), 215–218 (2019)
22. Hosseinzadeh, M., Ghafour, M.Y., Hama, H.K., Vo, B., Khoshnevis, A.: Multi-objective task and workflow scheduling approaches in cloud computing: a comprehensive review. J. Grid Comput. 1–30 (2020)
23. Hu, P., Dhelim, S., Ning, H., Qiu, T.: Survey on fog computing: architecture, key technologies, applications and open issues. J. Netw. Comput. Appl. **98**, 27–42 (2017)
24. Huedo, E., Montero, R.S., Moreno-Vozmediano, R., Vázquez, C., Holer, V., Llorente, I.M.: Opportunistic deployment of distributed edge clouds for latency-critical applications. J. Grid Comput. **19**(1), 1–16 (2021)
25. Jošilo, S., Dán, G.: Decentralized scheduling for offloading of periodic tasks in mobile edge computing. In: 2018 IFIP Networking Conference (IFIP Networking) and Workshops, pp. 1–9. IEEE (2018)
26. Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., Vahi, K.: Characterizing and profiling scientific workflows. Futur. Gener. Comput. Syst. **29**(3), 682–692 (2013)
27. Juve, G., Deelman, E., Vahi, K., Mehta, G., Berriman, B., Berman, B.P., Maechling, P.: Data sharing options for scientific workflows on amazon ec2. In: SC'10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–9. IEEE (2010)
28. Leyton-Brown, K., Shoham, Y.: Essentials of game theory: A concise multidisciplinary introduction. Synt. Lect. Artif. Intell. Mach Learn. **2**(1), 1–88 (2008)
29. Ligo project. https://pegasus.isi.edu/application-showcase/ligo/
30. Liu, Y., Xu, C., Zhan, Y., Liu, Z., Guan, J., Zhang, H.: Incentive mechanism for computation offloading using edge computing: A stackelberg game approach. Comput. Netw. **129**, 399–409 (2017)
31. Malawski, M., Juve, G., Deelman, E., Nabrzyski, J.: Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds. Futur. Gener. Comput. Syst. **48**, 1–18 (2015)
32. Mashayekhy, L., Grosu, D.: A merge-and-split mechanism for dynamic virtual organization formation in grids. IEEE Trans. Parall. Distribut. Syst. **25**(3), 540–549 (2014)
33. Mashayekhy, L., Nejad, M.M., Grosu, D.: Cloud federations in the sky: Formation game and mechanism. IEEE Trans. Cloud Comput. **3**(1), 14–27 (2015)
34. McChesney, J., Wang, N., Tanwer, A., de Lara, E., Varghese, B.: Defog: fog computing benchmarks. In: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, pp. 47–58 (2019)

35. Morales, L.E.P.: Efficient support for data-intensive scientific workflows on geo-distributed clouds. Ph.D thesis (2017)
36. Nisan, N., Ronen, A.: Algorithmic mechanism design. Games Econom. Behav. **35**(1-2), 166–196 (2001)
37. Osborne, M.J., et al.: An Introduction to Game Theory, vol. 3. Oxford University Press, New York (2004)
38. Montage project. http://montage.ipac.caltech.edu
39. Pegasus project. https://pegasus.isi.edu/application-showcase/
40. Pietri, I., Malawski, M., Juve, G., Deelman, E., Nabrzyski, J., Sakellariou, R.: Energy-constrained provisioning for scientific workflow ensembles. In: 2013 International Conference on Cloud and Green Computing, pp. 34–41. IEEE (2013)
41. Ren, J., Zhang, D., He, S., Zhang, Y., Li, T.: A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet. ACM Comput. Surv. (CSUR) **52**(6), 1–36 (2019)
42. Rodriguez, M.A., Buyya, R.: Scheduling dynamic workloads in multi-tenant scientific workflow as a service platforms. Futur. Gener. Comput. Syst. **79**, 739–750 (2018)
43. Shakarami, A., Ghobaei-Arani, M., Masdari, M., Hosseinzadeh, M.: A survey on the computation offloading approaches in mobile edge/cloud computing environment: A stochastic-based perspective. J. Grid Comput. 1–33 (2020)
44. Sipht project. http://newbio.cs.wisc.edu/sRNA/
45. Stavrinides, G.L., Karatza, H.D.: A hybrid approach to scheduling real-time iot workflows in fog and cloud environments. Multimed. Tools Appl. **78**(17), 24639–24655 (2019)
46. Tianze, L., Muqing, W., Min, Z., Wenxing, L.: An overhead-optimizing task scheduling strategy for ad-hoc based mobile edge computing. IEEE Access **5**, 5609–5622 (2017)
47. Toczé, K., Nadjm-Tehrani, S.: A taxonomy for management and optimization of multiple resources in edge computing. Wirel. Commun. Mob. Comput. 2018 (2018)
48. Velasquez, K., Abreu, D.P., Assis, M.R., Senna, C., Aranha, D.F., Bittencourt, L.F., Laranjeiro, N., Curado, M., Vieira, M., Monteiro, E., et al.: Fog orchestration for the internet of everything: state-of-the-art and research challenges. J. Int. Serv. Appl. **9**(1), 14 (2018)
49. Xie, Y., Zhu, Y., Wang, Y., Cheng, Y., Xu, R., Sani, A.S., Yuan, D., Yang, Y.: A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud–edge environment. Futur. Gener. Comput. Syst. **97**, 361–378 (2019)
50. Xu, X., Chen, Y., Yuan, Y., Huang, T., Zhang, X., Qi, L.: Blockchain-based cloudlet management for multimedia workflow in mobile cloud computing. Multimed. Tools Appl. **79**, 9819–9844 (2019)
51. Xu, X., Liu, Q., Luo, Y., Peng, K., Zhang, X., Meng, S., Qi, L.: A computation offloading method over big data for iot-enabled cloud-edge computing. Futur. Gener. Comput. Syst. **95**, 522–533 (2019)
52. Yi, S., Hao, Z., Zhang, Q., Zhang, Q., Shi, W., Li, Q.: Lavea: Latency-aware video analytics on edge computing platform. In: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, SEC '17. Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3132211.3134459
53. Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J., Jue, J.P.: All one needs to know about fog computing and related edge computing paradigms: a complete survey. J. Syst. Archit. **98**, 289–330 (2019)
54. Yu, J., Buyya, R.: A taxonomy of scientific workflow systems for grid computing. ACM Sigmod Record **34**(3), 44–49 (2005)
55. Zhang, K., Mao, Y., Leng, S., Maharjan, S., Zhang, Y.: Optimal delay constrained offloading for vehicular edge computing networks. In: 2017 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2017)
56. Zhou, B., Srirama, S.N., Buyya, R.: An auction-based incentive mechanism for heterogeneous mobile clouds. J. Syst. Softw. **152**, 151–164 (2019)