How to Cite:
Malathy, N., Kartheeswari, M., & Yogalakshmi, M. (2022). Energy efficient workflow scheduling for fog-enhanced IOT based healthcare application. *International Journal of Health Sciences*, *6*(S2), 8675–8698. https://doi.org/10.53730/ijhs.v6nS2.7244

Energy efficient workflow scheduling for fog-enhanced IOT based healthcare application

Malathy N
Assistant Professor (Senior Grade), Department of Information Technology, Mepco Schlenk Engineering College, Sivakasi, Tamilnadu
Email: malathy@mepcoeng.ac.in

Kartheeswari M
UG Student, Department of Information Technology, Mepco Schlenk Engineering College, Sivakasi, Tamilnadu
Email: kartheeswari343_it@mepcoeng.ac.in

Yogalakshmi M
UG Student, Department of Information Technology
Email: yogamani.wi_it@mepcoeng.ac.in

***Abstract*---**The Internet of Things has generated a tremendous amount of data. These files must be handled quickly and are typically kept in a cloud data center. Cloud computing can provide abundant resources to process such information IoT devices relating to data security, energy consumption, and data processing are driving the growth and interplay of fog computing and cloud computing. The problem of resource scheduling in a system designed to reduce task execution time and overall energy consumption. To partition the graph and determine task processing permutations and processor assignment, the tri EDA and division operator are employed. The comparison findings of a single application simulation reveal that the Pareto set produced by the proposed methodology can dominate a considerable number of those solutions produced by the heuristic technique and straightforward EDA. When it comes to multi-objective simulation, IoT devices can last much longer and perform similarly to other algorithms in terms of nodes' energy consumption and job completion time.

***Keywords*---**fog computing, IoT, resource scheduling, energy efficiency, makespan, evolutionary computation.

International Journal of Health Sciences ISSN 2550-6978 E-ISSN 2550-696X © 2022.
Manuscript submitted: 9 March 2022, Manuscript revised: 18 April 2022, Accepted for publication: 1 May 2022

Introduction

IoT describes a group of such objects that are embedded with sensors, actuators, software, and other technologies that are connected through wired or wireless and exchange the data with the other IoT devices It's used in a variety of real-world scenarios. The energy supply of IoT devices is limited, and data processing and communication range are constrained. In the Internet of Things, one of the most important factors that must be stabilized for QoS deployment, administration, and research optimization is quality of service (QoS). To ensure that the QoS guarantee is used to reduce limitations from equipment and network infrastructure. IoT systems must meet latency requirements from beginning to end as in real-time applications, both software and hardware must be deployed.QoS can assist reduce end-toenddelay.One of the finest alternatives for finishing time-constrained IoT applications is fog computing. Fog computing is an archetype that expands Cloud computing and network administration to the network's edge. In fog computing, the analysis along with the applications are placed among the edge tier and cloud tier. Computing in the fog using the concept of fog nodes and these nodes are very close to the data sources. Fog nodes are having a higher capacity to process and store the data obtained from IoT devices and decrease data transfer volume via the internet. Fog nodes process the data quicker by choice of sending data to the cloud server for decentralized processing. To fully move the core fog technology resources, the best utilization of spatially distributed resources is critical to improving the total performance of the system by reducing end-to-end delay, cutting bandwidth usage and costs, and improving user experience. Due to the challenges in dealing with the resource provisioning problem given by the complicated and independent fog computing infrastructure, this is not a little undertaking. Applications are launched by different users and bring additional challenges which are usually fought with each other. Task scheduling is a well-known major issue that reduces system performance. The planning of the activity on the problem of multiprocessor is the problem of the hard NP with communication delays. Multiobjective In fog computing, the task-scheduling problem is a phase as follows optimization task scheduling strategy. This issue with more factors is NP-hard, so finding a solution in a quadratic is impossible time. Optimization methods have been used to handle a variety of scheduling systems with high performance. The EDA Approach was a popular evolutionary algorithm for solving difficult scheduling problems.EDA is a stochastic optimization method that guides the search for the best solution to scheduling problems using an outside stochastic representation.EDA might be tweaked and the processes repeated to reach the same level of performance in this multi-objective DAG scheduling problem.EDA will be used in the proposed method to create the refining sequence, which will solve the work scheduling challenge while also lowering the makespan and power consumption objectives.

Literature Survey

Cloud computing is used to make fast the realization of the idea of the Internet of Things. IoT in the cloud has a lot to offer. By granting third-party access to infrastructure, cloud services can good accordance with the IoT sector and it is used to collect and for processing, the data was uploaded to the cloud. cloud computing is used to store IoT data. To manage the transmission of huge no of data as well as lowering the cloud's energy usage cloud-based IoT that it is very scalable.

Many research challenges have been increased and how the data was processed cost-effectively, close to the data source. As required the expansion of cloud computing and environmental consequence of data centers enchant them much attention. The efficient minimization of energy consumption and time by Priority-constrained parallel application scheduling. Data explosion may cause network traffic issues, fog computing is used to partition the task of breaking down data into smaller chunks. These can process the data on less powerful machines. In the Internet of Things, Fog computing is a decentralized data handling and resource distribution method (IoT). Fog-based computational resources must adhere to differences in IoT sources of data, the delay sensitivity of IoT networks, and the potentially highly large amounts of data released as well as absorbed by IoT devices. Despite existing foundations, [1] computing in the fog research is still in the early phases of development.In the Resource scheduling, approach modules on fogs node with one objective are utilized to shorten response times.[2] virtual machines in computing in the fog virtual host can share resources between them. Virtual allocation is used to reduce the network latency between them. These are not suitable for fog devices' key features like location-aware processing. End-user computers are increasingly being For IoT devices (IoT) as well as fog computing services, they're used as local servers. [3] Nano data centers are distributed system machines that can host and distribute material and programs in a peer-to-peer environment. Fog computing can complement centralized DCs to service-specific applications, such as IoT having sources of data on end-user locations, and can save energy if programs can be offloaded from centralized DCs and run on DCs.When it comes to data center maintenance costs, energy consumption is a critical factor to consider and it consists of how much amount of energy is consumed by the cooling system, not only energy consumed by the machine. It becomes a big issue. Computer Components are relocated to the network's edge. These are powered by a non-rechargeable battery. [4] The evolution of Fog computing minimizes the energy consumption by cloud servers. It performs computation on sensor data that consume more energy but it is used to reduce the energy consumption by fog nodes also. fog and cloud nodes could assist in energy saving compared to other cases.Fog Computing generally called edge computing, is a type of computing in the cloud that expands cloud computing by locating concentrated computer systems at users' residences, that store internet data and transfer it to mobile users via high-speed local contacts.As the result, computing in the fog creates a fog layer among edge users and the computing in the cloud, complementing the cloud by providing low latency, mobile users with high-speed services. The interaction and collaboration between the fog and the cloud are vital to examine in this fundamental framework. In the cloud computing environment, the tradeoff between energy usage and transmission latency is investigated. Provisioning of dynamic resources in an all-optical wide region Supporting energy-aware cloud system and fog system interacts with a Defined software Network. In Edge-cloud computing bi-objective optimization model focuses on delay and power consumption. [5]To achieve energy savings between clouds and edges, a dynamic resource provisioning technique is implemented. A software network is being used to reduce energy usage between the edge and the cloud, similar to the three-tier architecture.Under the same system design, a To manage the three-way tradeoffs among average response time, the average amount of application losses, and quantified near-optimal online solution is used. [6] The market for smart systems that are connected to IoT devices is quickly growing. To meet The merging of the

Fog system and Cloud system intends to harness the edge devices and remote data center-based-based computing resources to meet the Quality - of - service requirements of these intelligent sensors.Due to a lack of instance pricing and revenue-maximizing methodologies, it is difficult for service providers to make a comprehensive profit from such integration. When related expenses and allowances are deducted from revenue, the problem becomes even worse. The stringent revenue maximization goal of providers, on the other hand, has an impact on the budget of the user and the service quality of the system. To address these issues, we offer a profit-aware program placement criterion for integrated Fog–Cloud settings.[8] Some sorts of applications that aren't well suited to cloud computing, such as those that need minimal and expected delay or are spatially dispersed, can benefit from fog systems. Computing in the fog, is, on either hand, not a substitute for cloud technology, but rather a valuable addition. In the framework of the Internet of Things, The interaction and collaboration between both the cloud system and the fog system are the subjects of this study. [9] The large number of Sensor networks that are wireless devices necessitate dispersed analysis and signal processing.Organizing tasks in a logical fashion, that involves allocating jobs to sensor nodes and determining their implementation and communication order, is among the toughest tasks in enabling cooperative processing in wireless sensor networks.Because wireless nodes have limited resources, particularly in terms of energy, one of the most essential concerns In scheduling tasks in such conditions, the goal is to reduce the amount of energy used to maximize the system's operational lifetime. [10]Static scheduling of the program expressed by a directed acyclic graph on a multiprocessing system to reduce the program finishing time is a well-known difficulty in parallel computing.Because determining an ideal program is an NP-hard task in usually, the focus of researchers has shifted to the development of effective heuristics.Branch-and-bound, mixed-integer, searching, graph theory, randomized, evolutionary computation, and evolutionary techniques are just a few of the heuristic algorithms that have evolved using a variety of methodologies, algorithms, and their features, and it is to evaluate the corresponding benefits in terms of time complexity and performance.

System Models

Multiobjective Optimization Problem concept

A multi-objective optimization issue entails more than one conflicting goal, and thus necessitates a desirable trade-off between them. An equation can be used to define the problem.

min$z$= g(r)=$g_1(r)g_2(r)\ldots\ldots g_p(r)$ (1)

$r \in T^q$ is a q-dimensional decision vector, $z \in T^p$ is the vector objective.
The objective vector includes p individuals. If we are considering When there are many objectives, the sorting strategy that is used to solve the grade of solutions must be adjusted in comparison to when there is only one. To evaluate and rank the solutions, the non-dominated sorting approach is utilized. The best Pareto solutions are shown below.

Solution by the Pareto dominance$y_1$is dominated byy_2. It is denoted as

$(y_1 > y_2)$
a)$g_i(y_1) \leq g_i(y_2), \forall i \in \{1,2,\ldots p\}$
b)$g_i(y_1) < g_i(y_2), \exists\, i \in \{1,2,\ldots\ldots p\}$

If x is not dominated by any other solution, y is considered the optimal Pareto solution. The solution set containing all Pareto optimal solutions is called the optimal Pareto set. This set is obtained by a certain algorithm called the archive set. C metric[8]is a programming language that is used to handle multiobjective algorithms. The dominant connection between these solutions is shown in the employment of CM to care for A1 and A2.

$$c(A_1, A_2) = |\{y_2 \in A_2 | \exists\, y_1 \in A_1,\ y_2 < y_1\ or\ y_2 = y_1\}|/|A_2| \quad (2)$$

C(A1, A2) represents the proportion of solutions in A2 which are dominated by solutions in A1. C(A1, A2) & C(A2, A1) are used to compare the Pareto predominance of the archives set in principle. In the Pareto optimum metric, A1 is greater than A2 since C(A1, A2) is larger and C(A2, A1) is smaller.

Three-tier Architecture

In this assignment, the IoT system has a three-tier architecture model. This model consists of edge layer, fog, and cloud layers. The edge layer is placed on the bottom level. The IoT applications are placed in the things layer and are also capable of original data from the scope of control being collected and processed before being sent for further processing. The IoT applications in this layer are categorized into the various group's nearby places. The devices in this same cluster are supposed to be fully connected. The components in the edge layers are powered only by the battery. In our system, the next tier is the fog layer which is placed in the center of our system model. The fog tier is responsible for processing the calculation parts near the data centers. The IoT applications such as cameras, smartwatches, smart phones, lighting systems, and sensors are stated in the fog layer. Every network in fog layers is directly in connection with a cluster of objects that are placed in the edge layer. Additionally, the networks in the fog layers are completely in connection with other networks. Computation tasks are received from the things tier, and the fog layer shares tasks with the other nodes in this arrangement. The last tier of our architecture is in the cloud layers and fining. This tier's units are designed to be cloud nodes, which are typically housed in data centers. The fog nodes are all connected to the cloud nodes via a wide-area network (WAN), and the cloud nodes are all completely connected. When compared to fog nodes, cloud networks are more dependable and significant.

Model of the Application

Generally, all Internet of Things devices are known as Directed Acyclic Graph and the DAG can be represented as four parts <v, e, w, d>, in a certain application v denotes the tasks are not in contact, including computing In an IoT application, the function is important. Task dependence is e and communication workload and the non-connection tasks are w denotes non-communication task workload, d denotes the communication overhead. The pair of task i and j, d(l,j) indicates the

size of the data$e_{i,j}$. In DAG, after the ancestors of its tasks have been completed and received all of their input the remaining tasks will be executed. The tasks are only scheduled with precedence constraints. Additionally, once the tasks are initiated, they cannot be interrupted until all tasks are processed because the tasks are supposed to be non-preemptive. Two connected tasks (i,j) with communication time depend on the bandwidth B($m_i m_j$) related with corresponding nodes($m_i m_j$),($m_i \neq m_j$). In addition, the communication to computation ratio(CR) denotes the potential effect of connection delay on performances and it is determined as is divided by the average calculation time to get the average communication time. The higher the connection to computation ratio (CR), the more likely it is that the application will be communicated in detail. Generally, the sensor tool tasks of IoT applications are started from the things tier, allowing the DAG entry tasks to be processed on the edge layer. For the first job, we refer to the tasks that are not linked to parent tasks and are tasked with collecting raw data. As a result, all entry tasks are assigned to the Internet of things applications from the same group, but this situation is unlikely to last long.

Power Consumption model

To examine the power usage of the full three-layer IoT network, we must first define and summarise the major energy consumption drivers at each tier. To be clear, the energy usage of the cloud layer is difficult to predict because it is influenced by a variety of factors, including the design and operation of a cooling system, as well as the cloud services supplied by various suppliers. Furthermore, our system seeks to improve client time and power consumption performances. The computing in the cloud system is provided on a expect to be paid basis, and the power usage of cloud centers is difficult to reduce from the user perspective.

Table 1
Numerical notations of the Variables

	Representation	significance
Variables in statement	$y_{j,r,i}$	{0,1}$y_{j,r,i}$ = 1represent job k is qth job is integrated on node j, else $y_{j,r,i} = 0$
	$com_{r,i}$	$com_{r,i} \geq 0$ the time it takes for the qth task on the node to complete i,r=1,.....n
	g_j	{1,g} node identification of job j
	t	Amount of tasks in a single application
	g	The number of Connected devices, fog nodes, and cloud nodes in the three tiers refers to the number of Connected devices, fog nodes, and cloud nodes, respectively.
	$B(g_i, g_j)$	Bandwidth between node g_{i,g_j}
	cp_j	Bandwidth among networkg_{i,g_j}

Problem variables constraints	ps_i	Calculation time of job j
	Da(i,j)	Node i is processing speed and $\{ps_t, ps_f, ps_c\}$
	$i \rightarrow j$	Size of inter-task data between tasks I and j
Intermediate variables	$\delta(g_i, g_j)$	In the DAG, there is a route from I to j, and task I precedes task j.
	ES(i)	0 and 1 $\delta(g_i, g_j) = 1$ denotes $g_i \neq g_j$ and $\delta g_i, g_j = 0$ $denotes$ $g_i = g_j$
		Starting time of task i

As a result, the energy consumption of cloud data centers is not taken into account. In addition, IoT devices in the objects tier are often outfitted with non-rechargeable, limited-use batteries. As a result, reducing the power usage of IoT devices is critical to prolong the lifetime of the system, as a single IoT device failure can outcome in system death. In such systems, fog nodes are also placed close to the users. These nodes' energy usage could have a negative influence on residential regions. As a reason, the remaining of this research will concentrate solely on the power consumption of IoT applications and fog nodes. Communication and computing components account for the majority of the energy usage of IoT devices on the objects tier. IoT device energy consumption is divided into two parts: sending information and receiving data, and may be calculated from the following:

$$Energy_t(S,d) = Energy_{el} \times S + \varepsilon_{am} \times d \times e^y \quad (3)$$

$$Energy_r(S,d) = Energy_{el} \times d \quad (4)$$

D represents the transferred data magnitude and l denotes the Euclidean distance among the senders and receivers. E_{amp}andε_{amp} are energy dissipation ratio and transmitting amplifiers' energy dissipation, respectively, are equipment parameters. and x is set anywhere between two and four. For simplicity, we set x to 2 in this study. The energy consumed to process a data unit is indicated in the equation below, the task is largely related to the voltage required by the processors and is dependent on the time required to process a data unit of a task.

$$Energy_{co}(U_d, g) = MCU_d^2 + U_d \times \left(I_0 e^{U_d/vt}\right) \times \frac{M}{g} \quad (5)$$

$$g \approx L \times (U_d - c)$$

VT stands for voltage thermal, whereas C, n, K, and c represents processor-dependent factors. Finally, an IoT device's energy consumption can be calculated as follows:

$$Energy_{Things} = Energy_t + Energy_r + Energy_{co} \quad (6)$$

Both static and dynamic energy utilization affect fog node energy consumption. Static energy consumption refers to the fog nodes' basic energy consumption, whereas dynamic energy consumption is mostly determined by resource utilization. A fog node's energy consumption can be calculated as follows:

$$Energy_{Fog} = Energy_{Fog,s} + Energy_{Fog,d}$$

$$= Pow_{Fog,s} \times ti_{Fog} + \left(Pow_{Fog,max} - Pow_{Fog,s}\right) \times ti_{Fog} \times Ut_{fog} (7)$$

where $Pow_{Fog,s}$and $Pow_{Fog,max}$ denote the fog node's static and maximum power, respectively, and ti_{Fog}denotes the fog node's total processing time.

Problem Statement

The task scheduling is being used to decide the task graph, which is divided into two parts: task node allocation and task processing permutation. These two parts are used to minimize both the application's execution time, such as makespan, and the power consumption of fog nodes, as well as to extend the system's lifetime. The representation of math models is given in Table1 based on the models described previously. The designed goal is listed below**.**

$\min(D_{max}, Energy_{fog})$(8)

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

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

$$\sum_{j1=1}^{n} y_{j,r,i} \geq \sum_{j2=1}^{n} y_{j2,r+1,i} \forall r \leq n-1, i \qquad (11)$$

$$ES(j) = \{max_j \left\{ES(i) + \delta\left(m_i, m_j\right).\frac{Ds(i,j)}{Ba\left(m_{i,}m_j\right)}\right\}, \forall i \rightarrow j \qquad (12)$$

$$D_{r,i} \geq \max\{D_{r-1.i} + \sum_j (y_{j,r,i}.ES(j))\} + \sum_j (d_{j,r,i}.cp_j/ps_{mj}) \ \forall i, r \geq 1 \qquad (13)$$

$$D_{max} \geq D_{n,i}, \forall i \qquad (14)$$

$$Energy_{fog} = pow_{fog,s}.m_f.D_{max}\left(pow_{fog,max} - pow_{fog,s}\right)\sum_j \frac{w_j}{v_{mj}} \ \forall m_j \in fog \qquad (15)$$

Examination of the Lower BoundsAnalysis

The lower bound analysis is used to solve the task graph scheduling problem. DAG represents each as well as every IoT application. The Graph is represented as g = (v, e, w, d) a route from a node$u_0 and\ u_k$to in g is sequence$(v_0, v_1 \ldots\ldots v_k)$ of vertices connected by the edges $(u_0, u_{1,} \ldots\ldots u_k)$where x is the path's length, which is determined by adding the weight of its nodes.

Length(x) = $\sum_{N \in x.v} w_n + \sum_{e \in X} D(x)$ (16)

The computation length of the path is represented as x

$$Length_w(x) = \sum_{n \in x,v} W_n \ (17)$$

The critical path is represented as cp is the longest path in g. The computational critical path is represented as cp_w

Length(cp) = $max_{x \in g}${Length(x)} (18)

If homogeneous devices with processor speed v are assigned to the job graph, we can deduce that $C_{max} \geq max_{x \in g}(CP_w)/v$. The IoT system discussed in this paper, on the other hand, is a dispersed heterogeneous system with nodes with varying capabilities.$v_t, v_f, v_c$and are having three different processing speeds. Furthermore, the entry to the things layer is where the application's tasks are processed. Task 0 is the diagram's virtual entry node, with really no execution time or communication overhead. The real entering jobs are the successors to job 0, and they must be processed on the items layer. We also have

$$cp_w = \{0,1, \ldots., k-1, k, \ldots l, l+1, \ldots n+1\}$$

$$2 \leq k \leq nc_p + 1 \{ \min \sum_{j=1}^{k-1} \frac{w_j}{v_t} + \frac{D(k-1,k)}{B1} + \sum_{j=k}^{l} \frac{w_j}{v_f} + \frac{D(l,l+1)}{B2 \sum_{j=i+1}^{nc_p}} \frac{w_j}{v_c} \} \quad (19)$$

K-1 $\leq l \leq n_{cp}$

Process 0 is the diagram's virtual entry node, with no execution or communication overhead. 0 is successors are the true entrance tasks. and they must be allocated to the items layer for processing. We also have

$$cp_w = \{0,1, \ldots., k-1, k, \ldots l, l+1, \ldots n+1\}$$

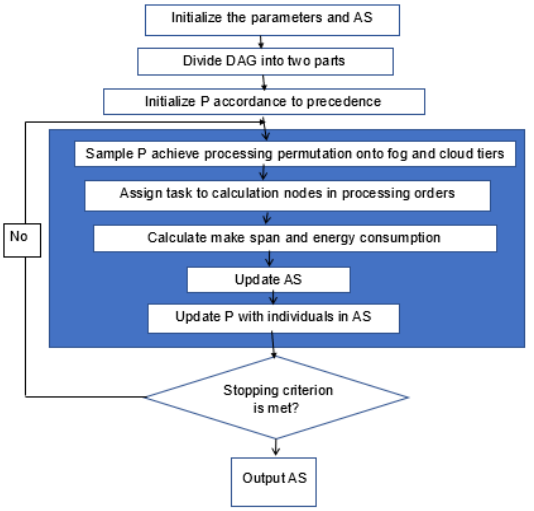
in this diagram, and there are total nc_p tasks in cp_w. An examination of lower bound analysis is given as follows, taking into account the system's conditions. where B1 is the bandwidth among the things tier and the fog tier, and B2 is the bandwidth among the edge layer and the fog layer. The fog tier and the cloud tier are two different types of tiers. And (k- 1, k) has been identified as cmm1 the cut line among the edge layer and the fog layer is cmm1 tier. We need to assign entry nodes to IoT devices because we need to assign entry nodes to IoT devices that have $K \geq 2$. (l,l+1) the cut edge is designated as cmm2 among the cloud system and the fog system. We may deduce that $l \geq k-1$. The fog system is related to both the edge layer and the cloud layer. Tasks before task k are applied to the edge layer, while tasks ending in (k, l) and $l, nc_p$are assigned to the fog system and cloud layers, respectively. The virtual exit vertex $nc_p + 1$ is used to unify the cases so that we may express all eventualities with just one equation. In Fig. 2, two cuts are shown, where a cut on the edge $nc_p, nc_p + 1$ signifies that all prior jobs are sent to the equal layer for processing. To put it another way, k=l+1 = nc_{p+1} implies that all $c_p$jobs are allocated to the edge layer; k=l+1 <nc_p means that all jobs after job k are uploaded to the cloud layer through the fog layer.

Methodology

Flowchart

This section introduces the proposed strategy, which aims to strike the right balance between energy savings and application completion time. The proposed algorithm is first described in general. The partition operator is then developed, along with the fictitious code, which we used in our modified EDA technique.

Following that, both regular techniques for EDA and our suggested evolutionary algorithms with the partitioning operator (EDA-p) are shown.



Partition Operator

A partition operator divides the DAG into two pieces to allocate the right chunk of every Internet of Things device to the edge layer. The DAG is split into two parts: one is allocated to the objects tiers and handled locally, and the other is forwarded to the fog system and cloud system for additional processing. The authors of [17]Tasks with high computational workloads and minimal connection overhead are uploaded to fog nodes and cloud nodes for processing, resulting in improved system performance and energy savings, according to a survey. It's critical to keep the amount of data sent between the objects and fog tiers to a minimum. However, data transmitting and collecting procedures use a significant level of power for these non-rechargeable IoT devices. As a result, minimizing data transfer among Internet of Things applications is similarly vital in this work, as the power depletion of IoT devices can have a negative impact on system performance and reduce the system lifetime. Furthermore, the calculation capacity of connected devices is inferior to that of fog and cloud nodes. When a significant amount of the duties is delegated to connected devices for machining, the application's completion time is likely to be significantly boosted. We create a simple processor model that comprises m_sBy integrating these parts, you can create a digital compute node that represents the computing resources that can be used both from fog and cloud tiers. This virtual computing node's capability is expressed as $v_f^{'} = v_f X m_f$. Our partition operator can use this model to determine. how to partition a DAG into two portions and effectively distribute relevant jobs into (ms+ 1) nodes, notably for things tier tasks This approach may be expanded to a situation with multiple applications with some tweaking.

Table 2
The Representations Used in the Algorithm

Notation	definition
M	The free to set off the Connected systems in the group and the virtualized computation node is represented by the machine time vector length.
C	The length of a completion time is 2 and represents the time it takes for a task to be completed in the edge stage and virtual computation node, respectively.
B	Battery energy The energy consumption on an IoT device induced by a certain job when processed in the IoT layer and virtual computation network is represented by a usage length of 2.
In(k)	$\sum_{u \in pre(k)} Ds(u,k)$
Ot(k)	$\sum_{v \in suc(k)} Ds(k,v)$
ES	For each task in the tier of the item and virtual computational, the earliest starting time vector
Send(Ds)	Transferring data-sized ds consumes a lot of battery power.
Receive(Ds)	Receiving data with a size of ds consumes a lot of battery power.
Process(Ds)	Processing job k uses a lot of battery power.

When a new application enters, the virtual computing node may not be accessible to handle extra tasks, as it was in the single application instance. When the graph partitioning is finished, the estimated available time and update the virtual computation node. The freshly arriving jobs are more usually to be given to connected devices before the virtual computing node the available for The greedy stochastic method is used to create algorithms. While this method may minimize the makespan and power consumption of fog nodes, it drains the battery life of IoT devices, shortening their lifetime. Let's start with some definitions before we go into the details of our partition operator. We start with a Pt: V-> {1, 2} mapping, Pt(k) = 1 indicates a job assigned to the items tier, while Pt(k) = 2 denotes an assigned task to the virtualized computation node. The boundary vertices of the task graph are represented as

$$\gamma: \{v \mid \exists (v, w) \in Edge: P(v) \neq P(w)\} \quad (20)$$

Also, pr(k) indicates the tasks set,

{u|(u,k)∈$Edge$}, su(i) I
γ: {v|∃(v, w) ∈ Edge: P(v) ≠ P(w)}ndicates task set. , {w|(i,w)∈$Edge$}
Property: For any of the node i in the DAG ,∀u∈$pr(i)$denotes task set. P(v)=1 and ∀v∈su(k). P (w)=2, if ∑v∈$pr(i)$ $T(v, i)$ < ∑w∈$su(i)$ $T(i, w)$.
∑(u,v)∈C,$Pt(k)$=1 $T(u, v)$ > ∑(u,v)∈C,$Pt(k)$=2 $T(u, v)$ (21)
Here, D: ={(v,w) ∈$Egde$| P(v)≠ $P(w)$}
Proof: It is not hard to see that k∈γ , Pt(k)=1 and ∃ (k, v) ∈$Edge$.
V ∈$succ(k)$and Pt(k) ≠ $Pt(v)$. When Pt(k)=2 then $\sum_{(u,v)\in C} D(u, v)$ is affected by Pt(k).

$$\sum_{v,w\in D,p(i)=1} T(v,w) = \sum_{(v,w)\in D(i,w)} T(v,w) + \sum_{w\in su(i)} T(i,w)$$

>$\sum_{(v,w)\in D(v,i)} T(v,w) + \sum_{v\in pr(i)} T(v,i)$
=$\sum_{v,w\in D,p(i)=2} T(v,w)$

Based on this feature, a criterion is created to determine not whether a job should be assigned to the virtual computation device. If a job meets the requirements$\sum_{v\in pr(i)}$ $T(v, i)$ < $\sum_{w\in su(i)} T(i, w)$ T(v, I wsu(i) T(i, w), it is assigned to the virtual computation node, which reduces the energy consumption caused by IoT device communication. Another criterion is intended to speed up task processing and minimize battery power consumption, as well as lessen the theoretical difficulty of the partitioning operator.

$P(v) \le P(w), \forall v \to w$ (22)

As a result, if the prior criterion is not used, all tasks with predecessors assigned to the virtual node would have the same allotment. In our technique, the value of each job is bottom level (b-level) is derived, where b-level is the longest from the current work to the exit task. In descending order of b-level, then selecting criteria apply to each task. As previously indicated, certain jobs are assigned directly to the virtual node, while others must be matched in terms of energy consumption and execution time. After all of the jobs have been assessed, the partition plan is defined.

Algorithm 1. Procedure of Partition Operator

For each task i in descending b-level list
If i∈ entry tasks
i is assigned to the IoT device with earliest available time;
Else If in(i) < out(i) or for any x ∈pre(i), there is x ∈ Fog Tier
i is assigned to virtual computation node;

Else
Find IoT devices with earliest available time;
C[0] :=**max**(M[s],task.ES[0]) + t_i/u_j ;

C[1] :=**max**(M[Fog],task.ES[1]) + t_i/v'_g;
Let **dout**(i) := $\sum_{v \in pre(i)} D(v,i) - \sum_{vasssignedonIoTdevices} D(v,i)$
B[Q] :=**Send**(**dout**(i))+**Received**(**dout**(i));
B[0] + =**Process**(i);
B[1] := **Send**(input(i));
IfC[0] < C[1] **and** B[0] < B[1]
k is assigned to IoT devices;
Else If C[0] > C[1] **and** B[0] > B[1]
i is assigned to the virtual node;
Else
If (B[0]-B[1])/ (B[0]+B[1]) + (C[0]-C[1])/C[0]+C[1]) > 0
i is assigned to the virtual node;
Else
i is assigned to the Things Tier;

END
Update M and Es of each task in suc(task)
END

EDA Scheme

Once the DAG has already been partitioned, the jobs are separated into two halves. When the partitioning operator is used, the assigned task to the things layer is sorted in b-level decreasing order. The workloads given to the virtual computing node are sorted using our suggested EDA sampling technique. Following the identification of these two processing sequences, tasks will be assigned to nodes based on the rule of earliest finish time first (EFTF). After that, the duration and energy usage of each application may be estimated. In this project, EDA is used to define the accordance with this standard for the jobs given to the fog and cloud layers. The traditional EDA approach is summarised here. To start, a probability model P is developed, and then people are sampled using P. The produced persons are next evaluated, and P is modified based on the outcomes of the examinations. After then, the individuals are sampled using the amended P value until the stopping limit is met. The details of our suggested EDA for this schedule challenge are as follows..

Things tier	i_1	i_2	....	i_l	....	i_t
Fog&Cloud Tier	j_1	j_2	....	j_k	....	j_f

Decoding and Encoding: Individuals are encoded in the manner illustrated in Figure 4, where T represents the number of tasks given to the objects tier and F represents the number of tasks assigned to the fog and cloud tiers. The processing sequence for the first row, which reflects the tasks assigned to the tier of the item, was chosen by the partitioning operator. The second sequence is generated by our proposed EDA, and it illustrates the processing sequence of the jobs assigned to the fog and cloud tiers. For workloads allocated to the fog and cloud layers, the compute nodes in both tiers are treated as unrelated processors, with EFTF deciding the processor assignment. In the other words, the jobs are sent to the

processing who can do them in the lowest period. When persons are assessed, the duties in the tier of the item are originally considered and prescribed schedule (decoded). The jobs are subsequently allocated to the fog and cloud tiers depending on the processing permutations provided by EDA.

Probability distribution: A probability model for task permutation P is formed based on the relative position connection between the jobs, taking into consideration the precedence limitations between the activities assigned to the fog and cloud tiers. P generates a permutation of fog and cloud job processing. (23), where pi;jg signifies the likelihood of task I will be put ahead of task j inside the permutations at the g-th iteration of evolution. Clearly, for $q_{j,k}(f) + q_{k,j}(f) = 1$ $task$ j and k that are ahead of or behind one other.

$Q(f) = \{q_{j,k}(f)\}_{S\times S}$(23)

Initialization of the probability model: In order to meet the precedence requirements of task pairings, the significance level q (j,k) is initialised for each given DAG (j,k). With a chance of 0.5, the order of task pairs with no restrictions is generated at random.

$q_{j,k}(0) = \{ 1, j \to k$ 0, k$\to j$
0.5, otherwise. (24)

Method of updating: For the single-objective optimization issue, the solutions created by EDA can be assessed by a single goal, such as makespan, and people with less makespan are used to update P. (g). Because AS is the set of elite answers under the Pareto ranking criterion, individuals in AS are utilized to modify P in the multi-objective optimization problem. The probability value is then adjusted using the population-based incremental learning approach (PBIL)

$q_{j,k}(f+1) = (1-\alpha)q_{j,k}(f) + \alpha \times \frac{\sum_l H_{j,k}^l(f)}{|A|}$(25)

$H_{j,k}^l(f) = \{1$, task j is before task k in the lth individual
0 , otherwise (26)

Where $\alpha \in (0,1)$ is the learning rate, |A| j is the archive set size, and $H_{j,k}^l(f)$is the indicator function corresponding to AS l-th answers.

Method of sampling: EDA generates new people by selecting an upgraded probability model. For each undetermined point in the fog and cloud processing permutation, a probability array is constructed. The array member represents the possibility that a given job will be assigned to this location. The product stated in pseudo code represents the chance that a job will lag after all other jobs. The array is then adjusted before being tested using the roulette wheel application.

Algorithm 2. Procedure of EDA Sampling Method

T is the available task set, initialized with tasks without parent tasks in fog and cloud tier;

pe is the label in task processing permutation, pe=0;

While T is not empty

Calculate a sampling probability s for each task in T;

For each task i in T
s(i) : = $\pi_{j\neq pe\ ,j\in T} p_{i.j}(f)$;
End
Normalize probability array s;
Sample s with Roulette Wheel Method to achieve the pe-th processing task i in the permutation;
Push the child task of i with all parents assigned into T;
pe + +;
End

Simulation

To evaluate its performance, we compare our proposed EDA-p to the specified algorithms, the heuristics approach, and the multi-objective EDA. For a fair comparison, the algorithms all are written in C++ and run on some computers.

Algorithm Comparison

To evaluate our proposed method, two benchmark algorithms are utilized, both of which are run in the same context as our algorithm. A heuristic used in single objective task network allocation optimization problems is the b-level heuristic technique. In the b-level heuristic technique, all jobs are ordered in decreasing order of b-level values. According to the EFTF rule, the jobs are then spread over the compute nodes, which include object, fog, and cloud nodes. The b-level heuristic approach generates a single objective, makespan, which is compared to the prior lower bound. In some circumstances, such as when n = 50 and CR = 0.1 and 0.5, the minimal relative percentage variation (RPD) is 0.39 and 0.17, correspondingly. We may conclude that the b-level heuristic strategy works well when CR is much less than 1.0 and that the lower limit is tight when the communication overhead between processes is modest. The estimation of the lower bound becomes less exact as the CR increases (when n =50; CR = 10, the lowest RPD is 127.70). Jobs that are not on the critical path are excluded from the lower bound.

Table 3
Factor Levels of Parameters

Parameters	Factor Level			
	1	2	3	4
N	25	30	35	40
α	0.057	0.072	0.10	0.135

Because of the large communication overhead, the scheduling strategy has a lot of idles, and the lower limit isn't as tight as it is when CR is low. The alternative method, known as simple EDA (sEDA), is a multi-objective strategy for developing a system-wide scheduling scheme. Unlike our suggested EDA-p, sEDA does not employ a partition operator. To construct a full completion of a given task permutation of fixed length, sEDA employs a probability model. Tasks are assigned to all three tiers following the EFTF rule. The probability model is then updated

using individuals in AS. The purpose of using sEDA would be to determine whether or not the partition operator's participation is effective**.**

Simulation Environment

Variables and testing environment settings are set to the values specified in [14]. The values for the parameters indicated in Equation (5) are as follows: E elec = 50nJ/b, amp = 10pJ/b/m2, VT = 26mV, C = 0.67nF; I o = 1.196mA, n = 21.26, K = 239.28 MHz/V, V dd = 1V, c =0.5V The processing speeds of IoT devices, fog nodes, and cloud nodes are 1, 5, and 10, respectively. The bandwidths within each layer are 1, 5, and 10, while the bandwidths between the objects tier and the fog tier, fog tier, and cloud tier are 1 and 8, respectively. Data that is uploaded to the cloud center must be sent through the Internet. . Because there is no universal mathematical equation for measuring transmission delay via the Internet, a fixed communication delay of 0.1s is introduced when transferring a generic unit of a communication job between the fog and cloud tiers. The typical distance between IoT devices is one meter, with a beginning battery power of 0.1J. To construct the things layer, various numbers of IoT devices (3, 3, 3, 4, 4, 4) are arranged in appropriate clusters. Six fog nodes are strategically placed to link to their respective clusters. This system takes into account two public cloud services. During the analysis of a single application case, IoT devices in a single cluster construct the application for each instance. Communication time and power consumption are much greater than when jobs are transferred to fog and cloud nodes for further processing because communication between object clusters must be managed by suitable fog nodes. As a consequence, the compute workloads that were previously transmitted across clusters are no longer required.

Single Application Study

The single application benchmark is provided by the standard task graph benchmark examples on the homogeneous DAG scheduling issue. It does not contain the inter-task data size, but the values of communication time contributing to normal distribution based on different CR settings are generated and utilized for testing, as in [13]. The test sets are n =50, 100, 300, and CR =0.1, 0.5, 1.0, 10.0, with each test set including 180 independent testing occurrences.As a consequence, 3 × 4 x 180=2160 examples are used to assess the effectiveness of the proposed method.

Table 5
Influence Trend Table

Level	N	α
1	0.099	0.035
2	0.19	0.075
3	0.085	0.035
4	0.199	0.055

Main Effects Plot

Average value

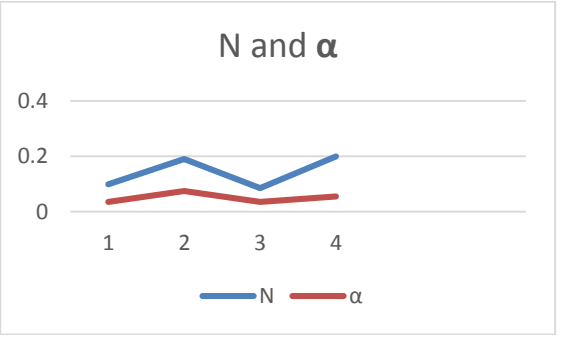


Fig.5.Effect trends of Parameters

Experiment with various choices. To compare their performance objectively, the halting threshold for both algorithms has been set to 30 generations. All of the EDA parameters are set to the same value in both sEDA and EDA-p. Because the probability model is updated by the whole AS, the fraction of the elite population does not require attention. The Taguchi technique of the design-of-experiment method (DOE) is used to calculate the other two major parameters of EDA, population size (N) and learning rate (). DOE is a statistical approach for investigating the influence of factors on algorithm performance at various levels. The experiment is conducted numerous times using orthogonal arrays with varying parameter settings. For each combination of n x CR (3 x 4 = 12), a random instance is picked. For each instance, 16 distinct N combinations are attempted separately, and the results are recorded as AS ci(ci = 1, 2,...16). It is best, according to Figure 5 and Table 5, to set N to a number that is neither too high nor too little.

Table 6
Results Compared with B-Level Heuristic Method

Task Number (n)	CR	Dominating level(%)	Non Dominating b-level(%)	Dominated b-b-level(%)
50	0.09	56.78	45.22	3.78
	0.45	89.89	25.56	1.48
	0.96	89.23	3.25	2.12
	10	92.56	1.45	3.88
100	0.9	51.56	22.89	26.56
	0.44	94.5	4	1.11
	1	98.32	0.23	0.26

	10.5	89.59	0.78	0.89
300	0.12	15	17.89	59.59
	0.51	78.56	6.45	1.56
	1	96.78	2.58	0
	10	90.89	0.59	1.32

A random instance is chosen for each combination of n x CR (3 x 4 = 12), For each instance, 16 different $N \times \alpha$ combinations are tried individually, and the results are saved as AS_{ci}(ci = 1, 2,...16). By combining $AS_1, AS_2, \ldots . AS$ the final AS (FAS) is obtained.

Table 7
Comparative Results of C Metric and Simple EDA

Task number(n)	CR	C(EDA-p,sEDA)	C(sEDA,EDA-p)
50	0.09	0.50	0.59
	0.45	0.6	0.20
	0.96	0.89	0.12
	10	0.59	0.23
100	0.9	0.26	0.15
	0.44	0.59	0.12
	1	0.84	0.03
	10.5	0.59	0.05
300	0.12	0.06	0.59
	0.51	0.95	0.05
	1	0.89	0.02
	10	0.84	0.01

The contribution of a particular combination of CON is then counted as CON = $\frac{|A'_{ci}|}{|FE|}$, Where $A'_{ci} = \{Y_1 \in A'_{ci} | \exists\, Y_{l'} \in FAS, Y_1 = X_{l'}$. Each combination's average CON is determined, and Table 4 shows the response value (RV) that was employed. Table 4 lists the DOE response values. Table 5 illustrates that the changes in response value under different values of N and an are almost equal, meaning that the impact of N on algorithm performance is substantially identical. Because the stopping requirement is 30 generations, the benchmark instances should employ a tiny an. According to the results, the simulation experiment settings should be N =30; =0.05. Various heuristic techniques are compared. Because the b-level heuristic technique generates one solution for each problem, our algorithm is compared to heuristic approaches in terms of the percentage of heuristic solutions dominated by our algorithm's solutions. In Table 6, the column "dominating b-level(percent)" indicates the proportion of times the b-level heuristic solution is dominated by any solution in the EDA-p. The percentage of times the b-level heuristic solution is not dominated by any EDA-p solution and the EDA-p solution is dominated by the b-level solution is referred to as "non-dominating b-level(percent)." The term

"dominated by the b-level(percent)" indicates that the b-level solution dominates at least one EDA-p solution. Table 6 computes and summarises the average value for each problem scale based on 180 examples. Table 6 indicates that in the great majority of circumstances, EDA-p beats the b-level heuristic. When CR exceeds or equals 1.0, the fraction of instances dominating the b-level solution increases. When CR is larger than or equal to 1.0, the Pareto set generated by EDA-p dominates more than 90% of the solutions generated by the b-level heuristic approach. As a result, it's possible to deduce that when CR ≥0.5, EDA-p outperforms the b-level heuristic. When the CR value is 0.1, however, EDA-p is unable to outperform the b-level heuristic technique. The fact that the amount of data transmitted is constrained may explain why the b-level technique performs effectively when CR is low As a consequence, when CR = 0.5, EDA-p beats the b-level heuristic. However, when the CR value is 0.1, EDA-p fails to beat the b-level heuristic approach. The fact that the amount of data conveyed is limited might explain why the b-level approach works well when the CR is low. As a result, it has little effect on the scheduling scheme, and the greedy EFTF rules employed in the b-level method perform similarly to the best makespan solution. When the CR grows, the greedy EFTF rule selects the processor with the earliest finish time. Because data transfers between tasks allocated to the things tier and their successors spend more time uploading intermediate results rather than processing the data locally, the greedy EFTF rules assign a significant number of tasks to the things tier. The objects layer, on the other hand, has a slow processing speed that may result in scheduling delays. Furthermore, when IoT devices are overburdened and tasks must be uploaded, data transfers may be increased. A straightforward EDA is compared. We compared our technique to a standard EDA in terms of C metrics. Table 6 summarises the average C metric values of 180 examples under each scale, where C(EDA-p, sEDA) denotes the percentage of simple EDA solutions that are dominated by or identical to the EDA solutions given in this paper. Table 7 shows that when CR = 0.1, the performance of EDA-p is inferior to that of simple EDA. The reason for this is that in simple EDA, the probability model creates the entire processing permutation and explores a larger solution space, whereas in EDA-p, the processor assignment of a subset of tasks is fixed and the solution space is narrower. When CR =0.1, sEDA can produce superior solutions in this approach. Under other CR values, however, EDA-p outperforms sEDA significantly. Because sEDA is the foundation of EDA-p except for the partitioning operator, it is clear that pre-partitioning the task graph is required, and this operator is effective in improving the scheduling performance of this problem in the vast majority of cases.

Multiple Application

A discrete-event simulator created in the C programming language is used to evaluate the proposed approach in a realistic environment. The simulations examine three performance metrics: system lifespan, makespan, and fog tier energy usage. "System lifetime" refers to the amount of time it takes for the first IoT device to run out of power [13], [14]. When the system fails, the lifespan of the system is recorded, and the simulation is kept running to analyze the other two metrics to compare the methodologies fairly. The total energy consumption of the fog node is computed using Equation, and the makespan is approximated using the time it takes for all apps to finish their final job.

To find an appropriate value for b, nine different values of 0.1, 0.2,..., 0.8, 0.9 of b are examined, with three optimization objectives in mind. The findings show that altering b does not influence makespan. To evaluate the proposed approach in a realistic context, a discrete-event simulator written in the C programming language is employed.

Table 8
Average Value Comparison with different CR

	CR	EDA-p	b-level	sEDA
lifetime	0.09	1258.86	389.48	400.34
	0.45	885.98	329.58	585.71
	0.96	785.41	454.17	484.11
	10	687.96	485.47	285.48
makespan	0.9	3250.78	5896.48	4520.59
	0.44	3987.12	1525.48	3553.68
	1	2588.89	1852.15	5550.91
	10.5	4325.9	7670.58	5965.16
energy	0.12	5658.15	2898.48	7857.79
	0.51	5353.89	3684.48	8955.91
	1	5494.58	9758.58	7854.95
	10	5398.88	5285.25	4485.89

The simulations look at three performance indicators: system lifespan, makespan, and fog tier energy utilization. The term "system lifetime" refers to the amount of time it takes for the first IoT device to lose power [14], [15]. When the system fails, the system lifespan is recorded, and the simulation is retained to compare the other two metrics fairly. The equation is used to calculate the total energy consumption of the fog node, and the makespan is calculated based on the time it takes for all of the applications to finish their final job.

Furthermore, the multi-objective algorithms in the multi-application simulation must select one solution from the Pareto set as the final scheduling scheme. The following method is used to minimize the weighted objectives: The Pareto set obtained by an algorithm is denoted by S, and the weighted value is denoted by B.
min B X $D_{max,j} + 1 - B\ X\ Energy_{fog}$ i ∈ S
(27)

To discover an appropriate value for b, nine possible values of 0.1, 0.2,..., 0.8, and 0.9 are investigated, with three optimization goals in mind. According to the data,

changing b does not affect makespan; nevertheless, raising b shortens the system's lifespan while increasing fog energy usage. As a result, in the simulation, b is set to 0.1. Benchmarks are used to compare. To begin, studies with a wide variety of CR values are conducted. For each round of the simulation, 100 random applications are produced and assigned to the system. Table 8 shows the average value of each statistic after 50 simulation cycles for each approach. Table 9 the last column displays the p-values of two-sample t-tests at the 95 percent confidence level. The alternative hypothesis for lifespan is that "the objective lifetime of EDA. p is not larger than the comparison algorithm 95 percent confidence level.

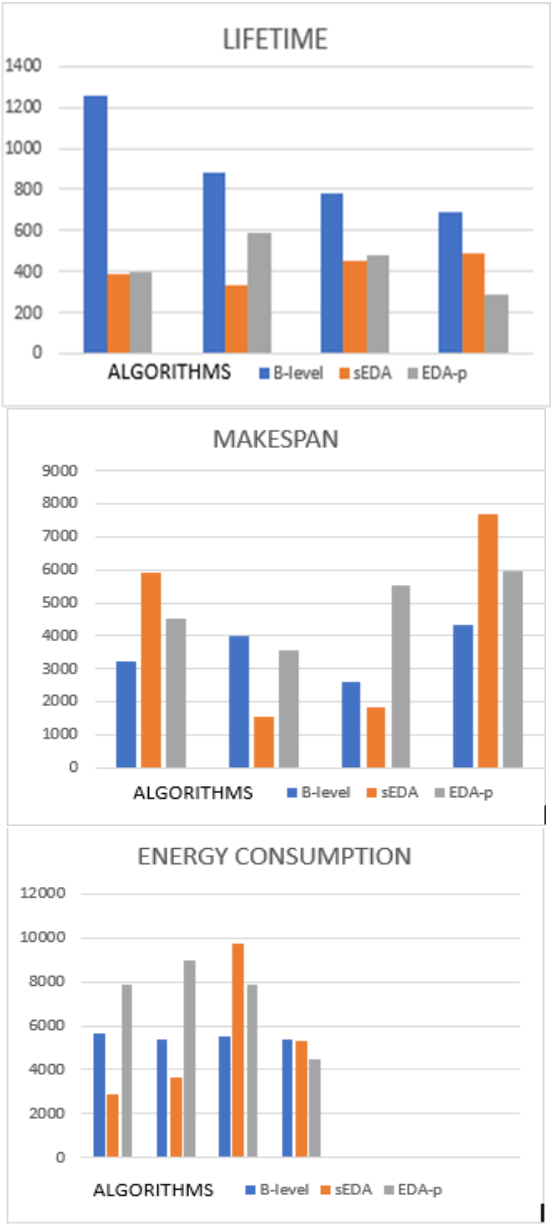


Fig.6. Clustered Column Chart of results comparison

At a 95% confidence level, the longevity and make span of solutions produced by EDA-p are far superior than those offered by other algorithms. Furthermore, when compared to the other method, the null hypothesis for the fog energy consumption goal could not be rejected. Table 9's last column displays the p-values of two sample t-tests at the 95 percent confidence level. The alternative hypothesis for lifespan states, "The objective lifetime of EDA. p is not larger than the comparison algorithm 95 percent confidence level," and "the objective of our proposed EDA is not smaller than the comparison algorithm 95 percent confidence level." The alternative hypothesis for the other two objectives states, "The objective of our proposed EDA is not smaller than the comparison algorithm 95 percent confidence level.Furthermore, when compared to the other method, the null hypothesis for the fog energy consumption goal could not be rejected.

Conclusion

The research develops an energy-efficient evolutionary technique for resource scheduling of three layers of IoT systems. To characterise and outline the problem, we first present a full system and computational model, followed by an attempt to mimic an actual scenario. There is a lower bound provided by the mathematical model. Following that, the partition operator is compared to an EDA scheduling technique. The pre-partition operator mitigates the detrimental effects of the greedy task assignment algorithm. EDA is also used to generate task permutations. Experiments on single and multi-application situations show that our technique is beneficial in reducing device makepan and energy consumption, as well as prolonging IoT device lifespan.

References

1. O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar, "Towards QoS-aware fog service placement," IEEE Fog Edge Comput., May 2017, pp. 89–96, doi: 10.1109/ICFEC.2017.12.
2. Aral, A., and Ovatman, T., "Network-aware embedding of virtual machine clusters into federated cloud infrastructure," J Syst. Softw., vol. 120, no. 1, Oct. 2016, pp. 89–104.
3. F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S. Tucker, "Fog computing may help to save energy in cloud computing," IEEE J. Sel. Areas Commun., vol. 34, no. 5, pp. 1728–1739, Mar. 2016, doi: 10.1109/JSAC.2016.2545559.
4. R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," IEEE Internet Things J., vol. 3, no. 6, pp. 1171–1181, May 2016, doi: 10.1109/JIOT.2016.2565516, IEEE Internet Things J., vol. 3, no. 6, pp. 1171–1181, May 2016, doi: 10.1109/
5. P. Borylo, A. Lason, J. Rzasa, A. Szymanski, and A. Jajszczyk, "Energy-aware fog and cloud interplay facilitated by wide area software definable networking," IEEE Commun., July 2016, pp. 1–7, doi: 10.1109/ICC.2016.7511451.
6. Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, and A. Y. Zomaya, "A dynamic tradeoff data processing framework for delay-sensitive applications in Cloud of Things systems," J. Parallel Distrib. Comput., vol. 112, Feb. 2018, pp. 53–66. Figure 6: Results comparison box plot.

7. B.-B. Li, L. Wang, and B. Liu, "An effective PSO-based hybrid algorithm for multiobjective permutation flow shop scheduling," IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. 38, no. 4 (June 2008), doi:10.1109/TSMCA.2008.923086.
8. "System modelling and performance evaluation of a three-tier cloud of things," Future Generation Computer Systems, vol. 70, no. 1, May 2017, pp. 104–125, by W. Li and colleagues.
9. Proc. 1st Edition MCC Workshop Mobile Cloud Comput., Aug. 2012, pp. 13–16, investigates "fog computing and its function in the internet of things."
10. ACM Comput. Surveys, vol. 31, no. 4, pp. 406–471, Dec. 1999, "Static scheduling strategies for allocating directed task graphs to multiprocessors."
11. S.-Y. Wang and L. Wang, "An estimation of a distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem," IEEE Trans. Syst. Man Cybern.: Syst., vol. 46, no. 1, pp. 139–149, April 2016, doi: 10.1109/TSMC.2015.2416127.
12. "Estimation of the distribution algorithm using multiobjective decomposition-based mallows models," A. Mendiburu, R. Santana, and A. Pozo. M. Zangari, A. Mendiburu, R. Santana, and A. Pozo. Inf. Sci., vol. 397, pp. 137–154, August 2017. "A case study of the permutation flowshop scheduling problem," Inf. Sci., vol. 397, pp. 137–154, August 2017.
13. 630–635, doi: 10.1109/COASE.2017.8256173, in Proc. IEEE Autom. Sci. Eng., Aug. 2017, pp. 630–635, doi: 10.1109/COASE.2017.8256173, in Proc. IEEE Autom. Sci. Eng., Aug. 2017, pp. 630–635, doi: 10.1109/COASE.2017.8256173, in Proc. IEEE Autom. Sci. Eng., Aug. C.-g. Wu, L. Wang, and J.-j. Wang published "A t-level driven search for estimation of distribution technique in addressing task graph allocation to multiprocessors" in Proc. IEEE Autom.
14. J Parallel Distributed Computing, vol. 117, July 2018, pp. 63–72. J Parallel Distrib. Comput., vol. 117, pp. 63–72, July 2018. [14] "A multi-model estimation of distribution technique for energy efficient scheduling under cloud computing system," J Parallel Distrib. Comput., vol. 117, pp. 63–72, July 2018. C.-G. Wu and colleagues published "A multi-model estimation of distribution strategy for energy efficient scheduling under cloud computing." J Parallel Distributed Computing, vol. 117, no. 1, pp. 63–72, July 2018.
15. W. Li et al., "System modelling and performance evaluation of a three-tier cloud of things," Future Generation Computer Systems, vol. 70, no. 1, May 2017, pp. 104–125.