IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 31, 2019, accepted August 13, 2019, date of publication August 19, 2019, date of current version August 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2936116

A Method Based on the Combination of Laxity and Ant Colony System for Cloud-Fog Task Scheduling

JIUYUN XU[1], (Member, IEEE), ZHUANGYUAN HAO[1], RURU ZHANG[2], AND XIAOTING SUN[1]
[1]College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China
[2]China Mobile (Suzhou) Software Technology Company, Suzhou 215010, China

Corresponding author: Jiuyun Xu (jiuyun.xu@computer.org)

ABSTRACT In today's Internet of Things research community, Cloud-fog framework is a potential technology for Internet of Things to support energy consumption of an IoT system and delay-sensitive applications that require almost real-time responses. However, how to schedule the computational tasks which is to offload to fog nodes or cloud nodes is not fully addressed until now. In this paper, in order to solve the complex task scheduling problem with some priority constraints of IoT applications taking into account the energy consumption and reducing energy consumption on the condition of satisfying the mix deadline, we formulate an associated task scheduling problem into a constrained optimization problem in cloud-fog environment. A laxity and ant colony system algorithm(LBP-ACS) is put forward to tackle this problem. In this algorithm, a strategy of task scheduling is not only considering the priority of a task, but also its finished deadline. In order to handle the sensitivity of task delay, the laxity-based priority algorithm is adopted to construct a task scheduling sequence with reasonable priority. Meanwhile, to minimize the total energy consumption, the constrained optimization algorithm based on ant colony system algorithm is used to obtain the approximate optimal scheduling scheme in the global. Compared with other algorithms, the experimental results show that the proposed algorithm can effectively reduce the energy consumption of processing all tasks, while ensuring reasonable scheduling length and reducing the failure rate of associated tasks scheduling with mixed deadlines.

INDEX TERMS IoT, energy consumption, task scheduling, ant colony algorithm, laxity.

I. INTRODUCTION

Due to potential computation, storage and processing capacity, cloud computing becomes primary computing paradigm to supported the IoT scenario and to leverage a massive heterogeneous set of devices can access internet anywhere, anytime [1]. In the coming era of the Internet of Things (IoT), it is estimated that above 50 billions of devices and smart objects with huge capacity for collecting and exchanging information intelligently will be interconnected in 2020 [2]. These large deal of devices will generate a tidal wave of data or service requests in IoT Scenario. Generally, many data stored to cloud where the resource is deployed far away from the end users over the Internet will not only pose heavy burden to network performance and network bandwidth but also result in unbearable transmission latency which is degraded quality of service (QoS) to end users [3]–[5]. Especially, it is not providing low-latency guaranteed to delay-sensitive applications which are very common in IoT scenarios [6]–[8].

Recently, Fog computing, which is proposed by Cisco in 2012 [9], is great attention for its potential in satisfying the requirements not yet well-addressed by the current Cloud Computing. The fog computing paradigm extends the computational resources available in the Cloud data center to the edge of the network as desired by IoT solutions. Fog computing aims to process part of the applications or IoT data locally on network edge, which can reduce the burden of data transferred via Internet and meet the needs of users. In the Cloud-Fog framework, Fog nodes are deployed at the edge of network, such as forest park, bus terminal and shopping center. Fog node can to pre-store cloud data and handle request between the IoT devices and the cloud data center. In order to reducing the delay and meet the needs of delay-sensitive applications, request will be processed in fog

The associate editor coordinating the review of this article and approving it for publication was Shuiguang Deng.

nodes that is one or two hops to the data sources, while latency tolerant and large-scale tasks can still be efficiently processed by the cloud.

A. MOTIVATION

In the IoT scenario represented by the intelligent transportation system, smart devices such as vehicles, traffic lights, mobile phones, sensors, CCTV surveillance cameras are connected to the fog device through a wireless network, and the fog device is connected to the cloud server through the optical fiber. The fog node receives the data in the traffic environment through the sensor, thereby detecting the speed of the nearby pedestrians and vehicles, and further interacting with the adjacent signal lights. Based on the above processing information, the fog device sends a warning message to the vehicle to avoid collision or congestion by adjusting the adjacent green light period. At the same time, smart devices collect traffic information such as peak hours and emergency locations to the cloud server for statistical analysis, and finally report the road condition information to the user. Throughout the process, the vehicle receives warning messages, smart lights to adjust the period and the user's smart device to receive traffic information, where the deadlines for these application tasks are different, and subtasks have dependencies. Through the cooperation of fog nodes and cloud nodes, which process collaboratively of applications with interdependence and mixing deadlines to meet low latency. At the same time, cloud server analyse, process traffic information and provide road condition information, finally ensuring traffic safety and stability. At the same time, taking into account the general trend of global energy consumption soaring, low energy consumption computing needs to be resolved. Optimizing energy consumption, on the one hand, it can reduce production costs. On the other hand, it can save energy and reduce emissions. Finally, it can achieve green computing and protect the environment. In the above scenario, the resources are dynamically utilized to ensure that the hybrid deadlines of DAG tasks are met through the collaborative calculation of cloud resources. Therefore, this paper will study the problem of optimized energy consumption scheduling for interdependent tasks with mixed deadlines in the cloud and fog computing system.

In this paper, we focus on associated task scheduling problem in cloud-fog environment. For the scheduling problems of complex tasks with priority constraints in IoT applications, a task scheduling strategy is proposed, which is the combination of laxity-based priority algorithm and ant colony system. In the process of calculating the priority, the limitation of the task deadline is considered. In order to enhance the sensitivity of task delay, the laxity-based priority algorithm is adopted to construct a task scheduling sequence with reasonable priority. At the same time, in order to minimize the total energy consumption, the constrained optimization algorithm based on ant colony system algorithm is used to obtain the approximate optimal scheduling scheme in the global. Compared with other algorithms, the experimental results show that the proposed algorithm can effectively reduce the energy consumption of processing all tasks, while ensuring reasonable scheduling length and reducing the failure rate of associated tasks scheduling with mixed deadlines.

B. THE CONTRIBUTIONS OF THIS PAPER

The contributions of this paper are as follows:

- When calculating the associated task priority, in order to enhance the sensitivity of task delay, we proposed the laxity-based priority algorithm to construct a task scheduling sequence with reasonable priority.
- In order to minimize the total energy consumption, the constrained optimization algorithm based on ant colony system algorithm is used to obtain the approximate optimal scheduling scheme in the global.
- Compared with other algorithms, the experimental results show that the proposed algorithm can effectively reduce the energy consumption of processing all tasks, while ensuring reasonable scheduling length and reducing the failure rate of associated tasks scheduling with mixed deadlines.

The remainder of the paper is organized as follows: Section II presents a survey of related work about associated task scheduling in cloud-fog environments. In section III, we firstly introduce the system architecture. Next, we formulate a mathematical formulation for associated tasks scheduling policy. Finally, the associated task scheduling strategy based on laxity and ant colony system is proposed. We describe some experimental results in section IV, followed by our conclusions in Section V.

II. RELATED WORK

As a complement to cloud computing, fog computing is a novel introduced paradigm, the number of task scheduling mechanisms specifically aiming at cloud and fog computing framework is quite limited so far. Although there has been a lot of work on task scheduling for cloud computing, it cannot be directly applied to the cloud computing framework.

From the viewpoint of IoT applications, Zhao *et al.* [10] pointed out that it is necessary to consider how to deploy the fog node resources for hybrid computing scenario, so as to realize efficient coordination of cloud and fog computing resources and how to implement appropriate scheduling of tasks and resources according to business needs.

In the cloud-fog architecture, in order to minimizing service delay and ensuring service quality, Souza *et al.* [11], [12] transform the QoS-aware service allocation problem into an integer optimization problem.

Xiuli et al. [13] introduced the cloud and fog network architecture into the field of car networking to solve the problem of high latency and not to support for mobility and location awareness in today's car networking. An improved particle swarm optimization algorithm is proposed to reduce delay and improve quality of service QoS. Reference [14] proposed an architecture based on fog regions and clouds, and designed an efficient task scheduling mechanism for heuristic

scheduling algorithm to minimize the task completion time and improve user experience.

Although there have some research on MEC, Deng *et al.* [15] first mathematically formulates the task offloading decision problem. It decomposes the primal problem into three sub-problems of corresponding subsystems, which can be independently solved. And author has compared the energy consumption and system delay between cloud computing, edge computing and cloud-edge computing. All of them solve a binary computation offloading problem in nature, namely the architecture they have proposed only includes cloud server or edge server. However, these works only consider independent tasks and ignore associated tasks. In IoT Scenario, each application is comprised of multiple interdependent tasks and each of which is specified by an amount of processing works.

Generally speaking, associated tasks have a priority constraint relationship with each other. When all necessary input data form its predecessor tasks arrive at the target resources, The subsequent task will start to be process [16]. An associated scheduling problem can be represented by a direct acyclic graph (DAG) [17], which is usually considered is an non-deterministic polynomial-time complete (NP-complete) problem. The heuristic algorithms is used to finding approximate optimal solutions. The heterogeneous earliest finish time (HEFT) algorithm is the most popular and widely used algorithm, which includes two main phases: a task prioritizing phase for computing the priorities of all tasks based on upward rank value and assigning the selected task to the processor which minimizes the task's finish time [18], [19]. Reference [20] proposed an extended HEFT cost-aware scheduling heuristic algorithm that recursively calculates the priority value of each task based on computational cost and communication cost. Then, the scheduler assigns each task to the cheapest virtual machine.

In [21], Pham and Huh propose a heuristic-based algorithm to deal with the DAG-based task scheduling problem. When the user's own fog device can not meet the demand, it enables the leasing cloud resource to handle the tasks. The main objective is achieving balance between the makespan and the monetary cost of cloud resources. However, this paper does not take into account the needs of latency sensitive applications that require task processed to complete within a certain delay. when formulating the task scheduling strategy, it is becoming increasingly more important to consider the deadline of the task.

In [22], Deng *et al.* studied the allocation of workload to reduce energy consumption and latency in the cloud and fog computing system. The cloud and fog system was divided into three independent sub-systems, using separately convex optimization technique, on linear integer programming and Hungarian method. These optimization algorithms solve the problem of traffic allocation in the independent fog-cloud subsystems in order to achieve the minimum transmission delay from foggy to cloud and power consumption. The result shows that fog computing improves performance for cloud computing by sharing the part computing burden, so bandwidth and transmission latency can reduce. However, this is less well suited to the fog computing infrastructure because the cloud data center is responsible for the allocation of work. So it can reduce overall performance.

From the viewpoint of service composition in Mobile Edge Computing, Deng *et al.* [23], [24] investigate service provisioning problem in distributed edges, and proposed some offloading strategies for solving optimal service provisioning problem. In [25], [26], authors carried up their research of service composition problem in mobile environment.

In this paper, we main deal with the associated tasks scheduling problem in cloud-fog computing, a task scheduling strategy is proposed, which is the combination of laxity-based priority algorithm and ant colony system. In the process of calculating the priority, the limitation of the task deadline is considered. In order to enhance the sensitivity of task delay, the laxity-based priority algorithm is adopted to construct a task scheduling sequence with reasonable priority. At the same time, in order to minimize the total energy consumption, the constrained optimization algorithm based on ant colony system algorithm is used to obtain the approximate optimal scheduling scheme in the global.

III. TASK SCHEDULING IN FOG-CLOUD ENVIRONMENT

A. SYSTEM ARCHITECTURE

In the cloud-fog computing environment, fog nodes at the edge of the network cooperate regionally together, connecting to the cloud nodes at the same time. The cooperation of the cloud and fog nodes achieve to satisfy mobile user's need. If fog nodes are limited to process the tasks, tasks are sent to cloud node. In this paper, in the fog-cloud computing system, we assume that the system consists of m fog nodes located in the edge of the network, u routers in the path between fog nodes and cloud nodes, and n cloud nodes located in the center of the network. The fog node communicates directly with the terminal devices and immediately forwards all received requests to the cloud-fog broker. The cloud-fog broker is responsible for analysing and estimating tasks and resources and then assigning tasks on the basis of the task scheduling policy. Since the fog nodes are close to the cloud-fog broker, the time consuming for data communication between each other is negligible. In order to ensure the performance and normal operation of the system in the cloud-fog computing environment, according to the different needs of users or applications, such as reducing energy consumption, reducing costs or minimizing the completion time, a task scheduling strategy that meets user requirements is formulated, and finally the task scheduling strategy is deployed in the fog cloud broker. We describe the operation of our scheduling model by summarizing the steps for running a scheduled service in figure 1.

Firstly, intelligent terminal devices send requests to its attached fog node at the edge of the network (step 1). Next, the fog node sends immediately the request data

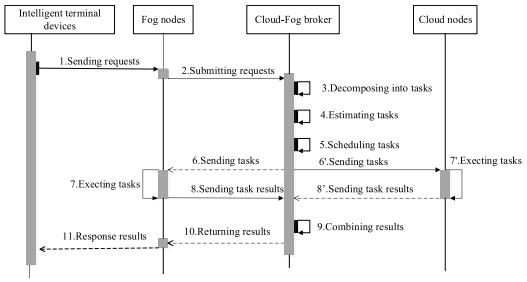

FIGURE 1. Model of the tasks scheduling in fog-cloud architecture.

and parameter information to the cloud-fog broker (step 2). In order to process in distributed way, each job is decomposed into a series of tasks (step 3) in the cloud-fog broker. At the same time, the number of instructions of the task and the usage of the required resources are estimated (step 4). Handling all information of tasks and resources, the cloud-fog broker runs a scheduling algorithm (step 5)to implement task allocation and get a scheduling scheme. According to the scheduling scheme, tasks are sent to the corresponding fog node or cloud node (step 6). The nodes are responsible for processing all tasks assigned (step 7) and then sending the processing results to the cloud-fog broker (step 8). After the tasks are completed, results of the tasks are combined in the cloud-fog broker (step 9). Finally the response results are sent to the end users (step 11) through the fog node (step 10).

B. TASK MODELLING BASED ON DIRECTED ACYCLIC GRAPH

In the IoT environment, an application consists of a series of interdependent tasks. Associated tasks with interdependencies are usually modelled as directed acyclic graphs (DAG), which are defined as follows.

Definition 1: A directed acyclic graph $G = (V, E)$, which describe a set of associated tasks and their priority constraints.

$V = \{v_1, v_2, \ldots, v_i, v_l\}$ $(\forall v_i \in V, l \in [1, l])$

- $v_{i-length}$ is the length of the computation task v_i. Considering that the tasks are all composed of instructions, the task length measured by the number of instructions. the unit is the MI(million instructions).
- $v_{i-deadline}$ denotes the deadline of the task v_i.

$E = \left\{e_{k,i} | v_k, v_i \in V\right\}$ represents a collection of dependencies between tasks. $e_{k,i}$ denotes that the task v_k is a predecessor task for task v_i.

C. THE FORMALIZATION OF ASSOCIATED TASK SCHEDULING PROBLEMS

In the DAG task map, a task that does not have any predecessors node is called an ingress node v_{entry}, and a task that does not have any successor node is called an egress node v_{exit}. The ingress node exists as the predecessor node of other task nodes. Other tasks can only be processed after the execution of the ingress node is completed. The execution result of the ingress node will transfer the data as input to the resource where the subsequent task is located.

Considering that the tasks in the DAG have a priority constraint, we assume that the task can be executed only if all the input data of the task available. The input data of the task is not only from the predecessor tasks in the DAG, but also some data resources (such as data storage) stored on the cloud nodes or fog nodes.

Assume task v_i is assigned to a resource node R_j, $c\left(e_i^{k,i}\right)$ represents the data transfer time from node R_f to R_j to execute task v_i, then $c\left(e_i^{k,i}\right)$ is defined as follows.

$$c_i = \left(d_i^m + \sum_{v_k \in exe(r_f)}^{v_i \in pred(w_i)} d_{k,i}\right) \times \left(\frac{1}{R_{j-bw}} + \frac{1}{R_{f-bw}}\right) \tag{1}$$

$$c\left(e_i^{k,i}\right) = \begin{cases} c_i, & \text{if } j = f \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where, d_i^m denotes the data where have stored in node R_f, and $d_{k,j}$ represents the data from all predecessor nodes of V_i which can be obtained through network traffic monitoring software. In this paper, we assume bandwidth, time delay and task size can be obtained by existing technical means. The R_{j-bw} and R_{f-bw} is the bandwidth of nodes R_j and R_f. The bandwidth is defined as the number of bits that can be transmitted per unit time.

When all necessary input data reaches the target cloud or fog node, the task will start to prepare, but also consider whether the target resource is idle. If the target resource is not idle, it needs to wait. Therefore, the earliest execution time of the task is determined by the predecessor task transmission time and the earliest idle time of the target processing node. The values of $EST\left(v_i, R_j\right)$ and $EFT\left(v_i, R_j\right)$ are computed as follows.

$$EST\left(v_i, R_j\right) = max\left\{avail\left(R_j\right), \max_{v_k \in pred(v_i)} \left(v_k, R_f\right) + c\left(e_i^{fj}\right)\right\} \tag{3}$$

$$EFT\left(v_i, R_j\right) = w\left(v_i, R_j\right) + EST\left(v_i, R_j\right) \tag{4}$$

where, $avail\left(R_j\right)$ denotes the the earliest time that node R_j completes the last assigned task and be ready to execute another task. $pred\ (v_i)$ is the predecessor node of the task v_i. $w\left(v_i, R_j\right)$ denotes the execution time of task v_i on node R_j. Then $w\left(v_i, R_j\right)$ is computed as follows. The (v_k, R_f) is the finish time of v_k on node R_f.

$$w\left(v_i, R_j\right) = \frac{v_{i-length}}{R_{j-mips}} \tag{5}$$

In the process of association task scheduling, if the task is assigned to the fog nodes, considering that the end user has only one or two hops to the fog nodes, the transmission energy consumption will be negligible, so the energy consumption

only includes the calculation energy consumption on the fog node. The energy consumption formula is as follows [27].

$$E_{ij}^{fog} = p_{idle}\,(a+1) \times EFT_{ij} + \int_{Com_{ij}} [p\,(t) - p_{idle}]\,dt \quad (6)$$

Here $a = t_{idle}/t_{act}$ denotes the ratio of free time to active time of the fog nodes. p_{idle} is the free power of the fog nodes. $p\,(t)$ is the power of the fog node at the moment t of executing the task.

If the task is assigned to the cloud node located in the network center, and the long-distance transmission needs to be performed by multiple core routing devices. The energy consumption includes the transmission energy of the core router and the computing energy consumption on the cloud node. The energy consumption formula is as follows [27].

$$E_i^{router} = r \times \left(\frac{< P_{idle} >}{U < C_{max} >} + E_b \right) \times N_{bit} \quad (7)$$

Here, $E_b = (P_{max} - P_{idle}) / (C_{max} U)$, P_{max} denotes the router's maximum power consumption, P_{idle} represents the router's free power consumption, C_{max} denotes the maximum load on the router and U represents the router's utilization, Let r denote the average number of routers in the path. N_{bit} denotes the number of bits of the task $task_i$ through the core router.

$$E_{ij}^{cloud} = (\alpha \times U_{cpu} + \beta \times U_{mem} + \gamma \times U_{bw}) \times P_{cloudj} \times EFT_{ij} + E_i^{router} \quad (8)$$

where U_{cpu} is CPU utilization of the virtual machine, U_{men} is the memory usage of the virtual machine, U_{bw} is the utilization of the bandwidth of the virtual machine, Let α, β, γ denote the coefficient among these three parts, which is $\alpha + \beta + \gamma = 1$.

Based on the requirements of user tasks and optimization goals, this paper presents a constraint associated tasks scheduling model in the cloud and fog framework, which is defined by:

$$\min\ F = \sum_i \sum_j \left\{ c_{ij} \times E_{ij}^{fog} + (1 - c_{ij}) \times E_{ij}^{cloud} \right\} \quad (9)$$

$$\text{s.t.} \begin{cases} EFT_{ij} \le v_{deadline} \\ 0 \le i \le l \\ 0 \le j \le q \\ (5), (8) - (10) \end{cases} \quad (10)$$

where, the value of c_{ij} does not determine whether task i is assigned to node j. $c_{ij} \in \{0, 1\}$ denotes that the i-*th* task has been already assigned to the resource-j. If the resource is a fog node, the value of c_{ij} is 1, otherwise, the value is 0, which means the task is assigned to the cloud node. F represents the total energy consumption for all tasks. For easy explanation, we illustrate by an example: There two task, task-1 and task-2, and two resource, resource-1 and resource-2, that need to be calculated for energy consumption. Besides, the task-1 is assigned to resource-1, and the task-2 is assigned to resource-2. In this situation, we assume resource-1 is a fog node and resource-2 is a cloud node, namely $c_{11} = 1$, $c_{22} = 0$. We use formula (9) to calculate energy consumption. That is $1 \times E_{11}^{fog} + (1-1) \times E_{11}^{cloud} + 0 \times E_{22}^{fog} + (1-0) \times E_{22}^{cloud}$. In addition, the tasks and resources in this paper are already the smallest unit. In this research work, we assume the resources and tasks are to be minimized without any further divided actions, and the relationship between the resources and tasks is one-to-one correspondence. That is to say, there is no task assigned to two resource nodes, and no resource node performs two tasks. So the variables of c_{12} and c_{21} don't need to consider in this paper. In our later experiment, we also eliminate these meaningless values.

D. ASSOCIATED TASK SCHEDULING STRATEGY

In order to solve the problem of associated task scheduling in the IoT with mixed deadlines, this paper proposes a combination of improved HEFT priority and ant colony-based scheduling strategy. Therefore, the scheduling strategy can achieve to select the most suitable resource for the task based on the DAG, and achieve the goal that meets the mixed delay of the task and achieve low energy consumption. The associated task scheduling strategy based on laxity and ant colony system (LBP-ACS) proposed in this chapter is put forward, which is divided into two parts: the laxity-based priority algorithm (LBPA) and the ant colony-based constrained optimization algorithm (COA-ACS). The first step uses the laxity -based priority algorithm (LBPA) to obtain the task priority sequence, and the second step uses the constrained optimization algorithm based on ant colony system (COA-ACS) to obtain the task scheduling scheme.

1) LAXITY-BASED PRIORITY ALGORITHM (LBPA) FOR OBTAINING PRIORITY SEQUENCES

The LBPA algorithm aims to calculate the laxity of each task by recursively, and then calculate the priority of each associated subtask according to the laxity, and finally convert the DAG-based task graph into an ordered task sequence. The laxity of a task is the shortest time that can be delayed before the deadline of the task,which indicates the urgency or time sensitivity of the task. The priority of the task is determined by the laxity of the task$laxity\,(v_i)$. The smaller the laxity of the task, the higher the priority of the task. That is to say, the time-sensitive task is preferentially scheduled. Let the time laxity of the task (v_i) is recursively defined by:

$$laxity = \min_{v_k \in succ(v_j)} \left\{ laxity\,(v_i) - \overline{c\,(k,i)} - \overline{w\,(v_i)} \right\} \quad (11)$$

$$laxity\,(v_i) = \begin{cases} v_{i-deadline} - \overline{(v_i)}, & \text{if} \quad v_i = v_{exit} \\ laxity, & \text{otherwise,} \end{cases} \quad (12)$$

where, $\overline{w\,(v_i)}$,$\overline{c\,(k,i)}$ denote the average calculation time of the task (v_i) and the average transmission time of the

Algorithm 1: LBPA

Input: Tasks Graph $G(V, E)$ and List of VM V_j
Output: A task priority sequence

```
Calculate (v_i), c(v_{k,i}) according to Formula 13, 14;
for i = l to 1 do
    Calculate laxity(v_i) according to Formula 12 ;
sort laxity(v_i) get the task priority sequence TaskList;
return The task priority sequence ← TaskList;
```

task (v_i), respectively. They are defined as follows.

$$\overline{w(v_i)} = \frac{v_{i-length}}{\sum_{i=1}^{q} R_{j-mip}/q} \tag{13}$$

$$\overline{c(k,i)} = \frac{c(k,i)}{\sum_{i=1}^{q} R_{j-bw}/q} \tag{14}$$

According to the formula 12, the time laxity of each task is calculated from the exit node of the task map to the ingress node in a recursive manner, then according to the time laxity of the task, the priority sequence of the task *TaskList* is obtained in ascending order.

The laxity-based priority algorithm (LBPA) is shown in 1.

2) CONSTRAINED OPTIMIZATION ALGORITHM BASED ON ANT COLONY SYSTEM (COA-ACS) FOR TASK ASSIGNMENT

Using the laxity-based priority algorithm (LBPA) above, the priority sequence of the task is obtained, and the associated task scheduling problem and the ant colony system algorithm are combined to propose a constrained optimization algorithm based on the ant colony system, which Choose the right cloud or fog resource for the task of the priority sequence. The scheme description of each ant for each trip is shown in figure 2. The horizontal axis represents the task of the task priority sequence *TaskList*, and the vertical axis represents the cloud or fog resource. If the task v_i is assigned to the resource R_j, it is represented as $edge(i, j)$ $(i = 1, 2, \ldots, m + n)$.

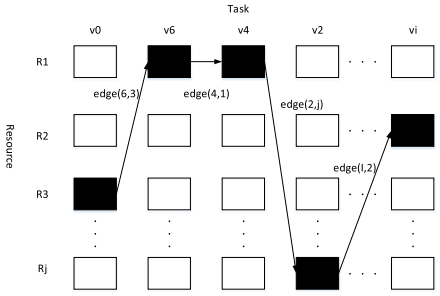

FIGURE 2. Illustration of a solution of an ant in search.

(1) Task Priority
According to the algorithm of the task priority LBSFC, we can calculate the priority of the task and obtain the sequence of the task execution.

(2) Initialization Pheromone
In the DAG-ACS algorithm, we first obtain the task allocation scheme $Z[i]$ through the greedy algorithm. Then, according to the calculation of the energy consumption formula of the resource of the cloud and fog. Finally we can calculate the initial pheromone τ_0 as shown in the following formula.

$$Z[i] = \sum_{i=0}^{l-1} argmin(F) \tag{15}$$

$$\tau_0 = 1/(1 \times F_z) \tag{16}$$

According to the above formula, the pheromone is initialized on the path e_{ij} formed by the each task and the virtual machine.

(2) Select Resources and Calculate Heuristics
In the iteration of the DAG-ACS algorithm, each ant k establishes a route of performing l (number of tasks) steps. When selecting a cloud or fog resource for each task based on the task's priority sequence, each ant has two ways to select resources, as shown in the following formula. For each task v_i, we first generate a random number $q \in [0, 1]$. If $q \leq q_0$, the ant will greedily choose the pheromone and the source with high heuristic information value. Otherwise, the ant will select the resource according to the roulette. The probability rule $p_{ij}^{(k)}(t)$ of the resource selection for the roulette is calculated as follows.

$$r_{v_i}(t) = \begin{cases} [\tau_{ij}(t)]^{\alpha} (\eta_{ij})^{\beta}, & q \leq q_0 \\ Roulette \quad selection, & \text{otherwise,} \end{cases} \tag{17}$$

$$p_{ij}^{(k)}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} (\eta_{ij})^{\beta}}{\sum_{k \in allowed_k} [\tau_{ik}(t)]^{\alpha} (\eta_{ik})^{\beta}}, \\ \quad \text{if } j \in allowed_k \\ 0, \quad \text{otherwise,} \end{cases} \tag{18}$$

where $\tau_{ij}(t)$ represents the pheromone on the path e_{ij} in the t^{th} iteration. α, β are respectively control parameters. $allowed_k$ denotes the virtual machine that the k^{th} ant can select.

The heuristic information η_{ij} denotes the expected value of processing the task v_i on the resource r_j. When selecting heuristic information, the task completion time and energy consumption are fully considered. η_{ij} calculation formula is as follows.

$$\eta_{ij} = \frac{1}{weight \times EFT_{ij} + (1 - weight) \times E_{ij}} \tag{19}$$

where EFT_{ij} and E_{ij} represent the completion time and energy consumption that the task v_i is assigned to the resource r_j, respectively. The weight denotes the weight of the task completion time.

(3) Local Pheromone Update
When constructing a task assignment scheme, the value of the pheromone is constantly changing since the ant will generate and volatilize of the pheromone on the

path e_{ij}. For example, if the ant selects the resource r_j for the task v_i, the pheromone on the path e_{ij} will partially evaporate. The calculation formula is shown following.

$$\tau_{ij}(t) = (1-\varepsilon) \times \tau_{ij}(t) + \varepsilon \times \tau_0 \tag{20}$$

where the parameter ε $(0 \leq \varepsilon \leq 1)$ is a pheromone volatilization factor.

(4) Global Pheromone Update
After each iteration completed, all ants have already built a task assignment strategy. According to the optimal allocation strategy, global pheromone updating is performed which is performed according to the following formula.

$$\tau_{ij}(t+1) = (1-\rho) \times \tau_{ij}(t) + \rho \times \Delta\tau_{ij}^{k} \tag{21}$$

where the parameter ρ is the pheromone volatility factor, $0 < \rho < 1$. $\Delta\tau_{ij}^{k}$ is computed as follows.

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{F_{best}(t)}, & \text{if } e_{ij} \text{ is best path} \\ 0, & \text{otherwise,} \end{cases} \tag{22}$$

where Q is an adaptive parameter. where F_{best} is the total energy consumption of the best-path in the first t ants searched.

(5) Algorithm Termination Condition
If the calculation has reached the expected value, the algorithm is terminated, and otherwise entering the next iteration. If the maximum number of iterations is reached, the algorithm terminates.

IV. EXPERIMENT AND ANALYSIS

In this experiment, in order to evaluate the performance of the proposed scheduling mechanism, we compare our algorithm with three others: Greedy for Energy(GfE), HEFT [18], and DEACO. The GfE algorithm only considers energy consumption metrics, ignoring task load balancing and time factors. The HEFT algorithm is a classic list scheduling algorithm, which only pays attention to the task completion time. According to the priority of the task, the task is assigned to the resource with the fastest processing speed, so that all tasks can be completed in the shortest possible time. We use Cloudsim [28] and extend the modular of task scheduler with fog for modeling and simulation of the fog-cloud computing infrastructure. All parameters are presented in Table 1. our experiment covers a random graph G = (V,E) with the increase of sizes from 20 to 100 and a set of heterogeneous Vms that are from 2 cloud nodes and 6 fog nodes. The I/O data of a task have a size from 5 to 6 MB.

Based on the parameters' settings in Tables 1, results of three metrics are obtained and illustrated in Figures 3-5.

Figure 3 shows the comparison of task scheduling length using LBP-ACS, GfE, HEFT and DEACO algorithms. It is shown that task scheduling length gets longer with the slightly increase of tasks number. In terms of schedule length, GfE algorithm gets the worst case, HEFT algorithm obtains the best result while LBP-ACS and DEACO are in the middle. Specifically, our algorithm is 9.7% better than DEACO. And compared with GfE, our LBP-ACS algorithm even achieves a far better performance, about 36.5%.

Algorithm 2: $COA - ACS$

Input: The task priority sequence $\leftarrow$ *TaskList* and List of VM V_j, K, N_{max}
Output: $solution = \{edge(i,j) \mid (i = 1, 2, \ldots, l; j = 1, 2, \ldots, m+n)\}$, minEnergy

for $Iter = 1$ **to** N_{max} **do**
  for $k = 1$ **to** K **do**
    Get a task allocation scheme $Z[i]$ by adopting a greedy algorithm ;
    Compute τ_0 according to Formula 16 ;
    Initialize η_{ij}, q_0 each ant select a path $edge(i,j)$ randomly ;
    while $i \leq l$ **do**
      Randomly generate a number q ;
      for $j = 1$ **to** n **do**
        if $q \leq q_0$ **then**
          Choose the $Vm_j \leftarrow$ $argmax\left\{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}\right\}$;
        Compute EFT_{ij}, E_{ij},η_{ij}, p_{ij} according to Formula $(5-3)$, $(5-16)$, $(5-15)$;
        Choose the Vm_j for v_i ;
      Update local pheromone according to Formula 20 ;
    Calculate best energy F_{best};
    Record the best path e_{ij} ;
    Update global pheromone according to Formula 22,21;
  $minEnergy \leftarrow F_{best}$;
return $solution \leftarrow$ best path $edge(i,j)$, $minEnergy$;

TABLE 1. Experiment parameters setting.

Entity Type	Parameter	Value
Tasks	length	[9000,15000]MI
	deadline	[5,220]s
Cloud Nodes	Processing Rate	[1000,1500]MIPS
	Bandwidth	[1.7,2]Mbps
Fog Nodes	Processing Rate	[700,900]MIPS
	Bandwidth	[90,100]Mbps
	Idle	10w
	Active	[18,25]w
	a	5
Routers	Idle	11070w
	Max	12300w
	Maxload	4480
	utilization	0.6
LBP-ACS	α	1
	β	1
	Q	100
	m	30
	t_{max}	100
	ρ	0.2
	ε	0.1

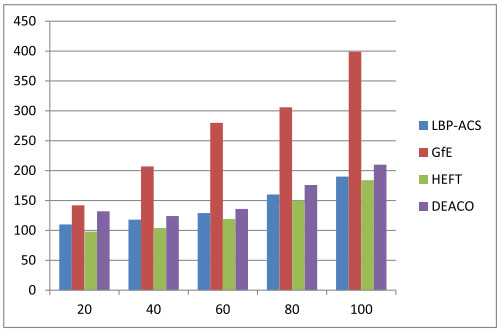

FIGURE 3. Scheduling Length of LBP-ACS vs. GfE. HEFT and DEACO.

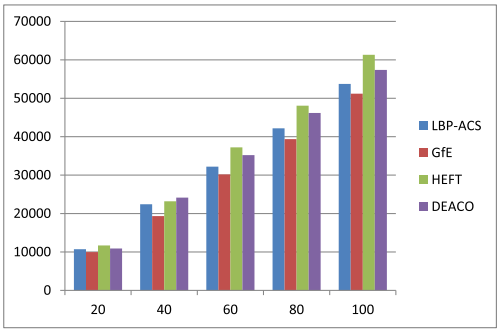

FIGURE 4. Energy Comparison of LBP-ACS vs. GfE. HEFT and DEACO.

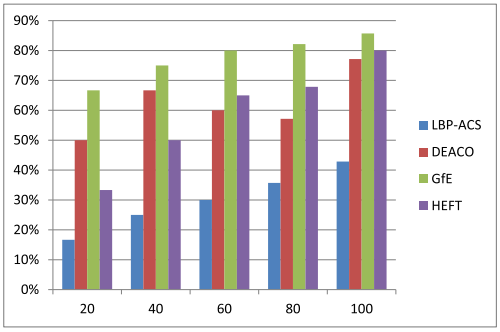

FIGURE 5. Failure Ratio of LBP-ACS vs. GfE. HEFT and DEACO.

Regarding the energy consumption for cloud or fog resources, as we can see that the energy consumption for cloud or fog resources gets bigger with the increase of tasks number in figure 4. It is observed that although HEFT provides the best performance, it has the highest energy consumption while the opposite is true for GfE algorithm. This is mainly due to the fact that the HEFT algorithm only focuses on the processing time, and blindly pursues reducing the task completion time, while ignoring the energy consumption index. Similarly, the GfE algorithm only focuses on the energy consumption index, and ignores the task deadline constraint. In contrast, our LBP-ACS algorithm conduces to the benefits of balance between schedule length and energy consumption for cloud or fog resources. Compared with HEFT algorithm, our LBP-ACS algorithm can save 11.1% of the energy consumption while performance reduction is not more than 6.8%. And compared with DEACO, our LBP-ACS algorithm can save 7.1% of the energy consumption, which means that our LBP-ACS algorithm has the advantage of reducing energy consumption together with its effectiveness.

In the experiment, we evaluated the performance in terms of a Failure ratio, defining the failure rate as the ratio of the number of tasks not completed within the deadline to the total number of scheduled tasks.

$$FA = \frac{Count_{fail}}{Count_{total}} \times 100\% \tag{23}$$

where, $Count_{fail}$ denotes the number of tasks that fail to schedule, which includes all tasks that fail to meet deadline constraints. $Count_{total}$ is all tasks with the deadline constraints.

Figure 5 shows that the failure ratio using different algorithms. As we can see that LBP-ACS algorithm outperformed all other algorithms, which has the lowest failure rate. as the other algorithms were not primarily designed to meet deadlines, whereas the LBP-ACS algorithm takes account of the deadlines for each end task. Although the HEFT algorithm minimizes the scheduling length of the task, it does not take into account the deadline constraints of task when calculating the priority. The delay-sensitive task is not preferentially scheduled, so the failure rate is higher than LBP-ACS algorithm. For the associated task schedule with deadline constraints, not only should the task completion time be minimized, but also tasks with sensitive deadlines should be prioritized.

In summary, the proposed LBP-ACS algorithm can effectively reduce the energy consumption while improving the task scheduling success rate compared with other algorithms in solving the associated task scheduling problem with mixed deadlines.

V. CONCLUSION AND FUTURE WORK

In this paper, in order to exert advantage of the cloud and fog computing, we adopt a cloud and fog cooperation architecture. For the purpose of obtaining the most benefit from such an architecture, one must allocate computing tasks strategically at each processing node of cloud or fog layer. For the scheduling problems of complex tasks with priority constraints in IoT applications. This paper addresses associated task scheduling in hybrid cloud-fog computing. The associated task scheduling strategy based on laxity and ant colony system is proposed in cloud-fog environment, which takes into account the energy consumption and tries to fulfil reduce energy consumption on the condition of satisfying the mix deadline. Simulations and numerical results have shown that our work can show a better performance than other existing methods.

In future work, on the one hand, we intend to deploy our proposal algorithm into real world systems. We consider an IoT deployment scenario with a user-defined analysis queries (tasks) that need to be performed on several fog nodes at the edge or public cloud nodes available to perform

the queries. With the planned implementation, we can thoroughly observe the performance in the real-world operation and find the shortcomings to improve our proposal. On the other hand, we should consider the scheduling of tasks that include independent tasks and associated tasks.

REFERENCES

[1] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the Internet of Things: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 414–454, 1st Quart., 2014.
[2] D. Evans, "The Internet of Things: How the next evolution of the Internet is changing everything," Cisco, San Jose, CA, USA, White Paper, 2011. [Online]. Available: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf
[3] K. Bierzynski, A. Escobar, and M. Eberl, "Cloud, fog and edge: Cooperation for the future?" in *Proc. 2nd Int. Conf. Fog Mobile Edge Comput.*, May 2017, pp. 62–67.
[4] P. Bellavista, J. Berrocal, A. Corradi, S. K. Das, L. Foschini, and A. Zanni, "A survey on fog computing for the Internet of Things," *Pervasive Mobile Comput.*, vol. 52, pp. 71–99, Jan. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574119218301111
[5] A. Kumari, S. Tanwar, S. Tyagi, and N. Kumar, "Fog computing for healthcare 4.0 environment: Opportunities and challenges," *Comput. Elect. Eng.*, vol. 72, pp. 1–13, Nov. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045790618303860
[6] T. V. N. Rao, A. Khan, M. Maschendra, and M. K. Kumar, "A paradigm shift from cloud to fog computing," *Int. J. Sci., Eng. Comput. Technol.*, vol. 5, no. 11, p. 385, Nov. 2015.
[7] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for Internet of Things and analytics," in *Big Data Internet Things: A Roadmap for Smart Environments*. Cham, Switzerland: Springer, 2014, pp. 169–186.
[8] A. A. Mutlag, M. K. A. Ghani, N. Arunkumar, M. A. Mohammed, and O. Mohd, "Enabling technologies for fog computing in healthcare IoT systems," *Future Gener. Comput. Syst.*, vol. 90, pp. 62–78, Jan. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X18314006
[9] P. Varshney and Y. Simmhan, "Demystifying fog computing: Characterizing architectures, applications and abstractions," in *Proc. IEEE 1st Int. Conf. Fog Edge Comput. (ICFEC)*, Feb. 2017, pp. 115–124.
[10] Y. Zhao, X. Wang, J. Zhang, and B. Mukherjee, "Optical networking for hybrid computing combining cloud and fog," in *Proc. Prog. Electromagn. Res. Symp.*, Aug. 2016, p. 4875.
[11] V. B. C. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, G. Ren, and G. Tashakor, "Handling service allocation in combined fog-cloud scenarios," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–5.
[12] V. B. Souza, X. Masip-Bruin, E. Marín-Tordera, W. Ramírez, and S. Sanchez, "Towards distributed service allocation in fog-to-cloud (F2C) scenarios," in *Proc. IEEE global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.
[13] X. He, Z. Ren, C. Shi, and J. Fang, "A novel load balancing strategy of software-defined cloud/fog networking in the Internet of Vehicles," *China Commun.*, vol. 13, pp. 140–149, Nov. 2016.
[14] D. Hoang and T. D. Dang, "FBRC: Optimization of task scheduling in fog-based region and cloud," in *Proc. IEEE Trustcom/BigDataSE/ICESS*, Aug. 2017, pp. 1109–1114.
[15] S. Deng, Z. Xiang, J. Yin, J. Taheri, and A. Y. Zomaya, "Composition-driven IoT service provisioning in distributed edges," *IEEE Access*, vol. 6, pp. 54258–54269, 2018.
[16] D. Kliazovich, J. E. Pecero, A. Tchernykh, P. Bouvry, S. U. Khan, and A. Y. Zomaya, "CA-DAG: Communication-aware directed acyclic graphs for modeling cloud computing applications," in *Proc. IEEE 6th Int. Conf. Cloud Comput.*, Jun./Jul. 2013, pp. 277–284.
[17] V. Kumar, C. Katti, and C. P. Saxena, "A novel task scheduling algorithm for heterogeneous computing," *Int. J. Comput. Appl.*, vol. 85, no. 18, pp. 35–39, Jan. 2014.
[18] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002.
[19] U. Boregowda and V. R. Chakravarthy, "A hybrid task scheduler for DAG applications on a cluster of processors," in *Proc. 4th Int. Conf. Adv. Comput. Commun.*, Aug. 2014, pp. 143–146.
[20] J. Li, S. Su, X. Cheng, Q. Huang, and Z. Zhang, "Cost-conscious scheduling for large graph processing in the cloud," in *Proc. IEEE Int. Conf. high Perform. Comput. Commun.*, Sep. 2011, pp. 808–813.
[21] X.-Q. Pham and E.-N. Huh, "Towards task scheduling in a cloud-fog computing system," in *Proc. 18th Asia–Pacific Netw. Oper. Manage. Symp.*, Oct. 2016, pp. 1–4.
[22] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.
[23] C. Zhang, H. Zhao, and S. Deng, "A density-based offloading strategy for IoT devices in edge computing systems," *IEEE Access*, vol. 6, pp. 73520–73530, 2018.
[24] S. Deng, H. Wu, W. Tan, Z. Xiang, and Z. Wu, "Mobile service selection for composition: An energy consumption perspective," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 3, pp. 1478–1490, Jul. 2017.
[25] S. Deng, L. Huang, J. Taheri, J. Yin, M. Zhou, and A. Y. Zomaya, "Mobility-aware service composition in mobile communities," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 555–568, Mar. 2017.
[26] S. Deng, L. Huang, G. Xu, X. Wu, and Z. Wu,, "On deep learning for trust-aware recommendations in social networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 5, pp. 1164–1177, May 2017.
[27] J. Xu, X. Sun, R. Zhang, H. Liang, and Q. Duan, "Fog-cloud task scheduling of energy consumption optimization with deadline consideration," *Int. J. Internet Manuf. Services*, to be published.
[28] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

JIUYUN XU (M'90) received the Ph.D. degree in computer science from the China University of Posts and Telecommunications, in 2004. He is currently a Professor with the China University of Petroleum (Eastern China). He has published in excess of 40 international conferences and journals papers. His research interests include service computing and the Internet of Things. He is a member of ACM, and a Senior Member of CCF. He is a reviewer for some prestigious journals, including IEEE TSC, IJIMS, and so on. He has also served on the technical program committees for numerous conferences, including ICWS, SCC, and so on.

ZHUANGYUAN HAO received the bachelor's degree in Internet of Things from Tsingdao Science and Technology University. He is currently pursuing the master's degree in computer technology with the China University of Petroleum. His research interest includes offloading issue in edge computing.

RURU ZHANG received the master's degree in computer technology from the China University of Petroleum, in 2016. She is currently a Software Engineer with the China Mobile R&D Center, Suzhou. Her research interests include service computing and cloud computing.

XIAOTING SUN received the bachelor's degree in computer application from Shandong Women's University. She is currently pursuing the master's degree in computer technology with the China University of Petroleum. Her research interest includes fog computing.