A Deadline-Aware Estimation of Distribution Algorithm for Resource Scheduling in Fog Computing Systems

Chu-ge Wu
Department of Automation
Tsinghua University
Beijing, China
wucg15@mails.tsinghua.edu.cn

Ling Wang
Department of Automation
Tsinghua University
Beijing, China
wangling@mail.tsinghua.edu.cn

***Abstract*—The Internet of Things (IoT) develops rapidly and has produced a large amount of data these years. A range of responsive IoT applications arise and are needed to be processed in a timely manner. Compared with traditional cloud computing system, fog computing is one of the promising solutions of processing the huge amount of local data and decreasing the end-to-end latency. Hard and soft deadlines are assigned to the tasks of applications to model the users' needs. In this work, the resource allocation and task scheduling problem under fog system is considered to minimize total tardiness of the tasks and meet the hard deadlines. A deadline-aware estimation of distributed algorithm (dEDA) with a repair procedure and local search is adopted to determine the task processing order and computing node allocation. The comparative results show that the solution produced by our proposed algorithm performs better than the algorithm without repair procedure or knowledge driven local search. In addition, the performance of our algorithm exceeds significantly the heuristic method on both total tardiness and successful rate metrics. Compared with the existing fog computing resource management algorithm, our algorithm performs much better under most situations.**

Keywords—Edge computing, Internet of Things, evolutionary computation, total tardiness, scheduling

I. INTRODUCTION

Internet of Things (IoT) gains the remarkable and rapid development these years and has been widely adopted into industry and daily life [1]. A wide range of applications are coming into being, including human-facing and machine-to-machine ones. IoT systems need to meet the desired end-to-end latency requirements and provide quality of service (QoS) and user experience guarantees with finite resource to support the real-time applications. Some applications are called responsive IoT applications as their requests are needed to be timely responded [2]. The typical IoT responsive applications includes but not limit to gaming, virtual reality [3], motion control in cars and mobile video streaming [4]. It can be seen that to optimize the latency has come into a major problem for IoT responsive applications.

Fog computing [5] is one of the solutions for the completion of responsive IoT applications since it aims at moving computation capability from remote cloud center(s) to the edge of the network. The use of fog computing enables the data management close to the sources and reduce the volume of data transmission over the Internet. In that case, the latency of applications as well as the battery power consumed to send and receive data are decreased respectively. Meanwhile, the user experience is improved. It is said that responsive IoT applications are considered to be one of the primary reasons for the adoption of fog computing [2]. However, to fully exploit the computing system structure and solve the resource scheduling problem is not a trivial task as the fog computing infrastructure is complex. In addition, different requirements developed by users brings difficulties to model the problem. It's known that the task scheduling on multiprocessors problem is NP-hard when two processors are used [6]. This problem focuses on the latency optimization for responsive IoT applications under fog computing environment. This problem which considers more factors can also be proved to be NP-hard, so there is no known polynomial algorithm for this problem until now.

A large range of evolutionary optimization algorithms have been successfully adopted to solve NP-hard optimization problems in the past two decades. The evolutionary optimization algorithms are good at both exploration and exploitation and then produce acceptable solution in limited time. Some kind of estimation of distribution algorithms (EDA) was adaptive to solve complex scheduling problems [7, 8] as it has the capacity to explore the search space and learn from the elite solutions. EDA is a population-based algorithm where the population is evaluated by sampling the probability model. And the probability model is updated with the elite individuals. In our previous work, our proposed EDA performed well in DAG scheduling problem under fog computing infrastructure [9] and compared with it, this work considers different objectives. Inspired by the successful application of EDA for this problem, the dEDA will be adopted to solve this problem and some knowledge driven operators according to the new objectives are embedded to the proposed algorithm.

The remainder of the paper is organized as follows: Section 2 reviews the related work from two aspects: fog computing system and total tardiness scheduling. Section 3 provides the problem statement including system model, application model and objectives in mathematic formulas. Then, the proposed algorithm is detailed in Section 4. In Section 5, the numerical

This research is supported by the National Natural Science Fund for Distinguished Young Scholars of China [No. 61525304] and the National Natural Science Foundation of China [No. 61873328].

studies are provided to demonstrate the effectiveness of the proposed algorithm. Finally, the paper draws some conclusions and outlines ideas for future work in Section 6.

II. Related Work

As a large amount of IoT applications raise and fog computing develops, many new research challenges have been raised on how to measure, model and optimize the end-to-end latency of the applications to meet the users' needs. [10] offers a detailed introduction to the fog computing and related algorithms. Resource management, cooperative offloading and load balancing are all needed to considered under fog computing environment. And QoS is considered to be one of the important objectives for the algorithms enabling real-time applications. In [2], the authors examine the properties of task latencies in fog computing by considering both traditional and wireless fog computing execution points. Similarly, the authors consider the fog computing service assignment problem in bus networks [11]. A model minimizing the communication costs and maximizing utility with tolerance time constraints is given to formulate the problem.

Except considering time metric with constrained optimization, to minimize tardiness is an approach to optimize latency based on the study on real-time transactions scheduling [12]. And in the scheduling area, the total tardiness is a well-known and efficient metric to measure whether a batch of tasks is processed on time and how much the tasks are delayed. [13] gives a survey on the total tardiness problems. It mainly introduces the single machine problem as well as the parallel machine and flowshop problem. And [14] surveys the weighted and unweighted tardiness problems. Many studies are proposed to solve the scheduling problems considering total tardiness metric. Some evolutionary algorithms, such as genetic algorithm (GA) [15], competitive memetic algorithm (CMA) [16] are adopted and proved to be efficient on scheduling problem with tardiness metric.

Based on these existing works, total tardiness is used to model the latency of the whole application in this work. In addition, some problem specific operators are designed for our problem.

III. Problem Statement

In this work, we assume a generalized three-tier IoT system [17] which is shown in Fig. 1. The system is composed of things, fog and cloud tiers.

The devices located in the things tier are responsible for collecting raw data and processing it accordingly before sending it for further processing. The devices in this tier are grouped into different clusters by their locations. And in the same cluster, devices are assumed to be fully connected.

Fog computing tier is located at the middle of the overall architecture to process the computation tasks near the data source. In this work, different devices such as laptop, Raspberry Pi, and local server are defined as fog nodes. Each fog node is connected with a cluster of things directly. On the other hand, the fog nodes are fully connected with each other. The fog node can share the computation tasks received from the IoT devices with all other nodes within this tier.

The third tier in the system is the cloud tier. The processing units 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 cloud nodes are fully connected. Compared with the fog nodes, the resources of the cloud nodes are more reliable and substantial.

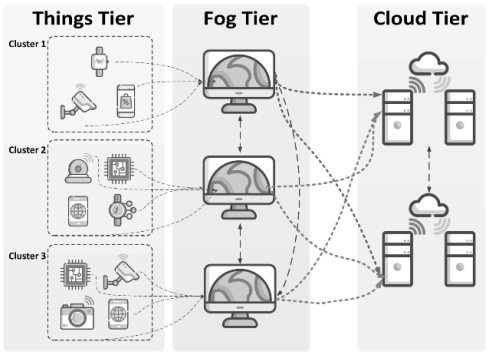

Fig. 1. An overview of the three-tier IoT system architecture

For the IoT applications, it is represented as a DAG (Directed Acyclic Graph) because most applications can be modelled as workflow. A given DAG can be defined as a five-tuple <***V***, ***E***, ***w***, ***D***, ***ddl***>, where ***V*** denotes the non-communication tasks, 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 precedence constraints between the tasks. ***w*** represents the workload of ***V*** and ***D*** represents the communication overhead of ***E*** . For a given task pair *i* and *j*, $D(i,j)$ denotes the data size of edge E_{ij}. In addition, it is known that the communication to computation ratio (*CCR*) indicates the impact of communication latency on performance and is calculated as average communication time divided by average computation time. ***ddl*** indicates the sub-deadline of each task in the application which is used to describe its expected completion time. As in the IoT system, some tasks must be finished on time, such as for the motion control in cars, otherwise it will lead to serious consequences. For these tasks, they are hard deadline constrained. Thus, the deadlines are divided into soft or hard deadlines and if a hard deadline is violated, then the application is considered as invalid.

For any task in the DAG, it can only start being processed after all its predecessors are finished as well as all the input data are received. And the tasks are assumed to be non-preemptive. The communication time of any two connected tasks (*i*, *j*) is dependent on the bandwidth $B(m_i, m_j)$ associated with the assigned computing nodes $(m_i, m_j), (m_i \neq m_j)$. Thus, the earliest start time of task *j* (EST_j) could be indicated as follows:

$$EST_j = \begin{cases} 0, & j = 0 \\ \max\limits_i \left\{ EST_i + \delta(m_i, m_j) \cdot \dfrac{D(i,j)}{B(m_i, m_j)} \right\}, & \forall\, i \to j \end{cases} \quad (1)$$

where task 0 denotes the root task of the DAG which is a virtual task. Its processing time is set 0 and it's the parent node of the entry nodes. $\delta(m_i, m_j)$ is an indicative function where $\delta(m_i, m_j) = 1$ denotes $m_i \neq m_j$ and $\delta(m_i, m_j) = 0$ denotes $m_i = m_j$. Because the IoT applications begin sensing data by the IoT devices, the entry tasks in DAG are processed in the things tier. And it's assumed that the entry tasks of a certain application are allocated in the same cluster.

Based on the system and application models, we use the total tardiness (*TT*) as our optimization objective to decrease task latency. *TT* of a certain application could be calculated as follows:

$$TT = \sum_{i=1}^{n} \max(C_i - ddl_i, 0) \quad (2)$$

where C_i indicates the completion time of task *i*. And the designed objective is given as follows:

$$\min \ TT$$

Subject to:

$$C_i \leq ddl_i, \forall i \in H \quad (3)$$

where *TT* is calculated as (2) and *H* denotes the task set of tasks with hard deadline.

In addition, as hard deadline constrained tasks need to be processed on time, the successful rate (*sr*) is introduced in this work. If (3) is satisfied, *sr* = 1, otherwise *sr* = 0. Compared with total tardiness, hard deadline satisfaction is more important. Thus, solution x_1 is considered to dominate solution x_2 (denoted as $x_1 \succ x_2$) if and only if:

$$sr(x_1) > sr(x_2) \quad (4)$$

$$TT(x_1) < TT(x_2), if \ sr(x_1) = sr(x_2) \quad (5)$$

IV. Details of Proposed Algorithm

The proposed algorithm that aims to minimize total tardiness of the DAG under the hard deadline constraints is introduced in detail in this section. Firstly, an overview of the proposed algorithm is given. Then, encoding and decoding procedure is introduced. After that the dEDA including initialization and repair procedure is presented. Finally, a local search aiming at decreasing total tardiness and increase successful rate is introduced.

A. Flowchart

The whole flowchart of our proposed algorithm is given as Fig 2. It can be seen from the flowchart that a basic EDA embedded with problem specific initialization, repairing decoding and local search procedure is given to solve the problem.

B. Encoding and Decoding Method

The individuals are encoded as a processing sequence, which is a permutation from 1 to n. The permutation is produced by our proposed dEDA.

When the individuals are decoded, the tasks are considered and assigned according to the processing permutation produced by dEDA. The computation nodes in the three tiers are treated as unrelated processors, where the processor assignment is determined by EFTF rule (earliest finish time first). According to EFTF, a task is assigned to the processor to achieve the minimum tardiness if the deadline cannot be met on any processors. In addition, if its deadline can be met on more than one processor, EFTF rule ensures that the task is assigned to the processor with minimum completion time.

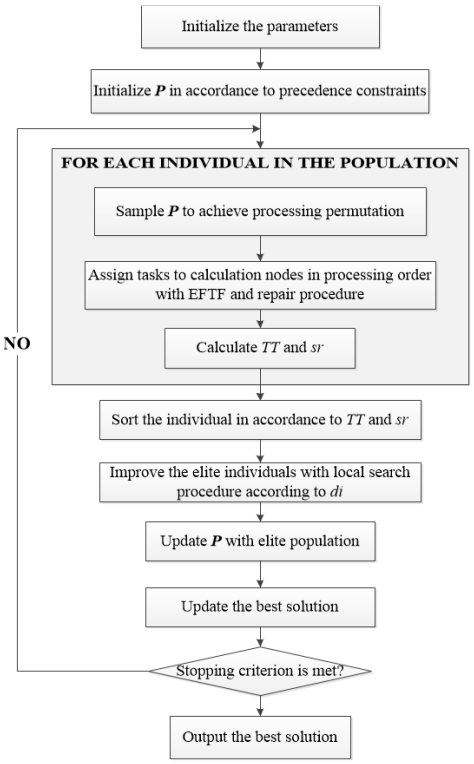

Fig. 2. A flowchart of the proposed algorithm

Algorithm 1: Procedure of decoding method

π is the task processing permutation;
For *i* = 0 to *n*
  m_{π_i}= **EFTF**(π_i);
  If $C_{\pi_i} > ddl_{\pi_i} \wedge \pi_i \in H$
    If $i > 1 \ \wedge \ \pi_{i-1} \notin H \wedge \pi_{i-1}$is not precedent of π_i
      swap(π_{i-1}, π_i);
      i--;
      Delete π_{i-1}, π_i from its processor respectively;
      continue;
    End If
  End If
End For

In addition, the tasks are expected to avoid missing hard deadlines. A repair procedure is embedded to improve the solution with tasks violating hard deadlines during decoding procedure. When a task with hard deadline is assigned to the

processors and its completion time is bigger than its deadline, it is considered to be move forward. The forward tasks in processing permutation are taken into account. If the forward task is not hard deadline constrained or precedent of the certain task, these two tasks are swapped and the decoding procedure is repeated from the certain task. The swap operator is repeated until the deadline is met or the condition is not satisfied. The detailed pseudo code of the proposed decoding method is presented in Algorithm 1.

C. dEDA Scheme

First of all, the standard EDA procedure is summarized as follows. Firstly, a probability model P is built according to the specific problem information. The individuals are sampled with P and the generated individuals are then evaluated. After that, P will be updated by learning from the elite individuals. And the sampling procedure is repeated with the updated P value until the stopping criterion is met. In this work, dEDA is employed to produce the task processing sequence. The details of dEDA for this scheduling problem are given as follows.

Probability model and initialization

Considering the precedence constraints between the DAG tasks, a task permutation probability model P considering the task relative position relationship is built and the task processing permutation is produced by P. The probability model is formulated as (6), where $p_{i,j}(g)$ represents the probability that task i is ahead of task j in the permutation at the g-th generation of the evolution. Clearly, $p_{i,j}(g) + p_{j,i}(g) = 1$ under any situation.

$$P(g) = \{p_{i,j}(g)\}_{n \times n} \tag{6}$$

When the model is initialized, if there is a precedence constraint between a pair of tasks, the probability value ($p_{i,j}$) is initialized as (7). In this way, the precedence constraint is settled down during the evaluation.

$$p_{i,j}(0) = \begin{cases} 1, i \to j \\ 0, j \to i \\ 0.5, otherwise \end{cases} \tag{7}$$

Updating method

EDA is evaluated by the elite individuals, the $N \times \eta\%$ best individuals, where $\eta\%$ is the elite population proportion. The probability updating scheme is presented as (8) and (9). For each individual in the elite population, the relationship between task pair (i, j) is recorded as $I_{i,j}^{k}(g)$ as (9). And the sum of indicators is divided by the amount of elite individuals to achieve the frequency of a certain task pair. The frequency is used to update the former probability as the population based incremental learning method (PBIL) [18]:

$$p_{i,j}(g+1) = (1-\alpha) \times p_{i,j}(g) + \alpha \times \textstyle\sum_k I_{i,j}^{k}(g) / (N \times \eta\%) \tag{8}$$

$$I_{i,j}^{k}(g) = \begin{cases} 1, \text{task } i \text{ is before task } j \text{ in the } k\text{th individual} \\ 0, \text{otherwise} \end{cases} \tag{9}$$

where $\alpha \in (0,1)$ denotes the learning rate and $I_{i,j}^{k}(g)$ is the indicator function corresponding to the k-th individual of the population.

Sampling method

In EDA, the new individuals are produced by sampling the updated probability model. In our work, a sampling method is designed to sample the relative position probability model and produce new individuals. For each position which has not been determined in the task 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. The detailed pseudo code can be referred in [9].

D. Local Search Method

To enhance the exploration ability of dEDA, the problem specific local search method is necessary to achieve better solutions. Thus, a local search method is designed in our proposed algorithm to decrease task tardiness and make more hard deadline constrained tasks to meet its deadline. The detailed pseudo code of the local search method is presented at first and then it is explained.

Algorithm 2: Local search method

```
s = 0;
While s < ls
  Calculate di for tasks within processing permutation π;
  For each task t according to descending order of di
    If t has been moved
      continue;
    Else
      Record that t has been moved;
      Find out t is the i-th task in permutation π;
      If i > 1 ∧ π_{i-1} is not precedent of π_i
        swap( π_{i-1}, π_i) to achieve π';
        Evaluate π' and calculate TT' and sr'.
        s++;
        If sr' > sr ∨ TT' < TT
          π := π';
          s = s + 5;
          break;
        Else
          i--;
        End If
      End If
    End If
  End For
End While
```

Firstly, a variable called delay impact (di) is defined for each task in each individual to describe how much the task delays its children tasks. The calculation of di value of task j is defined as follows:

$$di(j) = \sum_k \max\left((C_j - ddl_k) \times \delta_k, 0\right) \tag{10}$$

$$\delta_k = \begin{cases} 2, k \in H \\ 1, k \notin H \end{cases} \tag{11}$$

where k denotes the children tasks of task j. In (10), $C_j - ddl_k$ indicates the low bound of the tardiness of task k as task k must be processed after task j is completed. In addition, a coefficient (δ_k) larger than 1 is used to reinforce the delay impact on hard deadline constrained tasks.

The local search method is adopted according to the descending order of delay impact of the tasks for each elite individual. Before the stopping criterion is met, the unmoved task with biggest delay impact is moved forward to decrease the impact on its children tasks. The solution is evaluated after the swap operation and the old solution is replaced by the current solution if the current one is better. In this way, the solution is improved within the limited computing power.

V. Simulation

A. Comparison Algorithm

To evaluate the performance of our proposed algorithm, a modified heuristic method solving total tardiness problem are selected and modified as benchmark algorithm. For fair comparison, the algorithms are all coded in C++ language and run under the same running environment.

The heuristic algorithm is modified from the standard heuristic method [12] for total tardiness problem: earliest deadline first (EDF), shortest processing time first (SPF) and shortest t-level task first (STF). The heuristic methods are efficient for total tardiness problems and easy to be modified for a complicated system problem. The t-level in STF is calculated as the longest path from the entry task to the certain task [2]. EDF sorts the tasks in the ascending sequence of the tasks' due dates. Similarly, SPF and STF sort the tasks in ascending sequence of processing time and t-level values of the tasks respectively. These three heuristic algorithms are used altogether and the best solution is adopted as the heuristic solution. And the pseudo code of the heuristic algorithm is given as follows.

Algorithm 3: Modified Heuristic Algorithm

A := available task set, initialized with the root task;
While ***A*** is not empty
  If there are hard ddl constrained tasks in ***A***
    t := **EDF**/**SPF**/**STF** on hard deadline constrained tasks;
  Else
    t := **EDF**/**SPF**/**STF** on soft deadline constrained tasks;
  End If
End While
Find out available tasks in children tasks of t and push into ***A***;
Schedule t onto the system with **EFTF**.

The other comparison algorithm is our proposed EDA [9] with partition operator. The EDA with partition operator (pEDA) is designed for this three-tier IoT system to optimize both makespan and battery energy consumption as well as lifetime. Although the objectives are different from this work, pEDA optimizing the completion time of the application is also applicable to this work. Thus, pEDA is used as a comparison algorithm to demonstrate the efficiency of dEDA.

B. Simulation Environment and Experiment Settings

The testing environment setting and its variables are set as follows. The things tier is heterogeneous where different numbers ({3, 3, 3, 4, 4, 4}) IoT devices are set in corresponding clusters. And there are 6 fog nodes placed to link with their responsible cluster respectively. And 2 public cloud services are considered in this system.

The processing speed of the IoT devices, fog nodes and cloud nodes are set {1, 5, 10}. The bandwidth within each tier is set {1, 5, 10} and the bandwidths between the things and the fog, the fog and the cloud tiers are set as {1, 8} respectively. A fixed communication delay (0.1s) is added for transmitting a generalized unit of a communication task between the fog and the cloud tiers. And for each instance, the application is produced by the IoT devices in one cluster and it is processed on the certain one IoT device cluster.

A well-known pseudorandom task graph generator TGFF [19] 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. The task deadline is set as follows:

$$ddl_i = \left((Hb - Lb) \times rand(0,1) + Lb\right) \times tlvl_i \qquad (12)$$

where *Hb* and *Lb* are the high and low bound of the deadline assignment. In this work, *Hb* = 3 and *Lb* = 1.5. $tlvl_i$ is the t-level value of task i which is calculated as the largest sum of processing time of the task path from the entry task to task i. In addition, to discuss the performance of the algorithms under different situation, a parameter called *Loose* is set to control the deadlines. In this work, *Loose* is set as {1.0, 1.5, 2.0} and $ddl'_i = ddl_i/Loose$. For each *CCR*, 500 independent DAGs and corresponding deadline values are produced and for each DAG instance, 3 *Loose* values are used. Thus, 500×4×3 = 6000 instances are used to validate the efficiency of the proposed algorithm. In addition, the proportion of hard deadlines is set as 10%, i.e., 10% tasks in the application must be completed before its deadline, otherwise the application is considered as unsuccessful.

For the parameters of the proposed algorithm, the stopping criterion is set as 30 generations. Two key parameters of dEDA, the population size (N) and the learning rate (α), are set the same with [9] where N=10 and α=0.05. The proportion of elite population ($\eta\%$) and the steps of local search (ls) are set accordingly. In this work, $\eta\%$ is set 20% and $ls = n/3$. And the parameter setting of pEDA is set as [9].

C. Effect of Specific Procedures

To demonstrate the effectiveness of the repair procedure and local search method, the proposed algorithm is compared with the algorithm without the certain procedure. Thus, dEDA without repair decoding procedure (denoted as A1) and dEDA with random local search (denoted as A2) are employed and the comparison results are listed as follows. The relative percentage deviation (RPD) on both total tardiness (*TT*) and successful rate

(*sr*) of our proposed algorithm compared with A1 and A2 are listed in table I and II respectively.

$$RPD = \frac{EDA - alg}{alg} \times 100\% \quad (13)$$

In the tables, the first row of tables denotes three *Loose* values and the first column of the tables denotes four *CCR* values. And the average value of 500 instances under each problem scale is calculated and summarized.

TABLE I. RPD COMPARED WITH PROPOSED ALGORITHMS OF *TT*

Loose	**1.0**		**1.5**		**2.0**	
CCR	***A1***	***A2***	***A1***	***A2***	***A1***	***A2***
0.1	-6.26	-2.93	-11.50	-2.10	-12.08	-5.88
0.5	-0.89	-1.25	-4.54	-3.24	-6.00	-2.95
1.0	-0.71	-0.24	-1.93	-1.68	-2.24	-3.32
5.0	0.07	-2.94	0.10	-0.16	-3.25	-3.14

TABLE II. RPD COMPARED WITH PROPOSED ALGORITHMS OF *SR*

Loose	**1.0**		**1.5**		**2.0**	
CCR	***A1***	***A2***	***A1***	***A2***	***A1***	***A2***
0.1	0.21	0.21	0.00	0.00	1.15	0.23
0.5	0.00	0.00	0.00	0.00	0.22	0.45
1.0	0.23	0.00	1.01	0.76	0.92	0.31
5.0	20.93	4.00	20.18	2.24	19.67	5.80

First of all, as the deadline assignment rule (12) does not consider the transfer data between the tasks, as *CCR* increases, the transfer data volume increases and the t-level value becomes much less than the actual task completion time and the deadline turns tight. Therefore, when *CCR* is small, the deadlines are slack and the successful rate is high. When *CCR* is large, the deadlines becomes tight and the hard deadline constrained tasks might be more difficult to meet the deadlines and the successful rate decreases.

And then from Table I and II, it can be seen that our proposed algorithm exceeds A1 and A2 under most scenarios. For A1, the repair procedure in decoding is eliminated compared with our algorithm. From Table I and II, it can be seen that as *CCR* increases, the successful rate of our algorithm is much larger than A1. When *CCR* = 5.0, dEDA with repair procedure improves the successful rate of dEDA without repair procedure by around 20%. Thus, it could be concluded that the repair procedure is efficient to improve *sr*. And for A2, the delay impact leading local search is replaced by a random local search operator. The tasks are randomly chosen and swapped with other tasks and the former solution is replaced by the better one. From Table I and II, it can be seen that our proposed algorithm performs better than A2 on both objectives. It could be concluded that our knowledge driven local search is more efficient than the random search when the computation resource is limited.

D. Compared with Heurisitc Method

The comparison results of modified heuristic method and our proposed algorithm are listed as Table III and V. The average value of 500 instances under each problem scale is calculated and summarized respectively. In Table III, the *p*-values of two sample paired t-tests under 95% confidence level are listed. The alternative hypothesis is the solution of heuristic method is less than the our solution. And if the hypothesis is accepted, the "Sig" appears to be "Y", otherwise appears "N". When it comes to Table V, t-test is not suitable for the comparison of binary *sr* values.

TABLE III. COPARISON RESULTS OF *TT* BETWEEN HEURISTIC AND DEDA

Loose	**1.0**				**1.5**				**2.0**			
CCR	**Heuristic**	**dEDA**	***p*-value**	**Sig**	**Heuristic**	**dEDA**	***p*-value**	**Sig**	**Heuristic**	**dEDA**	***p*-value**	**Sig**
0.1	3.11	1.78	0.002	Y	6.41	4.32	0.000	Y	15.21	10.27	0.000	Y
0.5	1.58	1.28	0.000	Y	3.82	2.98	0.000	Y	12.36	8.61	0.000	Y
1.0	17.71	11.86	0.000	Y	39.42	25.36	0.000	Y	101.57	66.71	0.000	Y
5.0	1592.32	705.54	0.000	Y	2508.79	1178.18	0.000	Y	4022.57	2036.77	0.000	Y

TABLE IV. COPARISON RESULTS OF *TT* BETWEEN PEDA AND DEDA

Loose	**1.0**				**1.5**				**2.0**			
CCR	**pEDA**	**dEDA**	***p*-value**	**Sig**	**pEDA**	**dEDA**	***p*-value**	**Sig**	**pEDA**	**dEDA**	***p*-value**	**Sig**
0.1	3.63	1.78	0.002	Y	17.49	4.32	0.000	Y	17.13	10.27	0.000	Y
0.5	2.63	1.28	0.000	Y	15.69	2.98	0.000	Y	15.91	8.61	0.000	Y
1.0	14.68	11.86	0.000	Y	69.19	25.36	0.000	Y	69.14	66.71	0.000	Y
5.0	515.42	705.54	1.000	N	1450.12	1178.18	0.000	Y	1447.30	2036.77	1.000	N

TABLE V. COPARISON RESULTS OF *SR* BETWEEN HEURISTIC, PEDA AND DEDA

Loose	**1.0**			**1.5**			**2.0**		
CCR	**Heuristic**	**pEDA**	**dEDA**	**Heuristic**	**pEDA**	**dEDA**	**Heuristic**	**pEDA**	**dEDA**
0.1	**0.944**	0.916	**0.944**	0.920	0.832	**0.926**	0.854	0.828	**0.876**
0.5	0.982	0.978	**0.984**	0.952	0.862	**0.964**	0.872	0.862	**0.896**
1.0	0.856	0.862	**0.876**	0.764	0.639	**0.798**	0.608	0.641	**0.656**
5.0	0.248	0.285	**0.416**	0.150	0.106	**0.274**	0.078	0.100	**0.146**

Note: The bold values mean the best results

From Table III and V, it can be seen that dEDA performs better than heuristic method. For successful rate, it can be seen that as *CCR* and *loose* value increases, the deadlines turn tight, dEDA improves a part of applications to be valid. In addition, as

the deadline assignment rule doesn't consider communication delay, some hard deadline constrained tasks of the benchmark instances are unable to be processed on time under any scheduling strategy. Thus, the *sr* value of dEDA is very small

when *CCR* = 5.0. When it comes to total tardiness, it could be seen that *TT* values of dEDA are less than the heuristic solutions under 95% confidence level. Considering both metrics, the comparison results show that our proposed algorithm is an efficient algorithm for this problem compared to the modified heuristic method.

E. Compared with pEDA

The comparison results of pEDA and our proposed algorithm are listed as Table IV and V as before. From Table IV and V, it can be seen that our proposed algorithm performs better than pEDA under most situations. As the successful rate is the primary objective in this work, decoding method in dEDA is tailored to decrease the tardiness of hard-deadline constrained task, the average value of successful rate of dEDA is less than pEDA under all scales of instance. In addition, pEDA performs better than our proposed dEDA under some instances when *CCR* = 5.0. The possible reason is our proposed dEDA considers the successful rate first and the soft-deadline constrained tasks are delayed to ensure the timely completion of hard-deadline constrained tasks. On the other hand, the partition operator in pEDA is good at processing applications with high *CCR*. Thus, considering both metrics, the comparison results show that our proposed algorithm is efficient and it's also needed to balance the total tardiness and successful rate when *CCR* increases.

VI. CONCLUSION

In this work, a problem specific dEDA is proposed to address the latency optimization of IoT applications under the three tiers systems. At first, a model is given to formulate the problem. And then a dEDA scheduling algorithm compared with repairing decoding procedure and knowledge driven local search operator is designed. At last, the simulation experiments are conducted to demonstrate the efficiency of our designed operators and the whole algorithm in reducing total tardiness and increasing successful rate.

In the future, we plan to consider the reality and computing node capacity during the optimization. In addition, the prototype system is needed to achieve realistic benchmark instances and test our algorithm.

REFERENCES

[1] F. Mattern, C. Floerkemeier, K. Sachs, I. Petrov, and P. Guerrero, "From the Internet of Computers to the Internet of Things," in Lecture Notes in Computer Science, vol. 6462, Berlin, Heidelberg: Springer, 2010, pp. 242-259.

[2] M. Gorlatova, and M. Chiang, "Characterizing Task Completion Latencies in Fog Computing," ArXiv.org, ArXiv.org, Nov 6, 2018. (in press)

[3] D. Chu, "Toward immersive mobile virtual reality," Proceeding of the 3rd Workshop on Hot Topics in Wireless, 2016.

[4] T. X. Tran, P. Pandey, A. Hajisami, "Collaborative multi-bitrate video caching and processing in Mobile-Edge Computing networks," Annual Conference on Wireless On-demand Network Systems and Services (WONS), IEEE Congress on Jackson, pp.165-172, Feb 2017.

[5] 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.

[6] Y. K. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," ACM Computing Surveys, vol. 31, pp. 406-471, Dec 1999.

[7] S. Y. Wang and L. Wang, "An estimation of distribution algo-rithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem," IEEE Trans. systems, man, and cybernetics: systems, vol. 46, pp. 139-149, April 2016

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

[9] C. G. Wu, L. Li, L. Wang, and A. Zomaya, "Hybrid Evolutionary Scheduling for Energy-efficient Fog-enhanced Internet of Things," IEEE Trans on Cloud Computing, 2018. (in press)

[10] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, & A. Niakanlahiji, et al. "All one needs to know about fog computing and related edge computing paradigms: a complete survey", Journal of Systems Architecture, 2019. (in press)

[11] D. Ye, M. Wu., S. Tang, and R. Yu, "Scalable fog computing with service offloading in bus networks," Cyber Security and Cloud Computing (CSCloud), IEEE Conference on Beijing, pp. 247-251, June 2016

[12] R. Abbott, and H. Garcia-Molina, "Scheduling real-time transactions," ACM Sigmod Record, vol. 17, pp. 71-81, 1988.

[13] C. Koulamas, "The total tardiness problem: review and extensions," Operations research, vol. 42, pp. 1025-1041, 1994.

[14] T. Sen, J. M. Sulek, P. Dileepan, "Static scheduling research to minimize weighted and unweighted tardiness: a state-of-the-art survey," International Journal of Production Economics, vol 83, pp. 1-12, 2003.

[15] Y. Myungryun, and M. Gen, "Scheduling algorithm for real-time tasks using multiobjective hybrid genetic algorithm in heterogeneous multiprocessors system," Computers & Operations Research, vol. 34, pp. 3084-3098, 2007.

[16] J. Deng, L. Wang, S. Y. Wang, and X. L. Zheng, "A competitive memetic algorithm for the distributed two-stage assembly flow-shop scheduling problem," International Journal of Production Research, vol. 54, pp. 3561-3577, 2016.

[17] W. Li et al, "System modelling and performance evaluation of a three-tier Cloud of Things," Future Generation Computer Systems, vol. 70, pp. 104-125, May 2017.

[18] 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.

[19] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: task graphs for free," Proc. IEEE 6th international workshop on Hard-ware/software codesign Computer Society, pp. 97-101, Aug. 1998.