Hybrid Evolutionary Scheduling for Energy-efficient Fog-enhanced Internet of Things

Chu-ge Wu, Wei Li, *Senior Member, IEEE*, Ling Wang and Albert Y. Zomaya, *Fellow, IEEE*

Abstract—In recent years, the rapid development of the Internet of Things (IoT) has produced a large amount of data that needs to be processed in a timely manner. Traditional cloud computing systems can provide us with plentiful resources to process such data. However, the increasing requirements of IoT applications on data privacy, energy consumption savings and location-aware data processing pushes the emergence and the interplay of fog computing and cloud computing. This paper examines the resource scheduling issue under such a system to minimize makespan and energy consumption. A multi-objective estimation of distribution algorithm (EDA) as well as a partition operator is adopted to divide the graph and determine the task processing permutation and processor assignment. Single and multiple application simulation were both conducted. The comparative results show that the Pareto set produced by our proposed algorithm is able to dominate a large proportion of those solutions by the heuristic method and the simple EDA under single application simulation. When it comes to multi-application simulation, IoT devices can have a much longer lifetime with our proposed scheduling algorithm as well having similar performance to the other algorithms on fog node energy consumption and much better on makespan.

Index Terms—edge computing, Internet of Things, evolutionary computation, scheduling, energy efficiency, makespan.

1 Introduction

With the rapid development of Internet of Things (IoT), a tremendous number of heterogeneous, wireless or wired, sensing and actuation devices is employed in a wide range of real-world applications. The IoT devices generally have limited energy supply from battery, low data processing capability, short radio communication range and marginal reliability. To support the deployment of both hard and soft real-time applications, IoT systems need to meet desired end-to-end latency requirements and provide deterministic QoS guarantees with finite resources.

Fog computing [1] is one of the promising solutions for the completion of time-constrained IoT applications since it aims at moving computation and storage capabilities from remote cloud center(s) to the edge of the network. The use of fog computing enables processing of the collected data to start in close proximity to the sources and thus reduce the volume of data transmission over the Internet. To fully exploit the underlying fog computing infrastructure, the optimal use of spatially distributed resources is a key element to improve the overall system performance in terms of minimizing end-to-end delay, reducing bandwidth consumption and related costs, and maximizing user experience. However, this is not a trivial task due to the difficulties imposed by the complex and interdependent fog computing infrastructure on solving the resource scheduling problem. In addition, the applications initiated by different users bring extra challenges to this problem since different objectives are usually in conflict with each other and cannot be evaluated under a common measure. As is well known, the task scheduling on multiprocessors problem is NP-hard if more than two processors are used [2]. This problem with more factors considered can also be proved to be NP-hard, and thus it is impossible to obtain the optimal solution in a polynomial time.

In the past two decades, a wide range of evolutionary optimization algorithms have been successfully adopted to solve different types of scheduling problems with firm performance. Among the evolutionary algorithms, the Estimation of Distribution Algorithm (EDA) was often used to solve complex scheduling problems [3, 4] as it is a good candidate for tackling the scheduling and mapping problems. EDA is a population-based approach where the distribution of the promising solutions in the search space is described by a probability model. New solutions are generated by sampling the model. In our previous work, EDA performed well in DAG scheduling problems [5] and in generating energy efficient scheduling schemes for cloud computing systems [6]. As the problem in this work is a multi-objective DAG scheduling problem, EDA could be adjusted accordingly and the procedure repeated to achieve the same level of performance. Inspired by the successful applications of the EDA in the field of scheduling, the EDA will be embedded in the proposed algorithm to produce processing sequences and solve the task scheduling problem and jointly minimize the objectives of makespan and energy consumption.

The remainder of the paper is organized as follows: Section 2 reviews related work. Section 3 provides the background knowledge of multi-objective optimization followed by the models of the system, application, energy consumption, and ends up with the mathematical models for the resource scheduling problem. Then, the proposed

- *Chu-ge Wu and Ling Wang are with the Department of Automation, Tsinghua University, Beijing, 100084, P.R. China. E-mail: wucg15@mails.tsinghua.edu.cn; wangling@mail.tsinghua.edu.cn. (Corresponding author: Ling Wang.)*
- *Wei Li and Albert Y. Zomaya are with Centre for Distributed and High Performance Computing, School of Computer Science, The University of Sydney, NSW 2006. E-mail: liwei@it.usyd.edu.au; albert.zomaya@sydney.edu.au.*

xxxx-xxxx/0x/$xx.00 © 200x IEEE Published by the IEEE Computer Society

algorithm is presented in Section 4. In Section 5, the numerical studies are provided to demonstrate the effectiveness of the algorithm. Finally, the paper draws some conclusions and outlines ideas for future work in Section 6.

2 Related Work

The wide adoption of cloud computing accelerates the realization of the idea of IoT. Thanks to the availability and the flexibility of cloud infrastructures, IoT devices have become more visible to the public and the collected data can be directly uploaded to the cloud for further processing or storage. To manage the transmission of large volumes of data over the Internet and reduce the increasing energy consumption of the cloud infrastructures, many new research challenges have been raised over how to process the data in close proximity to the source of data in an efficient way while still meeting the needs of the applications. To deal with the network traffic issues caused by data explosion, fog computing aims to divide the job of data processing into smaller fragments so that they can be processed closer to the data sources on less powerful machines.

In [7], the authors proposed a resource scheduling approach for the placement of service modules on fog nodes, with a single objective of minimizing the response time among components. A virtual machine (VM) placement solution for distributed cloud environments is introduced in [8] for optimizing the VM allocation on physical nodes in order to minimize the network latency among them. However, these approaches are not applicable for our concern since the key features of fog devices like location aware processing are not included.

Power consumption is a crucial aspect in considering the maintenance costs for a datacenter. It involves not only the cost of energy consumed by machines, but also the energy consumed by the cooling devices. This issue becomes critical when the computing devices are moved to the edge of the network since it could be either powered by non-rechargeable batteries or could result in extra monetary cost if these devices are put into residential areas. In [9], the authors conducted a comparative study on energy consumption of fog computing and cloud computing paradigms. After extensive experimental studies, the authors highlighted multiple key factors at system design level that help fog nodes to consume less energy than the cloud nodes. One of the major findings is that close collaboration between fog node and cloud node could lead to energy saving compared to the standalone cases. Along with this finding, the authors in [10] proposed a bi-objective optimization model that considered delay and power consumption for workload allocation in an edge-cloud environment. The authors in [11] also presented a dynamic resource provisioning scheme for achieving energy efficient interplay between cloud and edge. However, the focus of the paper is to demonstrate that the use of a software defined network can help to minimize the energy consumption of the underlying networks between edge and cloud, which is not that different from the three-tier systems that we are working with. In [12], the authors proposed a quantified near-optimal online solution to balance the three-way tradeoff among average response time, average cost and average number of application loss under the same system architecture that we considered in this work. However, no energy consumption is addressed in this work.

3 System Modelling and Problem Statement

This section is divided into six subsections. First we provide a brief introduction to multi-objective optimization along with some definitions that will be used later. After that, we detail the models which are used in our proposed three-tier IoT system, including: system model, application model and energy consumption model. In the last, we present the problem statement and a lower bound analysis.

3.1 Basic concepts of multi-objective optimization problem

A multi-objective optimization problem generally involves more than one contradictory objective simultaneously and requires identification of a desired trade-off between such objectives. The problem can usually be defined by Equation (1).

$$\min y = f(x) = \left(f_1(x), f_2(x), \dots, f_q(x)\right) \quad (1)$$

where $x \in R^p$ is a p-dimensional decision vector and $y \in R^q$ is the objective vector that contains q individual objectives. When more than one objective is considered, the sorting method used to evaluate the quality of the solutions needs to be changed, compared to the single objective case. In this paper, the non-dominated sorting method is adopted to compare and rank the solutions. The essential knowledge of optimal Pareto solutions is provided as below.

Pareto dominance: A solution x_1 is considered to dominate solution x_2 (denoted as $x_1 \prec x_2$) if and only if:

(a) $f_i(x_1) \leq f_i(x_2), \forall i \in \{1,2, \dots, q\}$,

(b) $f_i(x_1) < f_i(x_2), \exists\, i \in \{1,2, \dots, q\}$.

Optimal Pareto solution: A solution x is called an optimal Pareto solution if it is not dominated by any other solution.

Optimal Pareto set/Pareto Front: The solution set containing all optimal Pareto solutions is defined as the optimal Pareto set/Pareto Front.

The optimal Pareto set obtained by a certain algorithm is defined as a **non-dominated solution set/archive set** (**AS**). Besides, C metric (CM) used to estimate multi-objective algorithms [13] is introduced. CM is used to compare the archive sets AS_1 and AS_2 which reflects the dominance relationship between the solutions in these two sets. CM is computed as (2).

$$C(AS_1, AS_2) = \left|\left\{x_2 \in AS_2 \middle| \begin{matrix}\exists x_1 \in AS_1, x_2 \prec x_1 \\ or\ x_2 = x_1\end{matrix}\right\}\right| / |AS_2| \quad (2)$$

where $C(AS_1, AS_2)$ reflects the percentage of the solutions in AS_2 that are dominated by or are the same as the solutions in AS_1. In principle, $C(AS_1, AS_2)$ and $C(AS_2, AS_1)$ are both calculated to compare the Pareto dominance of the archive sets and its algorithm. The larger $C(AS_1, AS_2)$ and smaller $C(AS_2, AS_1)$ means AS_1 is better than AS_2 under the Pareto optimal metric.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2889482, IEEE Transactions on Cloud Computing

3.2 System model

In this work, we assume a generalized three-tier IoT system [14] as depicted in Figure 1, where it is composed of things, fog and cloud tiers. The first tier is the things tier and it is the bottom tier in our system model. The devices located in this tier are responsible for collecting raw data from the area of interest and processing it accordingly before sending it for further processing. The devices in this tier are grouped into different clusters by their locations and the devices in the same cluster are assumed to be fully connected. Besides, all the devices are powered by battery unless otherwise specified. The second tier is the fog computing tier in our system, which is located at the middle of the overall architecture in order to process the computation tasks near the data source. In this work, different devices such as laptop, desktop, and Nano data centers are defined as fog nodes. Each fog node directly connects with a cluster of things located at the things tier. On the other hand, the nodes in the fog tier are fully connected with each other. The fog node can share the computation tasks received from the things tier with all other nodes within this tier. The third tier in the system is the cloud tier and processing units in this tier are modelled as cloud nodes, which are generally located in the data centers. In our model, each fog node is linked with cloud nodes via wide area network (WAN) and the nodes in the cloud are fully connected. Compared with the fog nodes, the resources of the cloud nodes are more reliable and substantial.

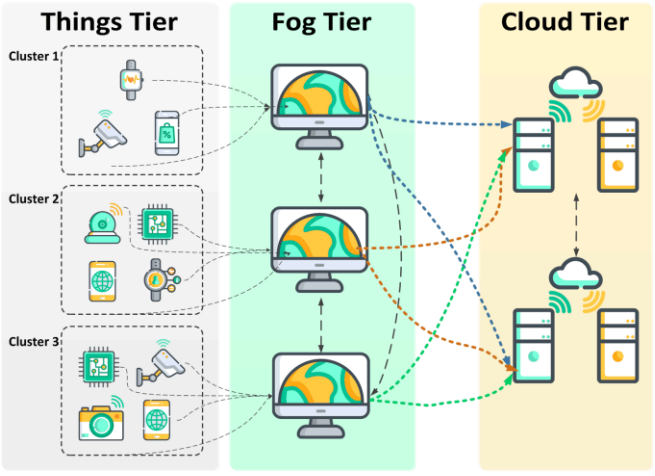

Figure 1. An overview of the three-tier IoT system architecture

3.3 Application model

Each IoT application is represented as a DAG (Directed Acyclic Graph). A given DAG can be defined as a four-tuple **<*V*, *E*, *w*, *D*>**, where ***V*** denotes the non-communication tasks of a certain application, such as a computing function or a sensing function in the IoT application. ***E*** is a set of directed and weighted edges used to represent the task dependence and the communication load among the non-communication tasks. ***w*** represents the workload of the non-communication tasks and ***D*** represents the communication overhead. For a given task pair i and j, $D(i,j)$ denotes the data size of edge E_{ij}. Any task in the DAG can only start being processed after all its predecessors are finished and all the input data is received. Thus, the tasks must be scheduled in accordance with the precedence constraints. In addition, the tasks are assumed to be non-preemptive, which means once they are initiated, they cannot be interrupted until the processing is completed. The communication time of any two connected tasks (i, j) is dependent on the bandwidth $B(m_i, m_j)$ associated with the assigned nodes$(m_i, m_j), (m_i \neq m_j)$. In addition, it is well known that the communication to computation ratio (*CCR*) indicates the potential impact of communication latency on performance and is calculated as average communication time divided by average computation time. The larger *CCR* is, the more likely the application is to be a communication intensive one. Otherwise, it is a computation intensive one.

In this work, the IoT applications are normally started from sensing tasks, which leads to the entry tasks in the DAG generally being processed at the things tier. By entry tasks, we refer to those tasks without parent tasks which are responsible for sensing or gathering raw data. Therefore, we assumed that all the entry tasks are allocated to IoT devices from the same cluster, but this condition is not likely to be permanent.

3.4 Energy consumption model

To study the energy consumption of the overall three-tier IoT system, we need to determine the major sources of energy consumption at each tier and sum them up. It is worth mentioning that the energy consumption of the cloud tier is difficult to accurately estimate since it is easily affected by multiple factors, such as the design and usage of the cooling system and the cloud services provided by different vendors. Besides, our algorithm aims at optimizing the time and energy consumption performance for users. The cloud computing service is brought to users in a pay-as-you-go manner, the energy consumption of cloud data center is hard to optimize from the users' end. Thus, the energy consumption of cloud data center is not considered in this work. More importantly, the IoT devices located at the things tier are often equipped with non-rechargeable batteries that have limited lifetime. Thus, it is necessary to minimize the energy consumption of the IoT devices to prolong the system lifetime since a single failure of an IoT device may cause the death of the entire system [15]. In addition, in such systems, fog nodes are placed in close proximity to the users (e.g. located at users' home) so that the energy consumption of these nodes may cause a negative impact on residential electricity tariffs. Thus, we focus primarily on evaluating the energy consumption of IoT devices and fog nodes in the remainder of this paper.

In the things tier, the energy consumption of IoT devices is mainly caused by communication component and computation component. The energy consumption of IoT devices consists of two parts, sending data and receiving data, which can be evaluated as follows:

$$E_{\text{tx}}(D,l) = E_{\text{elec}} \times D + \varepsilon_{\text{amp}} \times l \times d^x \quad (3)$$

$$E_{\text{rx}}(D,l) = E_{\text{elec}} \times l \quad (4)$$

where D represents the size of data to be transmitted and l represents the Euclidean distance between the sender and the receiver. E_{elec} and ε_{amp} are hardware-related parameters denoting ratio energy dissipation and transmission

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2889482, IEEE Transactions on Cloud Computing

amplifier energy dissipation and x can be varied from 2 to 4. In this paper, we set x to 2 for the sake of easy analysis.

The energy consumption of a processing a task is mainly related to the voltage used by the processor and is proportional to the time used for processing a data unit of a task, which can be mathematically represented by the following equation:

$$E_{\text{comp}}(V_{\text{dd}}, f) = NCV_{\text{dd}}^2 + V_{\text{dd}} \times \left(I_0 e^{\frac{V_{\text{dd}}}{nV_T}}\right) \times \frac{N}{f} \quad (5)$$
$$f \approx K \times (V_{\text{dd}} - c)$$

where V_{T} denotes the thermal voltage and C, I_0, n, K and c are processor dependent parameters.

In conclusion, the energy consumption of an IoT device can be evaluated as below:

$$E_{\text{Things}} = E_{\text{tx}} + E_{\text{rx}} + E_{\text{comp}} \quad (6)$$

The energy consumption of fog nodes depends on both static and dynamic energy consumption. Static energy consumption $E_{\text{Fog,s}}$ refers to the basic energy consumption of the fog nodes while dynamic energy consumption $E_{\text{Fog,d}}$ mainly depends on the resource utilization U_{Fog}.

The energy consumption of a fog node can be evaluated as follows,

$$E_{\text{Fog}} = E_{\text{Fog,s}} + E_{\text{Fog,d}} \quad (7)$$
$$= P_{\text{Fog,s}} \times t_{\text{Fog}} + (P_{\text{Fog,max}} - P_{\text{Fog,s}}) \times t_{\text{Fog}} \times U_{\text{Fog}}$$

where $P_{\text{Fog,s}}$ and $P_{\text{Fog,max}}$ refer to the static and maximum power of a fog node accordingly and t_{Fog} refers to the total processing time of the fog node.

3.5 Problem statement

The objective of the resource scheduling problem is to determine the task graph partition into two parts, task-node allocation and task processing permutation to minimize both the execution time of the application, a.k.a. makespan ($C_{\max}$) and the energy consumption of fog nodes, as well as to prolong the lifetime of the system.

Based on the models mentioned before, the notations of math models are given as Table I. The mathematical forms of the objective and the constraints are then given.

TABLE I. MATHEMATICAL NOTATIONS FOR THE MODELS

	Notation	Implication
Decision variables	$x_{j,r,i}$	$\{0, 1\}$, $x_{j,r,i} = 1$ denotes task j is the r-th task processed on node i, otherwise, $x_{j,i,r} = 0$;
	$C_{r,i}$	$C_{r,i} \geq 0$, the completion time of the r-th task on node i, r=1,…,n;
	m_{i}	$\{1, m\}$, node assignment of task j;
Problem variables & constraints	n	number of tasks in one application;
	m	number of nodes in three tiers, $\{m_{\text{t}}, m_{\text{f}}, m_{\text{c}}\}$ refers to the number of IoT devices, fog nodes and cloud nodes respectively;
	$B(m_{\text{i}}, m_{\text{j}})$	bandwidth between node m_{i} and m_{j};
	w_j	computation time of task j;
	v_{i}	processing speed of node i and $\{v_{\text{t}}, v_{\text{f}}, v_{\text{c}}\}$ refers to the speed of IoT devices, fog nodes and cloud nodes respectively;
	$D(i, j)$	inter-task data size between task i and j;
	$i \rightarrow j$	there is a path from i to j in the DAG and task i is precedent of task j;
Intermediate variables	$\delta(m_{\text{i}}, m_{\text{j}})$	$\{0,1\}$ $\delta(m_{\text{i}}, m_{\text{j}}) = 1$ denotes $m_{\text{i}} \neq m_{\text{j}}$, and $\delta(m_{\text{i}}, m_{\text{j}}) = 0$ denotes $m_{\text{i}} = m_{\text{j}}$;
	$EST(i)$	earliest start time of task i;

The designed objective is given as follows.

$$\min(C_{\max}, E_{\text{Fog}}) \quad (8)$$

Subject to:

$$\sum_{r=1}^{n}\sum_{i=1}^{n} x_{j,r,i} = 1, \forall j \quad (9)$$

$$\sum_{j=1}^{n} x_{j,r,i} \leq 1, \forall i, r \quad (10)$$

$$\sum_{j_1=1}^{n} x_{j_1,r,i} \geq \sum_{j_2=1}^{n} x_{j_2,r+1,i}, \forall r \leq n-1, i \quad (11)$$

$$EST(j) = \begin{cases} 0, & j = 0 \\ \max\limits_{i}\left\{EST(i) + \delta(m_i, m_j) \cdot \dfrac{D(i,j)}{B(m_i, m_j)}\right\}, & \forall\, i \rightarrow j \end{cases} \quad (12)$$

$$C_{r,i} \geq \max\left\{C_{r-1,i}, +\sum_j \left(x_{j,r,i} \cdot EST(j)\right)\right\} + \sum_j \left(x_{j,r,i} \cdot \frac{w_j}{v_{m_j}}\right) \quad (13)$$
$$\forall i, r \geq 1$$

$$C_{\max} \geq C_{n,i}, \forall i \quad (14)$$

$$E_{\text{Fog}} = P_{\text{Fog,s}} \cdot m_f \cdot C_{\max} + (P_{\text{Fog,max}} - P_{\text{Fog,s}}) \cdot \sum_j \frac{w_j}{v_{m_j}} \quad (15)$$
$$\forall m_j \in \text{Fog}$$

Equation (8) defines the designed objective as a bi-objective minimization problem. In the constraints, Equations (9) – (11) guarantee the validity of the generated scheduling scheme. More precisely, Equation (9) ensures that each task can be processed once and only once. Equation (10) ensures that all tasks are scheduled sequentially, so that only one task can be processed by the processor at a given time. Equation (11) ensures that the order of the queue is strictly met, which means if there is no r-th task in the processing queue, then there must be no (r+1)-th task in the queue. Equation (12) defines the earliest start time of each task. In Equation (13), the completion time of the tasks in the processing queue is calculated and Equation (14) defines makespan. From (7), E_{fog} can be easily derived and evaluated by (15).

3.6 Lower bound analysis

A lower bound analysis for the task graph scheduling problem is provided. As mentioned, each IoT application is represented by a DAG. Let $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{w}, \boldsymbol{D})$, a path from a vertex v_0 to v_{k} in $\boldsymbol{G}$ is a sequence $(v_0, v_1, \ldots, v_{\text{k}})$ of vertices connected by the edges $(v_{\text{i-1}}, v_{\text{i}}) \in \boldsymbol{E}$. And the length of a path x is the sum of the weights of its nodes and edges [16]:

$$\mathbf{len}(x) = \sum_{n \in x, \boldsymbol{V}} w_n + \sum_{e \in x, \boldsymbol{E}} D(e) \quad (16)$$

The computation length of a path x is the sum of the weight of the included nodes:

$$\mathbf{len_w}(x) = \sum_{n \in x, \boldsymbol{V}} w_n \quad (17)$$

A critical path cp of a task graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{w}, \boldsymbol{D})$ is the longest path in $\boldsymbol{G}$ as defined in (18). Similarly, the computation critical path cp_{w} can be defined as below.

$$\mathbf{len}(cp) = \max_{x \in \boldsymbol{G}}\{\mathbf{len}(x)\} \quad (18)$$

From Equations (12)-(14), we can derive that $C_{\max} \geq \mathbf{len_w}(cp_{\text{w}})/v$ if the task graph is assigned to the homogeneous machines with processing speed v. However, the IoT system discussed in this paper is a distributed heterogeneous system and the nodes within the system have different

processing speeds v_t, v_f and v_c. Besides, the entry tasks of the application are processed on the things tier. Thus, the above lower bound is no longer held and needs to be adjusted accordingly.

A sketch plot of a sample task graph is provided in Figure 2. In the figure, task 0 is the virtual entry node with zero execution time and zero communication overhead. The successors of task 0 are the real entry tasks, which are required to be assigned to the things tier for their processing. In this figure, we also have $cp_w = \{0, 1, \ldots, k\text{-}1, k, \ldots, l, l\text{+}1, \ldots, n\text{+}1\}$ and there are total n_{cp} tasks in cp_w. Considering the conditions of the system, a lower bound is given as follows:

$$\min_{\substack{2\le k\le n_{cp}+1\\ k-1\le l\le n_{cp}}} \left\{ \sum_{j=1}^{k-1}\frac{w_j}{v_t}+\frac{D(k-1,k)}{B_1}+ \sum_{j=k}^{l}\frac{w_j}{v_f}+\frac{D(l,l+1)}{B_2}\sum_{j=l+1}^{n_{cp}}\frac{w_j}{v_c} \right\} \quad (19)$$

where B_1 denotes the bandwidth between the things tier and the fog tier and B_2 denotes the bandwidth between the fog tier and the cloud tier. And $(k\text{-}1, k)$ identified as cmm_1 is the cut edge between the things tier and the fog tier. As entry nodes must be assigned to IoT devices, we have $k \geq 2$. $(l, l\text{+}1)$ identified as cmm_2 is the cut edge between the fog tier and the cloud tier. Since the fog tier is linked to both the things tier and the cloud tier, we can further derive that $l \geq k - 1$. The tasks before task k are assigned to the things tier and the tasks belonging to (k, l) and (l, n_{cp}) will be assigned to the fog tier and the cloud tier correspondingly.

To unify the cases, the virtual exit vertex $(n_{cp}\text{+}1)$ is employed so that we can use only one equation to represent all scenarios. Two cuts are illustrated in Figure 2, where if a cut is set on the edge $(n_{cp}, n_{cp}\text{+}1)$ means that all the tasks before are assigned to the same tier for processing. To be more specific, $k = l\text{+}1 = n_{cp}\text{+}1$ means that all the cp_w tasks are assigned to the things tier; $k = l\text{+}1 < n_{cp}$ means that all the tasks after task k are uploaded to the cloud tier via the fog tier; $k < l\text{+}1 = n_{cp}\text{+}1$ means that the tasks are assigned to the things tier and the fog tier. In this case, we only need to traverse $(n_{cp}\text{-}1)^2+(n_{cp}\text{-}2)^2+\ldots+2^2+1^2 = (n_{cp}\text{-}1)(n_{cp}\text{-}2)(2n_{cp}\text{-}1)/6$ cases and the lower bound of the time complexity is $O(n^3)$.

From Equation (15), it is easily seen that $E_{Fog} \geq P_{Fog,s} \cdot m_f \cdot C_{max}$, as the lower bound of C_{max} is given in (19), the lower bound (LB) of E_{Fog} could be calculated. In this way, $(LB, P_{Fog,s} \cdot m_f \cdot LB)$ is the lower bound for both objectives of the problem.

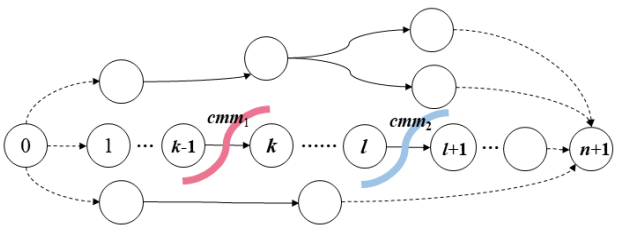

Figure 2. Sketch plot of a task graph and two cuts

4 Details of the Proposed Algorithm

The proposed algorithm that aims to achieve the desired trade-off between energy saving and shorten the application completion time is introduced in detail in this section. Firstly, an overview of the proposed algorithm is given. The partition operator that employed in our enhanced EDA approach is then introduced, and the pseudo code is also provided. After that, the procedures of both standard EDA and our proposed EDA with the partition operator (EDA-p) are introduced.

4.1 Flowchart

A two-phase scheduling algorithm is developed to find out the desired tradeoff for the objectives. Firstly, the task graph is divided into two non-overlapping sub-graphs and allocated to the thing tier and the virtual node (the computing capabilities provided by the remaining tiers of the system). And then, the proposed EDA is adopted to sort the tasks assigned to the fog tier and the cloud tier where a probability model P is initialized according to the precedence constraints between the tasks in these two tiers. The permutations of the population of EDA, which contains N individuals, are sampled by P. And then makespan and energy consumption is calculated, and each individual is used to update AS. After all the individuals are evaluated, P is evaluated with the individuals in AS. The flowchart of our proposed algorithm is illustrated in Figure 3.

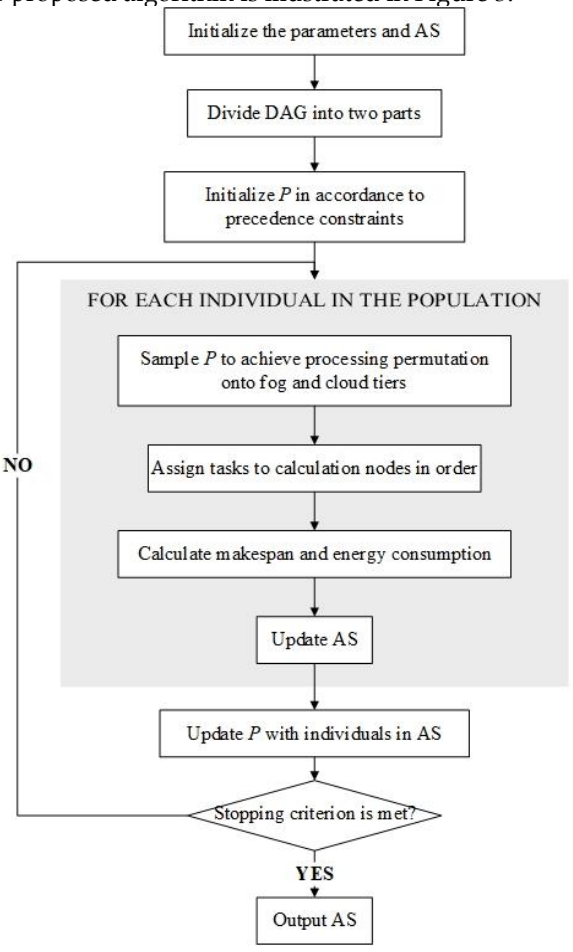

Figure 3. Flowchart of the proposed algorithm

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2889482, IEEE Transactions on Cloud Computing

4.2 Partition operator

To distribute the right part of each IoT application to the things tier, a partition operator is designed to divide the DAG into two parts. One part of the DAG is assigned on the things tiers and processed locally, and the remaining part of the DAG is sent to the fog and cloud tiers for further processing. In [17], the authors conducted a survey to show that the tasks with intensive computation workload and light communication overhead are uploaded to fog and cloud nodes for processing so that the overall system performance can be thus improved and energy can be saved as well. It is crucial to minimize the data transmitted between the things tier and the fog tier. However, the operations of both data sending and receiving consume a large amount of energy for these battery powered IoT devices. Thus, the minimization of the data transferred between IoT devices is equally important in our work since the energy depletion of these devices can affect the system performance in a negative way and shorten the system lifetime. In addition, the computation capability of IoT devices is not as good as the fog and the cloud nodes. It is to be expected that the completion time of the application can be extensively increased when a substantial portion of the tasks is assigned to IoT devices for processing. By combining these factors, we develop a simple processor model that includes m_s IoT devices and a virtual compute node that represents the computational resources that can be obtained from both fog and cloud tiers. The capability of this virtual computation node can be represented as $v'_f = v_f \times m_f$. With this model, our partition operator is able to decide how to partition a DAG into two parts and how to distribute the corresponding tasks onto (m_s+1) nodes effectively, particularly for those tasks on the things tier.

This model can be further extended to the multiple applications scenario with a proper adjustment. Compared to the single application case, it is possible that the virtual computation node is not available to handle more tasks when a new application arrives. Thus, the estimated available time (a.k.a. machine time) of the virtual computation node is updated when the graph partitioning is completed. By doing so, the partition operator can be adjusted according to the latest status of the IoT system when each new application arrives. For the algorithms based on the greedy heuristic approach, the newly arrival tasks are more likely to be allocated to the IoT devices before the virtual computation node becomes available. With such a solution, the makespan and energy consumption of the fog nodes may be decreased, but it is a waste of the battery power of IoT devices and thus shortens their lifetime.

Before providing the details of our partition operator, some definitions are made as follows. We first consider a mapping Pt: $V \rightarrow \{1,2\}$, where Pt(k) = 1 denotes task k is assigned to the things tier and Pt(k) = 2 denotes task k is assigned to the virtual computation node. The border vertices in the task graph are represented as [18]:

$$\gamma := \{u | \exists (u,v) \in Edge: Pt(u) \neq Pt(v)\} \quad (20)$$

Also, let **pred**(k) denote the task set, i.e. $\{u|(u,k) \in Edge\}$, and **succ**(k) denote the task set i.e. $\{v|(k,v) \in Edge\}$.

Property.

For any node k in the DAG, we have $\forall u \in \mathbf{pred}(k)$, Pt($u$) = 1, and $\forall v \in \mathbf{succ}(k)$, Pt($v$)=2. Then if $\sum_{u \in \mathbf{pred}(k)} D(u,k) < \sum_{v \in \mathbf{succ}(k)} D(k,v)$, we have

$$\sum_{(u,v) \in C, \mathrm{Pt}(k)=1} D(u,v) > \sum_{(u,v) \in C, \mathrm{Pt}(k)=2} D(u,v) \quad (21)$$

where $C := \{(u,v) \in \mathrm{Edge} | \mathrm{Pt}(u) \neq \mathrm{Pt}(v)\}$.

Proof.

It is not hard to see that $k \in \gamma$. As defined in (20), if Pt(k) = 1 and then we have $\exists (k, \mathrm{v}) \in$ Edge, where v belongs to **succ**(k) and Pt(k) ≠ Pt(v). Similarly, when Pt(k) = 2 then $\sum_{(u,v) \in C} D(u,v)$ is affected by Pt(k). And then we have:

$$\begin{aligned} &\sum_{(u,v) \in C, \mathrm{Pt}(k)=1} D(u,v) \\ &= \sum_{(u,v) \in C \setminus (k,v)} D(u,v) + \sum_{v \in \mathbf{succ}(x)} D(x,v) \\ &> \sum_{(u,v) \in C \setminus (u,k)} D(u,v) + \sum_{u \in \mathbf{pred}(x)} D(u,x) \\ &= \sum_{(u,v) \in C, \mathrm{Pt}(k)=2} D(u,v). \end{aligned}$$

Based on this property, a criterion is designed for making the decision on whether a task is assigned to the virtual computation node or not. If a certain task satisfies that $\sum_{u \in \mathbf{pred}(k)} D(u,k) < \sum_{v \in \mathbf{succ}(k)} D(k,v)$, then it is assigned to the virtual computation node and will most likely reduce the energy consumption caused by the communication between IoT devices.

In addition, to speed up the processing of the tasks and reduce the power drawn from the battery, as well as to decrease the computational complexity of the partition operator, another criterion is designed as follows.

$$Pt(u) \leq Pt(v), \qquad \forall\, u \rightarrow v \quad (22)$$

By doing so, all the tasks with the predecessors assigned to the virtual node remain with the same allocation without performing the previous criterion.

TABLE II. THE NOTATIONS USED IN PSEUDO CODE

Notation	Meaning
Mt	Machine time vector, length is m_s+1, represents the available time of the IoT devices in the cluster and the virtual computation node;
Ct	Completion time, length is 2, represents the completion time of a certain task processed in things tier and virtual computation node respectively;
Be	Battery energy consumption, length is 2, represents the energy consumption on IoT device caused by a certain task when processed in things tier and virtual computation node;
input(*k*)	$\sum_{u \in \mathrm{pred}(k)} D(u,k)$;
output(*k*)	$\sum_{v \in \mathrm{succ}(k)} D(k,v)$;
EST	Earliest starting time vector for each task in things tier and the virtual computation node respectively;
SendE(*D*)	Battery energy consumption of sending data sized D;
ReceiveE(*D*)	Battery energy consumption of receiving data sized D;
ProcessE(*k*)	Battery energy consumption of processing task k;

In our approach, the value of the bottom level (b-level) of each task is calculated, where b-level is the longest path from the current task to the exit task [2]. By descending order of b-level, each task is evaluated by the selection criteria. As introduced, some tasks are assigned to the virtual node directly and others are to find a desired balance between energy consumption and execution time. Once all

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2889482, IEEE Transactions on Cloud Computing

the tasks are evaluated, the partition strategy is determined. The detailed pseudo code is given as follows and the notations are given in Table II.

Algorithm 1: Procedure of partition operator

For each task k in descending b-level list
  If $k \in$ entry tasks
    k is assigned to the IoT device with earliest available time;
  Else If input(k) < **output**(k) **or** for any $x \in$ **pred**(k), there is $x \in$ Fog Tier
    k is assigned to virtual computation node;
  Else
    Find IoT device s with earliest available time;
    $Ct[0]$:= **max**($Mt[s]$, $task.EST[0]$) + t_k / v_s;
    $Ct[1]$:= **max**(Mt[Fog], $task.EST[1]$) + t_k / v'_f;
    Let **Dout**(k) := $\sum_{u \in \mathrm{pred}(k)} D(u,k) - \sum_{u \text{ assigned on IoT devices}} D(u,k)$
    $Be[0]$:= **SendE** (**Dout**(k)) + **ReceiveE**(**Dout**(k));
    $Be[0]$ += **ProcessE** (k);
    $Be[1]$:= **SendE** (**input**(k));
    If $Ct[0] < Ct[1]$ **and** $Be[0] < Be[1]$
      k is assigned to IoT device s;
    Else If $Ct[0] > Ct[1]$ **and** $Be[0] > Be[1]$
      k is assigned to the virtual node;
    Else
      If $(Ec[0] - Ec[1])/(Ec[0] + Ec[1]) + (Ct[0] - Ct[1])/(Ct[0] + Ct[1]) > 0$
        k is assigned to the virtual node;
      Else
        k is assigned to the Things Tier;
    End
  End
  Update Mt and EST of each task in **succ**($task$).
End

4.3 EDA scheme

Once the DAG is partitioned, the tasks are divided into two parts. The tasks assigned to the things tier are sorted by the b-level descending order when the partition operator is performed. The tasks assigned to the virtual computation node are sorted by our proposed EDA sampling method. When these two processing sequences are determined, the tasks will then be assigned to the nodes by the rule of earliest finish time first (EFTF). After that, the makespan and energy consumption of each application can be determined accordingly.

In this work, EDA is employed to produce the processing sequence of the tasks allocated to the fog tier and the cloud tier. The standard EDA procedure is summarized as follows. Firstly, a probability model P is built, and the individuals are sampled with P. The generated individuals are then evaluated and P will be updated according to the results of the evaluations. After that, the individuals are sampled with the updated P value until the stopping criterion is met. The details of our proposed EDA for this scheduling problem are given as follows.

Encoding and decoding

The individuals are encoded as in Figure 4, where T represents the number of tasks assigned to the things tier and F represents the number of tasks assigned to the fog tier and the cloud tier. The first row represents the tasks assigned to the things tier, and its processing sequence has been determined by the partition operator. The second sequence represents the processing sequence of the tasks assigned to the fog tier and the cloud tier, which is generated by our proposed EDA.

For the tasks assigned to the fog and cloud tiers, the computation nodes in both tiers are treated as unrelated processors where the processor assignment is determined by EFTF. In other words, the tasks are allocated to the processor with the smallest completion time. When the individuals are evaluated (decoded), the tasks in the things tier are first considered and assigned accordingly. Then the tasks in the fog tier and the cloud tier are assigned according to the processing permutation generated by EDA.

Things	i_1	i_2	…	i_l	…	i_T
Fog&Cloud	j_1	j_2	…	j_k	…	j_F

Figure 4. Example of encoding method

Probability model

Considering the precedence constraints between the tasks assigned to the fog tier and the cloud tier, a task permutation probability model P based on the relative position relationship between the tasks is built. A fog and cloud task processing permutation is produced by P. The relative position probability is designed as (22), where $p_{i,j}(g)$ represents the probability that task i is placed ahead of task j in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2889482, IEEE Transactions on Cloud Computing

the permutation at the g-th generation during the evolution. Clearly, $p_{i,j}(g)+ p_{j,i}(g) = 1$ for the cases of task i being ahead of or behind task j.

$$P(g) = \{p_{i,j}(g)\}_{T\times T} \quad (23)$$

Probability model initialization

For any given DAG, the probability value ($p_{i,j}$) is initialized as (23) so that the precedence constraints of task pairs (i, j) can be thus satisfied. For the task pairs without constraints, their order is given randomly and the probability is set at 0.5.

$$p_{i,j}(0) = \begin{cases} 1, i \to j \\ 0, j \to i \\ 0.5, \text{otherwise} \end{cases} \quad (24)$$

Updating method

For the single-objective optimization problem, the solutions produced by EDA can be evaluated by the single objective, such as makespan, and the individuals with smaller makespan are used to update $P(g)$. For the multi-objective optimization problem, the individuals in AS are used to update P as the AS is the set of elite solutions under Pareto ranking criterion. The probability value is then updated by using the population based incremental learning method (PBIL) [19]:

$$p_{i,j}(g+1) = (1-\alpha)p_{i,j}(g) + \alpha \times \sum_k I_{i,j}^k(\mathrm{g})/|AS| \quad (25)$$

$$I_{i,j}^k(\mathrm{g}) = \begin{cases} 1, \text{task } i \text{ is before task } j \text{ in the } k\text{th individual} \\ 0, \text{otherwise} \end{cases} \quad (26)$$

where $\alpha \in (0,1)$ denotes the learning rate, $|\mathrm{AS}|$ denotes the size of archive set, and $I_{i,j}^k(g)$ is the indicator function corresponding to the k-th solution of AS.

Sampling method

In EDA, the new individuals are produced by sampling the updated probability model. For each position which has not been determined in the fog and cloud processing permutation, a probability array is calculated. The element in the array denotes the probability that the certain task is assigned to this position. The product mentioned in pseudo code indicates the probability that a task is behind of all the other available tasks. Then the array is normalized and sampled by the roulette wheel method. A detailed pseudo code is given as follows.

Algorithm 2: Procedure of EDA sampling method

A is the available task set, initialized with tasks without parent tasks in fog and cloud tier;
l is the label in task processing permutation, $l = 0$;
While ***A*** is not empty
  Calculate a sampling probability *sp* for each task in ***A***;
  For each task t in ***A***
    $sp(t) \coloneqq \prod_{i\neq t, i\in A} p_{t,i}(g)$;
  End
  Normalize probability array *sp*;
  Sample *sp* with Roulette Wheel Method to achieve the l-th processing task t in the permutation;
  Push the child task of t with all parents assigned into A;
  l ++;
End

5 Simulation

To evaluate the performance of our proposed EDA-p, we compare it with the selected algorithms, the heuristic method and the multi-objective EDA. For fair comparison, the algorithms are all coded in C++ language and run on the same machines.

5.1 Comparison algorithms

To evaluate our proposed algorithm, two benchmark algorithms are employed and operated under the same running environment with our algorithm. One is the heuristic algorithm used in single objective task graph allocation optimization problem, the b-level heuristic method. In the b-level heuristic method, all the tasks are sorted in descending order of b-level values. And then the tasks are assigned to the computation nodes, including things, fog and cloud nodes, according to the EFTF rule.

We compare the single objective, makespan, generated by b-level heuristic method with the lower bound achieved before. It is apparent that in some instances, the difference is very small, such as when n=50, *CCR*=0.1 and 0.5, the minimum relative percentage deviation (RPD) are 0.39 and 0.17 accordingly. We could conclude that the b-level heuristic method performs well when *CCR* is markedly less than 1.0 as well as that the lower bound is tight when the communication overhead between the tasks is not very significant. And when it comes to the case that *CCR* is larger, the estimation to lower bound is not so tight (when n=50, *CCR*=10, the minimum RPD is 127.70). The lower bound does not consider the tasks outside the critical path. The significant communication overhead causes notable idles to occur in the scheduling scheme and makes the lower bound not as tight as when *CCR* is small.

The other algorithm is called simple EDA (sEDA) to serve as a multi-objective approach to generate a scheduling scheme for the entire system. Compared to our proposed EDA-p, no partition operator is employed in sEDA. The probability model to produce the permutation is adopted in sEDA to produce a whole task processing permutation with length n. The tasks are assigned to all three tiers according to the EFTF rule. The probability model is then updated with individuals in AS. The purpose of using sEDA is to prove whether the involvement of the partition operator is effective or not.

5.2 Simulation environment

The testing environment settings and the variables are set to the values used in [14]. The parameters mentioned in Equation (5) are set as follows: E_{elec} = 50nJ/b, $\varepsilon_{\mathrm{amp}}$ = 10pJ/b/m^2, V_{T} = 26mV, C = 0.67nF, I_0 = 1.196mA, n = 21.26, K = 239.28MHz/V, V_{dd} = 1V, and c = 0.5V. The processing speed of the IoT devices, fog nodes and cloud nodes are set as 1, 5 and 10 accordingly. The bandwidth within each tier is set to 1, 5 and 10 and the bandwidths between the things tier and the fog tier, the fog tier and the cloud tier are set to 1 and 8 respectively. The data uploaded to the cloud center needs to transfer via the Internet. Since no general analytical equation exists to measure the transmission delay over the Internet, a fixed communication delay is added for transmitting a generalized unit of a communication task

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2889482, IEEE Transactions on Cloud Computing

between the fog tier and the cloud tier, which is set as 0.1s based on the long-time observation of our network environment. The distance between the IoT devices is set as 1m on average and the initial battery power is set as 0.1J. To construct the things tier, different numbers of IoT devices ({3, 3, 3, 4, 4, 4}) are set in corresponding clusters. Six fog nodes are placed to link with their responsible cluster respectively. Two public cloud services are considered in this system.

During the study of the single application case, for each instance, the application is produced by the IoT devices in one cluster. As the communication between the things clusters needs to be relayed by the corresponding fog nodes, the communication time and the power consumption are much greater than if the tasks are forward to fog and cloud nodes for further processing. Thus, the computation tasks transferred between things clusters are eliminated. Therefore, only one cluster of IoT devices is considered in the single benchmark instance study.

5.3 Single application study

The single application benchmark is generated by the standard task graph benchmark instances on the homogeneous DAG scheduling problem provide by. As with instances in [20], it does not include the inter-task data size, the values of the communication time contributing to normal distribution according to different *CCR* settings are produced and used for experiments. For the test sets, n = {50, 100, 300} and *CCR* = {0.1, 0.5, 1.0, 10.0}, and each test set consists of 180 independent testing instances. Thus, 3×4×180 = 2160 instances are used to validate the efficiency of the proposed algorithm.

Experiment settings

To compare the performance of the algorithms fairly, the stopping criterion of both algorithms is set as 30 generations. All the parameters in EDA are set the same for both sEDA and EDA-p. As the probability model is updated by the whole AS, the percentage of elite population does not need attention. The other two key parameters of EDA, the population size (N) and the learning rate (a), are determined by the Taguchi method of design-of-experiment method (DOE) [21]. DOE is a statistical method, which is used to investigate the effect of parameters on the performance of the algorithm under different levels. Orthogonal arrays of different parameter levels are used to conduct the experiment with multiple runs. In this experiment, four levels are set for each parameter as shown in Table III and 4^2 full-factorial experiments are employed.

TABLE III. FACTOR LEVELS OF PARAMETERS

Parameters	Factor level			
	1	2	3	4
N	20	25	30	35
a	0.025	0.05	0.075	0.10

For each combination of n×*CCR* (3×4=12), an instance is randomly chosen. For each instance, 16 combinations of η × a are tested independently and the obtained AS_{ci} (ci = 1, 2, …, 16) are stored. The final AS (FAS) are obtained by integrating AS_1, AS_2,…, AS_{16}. Then, the contribution of a certain combination (*CON*) is counted as $CON(ci) = |AS'_{ci}|/|FE|$, where $AS'_{ci} = \{X_l \in AS_{ci} | \exists X_{l'} \in FAS, X_l = X_{l'}\}$. The average *CON* of each combination is calculated and used as the response value (RV), shown in Table IV.

TABLE IV. THE RV VALUE

Experiment Numbers	Factor		RV (%)	Experiment Numbers	Factor		RV (%)	Experiment Numbers	Factor		RV (%)	Experiment Numbers	Factor		RV (%)
	N	a			N	a			N	a			N	a	
1	1	1	6.66	5	2	1	5.49	9	3	1	5.30	13	4	1	7.51
2	1	2	5.58	6	2	2	3.62	10	3	2	13.19	14	4	2	6.85
3	1	3	5.39	7	2	3	9.79	11	3	3	1.77	15	4	3	7.00
4	1	4	3.47	8	2	4	6.05	12	3	4	8.62	16	4	4	3.72

Note: "RV" indicates the response value considered in design of experiment.

The response values of DOE are listed in Table IV and the main effects plot and the influence trend table is shown in Figure. 5 and Table V. From Table V, the variances of the response value under different levels of N and a have little difference, which means that the effect of N and a on the algorithm performance is nearly the same.

TABLE V. INFLUENCE TREND TABLE

Level	N	a
1	0.053	0.062
2	0.062	0.073
3	0.072	0.060
4	0.063	0.055
Delta	0.019	0.018
Rank	1	2

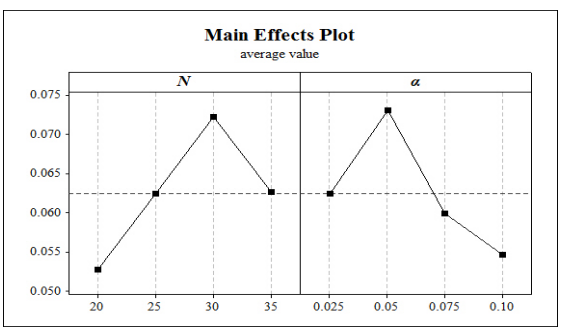

Figure 5. The effect trend of parameters

From Figure 5 and Table V, it is better for N to be set neither too big nor too small. And a small a is suitable for the benchmark instances as the stopping criterion is 30 generations. According to the results, N=30, a=0.05 are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2889482, IEEE Transactions on Cloud Computing

suggested as the parameters in the simulation experiments.

Comparison with heuristic methods

As the b-level heuristic method produces one solution for each problem, our algorithm is compared with the heuristic methods in terms of the percentage of the solutions of heuristics dominated by the solutions of our algorithm. The results are shown in Table VI.

In Table VI, the column "dominating b-level(%)" denotes the percentage of the instances in which the b-level heuristic solution is dominated by any solution in the Pareto set produced by EDA-p. the "non-dominating b-level(%)" denotes the percentage of instances in which the b-level heuristic solution is not dominated by any solution of EDA-p and no solution of EDA-p is dominated by b-level solution. The "dominated by b-level(%)" denotes that there is at least one solution produced by EDA-p dominated by the b-level solution. The average value of 180 instances under each problem scale is calculated and summarized in Table VI.

TABLE VI. RESULTS COMPARED WITH B-LEVEL HEURISTIC METHOD

Task number (n)	CCR	Dominating b-level (%)	Non-dominating b-level (%)	Dominated by b-level (%)
50	0.1	61.67	34.44	3.89
	0.5	75.56	22.78	1.67
	1.0	94.44	3.33	2.22
	10.0	94.44	1.67	3.89
100	0.1	48.33	26.11	25.56
	0.5	95.00	3.89	1.11
	1.0	98.89	0.56	0.56
	10.0	98.89	0.56	0.56
300	0.1	15.00	18.89	66.11
	0.5	88.89	9.44	1.67
	1.0	97.22	2.78	0.00
	10.0	98.33	0.56	1.11

From Table VI, it can be seen that EDA-p performs much better than the b-level heuristic under most scenarios. When *CCR* increases, the proportion of instances dominating the b-level solution is also increased, particularly when *CCR* is greater than or equal to 1.0, more than 90% of solutions produced by the b-level heuristic method are dominated by the Pareto set produced by EDA-p. Thus, it could be concluded that when $CCR \geq 0.5$, EDA-p performs significantly better than the b-level heuristic. However, EDA-p is not able to outperform the b-level heuristic method when *CCR* = 0.1. The possible reason for the b-level approach performing well when *CCR* is small is that the amount of data transmission is limited. As a result, it affects the scheduling scheme slightly and the greedy EFTF rules adopted in the b-level approach performs closely to the best solution of makespan. Along with *CCR* increases, the greedy EFTF rule chooses the processor with the earliest finish time. As the data transfers between tasks assigned to the things tier and their successors spend more time uploading the intermediate results rather than processing the data locally, the greedy EFTF rules assign a large part of tasks to the things tier. However, the processing speed of the things tier is low and this may slow down the scheduling. In addition, the data transfers might be larger when the IoT devices are overloaded and the tasks have to be uploaded.

Comparison with simple EDA

We compared our algorithm with simple EDA in terms of *C* metrics. The average *C* metric values of 180 instances under each scale are calculated and summarized in Table VII, where *C*(EDA-p, sEDA) reflects the percentage of the solutions in simple EDA that are dominated by or the same as the solutions in EDA proposed in this work.

TABLE VII. *C* METRIC COMPARATIVE RESULTS WITH SIMPLE EDA

Task number (n)	CCR	*C*(EDA-p, sEDA)	*C*(sEDA, EDA-p)
50	0.1	0.25	0.41
	0.5	0.50	0.22
	1.0	0.75	0.20
	10.0	0.81	0.17
100	0.1	0.18	0.60
	0.5	0.55	0.20
	1.0	0.94	0.02
	10.0	0.96	0.02
300	0.1	0.08	0.78
	0.5	0.68	0.08
	1.0	0.97	0.01
	10.0	0.96	0.01

From Table VII, it can be seen when *CCR*=0.1, the performance of EDA-p is not as good as simple EDA. The possible reason is the probability model in simple EDA generates the whole processing permutation and it explores a larger solution space while the processor assignment of a part of tasks in EDA-p is fixed and its solution space is smaller. In this way, sEDA can produce better solutions when *CCR* = 0.1. However, EDA-p performs much better than sEDA under other selected *CCR* values. As sEDA is the foundation of EDA-p except the partition operator, it could be seen that it is necessary to pre-partition the task graph and this operator is effective to improve the scheduling performance of this problem under most situations.

5.4 Multiple applications

To evaluate the performance of the proposed algorithm in a realistic environment, a discrete-event simulator is written in C programming language. In the simulations, three performances are considered: system lifetime, makespan and fog tier energy consumption. System lifetime refers to the time that the first IoT device is drained of its energy [22-23]. To compare the algorithms fairly, the system lifetime is recorded when the system dies, while the simulation is continued to evaluate the other two metrics. Makespan is calculated as the completion time of the last task of all of the applications and the whole energy consumption of fog node is computed as Equation (15).

To select the elite solutions to evaluate the evolutionary

algorithms during one application being processed, the makespan of a certain application begins at one of the tasks being processed until the last task is completed. The energy consumption of the fog node is calculated from the end time of the previous application to the end time of the existing application as in Equation (7).

To further evaluate the performance of the proposed algorithms, simulations are run on applications with 100 randomly generated DAGs. The arrival time of the applications is submitted to a random distribution of (10, 20). A well-known pseudorandom task graph generator TGFF [24] is used to produce application. Unless specifically stated, the task number n of a DAG is set to 200 with 50% variation and *CCR* is set as {0.1, 0.5, 1.0, 5.0} correspondingly. For each task of the application, the data size follows the normal distribution with the average value of a certain *CCR*. And the maximum number of successors and processors of a certain task are both set as 10.

In addition, in the multi-application simulation, it is necessary for the multi-objective algorithms to choose one solution from the Pareto set as the final scheduling scheme. The method used in this paper is to minimize the weighted objectives as follows where S represents the Pareto set produced by an algorithm and β is the weighted value.

$$\min_{i \in S} \beta \times C_{\max,i} + (1-\beta) \times E_{\text{fog},i} \qquad (26)$$

To find out an appreciate value for β, 9 different values {0.1, 0.2, …, 0.8, 0.9} of β are tested and three optimization objectives are considered respectively. The results show that the change of β does not significantly affect makespan, but when β increases, the lifetime of the system decreases and the fog energy consumption increases. Thus, β is set to 0.1 in the simulation.

Comparison with benchmarks

Firstly, experiments are carried out under different values of *CCR*. For each round of simulation, 100 random applications are produced and assigned to the system. 50 rounds of simulation are repeated for each algorithm and the average value of each metric is listed in Table VIII.

TABLE VIII. AVERAGE VALUE COMPARISON UNDER DIFFERENT *CCR*

	CCR	EDA-p	b-level	sEDA
lifetime	0.1	1248.66	382.06	319.34
	0.5	862.80	384.94	373.71
	1.0	679.51	241.86	228.11
	5.0	718.97	241.49	207.48
makespan	0.1	3500.37	3506.35	3540.59
	0.5	3524.99	3515.09	3523.68
	1.0	3538.18	3762.85	3610.91
	5.0	4338.80	9460.72	7835.16
energy	0.1	5657.96	5598.32	5377.79
	0.5	5353.79	5300.69	5144.91
	1.0	5494.57	5505.93	5182.95
	5.0	5390.15	5613.51	4543.89

Secondly, 50 rounds of 100-applications simulation are applied for each algorithm. The applications are produced with random task number and *CCR*. The average value of each metric is listed in Table X and the box plots are presented in Figure 6.

TABLE IX. AVERAGE VALUE COMPARISON

	EDA-p	b-level	*p*-value
lifetime	**642.23**	302.69	0.000
makespan	**3600.92**	5556.18	0.000
energy	**5362.57**	5375.86	0.336
	EDA-p	**sEDA**	***p*-value**
lifetime	**642.23**	249.19	0.000
makespan	**3600.92**	4671.93	0.000
energy	5362.57	**5011.42**	1.000

In Tables IX the last column shows the *p*-value of two sample *t*-tests under 95% confidence level. For lifetime, the alternative hypothesis is "the objective lifetime of EDA-p is not larger than the comparison algorithm 95% confidence level" and for the other two objectives, the alternative hypothesis is "the objective of EDA-p is not smaller than the comparison algorithm 95% confidence level".

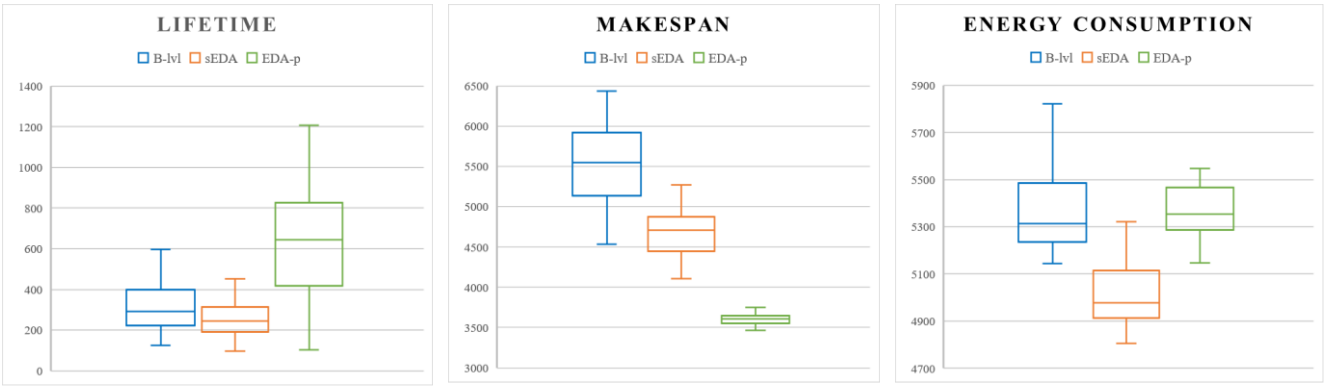

Figure 6. Box plot of results comparison

It can be seen that lifetime and makespan of solutions produced by EDA-p are significantly better than the other algorithms at 95% confidence level. In addition, the null hypothesis could not be refused for the fog energy consumption objective when compared with the other algorithms. It could be explained that for b-level and sEDA, a

large proportion of tasks is assigned to IoT devices when fog and cloud nodes are busy. This leads to the IoT devices running out of power very quickly while reducing the load of fog nodes. Thus, the energy consumption of fog nodes is smaller than solutions of EDA-p.

It can be seen from Figure 6 that EDA-p is much better than sEDA and the b-level heuristic method. Although considering makespan and fog energy consumption, during the evolution of the algorithm, EDA-p performs very well on lifetime and makespan metrics. In addition, EDA-p performance is similar to other algorithms on energy consumption metric. In conclusion, EDA-p is an efficient algorithm to solve the scheduling problem under the three-tier system.

6 Conclusion

In this work, an energy efficient evolutionary algorithm is proposed to address the resource scheduling of three tiers of IoT systems. At first, we give a detailed system and computation model to describe and formulate the problem and try to simulate the realistic problem. A lower bound of this is given according to the mathematic model. An EDA scheduling algorithm compared with partition operator is then designed. The pre-partition operator reduces the negative effects brought by the greedy task assignment rule. In addition, EDA is used for producing task permutation. The experiments conducted on both single and multi-application cases show that our algorithm is effective in reducing makespan and the energy consumption of devices, as well as prolonging the lifetime of IoT devices.

In the future, we plan to address applications with soft or hard deadlines. In addition, tests will be run on the prototype system built from standard hardware and software elements.

Acknowledgments

This research is supported by the National Key R&D Program of China [No. 2016YFB0901900], the National Natural Science Fund for Distinguished Young Scholars of China [No. 61525304] and the National Natural Science Foundation of China [No. 61873328].

References

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," *Proc. first edition of the MCC workshop on Mobile cloud computing*, pp. 13-16, Aug. 2012.

[2] Y.-K. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Computing Surveys*, vol. 31, no. 4, pp. 406-471, Dec. 1999.

[3] S.-Y. Wang and L. Wang, "An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem," *IEEE Trans. systems, man, and cybernetics: systems*, vol. 46, no. 1, pp. 139-149, April. 2016, doi: 10.1109/TSMC.2015.2416127.

[4] M. Zangari, A. Mendiburu, R. Santana, and A. Pozo, "Multiobjective decomposition-based Mallows Models estimation of distribution algorithm. A case of study for permutation flowshop scheduling problem," *Information Sciences*, vol. 397, pp. 137-154, Aug. 2017.

[5] C.-g. Wu, L. Wang, and J.-j. Wang, "A t-level driven search for estimation of distribution algorithm in solving task graph allocation to multiprocessors," *Proc. IEEE Automation Science and Engineering (CASE '17)*, pp. 630-635, Aug. 2017, doi: 10.1109/COASE.2017.8256173.

[6] C.-g. Wu and L. Wang, "A multi-model estimation of distribution algorithm for energy efficient scheduling under cloud computing system," *J Parallel and Distributed Computing*, vol. 117, pp. 63-72, July. 2018.

[7] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar, "Towards QoS-Aware Fog Service Placement," *Proc. IEEE Fog and Edge Computing (ICFEC '17)*, pp. 89-96, May. 2017, doi: 10.1109/ICFEC.2017.12.

[8] A. Aral and T. Ovatman, "Network-aware embedding of virtual machine clusters onto federated cloud infrastructure," *J Systems and Software*, vol. 120, no.1, pp. 89-104, Oct. 2016.

[9] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S. Tucker, "Fog Computing May Help to Save Energy in Cloud Computing," *IEEE J Selected Areas in Communications*, vol. 34, no. 5, pp. 1728-1739, March. 2016, doi: 10.1109/JSAC.2016.2545559.

[10] 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 of Things J*, vol. 3, no. 6, pp. 1171-1181, May. 2016, doi: 10.1109/JIOT.2016.2565516.

[11] P. Borylo, A. Lason, J. Rzasa, A. Szymanski, and A. Jajszczyk, "Energy-aware fog and cloud interplay supported by wide area software defined networking," *Proc. IEEE Communications (ICC '16)*, pp. 1-7, July. 2016, doi: 10.1109/ICC.2016.7511451.

[12] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, and A. Y. Zomaya, "A dynamic tradeoff data processing framework for delay-sensitive applications in Cloud of Things systems," *J Parallel and Distributed Computing*, vol. 112, pp. 53-66, Feb. 2018.

[13] B.-B. Li, L. Wang, and B. Liu, "An effective PSO-based hybrid algorithm for multiobjective permutation flow shop scheduling," *IEEE trans on systems, man, and cybernetics part A: systems and humans*, vol. 38, no. 4, pp. 818-831, Jun. 2008, doi: 10.1109/TSMCA.2008.923086.

[14] W. Li *et al.*, "System modelling and performance evaluation of a three-tier Cloud of Things," *Future Generation Computer Systems*, vol. 70, no.1, pp. 104-125, May. 2017.

[15] W. Li, F. C. Delicato, and A. Y. Zomaya, "Adaptive energy-efficient scheduling for hierarchical wireless sensor networks," *ACM trans. Sen. Netw.*, vol. 9, no. 3, pp. 1-34, May. 2013.

[16] O. Sinnen, *Task scheduling for parallel systems*. John Wiley & Sons, 2007.

[17] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129-140, Feb. 2013.

[18] P. Sanders and C. Schulz, "Engineering multilevel graph partitioning algorithms," *European Symposium on Algorithms*, Berlin: Springer-Verlag, pp. 469-480, 2011.

[19] S. Baluja, "Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning," Technical Report, Carnegie-Mellon Univ Pittsburgh Pa Dept of Computer Science, PA, USA, 1994.

[20] T. Tobita and H. Kasahara, "A standard task graph set for fair evaluation of multiprocessor scheduling algorithms," *J Scheduling*, vol. 5, no. 5, pp. 379-394, Sept. 2002.

[21] D.C. Montgomery, *Design and Analysis of Experiments*, New York:

Wiley, 1984.

[22] I. Dietrich and F. Dressler, "On the lifetime of wireless sensor networks," *ACM trans. Sen. Netw.,* vol. 5, no. 1, pp. 1-39, Feb. 2009.

[23] C. M. D. Farias *et al.*, "A Systematic Review of Shared Sensor Networks," *ACM Comput. Surv.,* vol. 48, no. 4, pp. 1-50, May. 2016.

[24] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: task graphs for free," *Proc. IEEE 6th international workshop on Hardware/software codesign Computer Society*, pp. 97-101, Aug. 1998, doi: 10.1109/HSC.1998.666245.

Chu-ge Wu received the B.Sc. degree in automation from Tsinghua University, Beijing, China, in 2015, where she is currently pursuing the Ph.D. degree in control theory and control engineering. She has published one refereed paper. Her current research interest includes scheduling and optimization algorithms for heterogeneous computing systems, intelligent optimization.

Wei Li received his PhD degree from School of Information Technologies at The University of Sydney. He is currently a research associate in Centre for Distributed and High Performance Computing, The University of Sydney. His research interests include Internet of Things, edge computing, sustainable computing, task scheduling, energy efficiency and optimization. He is the recipient of four IEEE or ACM conference best paper awards. He received the IEEE TCSC Award for Excellence in Scalable Computing for Early Career Researchers (2018) and the IEEE Outstanding Leadership Award (2018). He is a senior member of the IEEE Computer Society and the IEEE, and a member of the ACM.

Ling Wang received the B.Sc. in automation and Ph.D. degrees in control theory and control engineering from Tsinghua University, Beijing, China, in 1995 and 1999, respectively. Since 1999, he has been with the Department of Automation, Tsinghua University, where he became a full professor in 2008. His current research interests include intelligent optimization and production scheduling. He has authored five academic books and more than 260 refereed papers.

Professor Wang is a recipient of the National Natural Science Fund for Distinguished Young Scholars of China, the National Natural Science Award (Second Place) in 2014, the Science and Technology Award of Beijing City in 2008, the Natural Science Award (First Place in 2003, and Second Place in 2007) nominated by the Ministry of Education of China.

Albert Y. Zomaya is the Chair Professor of High Performance Computing and Networking in School of Information Technologies, The University of Sydney, and he also serves as the Director of Centre for Distributed and High Performance Computing. Professor Zomaya published more than 600 scientific papers and articles and is author, co-author or editor of more than 20 books. He is the Founding Editor in Chief of the IEEE Transactions on Sustainable Computing and serves as an associate editor for more than 20 leading journals. Professor Zomaya served as an Editor in Chief for the IEEE Transactions on Computers (2011-2014).

Professor Zomaya is the recipient of is the recipient of the IEEE Technical Committee on Parallel Processing Outstanding Service Award (2011), the IEEE Technical Committee on Scalable Computing Medal for Excellence in Scalable Computing (2011), the IEEE Computer Society Technical Achievement Award (2014), and the ACM SIGSIM MSWiM Reginald A. Fessenden Award (2017). He is a Chartered Engineer, a Fellow of AAAS, IEEE, and IET. Professor Zomaya's research interests are in the areas of parallel and distributed computing and complex systems.