Future Generation Computer Systems 117 (2021) 498–509

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

An evolutionary fuzzy scheduler for multi-objective resource allocation in fog computing

Chu-ge Wu [a], Wei Li [b], Ling Wang [a,*], Albert Y. Zomaya [b]

[a] Department of Automation, Tsinghua University, Beijing, 100084, PR China
[b] Centre for Distributed and High Performance Computing, School of Computer Science, The University of Sydney, NSW 2006, Australia

ARTICLE INFO

Article history:
Received 31 January 2020
Received in revised form 12 December 2020
Accepted 24 December 2020
Available online 30 December 2020

Keywords:
Edge computing
Internet of Things
Evolutionary computation
Estimation of distribution algorithm
Fuzzy scheduling
Agreement index
Robustness

ABSTRACT

With rapid development of the Internet of Things (IoT), a vast amount of raw data produced by IoT devices needs to be processed promptly. Compared to cloud computing, fog computing nodes are closer to data resource for decreasing the end-to-end transmission latency. Considering the limited resource of IoT devices, offloading computationally-intensive tasks to the servers with high computing capability is essential in the IoT–fog–cloud system to complete those tasks on time. In this work, we propose a fuzzy logical offloading strategy for IoT applications characterized by uncertain parameters to optimize both agreement index and robustness. A multi-objective Estimation of Distribution Algorithm (EDA) is designed to learn and optimize the fuzzy offloading strategy from a diversity of the applications. The algorithm partitions applications into independent clusters, so that each cluster can be allocated to the corresponding tier for further processing. Thus, system resources are saved by making scheduling decisions in a reduced search space. Simulation studies on benchmark problems and real-world cases are carried out to verify the efficiency of our proposed algorithm. Pareto sets produced by our algorithm outperformed classic heuristic solutions for 88.3% benchmark cases and dominated Pareto sets of two state-of-art multi-objective algorithms for 92.7% and 94.4% cases correspondingly.

1. Introduction

In recent years, the Internet of Things (IoT) paradigm has increasingly been applied to all spheres of our daily life [1]. The successful operation of many IoT applications is increasingly relied on timely and decisive response [2], such as gaming, virtual reality, motion control in cars, and mobile video streaming. To shorten the turnaround time, task offloading is a promising practice to exploit the nearby available computational resources to complete the tasks of those IoT applications [3]. Compared to cloud computing that generally handles on-demand heavy computational tasks, fog computing [4] can be seen as an intermediate tier to deliver just-in-time computing resources between cloud centres and IoT devices. The use of fog computing enables the processing close to data sources and reduces the transmission delay over the Internet, as well as guarantees the users' security and trust issues [5]. To achieve these goals with optimal efficiency, it is imperative to properly distribute tasks to devices in different tiers of the IoT–fog–cloud environment.

The IoT application tasks offloading onto the computing resources can be modelled as task scheduling problems on heterogeneous multi-processors and it is well-known that this kind of problem is generally an NP-hard problem [6]. For our problem, the processors are located at three tiers with different capacities while the dependent tasks are labelled with different due dates. This can be affirmatively proved to be NP-hard which has no known polynomial solution. Considering the successful application of evolutionary algorithms for large scale offline scheduling problems, an evolutionary algorithm is employed in this work. In addition, to properly distribute tasks to the processors located at different tiers of the system, the state of the processors is key to the task allocation. It is a noticeable resource consumption when all states of the processors are collected at the times. As such, to reduce the overhead of collecting the states of processors can greatly help to diminish the relevant system resource consumption. Besides, seeking for the task allocation on a reduced search space can also help to save system resources, so that a pre-partition method for the application without considering processor states is required.

Apart from the heterogeneous computing environment and response-time-sensitive requests from the applications, many uncertainties may exist [7] in the IoT–fog–cloud environment, which bring new challenges to the task scheduling problem. To simplify our analysis and highlight the essence of our argument, in this work, the structure of IoT applications is assumed to be known to the system. Also, the processing time of each task and the communication overhead between the tasks are not known exactly

* Corresponding author.
E-mail address: wangling@mail.tsinghua.edu.cn (L. Wang).

https://doi.org/10.1016/j.future.2020.12.019
0167-739X/

and have uncertainties. In this way, the fuzzy number is used to model the corresponding data since its effective representation of the vague states of information [8].

In addition, prior and posterior measures have been extensively studied on fuzzy scheduling issues to estimate the robustness of solutions [9]. The posterior measure can only perform when the scheduling solution is obtained, such as ε-robustness [10], where if a certain algorithm is ε-robust, then the relative deviation between the prediction and the real value is bounded by $1 \pm \varepsilon$. On the other hand, prior measures can be evaluated during the scheduling decision. As one of the prior measures, *ROB* represents the maximum possible difference between the solutions. Based on the credibility theory of fuzzy numbers, it can be proved that $ROB(\tilde{l})$ coincides with the entropy of $\tilde{l}$ [11]. The prior measure is chosen as an objective to improve the robustness of our solution since it can provide error estimation from the uncertainties in the decision-making progress in this work.

To optimize the agreement index of the applications as well as to guarantee the robustness of our scheduling solution, we model the task offloading and scheduling problem as a multi-objective offline optimization problem and find out a fitness trade-off between performance index and the robustness. The remainder of the paper is organized as follows: Section 2 reviews the related work. Section 3 provides the basic concepts on multi-objective optimization problem, fuzzy theory and the problem statement including system and application model, mathematic model and the robustness analysis. Then, the proposed algorithm is detailed described in Section 4. In Section 5, the numerical studies are provided to demonstrate the effectiveness of our proposed algorithm. Finally, the paper draws some conclusions and outlines ideas for future work in Section 6.

2. Related work

To optimize the IoT–fog–cloud system performance, resource allocation and computational task offloading are core aspects to consider. The resource allocation and computational task offloading problems turn to be more difficult and complicated since inherent heterogeneity, uncertainty and dynamic within the computing environment as well as the different objectives raised up by users and operators. As the most important performance index, different expressions of indexes are proposed to mathematically describe the time-sensitive request of IoT applications. An online strategy [12] is proposed to maximize the number of tasks meeting deadlines. Tardiness [13] is set as a penalty variable within the fitness value. Delay constraints are adopted in [14] to obtain feasible solutions. Apart from the performance index, the objectives, such as cost and energy consumption, are taken into consideration under different situations. Some state-of-art evolutionary algorithms [15–17] are adopted and perform well in multi-objective problems as it is capable of adjusting the solution accordingly.

Apart from optimization objectives, different kinds of scheduling models are proposed to emulate practical condition. Task queue model [18–21] is adopted to describe the task generating and caching situation in the gateway, fog and cloud nodes. In spite of the task queue model, independent task model [22,23] is used to model user requests. Quality of Experiment (QoE) aware application placement solution is presented in [22] where tasks are clustered by fuzziness method and labelled with linguistic classification. Inspired by this work, fuzzy logic is employed to realize fuzziness and de-fuzziness onto the application tasks in our work.

In addition, to model the precedence constraints between tasks, Direct Acyclic Graph (DAG) is used to model the application. Heterogeneous earliest finish time (HEFT) [27] produces scheduling schemes in heterogeneous processors environment for shortening application completion time. Tasks are sorted and opportunistically inserted to available processor idle time slot of towards the earliest finish time. HEFT is ranked as the best of twenty scheduling heuristic methods in terms of robustness and performance [28]. A mixed-integer linear programming is presented in [24] to model offloading the dependent tasks with deadline constraints in IoT–fog system. This work ignores IoT devices and takes the cloud computing tier as a processor with unlimited computing capability. To solve the multi-objective DAG task scheduling problem, multi-objective HEFT [29] is designed as a general framework to provide a set of solutions for the decision maker. New solutions are sorted according to crowding distance and best K solutions are chosen as Pareto set. Fuzzy dominance is introduced to improve the performance of MOHEFT in [25], which optimizes both makespan and cost index. Related works on fog resource allocation problem are summarized in Table 1. In this work, we consider a fuzzy DAG task scheduling problem where the agreement index is used to evaluate task latency. Meanwhile, robustness is another objective which is considered to guarantee the stability of the algorithm.

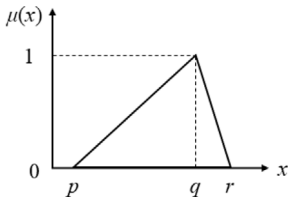


Fig. 1. Membership function of a triangular fuzzy number [30].

3. System modelling and problem statement

3.1. Basic concepts of fuzzy number

In the practices, IoT application data is hard to be precisely determined in advance. Thus, the task processing time, inter-task communication data and the deadline of tasks are modelled as fuzzy numbers in this work. Related fuzzy theory is introduced below.

Fuzzy set and membership function [30]: Fuzzy set $\tilde{A}$ in a universe of discourse X is characterized by a membership function $\mu_{\tilde{A}}(X)$. For each element x in X, a real number value function $\mu_{\tilde{A}}(x) \in (0, 1)$ denotes the membership grade of x in $\tilde{A}$, which indicates how likely an element x belongs to $\tilde{A}$.

$$\mu_{\tilde{l}}(x) = \begin{cases} 0, & x < p \text{ or } x \geq r \\ \dfrac{x-p}{q-p}, & p \leq x < q \\ \dfrac{x-r}{q-r}, & q \leq x < r \end{cases} \tag{1}$$

Fuzzy number: A fuzzy number $\tilde{l}$ is a fuzzy subset whose membership is convex and normal. According to the characteristic of membership functions, there are different types of fuzzy numbers, such as normal fuzzy number, triangular fuzzy number (TFN) and semi-trapezoid fuzzy number.

Triangular fuzzy number: TFN is chosen to model the task workload and data transfer overhead in this work. For TFN $\tilde{l}$, it is denoted as (p, q, r), where (p, q, r) represents correspond the smallest likely value, the most probable value, and the largest possible value of any fuzzy event. The membership function of a TFN is given below as Fig. 1.

Table 1
Literature classification for fog resource allocation problem.

Ref.	Performance metrics	Scheduling model	Approach (algorithm)
[18]	Response time, cost and number of application loss	Task offloading with queueing model	Lyapunov optimal method
[19]	Total execution time and resource cost		Multi-objective task scheduling algorithm with adaptive neighbourhood strategy
[20]	Weighted sum of latency and energy		Deep reinforcement learning method
[21]	Profit and response time constraint		SA-based migrating birds optimization algorithm
[22]	QoE	Independent task allocation model	Fuzzy logic based task clustering and linear optimization method
[23]	Energy consumption, profit with resource constraint		Convex optimization for power allocation and NSGA for scheduling
[24]	Deadline constrained makespan	DAG task offloading and scheduling model	MILP enhanced greedy task allocation method
[25]	Response time and cost		Fuzzy dominance based multi-objective heuristic method
[26]	Response time, fog energy consumption and system lifetime		Task offloading heuristic method and EDA for task scheduling
This work	Agreement index and robustness		EDA evolution for fuzzy logical offloading strategy

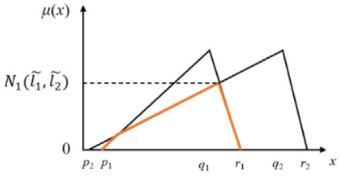


Fig. 2. Nearness calculation of TFNs [30].

The basic operations of TFN are shown as below. The sum of two TFNs $\widetilde{l_1} = (p_1, q_1, r_1)$ and $\widetilde{l_2} = (p_2, q_2, r_2)$ can be calculated as:

$$\widetilde{l_1} + \widetilde{l_2} = (p_1 + p_2, q_1 + q_2, r_1 + r_2) \tag{2}$$

The maximum number of two TFNs can be calculated as (**max(p_1, p_2), max(q_1, q_2), max(r_1, r_2)**). In addition, to compare $\widetilde{l_1}$ and $\widetilde{l_2}$, signed distance is used for de-fuzzification to achieve the total cost estimate in the fuzzy sense. The signed distance of a TFN can be calculated as (3). In the rest of the paper, we will use it to rank and compare the given TFNs.

$$dis\,(l) = \frac{p + 2q + r}{4} \tag{3}$$

Nearness degree: Nearness degree is a metric to measure the similarity of two fuzzy sets, which can be denoted as N (A, B). Considering the meaning of nearness degree, one of the nearness degree calculation methods is chosen to measure the closeness between the fuzzy numbers in this work which is formulated as (4).

$$N_1\,(B, A) = \vee_{x \in X}\,(\mu_B\,(x) \wedge \mu_A\,(x)) \tag{4}$$

where for $\wedge$ and $\vee$, we have $a \vee b = \max(a, b)$ and $a \wedge b = \min(a, b)$.

Based on (4), for TFNs $\widetilde{l_1}$ and $\widetilde{l_2}$, $N_1\left(\widetilde{l_1}, \widetilde{l_2}\right)$ is calculated as the minimum point of intersection between two membership function curves as Fig. 2 shows.

3.2. Basic concepts of multi-objective optimization problem

The multi-objective optimization problem (MOP) is adopted to model the problem in this work to balance the performance and robustness of the solution. MOP aims to find out the optimal solution for more than one objective while the objectives are contradictive. Despite the optimization of each single objective, the trade-off between objectives is required to address during the decision making. The problem can usually be defined as follows.

$$\min f\,(x) = \left(f_1\,(x), f_2\,(x), \ldots, f_q\,(x)\right) \tag{5}$$

where $f(x) \in R^q$ is the objective vector that contains q individual objectives. To evaluate solutions, non-dominated sorting method is adopted.

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

$$f_i\,(x_1) \leq f_i\,(x_2)\,, \forall i \in \{1, 2, \ldots, q\} \tag{6}$$

$$f_i\,(x_1) < f_i\,(x_2)\,, \exists i \in \{1, 2, \ldots, q\} \tag{7}$$

Optimal Pareto solution: If one solution is not dominated by any other solutions, it is called the optimal Pareto solution.

Non-dominated solution set/archive set (AS): For a specific algorithm, the archive set consists of all optimal Pareto solutions of it. To reflect the dominance relationship between the solutions in two archive sets, say, AS_1 and AS_2, **converge metric (*C* metric)** is widely used, which is calculated as (8):

$$C\,(AS_1, AS_2) = \frac{\left|\left\{x_2 \in AS_2 \,\middle|\, \begin{array}{c}\exists x_1 \in AS_1, x_2 \prec x \\ or\ x_2 = x_1\end{array}\right\}\right|}{|AS_2|} \tag{8}$$

where $C(AS_1, AS_2)$ reflects the percentage of the solutions in AS_2 that are dominated by or identical to the solutions in AS_1.

Another indicator, **Inversed Generation Distance (*IGD*)**, is also widely used to compare archive sets. *IGD* estimates the distance from the obtained Pareto front to the true Pareto front. For each solution, the minimum Euclidean distance between the solution and the true Pareto front is calculated. A set of distances are calculated and the average value of them is *IGD*, which gives a measure of both diversity and convergence.

3.3. System and application models

A generalized three-tier IoT system [31] consisted of IoT devices, fog devices and cloud servers is considered in this work as Fig. 3.

The first tier is called things tier where various IoT devices are responsible for sensing, collecting raw data and processing the data accordingly before sending out for further computation. In this work, IoT devices are homogeneous and equipped with both sensing and actuating components. Devices are grouped into clusters according to their location. For each group, devices are fully connected.

The second tier is named fog tier. In this work, it is assumed that each fog node is connected with a certain clusters directly and fog nodes are fully connected. The cloud servers are the main computing component of the third tier [32]. Fog nodes are linked with cloud tier via wide area network. The more discussion on

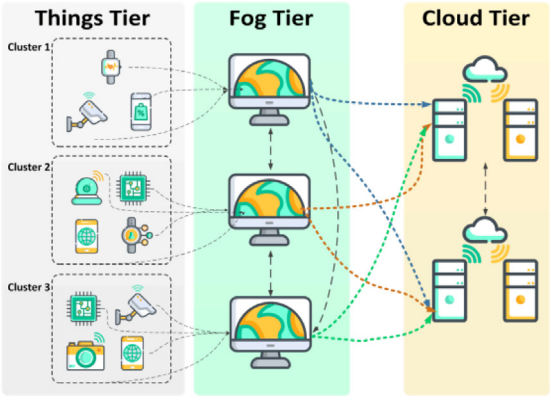


Fig. 3. An overview of the three-tier IoT system architecture [26].

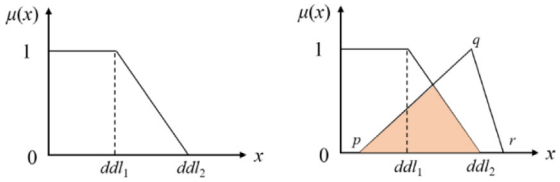


Fig. 4. $\widetilde{ddl}(j)$ and the calculation of AI_j [30].

the needs of the employment of fog computing can be found in [31].

DAG is adopted to model IoT applications which consist of dependent tasks. Tasks are assumed to be non-preemptive and all the tasks can be processed after all its precedent tasks be finished and its input data is received in this work. A five-tuple vector $\langle V, E, \tilde{w}, \tilde{D}, \widetilde{ddl}\rangle$ is used to model a DAG application, where V denotes a set of non-communication tasks. $E = \{E_{ij}\}$ is a set of directed and weighted edges representing the precedence constraints. To model the uncertainty of IoT applications, processing time, communication overhead and due date are described with fuzzy numbers where $\tilde{w}_j$ represents the fuzzy processing time of v_j and $\widetilde{D(i,j)}$ represents communication overhead of E_{ij}.

On the other hand, due time is assigned to each task in the applications to model the needs to users. $\widetilde{ddl}(j)$ indicates the due time of task j and it is represented by a right semi-trapezoid fuzzy number (ddl_1, ddl_2) as shown on the left of Fig. 4.

Similar to the tardiness index in crisp scheduling problem, **Agreement Index (*AI*)** [33] is introduced to measure and evaluate the tardiness of the scheduled task. AI_j is calculated as (9) where $\widetilde{C}_j$ denotes the completion time of task j. It is illustrated as the right of Fig. 4 and obviously, $AI_j \in (0, 1)$.

$$AI_j = \frac{area\ \widetilde{C}_j \cap \widetilde{ddl}_j}{area\ \widetilde{C}_j} \tag{9}$$

In addition, a priori robustness index, *ROB*, is used to evaluate the robustness of a solution. It represents the maximum possible deviation of the fuzzy interval. For TFN $\tilde{l} = (p, q, r)$, it can be calculated as (10):

$$ROB\left(\tilde{l}\right) = \max\left(q - p, r - q\right) \tag{10}$$

3.4. Fuzzy scheduling model

The objective of the computational offloading problem is to generate a robust schedule with high agreement index, i.e. to maximize the total agreement index and minimize the robustness index of a certain IoT application simultaneously. Based on the system and application models, the notations are given in Table 2. The mathematic formulation and constraints are given afterwards. The model regards the mathematic model in [24], where the crisp scheduling model is transferred to fuzzy one.

$$\min(1 - (\sum_{j=1}^{n} AI_j)/n, ROb_{\tilde{C}_{\max}}) \tag{11}$$

Subject to:

$$\sum_{r=1}^{n}\sum_{i=1}^{m} x_{j,i,r} = 1, \forall j \tag{12}$$

$$\sum_{j=1}^{n} x_{j,i,r} \le 1, \forall i, r \tag{13}$$

$$\sum_{j_1=1}^{n} x_{j_1,i,r} \ge \sum_{j_2=1}^{n} x_{j_2,i,r+1}, \forall r \le n-1, i \tag{14}$$

$$\tilde{C}_{j_1} - \tilde{C}_{j_2} + \tilde{M}\left(2 - x_{j_1,i,r+1} - x_{j_2,i,r}\right) \ge \frac{\tilde{w}_{j_1}}{v_i}, \forall j_1, j_2, r \le n-1, i \tag{15}$$

$$\tilde{C}_{j_1} - \tilde{C}_{j_2} \ge \sum_{r=1}^{n}\sum_{i_1=1}^{m} x_{j_1,i_1,r} \cdot \frac{\tilde{w}_{j_1}}{v_{i_1}} + \sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\sum_{r=1}^{n}\sum_{t=1}^{n} x_{j_1,i_1,r} \cdot x_{j_2,i_2,r} \cdot \frac{\tilde{D}(j_1,j_2)}{B(i_1,i_2)}, \forall j_1 \to j_2 \tag{16}$$

$$\tilde{C}_{\max} \ge \tilde{C}_j, \forall j \tag{17}$$

where (11) defines the bi-objective optimization problem. Eqs. (12)–(14) guarantee the validity of the generated solution. More precisely, (12) ensures that each task can be processed once and only once. At a given position of the processor, no more than one task can be processed according to (13). The order of the queue is strictly met according to (14), which means $(r+1)$th task does not exist in the queue if there is no the rth task on the certain processor. As the completion time and data size are all fuzzy numbers, the operators in (15) and (16) are the fuzzy operators. Eq. (15) ensures that for two tasks on the same processor, the $(r+1)$th task should not be processed before the completion time of the rth task. Eq. (16) ensures the validity of precedence constraints between the tasks and (17) presents how makespan is calculated.

4. The details of the proposed algorithm

The proposed algorithm is designed to achieve a robust scheduling solution and its detail is given in this section. First of all, the flowchart of the algorithm is presented to provide an overview. And then, a fuzzy clustering method, which is used to pre classify the task graph and its specific operators are explained in details. After that, an enhanced EDA proposed for fuzzy partition rule learning is introduced. In addition, two local intensification methods designed for opportunistic improving the system performance are presented. A complexity analysis is offered at the end.

4.1. Flowchart

Our proposed algorithm is illustrated in Fig. 5. IoT application is clustered through fuzzy pre partition method in accordance to task characteristics. After have been clustered into L groups, these groups of tasks are assigned to corresponding tiers according to the "cluster-tier allocation rule", which is produced, evolved and optimized by fuzzy EDA.

Table 2
Notations for the models.

	Notation	Implication
Decision variables	$x_{j,i,r}$	{0, 1}, xj,r,i = 1 denotes task j is processed on node i in position r, otherwise, xj,i,r = 0;
	$\tilde{C}_j$	triangle fuzzy number, the completion time of task j, j = 1,…,n;
	m_j	{1, …, m}, node assignment of task j;
Problem variables & constraints	n	Number of tasks in a certain application;
	m	Number of nodes in three tiers;
	$B(m_i, m_j)$	Bandwidth between node mi and mj;
	$\tilde{w}_j$	Triangular fuzzy number, the computation time of task j;
	v_i	Processing speed of node i;
	$\tilde{D}(j_1, j_2)$	Triangular fuzzy number, the inter-task data size between task j1 and j2;
	$j_1 \rightarrow j_2$	Task j1 is precedent of task j2 in the task graph;

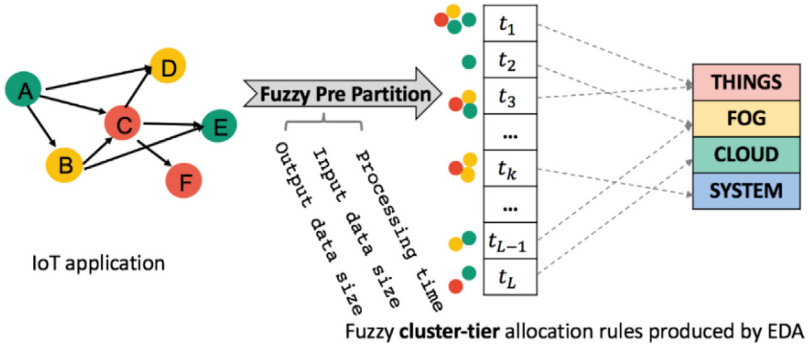


Fig. 5. Diagram of the proposed algorithm.

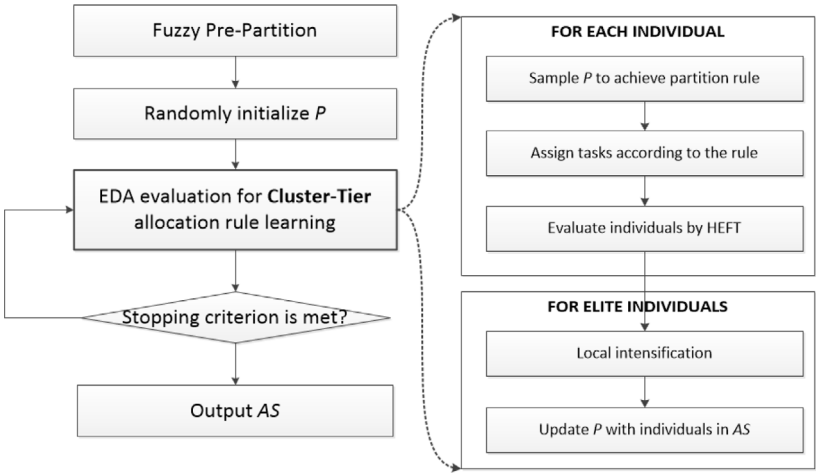


Fig. 6. Flow chart of the proposed algorithm.

The flow chart of the whole algorithm is provided in Fig. 6. It can be seen that the second part of the algorithm is fuzzy partition learning method, which is an enhanced EDA with a knowledge driven local intensification procedure. The AS containing Pareto solutions are outputted as the results.

4.2. Fuzzy pre partition

Fuzzy logic combines both numerical and linguistic information obtained from human experts. Task clustering is the first and essential step of the fuzzy logic based partition strategy that aims to partition the application without knowing the latest status of the processors. In this work, based on three features of tasks, all the tasks are classified into different clusters through the fuzzy nearness calculation method.

As the computational workload directly influences the computing resource requirement, and the communication overhead influences the latency between different tiers, three features of tasks: **computational processing time, input and output communication overhead**, are used to cluster the tasks. The input and output data size of task k is calculated by (18) and (19), respectively:

$$\widetilde{D_{in}}(k) = \sum_j \widetilde{D(j,k)}, \forall j \in Parents(k) \tag{18}$$

$$\widetilde{D_{out}}(k) = \sum_j \widetilde{D(k,j)}, \forall j \in Children(k) \tag{19}$$

where $Parents(k)$ and $Children(k)$ denote the sets of parent and children tasks of task k. Then, the tasks are labelled by these three features:

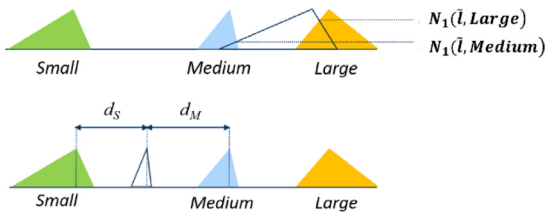


Fig. 7. (a) Maximum nearness value (Upper) (b) Minimum distance (Below).

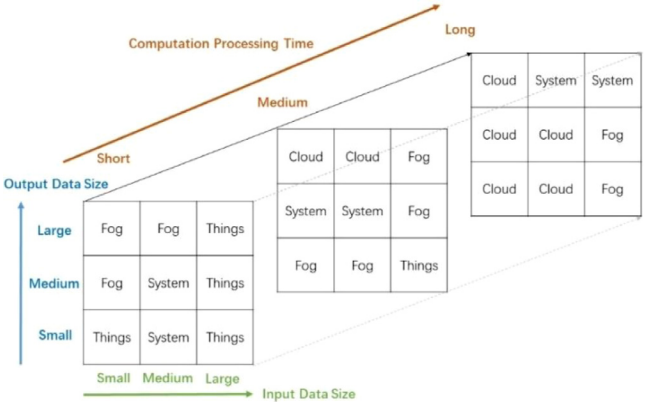


Fig. 8. Fuzzy Rule for task–tier assignment.

Computational processing time: $\tilde{w} \in \{Short, Moderate, Long\}$; **Input/Output** data **size**: $\widetilde{D_{in}}/\widetilde{D_{out}} \in \{Small, Medium, Large\}$ (see Fig. 7).

For each dimension, the tasks are sorted by fuzzy dominance to label the tasks. The detailed pseudo code is provided as Method 1. As shown in Method 1, for each dimension, tasks are sorted in the descending order of $\tilde{l}$ and the minimum, average and maximum values are chosen as the centres of corresponding clusters. In this way, all the tasks are labelled with three levels.

Method 1: Procedure of fuzzy pre partition

For r in {computational processing time, input data size, output data size}
  $\tilde{l} = \tilde{w}/\widetilde{D_{in}}/\widetilde{D_{out}}$ and sort the tasks in accordance to the descending order of $\tilde{l}$;
  $Large := \mathbf{max}(\tilde{l})$; $Medium := \mathbf{avg}(\tilde{l})$; $Small := \mathbf{min}(\tilde{l})$;
  For each task k
    $N_{1L}(k) := N_1(\tilde{l}, Large)$; $N_{1M}(k) := N_1(\tilde{l}, Medium)$; $N_{1S}(k) := N_1(\tilde{l}, Small)$.
    If $\nexists N_{1x}(k) > 0, \forall x$ (As shown in Fig. 7 (b))
      $d_s(k) := dis(Small, \tilde{l})$; $d_M(k) := dis(Small, \tilde{l})$; $d_L(k) := dis(Small, \tilde{l})$.
      $Level_t(k) = \mathbf{argmin}(d_s(k), d_M(k), d_L(k))$;
    Else (As shown in Fig. 7 (a))
      $Level_t(k) = \mathbf{argmax}(N_{1S}(k), N_{1M}(k), N_{1L}(k))$;
    End
  End
End

4.3. "Cluster-Tier" allocation rule

After the pre partition, each task is labelled and all tasks are clustered into $3 \times 3 \times 3 = 27$ clusters. Then, it is needed to learn and determine the "cluster-tier" allocation rule as illustrated in Fig. 8. For each task cluster, there are four choices: **things tier, fog tier, cloud tier and the whole system** and the solution space of tier assignment is 4^{27}, which is too large for the Brute-force approach. Thus, an EDA is employed to produce, evolve and optimize this rule.

Once the task–tier assignment rule is determined, the tasks will be assigned to the processors of corresponding tier by the insert based rule of heterogeneous earliest finish time first [27]. After that, the *AI* and *ROB* of each application can be achieved accordingly. The details of our proposed EDA for fuzzy task–tier assignment rule learning are given in next section.

Category	1	2	…	k	…	L
Tier Assignment	t_1	t_2	…	t_k	…	t_L

Fig. 9. Example of encoding method.

4.4. EDA for the allocation rule learning

EDA is a population based evolutionary algorithm which builds a probability model to describe the solution space and learns from the elite individuals. Many kinds of statistics sampling and updating methods could be chosen accordingly to learn, evaluate and build the possibility model. Standard EDA procedure is summarized as follows. Firstly, probability model P is built, and the individuals are sampled with P. The generated individuals are then evaluated. Then, P is updated in accordance to the elite individuals. After that, new individuals are sampled with the updated P value until the stopping criterion is met.

Encoding and decoding

Individuals are encoded according to the tier assignment rule as Fig. 9, where t_k represents the tier assignment of the kth cluster. We have $t_k \in \{0, 1, 2, 3\}$ where {0, 1, 2} refers to the things, fog, and cloud tier respectively and 3 represents the whole system, which represents the case of no exact tier assignment is recommended for this cluster of tasks.

HEFT is employed as the decoding method to schedule the tasks to corresponding processors after the tier assignment is determined. According to HEFT, tasks are sorted in sequence of the application DAG topological structure and all the computational nodes are considered as heterogeneous parallel processors. Each task is assigned to the earliest available processor or the valid idle space.

Probability model and its initialization

Tier assignment probability model P is designed as (20), where $p_{k,j}(g)$ represents the probability that the kth category tasks should be encoded as j at the gth generation of the evolution. For each k, we have $\sum_j p_{k,j}(g) = 1$. In addition, P is initialized randomly, e.g. $p_{k,j}(0) = 0.25$ for each k, j.

$$P(g) = \left\{p_{k,j}(g)\right\}_{L\times 4} \tag{20}$$

Sampling and updating method

The kth position of the encoding string in Fig. 9 is generated by sampling the kth row with Roulette Wheel Method (**RWM**). In addition, the probability model is updated with individuals in AS, which is the set of Pareto solution achieved until the current generation. With individuals in AS, the probability model learns from both history data and current elite individuals. Population based incremental learning method (PBIL) [34] is used for updating as (21), where $\alpha \in (0, 1)$ denotes the learning rate and $I^i_{k,j}(g)$ is the indicator function corresponding to the ith individual of AS.

The pseudo code of the proposed EDA is presented as Method 2.

$$p_{i,j}(g+1) = (1-\alpha) \times p_{i,j}(g) + \alpha \times \sum_i I^i_{k,j}(g) / |AS| \tag{21}$$

$$I^i_{k,j}(g) = \begin{cases} 1, & k\text{th task category is assigned to} \\ & j \text{ in the } k\text{th individual} \\ 0, & \text{otherwise} \end{cases}$$

Method 2: EDA for the allocation rule learning

$P(g)$: probability model at the g-th generation;
G: the number of evolution generations;
$P_{k,j}(0) = 0.25,\ \forall\ k, j$
While $g < G$
  For l from 1 to N
    For k from 1 to L
    $t_k = \mathbf{RWM}(p_{k,1}(g), p_{k,2}(g), p_{k,3}(g), p_{k,4}(g))$;
    End
    Decode the l-th individual with HEFT;
    Update AS with the l-th individual;
  End
  For individual x in AS
   Implement Method 3 for x;
  End
  Update P through (20); g++;
End

Local Intensification

To enhance the exploration of the proposed algorithm, a problem specific local search methods are designed to achieve better solutions. Here, we adjusted the tier assignment of tasks with small *AI* within the limited local search calculation budget. The encoding string is altered directly when possible. For the specific solution in AS, to decrease the task latency, all the clusters of tasks are sorted in the ascending sequence of average *AI*. In this order, other tier assignment choices are tried until the stopping criterion is met.

After the evaluation of individuals in population, local search procedure is used for each solution in AS. AS is updated by the new solutions produced by the local search.

Method 3: Local intensification procedure

T={0,1,2,3} as encoding section introduced;
For each solution x ($\overline{t_0 t_1 \ldots t_k \ldots t_L}$)in AS
  For tasks in each cluster i
    $avgAI(i) = \sum_{j \in i} AI_j / |cluster_i|$
  End
  Sort task clusters i in ascending order of $avgAI(i)$; $g = 0$; $k = 0$
  While $g < LS$:
    $k :=$ **argmin**$_i(avgAI(i))$ (for i has not been considered); $t_k :=$ the tier assignment of k
    For S **in** $T \backslash t_k$:
      g++; $t_k := S$; Evaluate the new solution x' ($\overline{t_0 t_1 \ldots S \ldots t_L}$) with HEFT;
      If $x' \succ x$: Update AS with x'; $x := x'$;
      Else If $x \succ x'$: break;
      Else:
        If rand(0, 1) < 0.5:
          Update AS with x'; $x := x'$;
        End
      End
    End
    k has been considered;
  End
End

Complexity analysis

The time complexity is analysed according to the flowchart in Fig. 6, and listed as follows:

(1) Fuzzy pre-partition operator: $O(n)$;

(2) Initialization operator: the time complexity is bounded by a constant.

(3) Evaluation of EDA (total G generations):

Solution evaluation: HEFT has an $O(n \cdot M_{p,c} \cdot m)$ time complexity for m processors, and $M_{p,c}$ denotes the max parents or children number. For each generation, time complexity is $O(n \cdot M_{p,c} \cdot m \cdot N)$.

Local intensification: LS denotes the step length for each individual in AS. The total computation complexity can be calculated as $O(|\ AS\ |\cdot n \cdot M_{p,c} \cdot m \cdot LS)$. It is ruled that $|\ \mathrm{AS}\ | \leq N$, then the complexity is transferred to $O(n \cdot M_{p,c} \cdot m \cdot N \cdot LS)$.

Probability model and AS updating: probability model updating is bounded by constant. For AS updating, suppose $|\ \mathrm{AS}\ | = N$, then the number of comparison between the solutions is N^2.

In conclusion, the time complexity is estimated as $O(\max(n \cdot M_{p,c} \cdot m \cdot N \cdot LS, N^2) \cdot G)$.

The space complexity of our algorithm can be analysed as follows. Each solution needs an array of length L to store its fuzzy rule and an array of length n to store its corresponding tier assignment for the local intensification. For the probability model, a $4L$ matrix is needed. To store the Pareto solutions, the scale of AS is up to N. Thus, the space complexity of our proposed algorithm is estimated as $O((n + L) \cdot N)$.

5. Simulation

To evaluate the performance of our proposed algorithm, we compare it with the selected algorithms, HEFT [27] and two novel multi-objective intelligent optimization algorithms: MOEA/D [35] and FDHEFT [25]. For fair comparison, the algorithms are all coded in C++ and run on the same processor.

5.1. Comparison algorithms

HEFT is a popular heuristic algorithm to optimize workflow scheduling problems. As HEFT is a single objective scheduling algorithm, which produces a single solution once, the solution produced by HEFT is adopted as a baseline to evaluate the performance of AS produced by our EDA.

On the other hand, multi-objective evolutionary algorithm based on decomposition (MOEA/D) is chosen as it is a state-of-the-art multi-objective evolutionary algorithm. The multi-objective problem is converted into a number of single-objective sub-problems by using a decomposition approach [36]. FDHEFT is an algorithm tailored to solve multi-objective workflow scheduling problem with fuzzy logic method. It outperformed ε-fuzzy particle swarm optimization algorithm [37], multi-objective HEFT [38], and strength Pareto evaluation algorithm [39]. In this work, we modified the objectives of the algorithm to optimize agreement index and *ROB* and adopted it as a comparison algorithm.

5.2. Parameter settings

The system variables are set in advance as it is known before the scheduling. The detailed environment settings can be referred to [26]. In addition, to measure the impact of data transfer latency on computing performance, **communication to computation ratio (*CCR*)** is used in the simulation experiments. It is calculated as average communication time divided by average computation time.

Five parameters are considered in our experiments, which are the population size (N), corresponding learning rate (α), and stopping criterion related parameter and the steps of local search operator (LS). To compare the algorithms fairly, the stopping criterion, e.g. the running time, set as $n/10$ (s), where n denotes the number of tasks. LS is set by experience, which is proportional to the length of encoding string.

Population size (N) and learning rate (α) are selected by the Taguchi method of the design-of-experiment method (DOE) [40], which is a statistical method and it is used to investigate how the change of parameters affects the performance of the proposed algorithm. Orthogonal arrays of four-parameter levels are

Table 3
Factor levels of parameters.

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

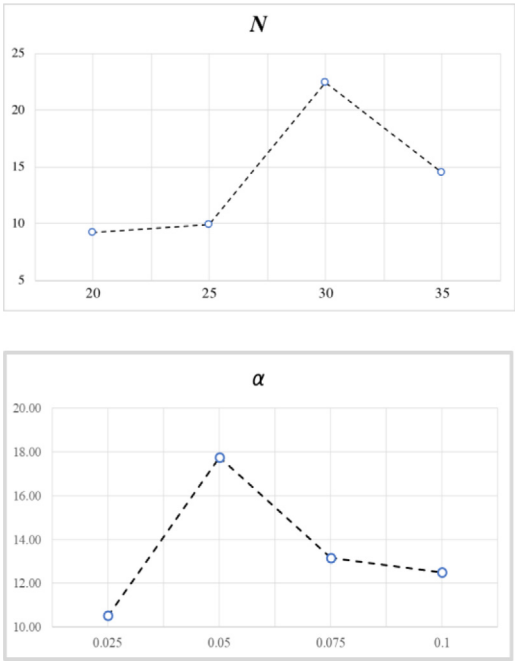


Fig. 10. Main effect plot.

employed as shown in Table 3, and 4^2 full-factorial experiments are employed.

For each combination of *Loose*×*CCR* (3×4 = 12), a random instance is produced, where *Loose* is due time related parameter. Under each pair of (*Loose*, *CCR*), 16 combinations of (N, α) are set as parameters independently and the obtained AS_{ci} ($ci = 1, 2, \ldots, 16$) are stored. The final AS (*FAS*) are obtained by integrating $AS_1, AS_2, \ldots, AS_{16}$. Then, the contribution of a certain combination (*CON*) is carried out, where $CON\,(ci) = \left|AS'_{ci}\right| / |FE|$ and $AS'_{ci} = \{X_l \in AS_{ci} | \exists X_{l'} \in FAS, X_l = X_{l'}\}$. The average *CON* of each combination is used as the response value (RV) and the main effect plot is generated as Fig. 10.

From Table 4 and Fig. 10, it can be seen it is better for N to be set neither too big nor too small. Thus, $N = 30$ and $\alpha = 0.05$ are suggested as the parameters in the simulation experiments under the certain stopping criterion.

Table 5
Simulation settings.

	Notation	Parameter setting in this paper
Randomly generated DAG parameters	n	200 with 50% variation;
	CCR	{0.1, 0.5, 1.0, 5.0};
	Loose	{1.0, 1.5, 2.0};
Algorithm parameters	N	30, population size;
	α	0.05, learning rate;
	LS	Set as the length of the encoded string;
	Running time	The algorithm is stopped after n/10 (s).

5.3. Randomly generated application graphs

A pseudorandom task graph generator TGFF [41] is used to produce randomly generated DAGs. The DAG structure and computation workload for each task is determined by TGFF and for the transfer data size of each task, it is produced according to the normal distribution of a certain *CCR*. The maximum number of successors and processors of a certain task are both set as 10. After the DAG is achieved, the crisp task processing time and communication data size is transferred into triangle fuzzy number as (22).

$$\tilde{X} = (0.9X, X, 1.1X) \tag{22}$$

The crisp due time of task is set by (23):

$$ddl\,(i) = tlvl\,(i)\,/Loose \tag{23}$$

where $tlvl(i)$ is the largest sum of processing time of the task path from the entry task to task i. *Loose* controls the due dates where a larger *Loose* leads to tight due times. Fuzzy due times is achieved as follows:

$$\begin{aligned} ddl_1\,(i) &= ddl\,(i)\,, \\ ddl_2(i) &= ddl\,(i) \times (1 + rand\,(0, 1)) \end{aligned} \tag{24}$$

Specific parameters of our simulation settings are listed in Table 5. For each level of *CCR* and *Loose*, 50 DAGs are produced randomly, $50 \times 4 \times 3 = 600$ instances are thus used to validate the efficiency of the proposed algorithm.

Effects of task partition operator

As our algorithm produces the fuzzy "cluster-tier" allocation rule and the tasks are assigned to specific tiers rather than scheduled within the whole system. To verify the efficiency of the partition operator, we compare the results produced by our proposed algorithm with HEFT solutions. The comparison results are listed in Table 6 where the average value of 600 instances under each situation is calculated and HEFT solution is compared AS produced by our algorithm.

The column "dominating HEFT (%)" denotes the percentage of the instances in which HEFT solution is dominated by any solution in our AS. Similarly, "equals to HEFT (%)" denotes our AS contains only one solution and it equals to the HEFT solution,

Table 4
The RV value.

Experiment numbers	Factor		RV (%)	Experiment numbers	Factor		RV (%)	Experiment numbers	Factor		RV (%)	Experiment numbers	Factor		RV (%)
	N	α			N	α			N	α			N	α	
1	1	1	7.89	5	2	1	7.89	9	3	1	15.79	13	4	1	10.53
2	1	2	15.79	6	2	2	7.89	10	3	2	28.95	14	4	2	18.42
3	1	3	5.26	7	2	3	7.89	11	3	3	23.68	15	4	3	15.79
4	1	4	7.89	8	2	4	15.79	12	3	4	15.79	16	4	4	10.53

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

Table 6
Results compared with HEFT heuristic method.

Deadline loose	*CCR*	Dominating HEFT (%)	Equals to HEFT (%)	Non-dominating HEFT (%)	Dominated by HEFT (%)
1.0	0.1	66.8	11.4	2.0	19.8
	0.5	70.0	12.6	2.8	14.6
	1.0	96.0	1.4	0.8	1.8
	5.0	100.0	0.0	0.0	0.0
1.5	0.1	77.0	7.2	4.0	11.8
	0.5	86.4	4.8	3.6	5.2
	1.0	96.4	1.4	1.4	0.8
	5.0	100.0	0.0	0.0	0.0
2.0	0.1	79.6	7.6	5.6	7.2
	0.5	93.4	2.6	1.0	3.0
	1.0	99.2	0.8	0.0	0.0
	5.0	100.0	0.0	0.0	0.0

Table 7
C metrics results compared with multi-objective algorithms.

Deadline loose	*CCR*	*C*(fEDA, MOEA/D)	*C*(MOEA/D, fEDA)	*C*(fEDA, FDHEFT)	*C*(FDHEFT, fEDA)
1.0	0.1	0.885	0.057	0.858	0.000
	0.5	0.970	0.000	1.000	0.000
	1.0	0.967	0.000	1.000	0.000
	5.0	0.950	0.000	1.000	0.000
1.5	0.1	0.950	0.000	0.917	0.000
	0.5	0.910	0.000	0.777	0.050
	1.0	0.810	0.075	0.852	0.050
	5.0	0.979	0.000	1.000	0.000
2.0	0.1	0.907	0.050	1.000	0.000
	0.5	0.922	0.000	1.000	0.000
	1.0	0.879	0.067	0.930	0.000
	5.0	0.994	0.000	1.000	0.000

"non-dominating HEFT (%)" denotes HEFT solution and AS solutions are non-dominated and "dominated by HEFT(%)" denotes that at least one of our AS is dominated by HEFT solution.

The results show that a large proportion of HEFT solutions is dominated by our AS. For different *CCR*, it can be seen that when *CCR* is 0.1, the proportion of instances where EDA dominates the HEFT solution is around 70%. As *CCR* increases, the proportion of EDA dominating HEFT solution increases to 100%. Our algorithm performs better for applications with heavy communication overhead and the possible reason is that the fuzzy partition improves the robustness of HEFT. When *CCR* is large, the communication latency between different processors and tiers increases, the pre task partition according to their corresponding characteristics plays an important role. In addition, under the same *CCR*, when the due time turns earlier, more HEFT solutions are dominated by our AS and the proportion of solutions dominated by HEFT solution decreases. The main reason is that when due time is loose, many tasks can be finished before its due time and its *AI* value turns to be 1.0 and the domination relationship depends on the robustness index. When the due time turns tight, our algorithm is more efficient on *AI* objective, so most HEFT solutions are dominated.

Comparison with multi-objective algorithms

We compared our algorithm with the novel MOEA/D in terms of *C* metrics. The average *C* metric values of 600 instances under different situations are summarized in Table 7. Fuzzy EDA (fEDA), MOEA/D and FOHEFT refer to our proposed algorithm and these two comparison algorithms correspondingly.

From Table 6, it shows that *C*(fEDA, MOEA/D) is much larger than *C*(MOEA/D, fEDA), which means most MOEA/D solutions are dominated by ours. This reveals that, fEDA is good at learning the "cluster-tier" allocation rule and the tailored knowledge driven local search operator is more efficient than genetic operators on this specific problem. Compared with FDHEFT, it can be seen that our solutions dominate FDHEFT solutions on this problem.

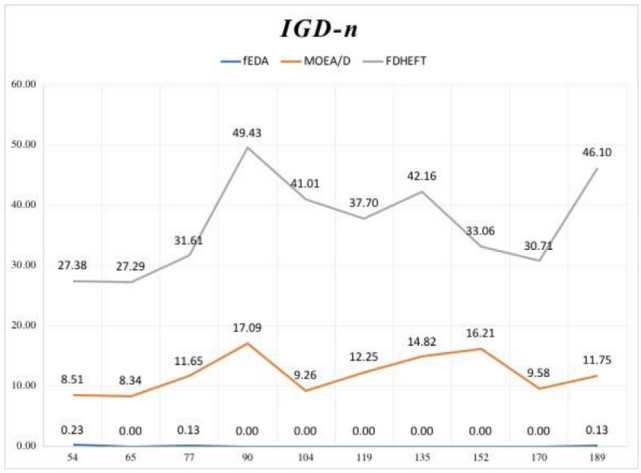
In addition, *IGD* is used as a metric to evaluate the performance of algorithms as a supplement to *C* metrics. As mentioned before, a small *IGD* means denotes a good multi-objective solution. Compared to MOEA/D and FOHEFT, it can be seen that solutions produced by fEDA performs better on both *AI* and robustness according to *IGD* values presented in Table 8.

5.4. Dags from real world problems

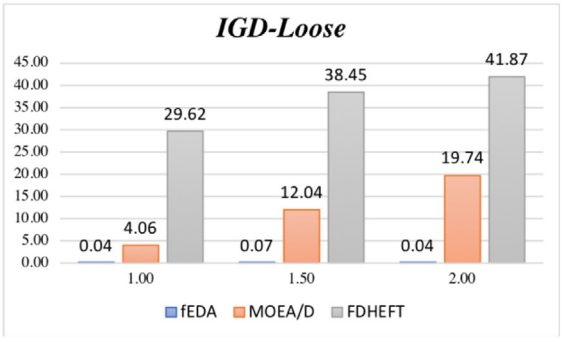
Besides the random produced DAGs, we consider two kinds of application DAGs from real world problems: Gauss elimination algorithm [42] and FFT [43] (Fast Fourier Transformation). The task number of Gauss elimination algorithm program is related to the matrix size (b) and equals to $(b^2+b-2)/2$. The task number of FFT can be calculated as $(2b-1+blog_2b)$, where b denotes the size of array and $b=2^k$. It is referred to [27] for more details.

For Gauss Elimination application, 10 different scales of DAGs are produced (b = 10–19). For FFT application, 5 different scales of DAGs are produced (k = 3–7). For each scale of DAG, *CCR* and *Loose* combinations in Table 5 are used to generate different cases. In this way, $10 \times 4 \times 3 = 120$ and $5 \times 4 \times 3 = 60$ instances are produced as a benchmark set respectively. Here, *IGD* values are calculated and presented as follows. The plots show how the average *IGD* value varies in accordance *n*, *CCR* and *Loose*.

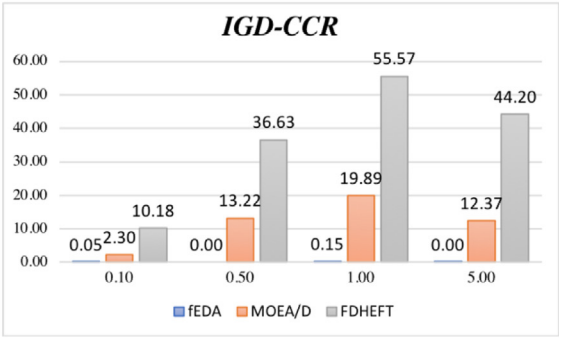
For Gaussian Elimination Application results presented in Fig. 11, it can be seen that fEDA performs much better on *IGD*



(a) '*IGD-n*' plot



(b) '*IGD-Loose*' histogram



(c) '*IGD-CCR*' histogram

Fig. 11. Variable dependent IGD value plots of Gaussian Elimination Application.

Table 8
IGD values compared with multi-objective algorithms.

Deadline loose	*CCR*	fEDA	MOEA/D	FDHEFT
1.0	0.1	0.057	0.616	0.901
	0.5	0.000	22.328	44.518
	1.0	0.070	6.349	66.218
	5.0	0.086	3.920	66.544
1.5	0.1	0.000	1.793	5.726
	0.5	0.097	0.873	6.192
	1.0	0.116	19.259	26.319
	5.0	0.094	18.856	63.640
2.0	0.1	0.397	2.853	36.802
	0.5	0.001	1.659	32.510
	1.0	0.049	1.019	8.347
	5.0	0.000	6.207	11.802

values compared with MOEA/D and FDHEFT. As the task number increases from 50 to 200, *IGD* values of fEDA equals to 0.0 under most situations, which means that the solutions produced by fEDA accounts for a large proportion of the general Pareto front. From Fig. 11(b) and (c), it can be seen that under different kinds of due time constraints and *CCR*, fEDA provides a better performance compared with the other algorithms.

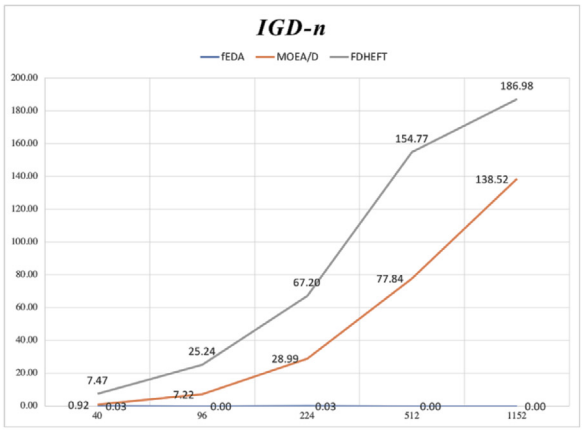
Fore FFT applications, there are more precedence constraints within the DAGs compared with Gaussian elimination application, which imposes more difficulty on scheduling and resource allocation. As presented in Fig. 11, it can be seen that fEDA performs much better. As the task number increases from 40 to 1200, *IGD* values of fEDA equals to 0 except for the smallest-scale instance. When $n = 40$, $IGD = (0.03, 0.92, 7.47)$, which denotes that when the problem scale is small, the performance gap is not obvious. As n increases, the gap turns larger, which shows that our algorithm is effective as data scale increases. From Fig. 11(b) and (c), our algorithm performs better as the due time constraints are moderate and *CCR* is large.

There are more precedence constraints within the DAG of FFT Applications compared with Gaussian Elimination Application. This imposes more difficulty on scheduling and resource allocation. As presented in Fig. 12, it can be seen that fEDA performs much better on *IGD* values. As the task number increases from 40 to 1200, *IGD* values of fEDA equals to 0 except for the smallest-scale instance. When $n = 40$, $IGD = (0.03, 0.92, 7.47)$, which denotes that when the problem scale is small, the performance gap is not obvious. As n increases, the gap turns to be larger, which shows that our algorithm is effective as data scale increases. From Fig. 12(b) and (c), our algorithm performs better as the deadline constraints are not very tight and *CCR* is large.

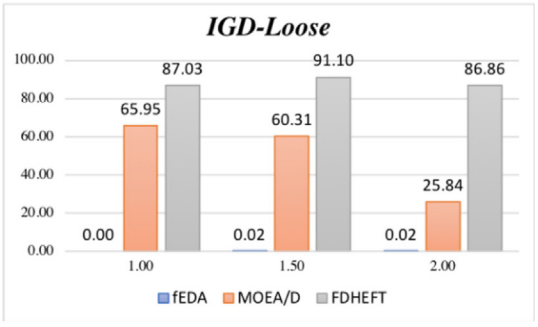
6. Conclusions

In this work, an evolutionary algorithm guided fuzzy rule partition method is proposed to solve the uncertain computation offloading problem in IoT–fog–cloud system. The main contributions are listed as follows:

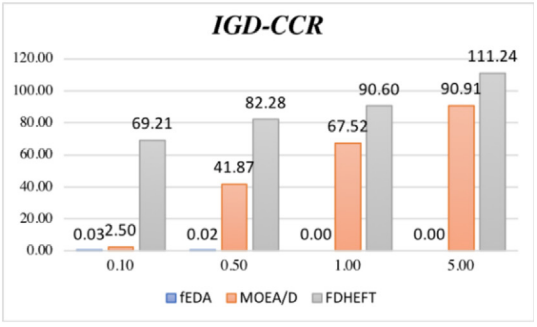
1. A detailed system, application and mathematic models are given to formally formulate the problem. Fuzzy number is



(a) *IGD-n*' plot



(b) '*IGD-Loose*' histogram



(c) '*IGD-CCR*' histogram

Fig. 12. Variable dependent IGD value plots of FFT Application.

adopted to model the uncertain characteristic of workloads and transfer data.

2. A fuzzy logic based partition method is introduced to cluster and assign different types of tasks of the application to different tiers. Based on this algorithm, an optimal DAG task offloading scheme is achieved. Tasks are offloaded to specific tier and a large amount of computation and scheduling capability is saved.

3. Knowledge based local search operators are introduced to efficiently improve the performance of the fuzzy logic based partition method.

Finally, simulation experiments conducted on different due date constraints and *CCR* cases show that our algorithm performs well on both agreement index and robustness objectives.

In the future, we plan to combine the offline training and online model due to the time complexity of our proposed algorithm is relatively high. In this way, the algorithm could be expanded to solve large scale and online problems in a timely manner. In addition, a prototype system will be built to validate our design.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

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].

References

[1] F. Mattern, C. Floerkemeier, K. Sachs, I. Petrov, P. Guerrero, From the internet of computers to the internet of things, in: Lecture Notes in Computer Science, vol. 6462, Springer, Berlin, Heidelberg, 2010, pp. 242–259.
[2] M. Gorlatova, H. Inaltekin, M. Chiang, Characterizing task completion latencies in fog computing, arXiv preprint arXiv:1811.02638.
[3] M. Aazam, S. Zeadally, K.A. Harras, Offloading in fog computing for IoT: review, enabling technologies, and research opportunities, Future Gener. Comput. Syst. 87 (2018) 278–289.
[4] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J.P. Jue, All one needs to know about fog computing and related edge computing paradigms: A complete survey, J. Syst. Archit. 98 (2019) 289–330.
[5] P.Y. Zhang, M.C. Zhou, G. Fortino, Security and trust issues in fog computing: a survey, Future Gener. Comput. Syst. 88 (2018) 16–27.
[6] Y.K. Kwok, I. Ahmad, Static scheduling algorithms for allocating directed task graphs to multiprocessors, ACM Comput. Surv. 31 (4) (1999) 406–471.
[7] A. Tchernykh, U. Schwiegelsohn, E.G. Talbi, M. Babenko, Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability, J. Comput. Sci. 36 (2019) 1–9.
[8] R. Sowinski, M. Hapke, Scheduling Under Fuzziness, Physica-Verlag, 2000.
[9] J.J. Palacios, I. González-Rodríguez, C.R. Vela, J. Puente, Robust multiobjective optimisation for fuzzy job shop problems, Appl. Soft Comput. 56 (2017) 604–616.
[10] J. Bidot, T. Vidal, P. Laborie, J. Beck, A theoretic and practical framework for scheduling in a stochastic environment, J. Sched. 12 (3) (2009) 315.
[11] P. Li, B. Liu, Entropy of credibility distributions for fuzzy variables, IEEE Trans. Fuzzy Syst. 16 (1) (2009) 123–129.
[12] J. Meng, H. Tan, X.Y. Li, Z. Han, B. Li, Online deadline-aware task dispatching and scheduling in edge computing, IEEE Trans. Parallel Distrib. Syst. 31 (2020) 1270–1286.
[13] N. Verba, K.M. Chao, J. Lewandowski, N. Shah, A. James, F. Tian, Modeling. industry, Modeling industry 4.0 based fog computing environments for application analysis and deployment, Future Gener. Comput. Syst. 91 (2019) 48–60.
[14] H. Yuan, J. Bi, M. Zhou, Q. Liu, A.C. Ammari, Biobjective task scheduling for distributed green data centers, IEEE Trans. Autom. Sci. Eng. (2020) 1–12.

[15] H. Yuan, J. Bi, W. Tan, M. Zhou, B.H. Li, J. Li, TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds, IEEE Trans. Cybern. 47 (2017) 3658–3668.
[16] Z. Lv, L. Wang, Z. Han, J. Zhao, W. Wang, Surrogate-assisted particle swarm optimization algorithm with Pareto active learning for expensive multi-objective optimization, IEEE/CAA J. Autom. Sin. 6 (2019) 838–849.
[17] Q. Kang, X. Song, M. Zhou, L. Li, A collaborative resource allocation strategy for decomposition-based multiobjective evolutionary algorithms, IEEE Trans. Syst. Man Cybern : Syst. 49 (2019) 2416–2423.
[18] Y. Nan, W. Li, W. Bao, F.C. Delicato, P.F. Pires, A.Y. Zomaya, A dynamic tradeoff data processing framework for delay-sensitive applications in cloud of things systems, J. Parallel Distrib. Comput. 112 (2018) 53–66.
[19] M. Yang, H. Ma, S. Wei, Y. Zeng, Y. Chen, Y. Hu, A multi-objective task scheduling method for fog computing in cyber-physical-social services, IEEE Access 8 (2020) 65085–65095.
[20] W. Zhan, et al., Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing, IEEE Internet Things J. 7 (6) (2020) 5449–5465.
[21] H. Yuan, M. Zhou, Profit-maximized collaborative computation offloading and resource allocation in distributed cloud and edge computing systems, IEEE Trans. Autom. Sci. Eng. (2020) 1–11.
[22] R. Mahmud, S.N. Srirama, K. Ramamohanarao, R. Buyya, Quality of experience (QoE)-aware placement of applications in Fog computing environments, J. Parallel Distrib. Comput. 132 (2019) 190–203.
[23] S. Hu, G. Li, Dynamic request scheduling optimization in mobile edge computing for IoT applications, IEEE Internet Things J. 7 (2) (2020) 1426–1437.
[24] S. Sundar, B. Liang, Offloading dependent tasks with communication delay and deadline constraint, in: IEEE Conference on Computer Communications, INFOCOM 2018, IEEE, Honolulu, Hl, USA, 2018, pp. 37–45.
[25] X. Zhou, G. Zhang, J. Sun, et al., Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT, Future Gener. Comput. Syst. 93 (2019) 278–289.
[26] C.G. Wu, W. Li, L. Wang, A.Y. Zomaya, Hybrid evolutionary scheduling for energy-efficient fog-enhanced internet of things, IEEE Trans. Cloud Comput. (2018) 1–8.
[27] H. Topcuoglu, S. Hariri, M.Y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE Trans. Parallel Distrib. Syst. 13 (3) (2002) 260–274.
[28] L.C. Canon, E. Jeannot, R. Sakellariou, W. Zheng, Comparative Evaluation of the Robustness of Dag Scheduling Heuristics, Grid Computing: Achievements and Prospects, Springer, Boston, MA, 2008, pp. 73–84.
[29] J.J. Durillo, H.M. Fard, R. Prodan, MOHEFT: A multi-objective list-based method for workflow scheduling, in: 4th International Conference on Cloud Computing Technology and Science Proceeding, IEEE, Taipei, 2012, pp. 185–192.
[30] L.A. Zadeh, Fuzzy sets, in: Fuzzy Sets Fuzzy Logic and Fuzzy Systems: Selected Papers By Lotfi a. Zadeh, World Scientific Publishing Co. Inc., USA, 1996, pp. 19–34.
[31] W. Li, et al., System modelling and performance evaluation of a three-tier cloud of things, Future Gener. Comput. Syst. 70 (2017) 104–125.
[32] M.H. Ghahramani, M. Zhou, C.T. Hon, Toward cloud computing QoS architecture: analysis of cloud systems and cloud services, IEEE/CAA J. Autom. Sin. 4 (2017) 6–18.
[33] M. Sakawa, R. Kubota, Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms, European J. Oper. Res. 120 (2000) 393–407.
[34] 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.
[35] E.D. Jiang, L. Wang, An improved multi-objective evolutionary algorithm based on decomposition for energy-efficient permutation flow shop scheduling problem with sequence-dependent setup time, Int. J. Prod. Res. 57 (6) (2019) 1756–1771.
[36] Q. Zhang, H. Li, MOEA/D: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.
[37] R. Garg, A.K. Singh, Multi-objective workflow grid scheduling using ε-fuzzy dominance sort based discrete particle swarm optimization, J. Supercomput. 68 (2) (2014) 709–732.
[38] J.J. Durillo, R. Prodan, Multi-objective workflow scheduling in Amazon EC2, Clust. Comput. 17 (2) (2014) 169–189.
[39] J. Yu, M. Kirley, R. Buyya, Multi-objective planning for workflow execution on grids, in: IEEE/ACM International Conference on Grid Computing, 2007, pp. 10–17.
[40] D.C. Montgomery, Design and Analysis of Experiments, Wiley, New York, 1984.
[41] R.P. Dick, D.L. Rhodes, W. Wolf, TGFF: task graphs for free, in: 6th International Workshop on Hardware/Software Codesign Computer Society, 1998, pp. 97–101.
[42] M. Cosnard, M. Marrakchi, Y. Robert, D. Trystram, Parallel Gaussian elimination on an MIMD computer, 6 (1988) 275–296.
[43] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, 1990.

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. Her current research interest includes scheduling and optimization algorithms for heterogeneous computing systems, intelligent optimization.

Wei Li received his Ph.D. 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. 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 300 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. Professor Wang now is the Editor-in-Chief of International Journal of Automation and Control, and the Associate Editor of IEEE Transactions on Evolutionary Computation, Swarm and Evolutionary Computation, etc.

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 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.