Scientific Workflow Scheduling in Mobile Edge Computing Based on a Discrete Butterfly Optimization Algorithm

Jan Lansky
Department of Computer Science and Mathematics, Faculty of Economic Studies, University of Finance and Administration, Prague
Mokhtar Mohammadi
Department of Information Technology, Lebanese French University, Erbil, Kurdistan Region
Adil Hussein Mohammed
Department of Communication and computer engineering, Cihan university-Erbil , Erbil, Kurdistan region
Sarkhel H.Taher Karim
Computer Department, College of Science, University of Halabja, Halabja
Shima Rashidi
Department of Computer Science, University of Human Development, sulaymaniyah
Amir Masoud Rahmani
Department of computer science, Khazar University, Baku
Mehdi Hosseinzadeh (mehdihosseinzadeh@duytan.edu.vn)
Institute of Research and Development, Duy Tan University, Da Nang 550000

Research Article

Keywords: MEC, Workflow, Optimization, Data-Intensive, Energy.

Posted Date: February 19th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-208986/v1

Scientific Workflow Scheduling in Mobile Edge Computing Based on a Discrete Butterfly Optimization Algorithm

J. Lansky[1], M. Mohammadi[2], A. H. Mohammed[3], S. H. T. Karim[4,5], S. Rashidi[6], A. M. Rahmani[7,8], And M. Hosseinzadeh[9,10*]

[1]Department of Computer Science and Mathematics, Faculty of Economic Studies, University of Finance and Administration, Prague, Czech Republic
[2]Department of Information Technology, Lebanese French University, Erbil, Kurdistan Region, Iraq
[3]Department of Communication and computer engineering, Cihan university-Erbil , Erbil, Kurdistan region, Iraq
[4]Computer Department, College of Science, University of Halabja, Halabja, Iraq
[5]Sulaimani Polytechnic University, Technical College of Informatics, Computer Networks Department, Sulaymaniyah, Iraq
[6]Department of Computer Science, University of Human Development, sulaymaniyah, Iraq
[7]Future Technology Research Center, National Yunlin University of Science and Technology, 123 University Road, Section 3, Douliou, Yunlin 64002, Taiwan, Republic of China
[8]Department of computer science, Khazar University, Baku, Azerbaijan
[9]Institute of Research and Development, Duy Tan University, Da Nang 550000, Vietnam
[10]Mental Health Research Center, Psychosocial Health Research Institute, Iran University of Medical Sciences, Tehran, Iran

ABSTRACT Mobile Edge Computing (MEC) is an interesting technology aimed at providing various processing and storage resources at the edge of the Internet of things (IoT) networks. However, MECs contain limited resources, and they should be managed effectively to improve resource utilization. Workflow scheduling is a process that tries to map the workflow tasks to the most proper set of computing resources regarding some objectives. For this purpose, this paper presents DBOA, a discrete version of the Butterfly Optimization Algorithm (BOA) that applies the Levy flight to improve its convergence speed and prevent the local optima problem. Then, DBOA is applied for DVFS-based data-intensive workflow scheduling and data placement in MEC environments. This scheme also employs the HEFT algorithm's task prioritization method to find the task execution order in the scientific workflows. For evaluating the performance of the proposed scheduling scheme, extensive simulations are conducted on various well-known scientific workflows with different sizes. The obtained experimental results indicate that this method can outperform other algorithms regarding energy consumption, data access overheads, etc.

Keywords: MEC, Workflow, Optimization, Data-Intensive, Energy.

* Corresponding author: M.Hosseinzadeh (e-mail: hosseinzadeh.m@iums.ac.ir).

I.Introduction:

Internet of things (IoT) is a collection of wirelessly interrelated devices, such as home appliances, mobile phones, vehicles, sensors, RFID tags, and so on[1]. These things can cooperate using some unique addressing method, but they suffer from low computation resources and limited battery po wer. IoT devices may also continuously produce a considerable amount of data and transfer them to remote cloud computing data centers. Nevertheless, this incurs high traffic and long delays. Therefore, there is a need for more computational and storage resources at the IoT networks' edge. MEC is new technology to answer such requirements and provide a distributed computational model to facilitate task processing, networking, and data storage at the edge of IoT networks [2]. MEC can meet IoT applications' resource requirements and can reduce communication overheads and latency [3]. Each MEC server is a virtualized environment equipped with a computational device, a wireless communication unit, and data storage cards. Several virtual resources resided in the MEC servers which can serve mobile devices (MDs) or fixed IoT devices. Typically, an MD can communicate directly with MEC servers through a single-hop wireless connection using a wireless interface, namely 4G LTE devices, WiFi, Bluetooth, etc. On the other hand, the MEC servers themselves are connected to the cloud infrastructure via the Internet [4]. Figure 1 depicts MEC's architecture in which IoT devices are connected to the MEC using wireless links [5]. Efficient resource management is a challenging issue in MEC and can provide high-quality services to the MDs. In MEC environments, scheduling approaches assign tasks to the most appropriate set of MEC servers [6]. Such MEC scheduling schemes manage MEC virtual resources and focus on the constraints and deadlines of the workflows.

Generally, regarding their architecture, the MEC scheduling schemes can be classified as centralized and decentralized scheduling approaches. Besides, concerning the handling task methods, the scheduling methods can be classified as online and offline scheduling approaches. Furthermore, these scheduling schemes can be classified into the heuristic and metaheuristic categories based on their algorithm. The latter category approaches may use a single objective or multi-objective optimization algorithms. In the scheduling process, some of the studied schemes consider homogenous VMs for MEC, and the others use heterogeneous VMs. All MEC scheduling approaches often pursue goals such as minimizing cost, makespan, communication overhead, and energy consumption while maximizing reliability, security, availability, etc. Energy consumption is one of the critical factors that is considered in most MEC scheduling schemes.

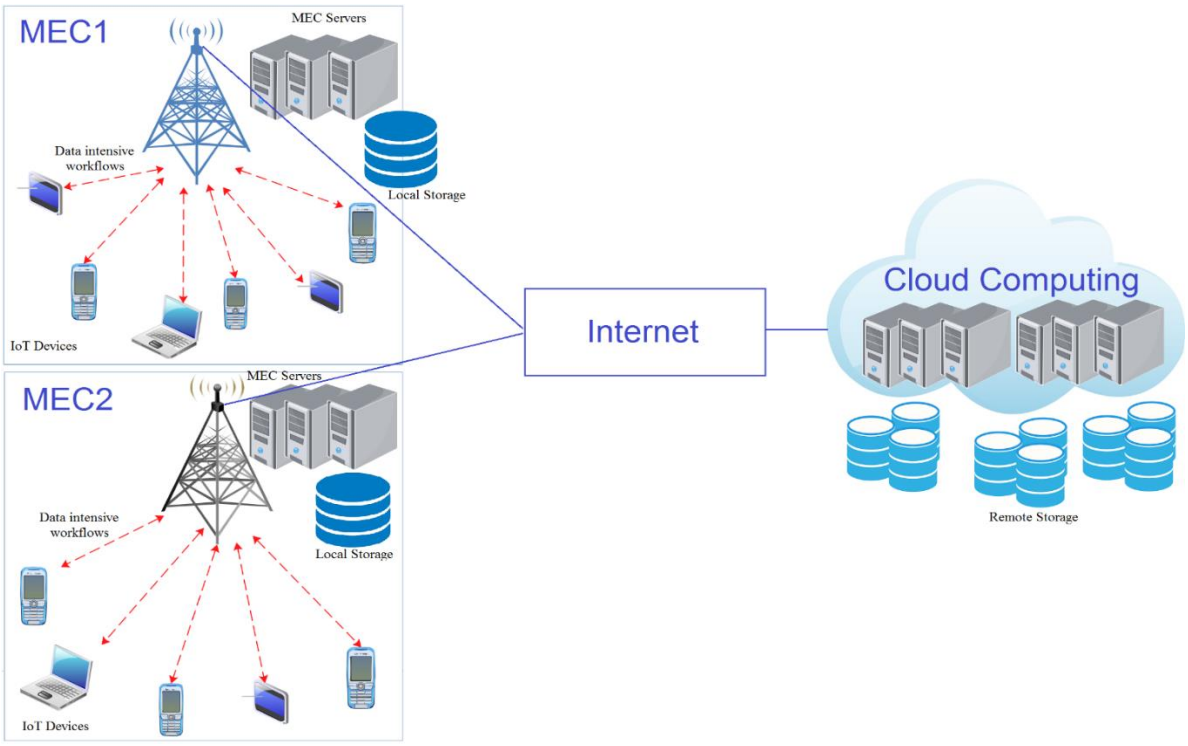


Figure 1: MEC architecture

Three major energy models have been incorporated in MEC environment which are: the conventional energy model, the renewable energy model, and the dynamic voltage and frequency scaling (DVFS)-based energy model. The latter is focused on this article. DVFS is an exciting method that can reduce the processor frequency to mitigate its power usage. However, reducing the processor frequency increases its execution time; thus, such reductions of frequency should be handled while considering deadlines and various QoS factors. Although numerous DVFS-based researches have been carried in different cloud computing platforms, very few MEC scheduling schemes are provided in the literature.

One of the essential issues in scientific workflow scheduling is the data placement strategy that can mitigate data transmission and data storage costs in the MECs. Both workflow scheduling and data placement problems are proven to be NP-hard problems [7, 8]. However, only a few research has been conducted for optimal data placement and data-intensive workflow scheduling strategies in MEC environments. In this context, several heuristics and metaheuristic scheduling methods are widely studied by scheduling independent tasks. Still, fewer researches have been presented for workflow scheduling in the MEC context. Different optimization algorithms, such as PSO, genetic algorithm, etc., are used in the metaheuristic fog scheduling approaches. In the optimization algorithm context, the BOA algorithm is a population-based algorithm inspired by the butterflies' movement in their food foraging behaviors. It is used for solving continuous optimization problems that are firstly introduced in [9]. Furthermore, up to now, the BOA algorithm is successfully incorporated in different domains such as features selection [10, 11], optimization algorithms [12-16], engineering optimization problems [17-21], enhancing WSNs [22], localization problems [23], artificial neural networks [24, 25], etc.

However, despite the BOA's success in the continuous optimization context, it cannot be used without modifying the discrete problems. Thus, regarding the scheduling problem's discrete nature, there is a need for some improvement and changes in this algorithm to make it usable in discrete optimization problems. However, despite the BOA's success in the continuous optimization context, it cannot be used without modification. Thus, regarding the scheduling problem's discrete nature, there is a need for some improvements in this algorithm.

This article presents DBOA, a discrete version of the BOA algorithm, and uses genetic algorithm operators such as mutation and crossover to solve discrete optimization problems. Afterward, a workflow scheduling and data placement framework based on the DBOA is provided for the MEC environments. For this purpose, a discrete encoding is used in this scheme, which will be processed by the DBOA to place the data replicas and the workflow tasks in the best available MEC sites and VMs, respectively. Furthermore, this scheme uses the HEFT scheduling algorithm's task prioritization method for finding the order of the execution of the task in the scientific workflows and then use the DBOA for assigning tasks for proper MEC virtual machines. The experiments on the various scientific workflows indicate that this proposed scheme can reduce the scheduling process's energy consumption and data access overheads.

The rest of this scheduling article is organized as follows: Section 2 gives an interesting review of the research background and previously published scheduling approaches on the MEC, and Section 3 discusses the BOA algorithm. Section 4 presents the proposed discrete version of the BOA algorithm, and Section 5 introduces the proposed workflow scheduling framework. Moreover, Section 6 reports the experimental results, and finally, Section 7 puts forward the concluding remarks and directions of future studies.

II. Research background

This section studies some of the primary schemes proposed for task and workflow scheduling and data placement in the different MEC environments.

A. SDN and NFV-based MEC schemes

This subsection discusses the MEC solutions provided using techniques such as big data, SDN (Software-Defined Networks), and NFV (Network Function Virtualization). For instance, in [26], Li et al. introduced a data migration-based heterogeneous task scheduling to reduce the task execution time and energy consumption of data centers. This scheme considers the data locality property, indicating the relationship between the tasks, data blocks, and servers. The authors evaluated their algorithm by comparing the cost of remote task execution and data migration. The experiment results show that the data migration method can decrease the task execution time.

The approach proposed in [27] introduces a computing migration solution for the next generation networks. Also, it presents a MEC strategy based on SDN and NFV technologies as well as multi-attribute decision making and computing migration. The authors conducted their experiments on MATLAB and showed that the multi-attribute decision making based on SDN and NFV could select the appropriate MEC center, reduce the server response time, and improve the QoS experience.

In [28], the authors attempted to deal with resource allocation on NFV-enabled MECs, aiming to minimize the mobile services' latency and costs of the MECs. They proposed a resource allocation method consisting of a fast heuristic-based incremental allocation method to allocate resources dynamically based on operational cost. They exhibited that simulations that this scheme can allocate resources to guarantee applications' low latency requirements while saving cost compared to the fixed MECs.

The scheme proposed in [29] by Pham tries to optimize gateway placement and multihop routing in the NFV-enabled IoT(NIoT) and the service placement in the MEC and cloud layers. They developed some approximation algorithms such as SP1A, SP2A, GMA for dealing with a large NIoT system and optimizing routing, resource allocation for service functions, and gateways deployment. The authors indicated that their approximation algorithms could reduce the computation time and achieves near-optimal results.

In [30], the authors introduced an open-source NFV/SDN-based MEC, which handles traffic management and application provisioning in the MECs. In this scheme, the MEC applications are managed as VNFs on the virtual environments provided using the Juju VNF Manager. In this scheme, an SDN controller is used to manage traffic on the MEC, and the control plane is used to find appropriate traffic management states. They evaluated their approach in two use-cases, in which, in the first case, the MEC caching is used to improve the user QoS and latency. In contrast, in the second case, the public safety communication method provides a communication method for rescue teams when the network core and a public data network are not available.

In [31], the authors introduced ADE 2 WiNFV, a new network system that applies the software-defined wireless network virtualization (WiNV) in the WiFi. This scheme can combine the software-defined WiNV with NFV-based MEC to handle application based end-to-end slicing in heterogeneous networks.

Development in the IoT domain enables remote monitoring in the E-Health systems. For instance, in [32], the authors tried to securely integrate big data processing with cloud M2M systems based on Remote Telemetry Units and propose an E-Health architecture built on Exalead CloudView, a search-based application.

B. Scheduling schemes in MECs

Some of the recent scheduling schemes, such as [33-38], have investigated the task scheduling problem in MEC environments.

In [39], the authors presented a task scheduling approach in which computational tasks should be offloaded to the MEC servers. This task scheduling scheme tries to reduce power usage while meeting all tasks' deadlines and considering MDs' mobility. The resources of heterogeneous MEC servers and computation demands of tasks. They optimized computation tasks to minimize MDs and MEC servers' energy consumption and satisfy tasks' deadlines. The authors also analyzed their algorithm's performances and showed that it could reduce energy consumption.

The proposed approach in [40] studies the workflow scheduling in MEC by formulating the scheduling as an integer problem, aiming to handle different tasks while mitigating the makespan. This scheme uses a greedy method to construct the IGS (Improved Greedy Search) method to deal with constraints. Also, they proposed an improved heuristic algorithm that uses IGS for initialization and applies a two-layer scheme to improve initialized solutions. The authors showed the IGS achieves a high probability of generating feasible solutions, and ICH can better reduce the makespan.

In [41], Cao et al. presented UARP, an uncertainty-aware resource provisioning method for scheduling workflow in the software-defined network-based MECs. The UARP applies the NSGA-III optimization algorithm to elaborate on the workflow scheduling strategy. The authors showed that the UARP could reduce uncertainty, processing time, and energy consumption.

In [42], the authors proposed PIOTS, a pattern-identified online task scheduling mechanism for the networking infrastructure, where multitier MEC is provided to handle the offloaded tasks. They used the pattern of IoT tasks to train a self-organizing map, which represents the task pattern features in defined dimensions. Then, optimal task scheduling on MECs is conducted using SOM neurons with the Hungarian method. The authors showed that the PIOTS method could provide better service capability, computation performance, and task processing latency for handling IoT tasks.

The scheduling approach proposed in [43] provides a location-aware and proximity-aware multiple workflow scheduling on the MEC servers. This approach is capable of minimizing monetary costs with user-required workflow completion deadlines. It employs the discrete firefly algorithm for finding optimal scheduling solutions. For the validation purpose, the authors show that their approach can

outperform other schemes based on a real-world dataset of edge resource locations and multiple scientific workflows.

In [44], Liu et al. proposed a Markov decision process-based method to schedule tasks regarding factors such as buffer queueing state, the local processing unit's execution state, and the transmission unit state. By analyzing each task's delay, and the average power consumption at the mobile device, a power-constrained delay minimization problem is formulated. Also, a one-dimensional search algorithm to find the optimal task scheduling policy is proposed. Simulation results demonstrate the capability of the proposed optimal stochastic task scheduling policy in achieving a shorter average execution delay compared to the baseline policies.

The scheduling method in [45] tries to reduce the task execution delay in MEN(Mobile Edge Network) by considering the task properties, user mobility, and the network's constraints. It formalizes this scheduling problem as a constraint satisfaction problem and introduces a lightweight heuristic solution. Based on the conducted experiments, the authors showed that this scheme could reduce the task execution delay in MENs and mitigate the end-to-end delay for MEC tasks.

In [46], the authors tried to combine the optimal placement of data blocks and scheduling tasks to reduce the delay and response time and increase users' satisfaction in MEC. They considered the data blocks' popularity, storage capacity, and the MEC servers' replacement ratios for optimal data placement. This placement approach can prevent replacing the data blocks to reduce the bandwidth overhead. In this optimal task scheduling method, the containers are applied as resource units to use MEC servers' data storage and increase their performance. The authors conducted some experiments and exhibited that this scheduling algorithm's performance is better than the other approaches.

In [47], Lin et al. presented GA-DPSO, a discrete PSO algorithm with GA operators to optimize data transmission of scientific workflows' data on MEC and cloud computing. The GA's mutation and crossover operators are applied to prevent local optima problem, premature convergence of PSO, and reduce the data transmission time. This scheme considers factors such as bandwidth between DCs, the number of edge DCs, and their storage capacity in the scheduling and placement processes. The experimental results show that this scheme can reduce the data transmission time during workflow execution on MEC and cloud.

In the rest of this paper, the proposed hybrid optimization algorithm to solve workflow scheduling and data placement issues is presented.

BOA algorithm

In the BOA algorithm, each butterfly is a search agent whose fitness varies with the butterfly's movement from one location to another. Also, each butterfly participates in the optimization process by generating a fragrance based on its intensity. The fragrance can be emitted over the area, and butterflies can sense it, and by this method, they can share their data. When a butterfly can sense fragrance from other butterflies, it moves toward them (global search); otherwise, it will move randomly (local search). Then, every butterfly moves randomly toward the best butterfly emitting more fragrance. The BOA algorithm has three main phases, denoted as initialization, iteration, and final steps. In each run of BOA, the initialization step is executed. The searching step is performed iteratively, and in the last step, the algorithm is terminated when the best solution is found.

In the initialization phase, the algorithm defines the objective function and its solution space. After setting the values, the algorithm proceeds to create an initial population of butterflies for optimization. As the total number of butterflies remains unchanged during the BOA simulation, a fixed size memory is allocated to store their information. The positions of butterflies are randomly generated in the search

space, with their fragrance and fitness values calculated and stored. This finishes the initialization phase, and the algorithm starts the iteration phase, which performs the search with the artificial butterflies created. In each iteration in the second phase of the BOA, all butterflies move to new positions in solution space, and their fitness values are computed and evaluated. Afterward, the butterflies generate fragrance using Equation 1.

$$f = cI^a \quad (1)$$

In which parameter *f* is the fragrance value, *c* is the sensory modality, *I* denotes the stimulus intensity, and the power exponent depends on modality, which accounts for the varying degree of absorption. The parameters *a* and *c* are between [0, 1], affecting the algorithm's convergence speed. When $a = 1$, there will be no fragrance absorption, and the amount of fragrance emitted by a butterfly is sensed in the same capacity by the other butterflies. Thus, a butterfly emitting fragrance can be sensed from anywhere in the domain, and a single global optimum can be reached. On the other hand, if $a = 0$, the fragrance emitted by any butterfly cannot be sensed by the other butterflies. Another important parameter is *c,* which determines the speed of convergence and how the BOA algorithm behaves. In the BOA algorithm, butterflies can search for the mating partner and food at the global and local scales. In the global search phase, each butterfly moves toward the best butterfly by using Equation 2, in which X_i^t is the i[th] butterfly in iteration *t, g** is the best butterfly in the current iteration, and f_i is the butterfly's fragrance, while *r* is a random number between 0 and 1. In this algorithm, the local search phase can be performed using Equation 3, in which *r* a random number between 0 and 1, while X_j^t and X_k^t are j[th] and k[th] butterflies in the same swarm. Figure 2 depicts the pseudo-code of the BOA algorithm. In this algorithm, probability *p* is used to switch between the global search and local search.

$$X_i^{t+1} = X_i^t + \left(r^2 \times g^* \ * - X_i^t\right) \times f_i \quad (2)$$

$$X_i^{t+1} = X_i^t + \left(r^2 \times X_j^t - X_k^t\right) \times f_i \quad (3)$$

```
Produce initial solutions containing n butterflies
Compute their Intensity
Define c, a, p, sensor modality, power exponent, and modality.
For i=1 to Max_iteration
    Compute the fragrance of the population
    Find the best butterfly
    For each butterfly do
        Produce a random number called r
        If r < p Then
            Go towards the best butterfly using Equation 2
        Else
            Make a random move using Equation 3
        End
    Next
    Update the power exponent a
Next i
Return the best solution
```

Figure 2. BOA pseudo-code

III. The proposed discrete BOA algorithm

This section presents the proposed DBOA algorithm, in which its contributions can be listed as follows:

Using chaotic maps to produce a discrete random initial population and create various random numbers applied in the DBOA algorithm.

Introducing new equations to convert the discrete population to another set of discrete solutions. For this purpose, the methods of global and local searches are changed.

Adding another step to the BOA algorithm to conduct Levy flight-based search and prevent local optima problem.

To provide a discrete version of the BOA algorithm, a method to create a discrete initial population is introduced. Then, the required operators to modify these discrete solutions and convert them into the other discrete solutions are designed. For producing a discrete population, Equation 4 is used to make initial discrete solutions in the DBOA algorithm, in which *Chaotic_MAP()* is one of the chaotic maps [48, 49] that is used to produce random numbers between [0, 1]. The Nvm denotes the maximum number of the VMs in the MEC, *Ndvfs* is the number of DVFS levels. Also, $Ndatafrag$ is the number of data fragments, *Ntask* is the number of tasks, and *Nmec* is the number of MEC environments. Equation 5 indicates the logistic map, and Equation 6 defines the piecewise map. Equation 7 indicates the Intermittency map, Equation 8 shows the circle map, Equation 9 defines the iterative map, and Equation 10 describes the Sine chaotic map.

In this section, genetic operators such as swap, crossover, and mutation are used to modify Equations 2 and 3 in the basic BOA algorithm. In the proposed algorithm, Equation 11 tries to compute the expression $r2 \times g^*$, which is part of the Equation 2.

$$\begin{aligned} P_i^j &= Floor(\ Nvm * Chaotic\_MAP()) \\ P_i^{j+1} &= Floor(\ Ndvfs * Chaotic\_MAP()) \\ P_i^k &= Floor(\ Ndatafrag * Chaotic\_MAP()),\ where\ (Ntask+1) \le k \le (Ntask+Nmec) \end{aligned} \tag{4}$$

$$K_{i+1} = aX_i(1 - X_i) \tag{5}$$

$$X_{i+1} = \begin{cases} \dfrac{X_i}{P} & 0 \le X_i < P \\ \dfrac{X_i - P}{0.5 - P} & P \le X_i < 0.5 \\ \dfrac{1 - P - X_i}{0.5 - P} & 0.5 \le X_i < 1 - P \\ \dfrac{1 - X_i}{P} & 1 - P \le X_i < 1 \end{cases} \tag{6}$$

$$X_{i+1} = \begin{cases} \varepsilon + X_i + C(X_i)^n & 0 < X_i \le P \\ \dfrac{X_i - P}{1 - P} & P < X_i < 1 \end{cases} \tag{7}$$

$$X_{i+1} = X_i + b - \left(\frac{a}{2\pi}\right) sin(2\pi X_i)\ mod(1) \tag{8}$$

$$X_{i+1} = sin\left(\frac{a\pi}{X_i}\right) \tag{9}$$

$$X_{i+1} = \frac{a}{4} sin(\pi X_i) \tag{10}$$

$$r^2 \times g^* = \begin{cases} g_j^* & r^2 \ge RandV1 \\ ((g_j^* + \Delta g_j^*)\ mod\ UB)\ mod\ LB & r^2 < RandV1 \end{cases} \tag{11}$$

$$\Delta g_j^* = g_j^* \times (1 - \frac{r_i}{r_{max}}) \times (-1)^j \quad (12)$$

$$(r^2 \times g^* - X_i^t) \times f_i = \begin{cases} r^2 \times g^* & f_i \geq RandV2 \\ X_i^t & f_i < RandV2 \end{cases} \quad (13)$$

In Equation 11, the *RandV1* is a real random number between [0,1] achieved by using the chaotic maps and Δg^* is the offset value which should be added to the best butterfly to achieve a solution near it. The Δg^* can be computed using equation 12, in which, the Δg^* has larger changes in the first rounds to benefit from more exploration rate, and the lowest changes will be made in the last rounds to profit from exploitation. The next expression to be computed is $(r^2 \times g^* - X_i^t) \times f_i$, which is computed using Equation 13. This equation performs a two-point crossover to compute the required expression and uses a random variable *RandV2,* which is between 0 and 1. Then Equation 14 computes the x_i^{t+1} by performing a multi-point cross over. Furthermore, Equation 15 indicates a two-point crossover operator. Afterward, in the DBOA algorithm, Equation 3 is changed to produce discrete results. For this purpose, Equation 16 is used, in which *UB* indicates the upper bound and *LB* is the lower bound of a dimension in the solution.

Besides, in Equation 16, ΔX_j^t is the discrete changes which should be made on the X_j^t the solution and the *RandV5* is a random variable between [0, 1]. For computing the ΔX_j^t Equation 17 is used, in which r_i and r_{max} are the current round and the maximum round of the algorithm.

Another expression which should be changed is $(r^2 \times X_j^t - X_k^t) \times f_i$ that is modified as Equation 18, in which the *RandV6* is a random variable between [0, 1] and X_k^t is the k[th] butterflies in the same swarm as X_j^t. Afterward, for computing the $X_i^t + (r^2 \times X_j^t - X_k^t) \times f_i$, Equation 19 is applied, which performs a multi-point crossover operation; in this equation, *NC* is a random number between [1, *SolutionLength*], and *SolutionLength* indicates the length of each solution. The proposed DBOA algorithm applies the Levy flight method to better search the problem space, increase its convergence speed, and prevent the local optima problem. The Levy flight can help for generating new butterflies in the problem space to achieve these objectives.

$$X_i^{t+1} = CrossOver(X_i^t, (r^2 \times g^* - X_i^t) \times f_i, NC)$$
$$NC = Rand(1, SolutionLength) \quad (14)$$

$$X_i^{t+1} = \begin{cases} X_i^t & if\ j \geq RandV4 \\ (r^2 \times g^* - X_i^t) \times f_i & if\ RandV3 \leq j < RandV4 \\ X_i^t & if\ j < RandV3 \end{cases} \quad (15)$$

$$r^2 \times X_j^t = \begin{cases} X_j^t & r^2 \geq RandV5 \\ ((X_j^t + \Delta X_j^t)\ mod\ UB)\ mod\ LB & r^2 < RandV5 \end{cases} \quad (16)$$

$$\Delta X_{jk}^t = X_{jk}^t \times (1 - \frac{r_i}{r_{max}}) \times (-1)^k \quad (17)$$

$$(r^2 \times X_j^t - X_k^t) \times f_i = \begin{cases} r^2 \times X_j^t & f_i \geq RandV6 \\ X_k^t & f_i < RandV6 \end{cases} \quad (18)$$

$$X_i^{t+1} = CrossOver(X_i^t, (r^2 \times X_j^t - X_k^t) \times f_i, NC) \quad (19)$$

NC=Rand(1,SolutionLength)

$$L(s,\gamma,\mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}} \exp\left(\frac{\gamma}{2(s-\mu)}\right) \frac{1}{(s-\mu)^{\frac{3}{2}}} & 0 < \mu < \infty \\ 0 & otherwise \end{cases} \quad (20)$$

$$F(k) = \exp(-\alpha|k|^{\beta}), \quad (21)$$

Generally, Levy flight is a random process with non-Gaussian distribution, proposed by the French mathematician Levy in the 1930s. Typically, for Levy flight distribution, Equation 20 can be used, in which s and μ are samples and transmission parameters. Also, in this equation, the parameter γ controls the scale of the Levy flight distribution. The definition of Levy flight distribution in the Fourier transform is expressed using Equation 21, in which α is the scale parameter, and β is (0,2]. Furthermore, using the Mangata algorithm, the step length s can be computed using Equation 22, in which parameters v and u have Gaussian distribution and can be computed using Equations 23 and 24. Afterward, the stepsize parameter, which determines the search space step size, can be computed using Equation 25, and s depends on the problem dimension. Also, Equation 26 is added to the butterfly algorithm to perform a better local search and mitigate the local optima problem, in which, $Levy(X_j^t)$ is the Levy flight of the jth solution in the population and can be computed using Equation 27, in which the step is calculated using Equation 28, $\otimes$ is an element-based multiplication operator, and random(size(X_j^t)) is a random solution that can be computed by chaotic operators. Figure 3 depicts the pseudo-code of the DBOA algorithm.

$$s = \frac{u}{|v|^{1/\beta}} \quad (22)$$

$$u \sim N(0, \delta_u^2), \qquad v \sim N(0, \delta_v^2) \quad (23)$$

$$\sigma_u = \left\{\frac{\Gamma(1+\beta)\sin()}{\Gamma[\frac{(1+\beta)}{2}]\beta 2^{(\beta-1)/2}}\right\}^{1/\beta} \quad (24)$$

*Stepsize= scale*s* (25)

$$X_i^{t+1} = Levy(X_j^t) + \left(r^2 \times X_j^t - X_k^t\right) \times f_i \quad (26)$$

$Levy(X_j^t) = X_j^t + step \otimes random(size(X_j^t))$ (27)

$Step = stepsize \otimes X_j^t$ (28)

IV.Workflow Scheduling Using DBOA Algorithm

This scheme's primary goal is to assign proper VMs to the workflow tasks to minimize energy consumption and makespan of the workflow scheduling process. This section explains how the proposed DBOA algorithm is used for efficient workflow scheduling in MEC environments. For this purpose, a formal formulation of the problem is provided, and then the proposed approach to deal with it is discussed.

Function Discrete BOA (DBOA)
Input: Number of solutions Maximum number of iterations **Output:** The best solution

```
For i= 1 : Npopulation
    For j=1 to Ntask step 2
      X_j^i = Floor(Nvm ∗Chaotic_MAP())
      X_j^i = Floor(Ndvfs ∗Chaotic_MAP())
    End
    For j=1 to Nreplica
      X_j^i = Floor(Nmec ∗Chaotic_MAP())
    End
End

Compute their Intensity
Define c, a, p, sensor modality, power exponent, and modality.

For i=1 to Max_iteration
  Compute the fragrance of the population
  Find the best butterfly
  For each butterfly do
    Produce a random number called r
    If r < p Then
      Go towards the best butterfly using
      Equation 11 to 15
    Else
      If randLevy > 0.5
        Make a random move using Equation
        16 to 19
      Else
        Make a random move based on the
        Levy Flight and using Equation 26
      End
    End
  Next
  Update the power exponent a
Next i
Return the best solution
```

Figure 3: DBOA pseudo-code

A. DVFS

This subsection provides a formal definition of the energy model considered in this scheduling approach. Typically, the DVFS method reduces the CPU operational frequency and voltage, mitigating processors' energy consumption in the task execution. DVFS method can be incorporated in all computing systems, ranging from mobile systems to cloud computing DCs. However, reducing the CPU frequency mitigates its speed, and as a result, the QoS constraints of the application should be considered in using the DVFS. Thus, the main task of a DVFS-based scheduling framework is to determine the minimum necessary operating frequency according to each task's deadline.

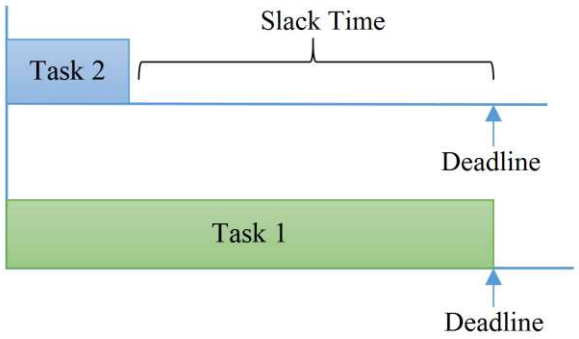


Figure 4: Slack time

Typically, when more frequency is used in DVFS-based scheduling [50, 51], the task can be executed faster, but it takes longer to finish the task when a lower frequency level is utilized. Figure 4 shows the slack time, which can be used by DVFS-based scheduling. As shown in Equation 29, the slack time is the period between the deadline ($TaskDeadline_i$) and the task finish time $TaskFinishTime_i$ (F_{max}).

$$SlackTime = TaskDeadline_i - TaskFinishTime_i\ (F_{max}) \quad (29)$$

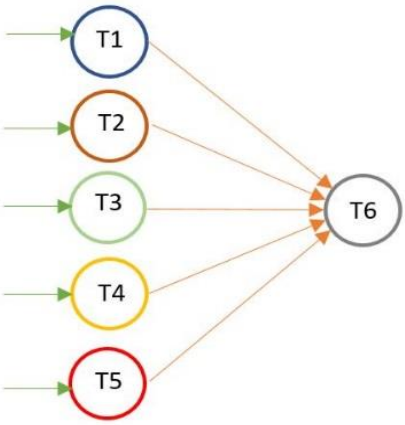


Figure 5: Data aggregation point in a workflow

Figure 5 depicts some parts of a workflow that perform data aggregation and should be scheduled on the MEC environment. As shown in this figure, T1, T2, T3, T4, and T5 should be executed and completed before T6 starts to be executed. However, in some cases, T1 to T5 may be heterogeneous and have different execution times. For example, when the T1 is longer than other tasks, the VMs which run T2 to T5 will be idle after executing their task. Using the DVFS, the frequency of the VM1 to VM5's CPU can be mitigated, and the execution speed for extending the execution time of these tasks can be reduced while meeting the deadline determined by the task T1.

Also, the deadlines may be specified by the user or by the scheduling algorithm itself. The primary goal of DVFS-based scheduling is to detect the least possible frequency for each task's VM regarding its deadline. In this case, while the deadline is completed, the least power is consumed for tasks and workflow executions. In this scheme, it is assumed that power consumption consists of static and dynamic and energy consumptions. Typically, static energy consumption is ignored since dynamic energy consumption is more time-consuming and expensive.

Typically, in the idle periods of VMs, their voltage should be set to the lowest level to save energy. The energy consumption of idle periods for all available processors can be defined using Equation 30, in which v_{jmin} and f_{jmin} are the minimum voltage and frequency of the jth VM, and IT_j is considered to

be the idle time of the j[th] VM. The $Power_{idle}$ or idle time power consumption is computed using Equation 31. By using Equation 32, the energy consumption of the VMs in their busy periods can be computed, in which *K* is the constant of dynamic power consumption and depends on the capacities of the devices. Also, v_{js}^2 is the s[th] level's voltage in the j[th] VM, and f_{js} is the frequency of processor j[th] VM, at the s[th] voltage level, and ET_{ij} is the execution time of the i[th] task on the j[th] VM. According to these equations, the total energy consumption required for workflow scheduling in a MEC environment can be computed using Equation 33.

Problem formulation

This section provides a formal definition of the considered workflow model and the environment incorporated to run workflows. In this scheme, a set of MECs is considered, which are denoted as MEC={MEC_1, MEC_2, MEC_3,…}. Also, it is assumed that each MEC contains a collection of computational resources in the form of VMs, denoted by $VM=\{VM_1, VM_2, VM_3, ...\}$. The VMs can also work at different levels, such *as* $FDVFS=\{FDVFS_1, FDVFS_2, FDVFS_3, ...\}$, in which the $FDVFS_i$ indicates the i[th] setting for the frequency and voltage of the MEC processor and $FDVFS_i=(Frequency_i, Voltage_i)$. Furthermore, it is assumed that $FDVFS_i<FDVFS_j$ in which i<j, or to be more specific, $Frequency_i<Frequency_j$ and $Voltage_i< Voltage_j$. Besides, a dataset on the cloud computing data center is considered in which several fragments of it can be placed on the MEC environments and are denoted as $Dataset=\{Frg0, Frag1, Frag2, ...\}$. However, each MEC's total fragments should be less than or equal to its storage capacity ($MECStorage_i$). The number of VMs that each MEC can allocate for the IoT tasks are denoted as $NVMmec_i$.

$$E_{Idle} = \sum_{j=1}^{p} Power_{idle} * IT_j \quad (30)$$

$$Power_{idle} = k * v_{jmin}^2 * f_{jmin} \quad (31)$$

$$E_{busy} = \sum_{j=1}^{n} Power_{dynamici} * ET_{ij} \quad (32)$$

$$E_{total} = E_{Busy} + E_{Idle} \quad (33)$$

However, as shown in Equation 34, the size of the total storage in MECs are assumed to be far less than the size of the dataset, which is denoted by $Size(dataset)$. Each submitted workflow to the MEC environment is considered a directed acyclic graph (DAG). In this scheme, the set of workflows submitted to the MEC is indicated by $W=\{W_1, W_2, W_3, ...\}$, in which each of them contains some tasks, $W_i=\{T_1, T_2, T_3, ...\}$. Moreover, each workflow is considered a DAG, in which each node represents a task, and edges specify the data or control dependencies between tasks. Also, E_{ij} defines the edge between the T_i and T_j, when $T_i \neq T_j$. This indicates that the child task only can be executed after all of its parent tasks are fully executed, and their output data have been received. Control dependencies only transfer the configuration parameters needed to run the child task and transfer fewer data than data dependencies.

However, the transferred data in the data dependencies are used as input data to the child process. Equation 35 indicates how to compute the average computation time of the T_i; in this equation $Exection\_Time(T_i, FDVFS_k)$ or the execution time of the T_i in the k[th] DVFS level,

in which $VM(j, FDVFS_k)$ specifies the speed of the j[th] VM using k[th] DVFS level, and $Task\_len(T_i)$ specifies the length of the task in terms of the millions of instructions per second. Also, the average time to execute the task T_i on the j[th] VM can be computed using Equation 38, in which *Ndvfs* is the number of DVFS levels in the VM.

Furthermore, the task T_i's average execution time on all VMs can be computed using Equation 37. In this scheme, the earliest start time of each task can be computed using Equation 38, where $avail(VM_j)$ is the time which j[th] VM becomes available to execute the requested task. Also, the communication time of the data transfer between the T_i and T_j can be computed using Equation 39, in which $Bandwidth(VM(T_i), VM(T_j))$ is the bandwidth between two VMs which execute the T_i and T_j tasks and $Data(T_i, T_j)$ denotes the amount of data that should be transferred between these tasks. Furthermore, in this scheme, the finish time of each task can be computed using Equation 40, in which $deadline_{wi}$ denotes the deadline of the i[th] workflow. Moreover, the makespan of the workflow w_i can be calculated using Equation 41.

$$\sum_{i=1}^{Nmec} MECStorage_i < Size(dataset) \tag{34}$$

$$Exection\_Time(T_i, FDVFS_k) = \frac{Task\_len(T_i)}{VM(j, FDVFS_k)} \tag{35}$$

$$Ave(Exection\_Time(T_i, VMj)) = \frac{1}{Ndvfs} \sum_{k=1}^{Ndvfs} Exection\_Time(T_i, F\_DVFS_k) \tag{36}$$

$$Ave(Exection\_Time(T_i)) = \frac{1}{Nvm} \sum_{j=1}^{Nvm} Ave\big(Exection\_Time(T_i, VM_j)\big)_{,,} \tag{37}$$

$$ESTime(T_i, VM_j) = \begin{cases} 0 & IF\ \ T_1\ is\ an\ entry\ task \\ \max\{avail(VM_j), \max\{FT(Tj) + Com\_Time(T_j, T_i)\} & otherwise \end{cases} \tag{38}$$

$$Com\_Time(T_i, T_j) = \begin{cases} 0 & IF\ \ VM(T_i) = VM(T_j) \\ \dfrac{Data(T_i, T_j)}{Bandwidth(VM(Ti), VM(Tj))} & otherwise \end{cases} \tag{39}$$

$$FT(T_i, VM_j) = \begin{cases} deadline(w_i) & if\ \ T_i\ \ is\ an\ exit\ task \\ ESTime(T_i, VM_j) + Ave(Exection\_Time(T_i)) & otherwise \end{cases} \tag{40}$$

$$makespan(w_i) = \{\ max(FT(T_i))\ \mid T_i\ \in\ w_i\ \} \tag{41}$$

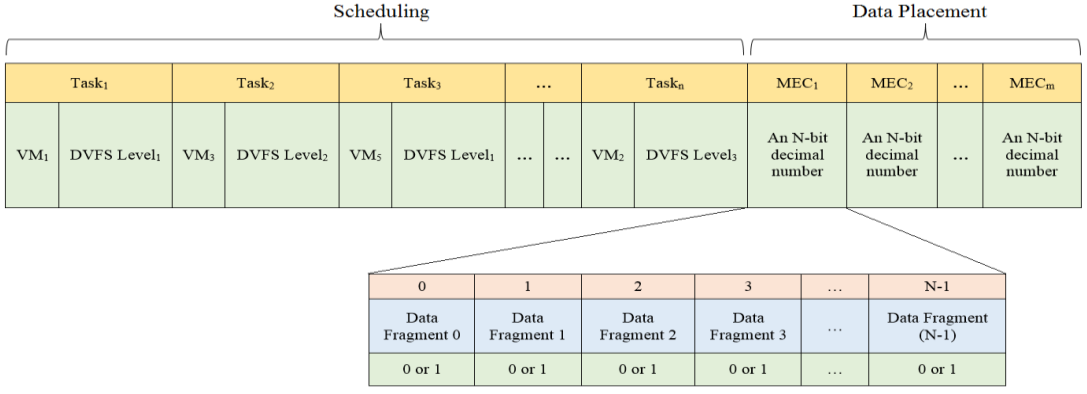


Figure 6: A sample encoding in the proposed scheduling scheme

$$Fitness= \min \begin{cases} 1 & if\ \ makespan(w_i) > deadline(w_i)\ or\ an\ MEC\ has\ more\ fragments\ than\ its\ capacity \\ \alpha * ExecutionTimeRatio + \beta * AppiledStorageRatio & if\ \ makespan(w_i) < deadline(w_i) \end{cases} \quad (42)$$

$\alpha+\beta=1$

$0 \leq ExecutionTime \leq 1$

$$For\ each\ MEC\ environment \sum_{i=Ntask+1}^{Ntask+Nmec} solution(i) < MECStorage_i \quad (43)$$

$$ExecutionTimeRatio = \frac{makespan(w_i)}{deadline(w_i)} \quad (44)$$

$$AppiledStorageRatio = \frac{AppiledFrags(w_i)}{TotalStorage} \quad (45)$$

$$TotalStorage = \sum_{i=1}^{Nmec} MECStorage_i \quad (46)$$

$$AppiledFrags(w_i) = \sum_{j=Ntask+1}^{Ntask+Nmec} count(solution(j)) * FragSize \quad (47)$$

$$Rank(Ti) = Ave(Exection\_Time(Ti)) + \{\max(Com\_Time(Ti,Tj) + Rank(Ti)) | Tj \in \text{Successor(Ti)}\} \quad (48)$$

B. Encoding and fitness function

Figure 6 indicates a sample encoding in the proposed scheduling scheme. As shown in this figure, each solution can be represented by a two-dimensional array in which for each task, its VM, and the VM's DVFS level should be determined. This scheme assumes that each VM can be tuned to several DVFS levels. Also, both the VM's number and DVFS levels are discrete numbers. Equations 42 and 43 indicate the fitness function and its constraints considered in this fog scheduling scheme, in which the *ExecutionTime* parameter can be computed using Equation 44. Also, in Equation 44, $makespan(w_i)$ is the makespan of the workflow w_i that is achieved using the proposed algorithm, and $deadline(w_i)$ is its deadline. Furthermore, in Equation 45, the $AppiledStorageRatio$ parameter can be computed using Equations 46 and 47, where $FragSize$ is the size of each data fragment in terms of byte and $count(solution(i))$ is used to count the number of bits with 1 in the j[th] entry of the solution.

C. Finding task order

List-based scheduling methods first compute the workflow tasks' priorities in a DAG and rank them in non-increasing order. HEFT is one of the popular list scheduling methods provided in the literature [52].

Procedure MEC_Scheduling()
Input: Workflow wi DVFS levels Deadline MEC environments specifications

```
Determine the number of VMs in the MEC environment
Determine the number of DVFS levels
Determine the workflow deadline

Read workflow data from its DAX file
Compute the rank of workflow tasks using the HEFT ranking method
Sort tasks in increasing order of their rank
Compute the workflow slack time
Allocate the slack time to each workflow levels
Set the objective function
Use the DBOA optimization algorithm to obtain the best solution
For each task in workflow
    Find the VMi and DVFS_Leveli from the best solution
    Find the predecessor set of Ti
    If Ti's predecessor is empty Then
        While ( VMi is not idle )
            Wait
        End
    Else
        While ( VMi is not idle or all Ti's predecessor are not executed )
            Wait
        End
    End
    Allocate the DVFS levels to the VMs
    Execute task Ti on the VMi

End
```

Figure 7: DVFS-based workflow scheduling using DBOA

For finding the order of task execution in the scientific workflows, this scheme benefits from the task prioritization method provided in the HEFT or Heterogeneous Earliest Finish Time Algorithm. HEFT is a heuristic scheduling method for inter-dependent tasks onto a network of heterogeneous workers taking communication time into account. For inputs, HEFT takes a set of tasks, represented as a directed acyclic graph, a collection of workers, the times to execute each worker, and the times to communicate the results from each job to its children between each pair of workers. It descends from list scheduling algorithms. HEFT algorithm first determines the priorities of the tasks and then assigns tasks to the workers. The rank of each task indicates its execution turn in the workflow scheduling. Thus, tasks with the lower rank will be executed first, and tasks with the higher ranks will be executed later and will have the lowest priority for execution. Equation 48 indicates how the rank can be computed for each workflow task, where T_i is the ith task in the workflow and $Ave(Exection\_Time(T_i))$ is the average execution cost of the ith task. Also, $\text{Successor}(T_i)$ specifies the successor tasks of the T_i and $Com\_Time(T_i, T_j)$ specifies the communication cost between the T_i and T_j.

DBOA-based workflow scheduling

The pseudo-code of the DVFS-based workflow scheduling using DBOA is shown in Figure 7. As shown in this algorithm first, the required parameters for running the algorithm are tuned first. The HEFT algorithm's task prioritization method is used to find the workflow's task execution order. Afterward, the DBOA algorithm is used to find the best possible location for the tasks and data replicas. Then, tasks are allocated to the required VM for execution by the order and setting specified in the best solution.

Simulation Results

This section presents the results of the experiments conducted to evaluate the performance of the proposed scheduling framework. MEC has several interesting simulation frameworks, such as iFogSim, FogNetSim++, MobFogSim, and EdgeCloudSim. This scheduling scheme uses the iFogSim simulator

to conduct the required simulations that are an efficient open-source tool for modeling and simulating resource management in IoT and MEC networks. The iFogSim simulator works with the CloudSim, another open-source java-based simulator simulating cloud computing environments and managing its resources. The iFogSim simulator applies the CloudSim to deal with the events among MEC components. The proposed workflow scheduling algorithm is evaluated on the Epigenomics, SIPHT, and Montage, LIGO scientific workflows. Figure 8 depicts the structure of the five well-known scientific workflows.

Epigenomics is a data processing workflow that indicates the execution of the genome sequencing operations and is applied by the Epigenome Center. In this workflow, the DNA sequence data is produced by a genetic analysis system, split into many chunks that can be processed in parallel. Furthermore, each data chunk is converted to a format required by the sequence aligner. Then noisy sequences are filtered. Besides, a map to detect the density of sequence at every genome position is created.

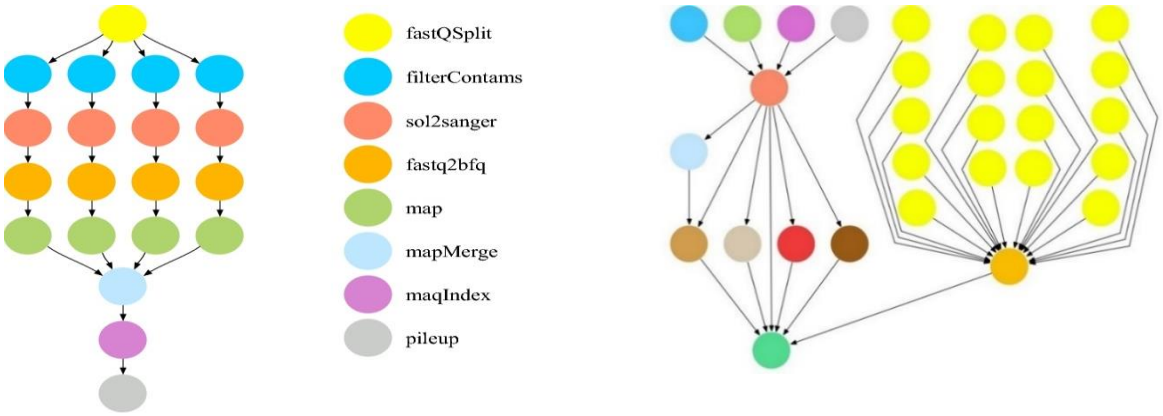
SIPHT is a program that uses a workflow for automating the search for sRNA encoding-genes for all of the bacterial replicons. It is created for a bioinformatics project at Harvard University in searching for untranslated RNAs for regulating processes like virulence or secretion in bacteria.

The NASAIPAC creates Montage as an open-source toolkit to generate custom mosaics of the sky, and it is presented as a workflow that can be run in various Grid, cloud, and even MEC environments.

LIGO workflow is applied to produce and evaluate the gravitational waveforms for the compact binary systems.

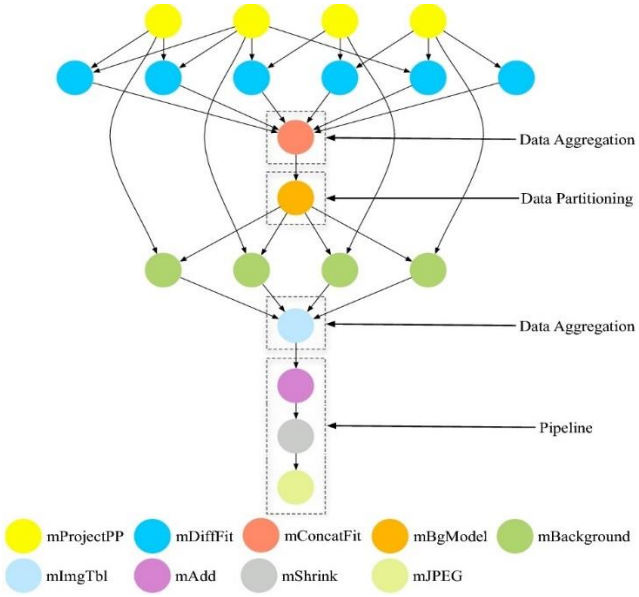
The earthquake center of southern California typically employs CyberShake workflow to analyze earthquake effects.

For evaluation of the scientific workflows, in [34], the authors provided a workflow generator tool that creates arbitrary size scientific workflows XML format that contains data about the task size and the amount of communication between dependent tasks.

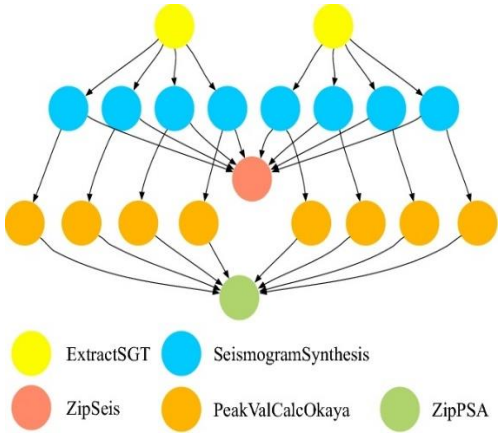


(a) Epigenome workflow

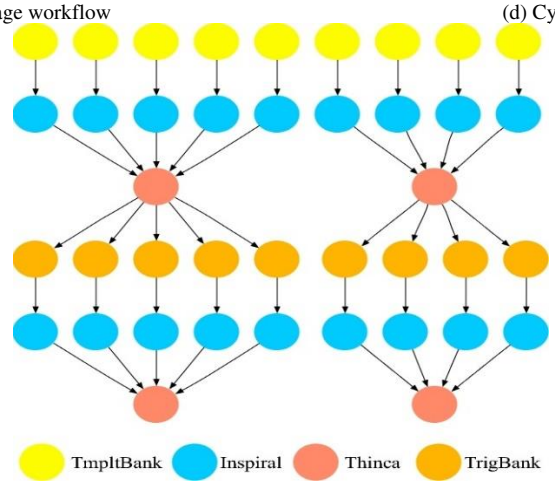
(b) SIPHT workflow



(c) Montage workflow



(d) CyberShake workflow



(e) LIGO workflow

Figure 8: Five well-known scientific workflows

Table 1: Simulation parameters	
Parameters	**Values**
Available VMs	15, 30, 60, 200 VMs
Mobile devices	50 devices
Area	1000*1000
Mobility model	Random way-point
VM's type	Homogeneous
VM' bandwidth	1000 Mbps
Epigenomics workflows	46 Tasks 100 Tasks 997 Tasks
SIPTH workflows	50 Tasks 100 Tasks 1000 Tasks

Montage workflows	50 Tasks 100 Tasks 1000 Tasks
LIGO workflows	1000 Tasks 100 Tasks 50 Tasks
CyberShake workflows	1000 Tasks 100 Tasks 50 Tasks
Algorithm runs	40
Deadline	Makespan+(Makespan/10) Makespan+2*(Makespan/10)
DVFS levels	3 and 5 levels
Number of solutions	60 Solutions
Max iteration	900 rounds

For conducting the required experiments and evaluate the proposed algorithm, as shown in Table 1, three workflows with 50, 100, and 1000 tasks are utilized for the LIGO, CyberShake, and Montage scientific workflows. Regarding the Epigenomics, three workflows with 46, 100, and 997 tasks are used, which the workflow generator provides their DAX. Besides, for SIPHT scientific workflow, three workflows with 60, 100, and 1000 tasks are utilized. Furthermore, the depicted results are averaged from 40 different runs of the investigated optimization algorithms.

D. Energy consumption improvements

This subsection presents the improvements which have been achieved in the energy consumption context. In these experiments, we use two sets of DVFS levels in the simulation scenarios. In these experiments, the first case applies three DVFS levels, specified in Table 2, and the second case applies five DVFS levels, as shown in Table 3.

Table 2: DVFS setting

Level	Voltage	CPU Speed
0	1.5	100%
1	1.4	90%
2	1.2	70%
3	1.0	50%

Table 4: The energy consumption ratio with three DVFS levels

Workflow	Size	DBOA	ACO	PSO	DE	[53]	[54]
LIGO	1000	0.631	0.690	0.721	0.783	0.735	0.752
LIGO	100	0.642	0.717	0.754	0.824	0.758	0.761
LIGO	50	0.653	0.712	0.768	0.831	0.761	0.779
Epigenomics	997	0.589	0.609	0.628	0.637	0.641	0.652
Epigenomics	100	0.612	0.611	0.631	0.642	0.649	0.663
Epigenomics	46	0.623	0.630	0.646	0.658	0.661	0.668
SIPHT	1000	0.642	0.652	0.668	0.681	0.714	0.721

SIPHT	100	0.658	0.678	0.676	0.699	0.735	0.728
SIPHT	60	0.669	0.689	0.685	0.724	0.741	0.732
CyberShake	1000	0.688	0.691	0.714	0.723	0.729	0.738
CyberShake	100	0.694	0.723	0.713	0.726	0.734	0.745
CyberShake	50	0.699	0.710	0.721	0.731	0.739	0.749
Montage	1000	0.735	0.767	0.785	0.788	0.816	0.831
Montage	100	0.761	0.791	0.795	0.792	0.832	0.844
Montage	50	0.783	0.818	0.827	0.827	0.857	0.859

Table 3: DVFS setting

Level	Voltage	CPU Speed
0	1.5	100%
1	1.4	90%
2	1.3	80%
3	1.2	70%
4	1.1	60%
5	1.0	50%

To indicate the improvements achieved by our scheme, we consider the ratio of the DVFS-based workflow scheduling energy consumption to the non-DVFS based workflow scheduling using different optimization algorithms.

Table 4 exhibits the percentage of the energy consumption ratio in workflow scheduling conducted using three DVFS levels and by algorithms such as DBOA, ACO, PSO(particle Swarm Optimization), DE (Differential Evolution), and two variants of the DE presented in [53] and [54]. In this table, the simulation results are provided for three different size scientific workflows. As can be concluded from this table, it can better mitigate the energy required to schedule different size workflows. The reason for this improvement is that our proposed discrete optimization algorithm is more adaptable to the scheduling problem and can achieve better VMs and better DVFS settings for workflow scheduling. At last, from these experiments, the following items can be concluded about the amount of energy consumption reduction in DVFS-based scheduling:

Workflow scheduling deadline.

Number of VMs applied in the scheduling.

Workflow structure.

Deadline distribution policy is applied to distribute the slack time among different workflow levels.

The number of rounds in which the optimization algorithms are executed.

Storage considered for each MEC environment.

Dataset’s fragment size.

The bandwidth of the broker, MECs, and VMs.

The number of DVFS levels considered for VMs.

Our proposed scheduling scheme attempts to reduce the energy consumption of the MEC’s in scheduling the data-intensive workflows by minimizing the number of VMs, keeping interacting tasks to the same VM as much as possible, and placing proper data fragments at the best MECs.

Table 5 shows the total energy consumption ratio when five DVFS levels are applied in the workflow scheduling process. As shown in this table, better results can be achieved by incorporating more DVFS levels, regardless of the optimization algorithm used in the scheduling. However, the proposed DBOA algorithm outperforms other algorithms such as ACO, PSO, DE, and its two variants, even with more DVFS levels. As shown in Table 1, 60 particles or solutions are considered for each optimization algorithm in these experiments. Besides, algorithms are run for a maximum of 900 rounds.

Table 5: The energy consumption ratio in workflow scheduling with five DVFS levels

Workflow	Size	DBOA	ACO	PSO	DE	[53]	[54]
LIGO	1000	0.586	0.611	0.632	0.654	0.689	0.705
LIGO	100	0.592	0.619	0.643	0.669	0.692	0.719
LIGO	50	0.605	0.624	0.651	0.673	0.699	0.728
Epigenomics	997	0.534	0.587	0.603	0.615	0.628	0.631
Epigenomics	100	0.539	0.591	0.611	0.622	0.631	0.634
Epigenomics	46	0.543	0.599	0.617	0.631	0.639	0.638
SIPHT	1000	0.602	0.629	0.635	0.651	0.667	0.647
SIPHT	100	0.611	0.634	0.638	0.655	0.671	0.649
SIPHT	60	0.617	0.638	0.642	0.658	0.673	0.651
CyberShake	1000	0.618	0.627	0.635	0.652	0.664	0.681
CyberShake	100	0.621	0.630	0.639	0.656	0.666	0.684
CyberShake	50	0.622	0.632	0.641	0.658	0.671	0.686
Montage	1000	0.635	0.672	0.685	0.685	0.716	0.731
Montage	100	0.637	0.674	0.686	0.686	0.718	0.733
Montage	50	0.638	0.677	0.689	0.689	0.721	0.736

E. Communication overhead

Communication overhead is one of the important metrics that most scheduling schemes try to reduce by better assignment of the workflow tasks to the MEC's VMs. As outlined in the previous section, this scheme can further reduce the communication overheads by placing proper data fragments in the locations they will be used. In this subsection, we try to indicate the effectiveness of the proposed discrete optimization algorithm (DBOA) to reduce the communication overheads of the data-intensive workflows. For this purpose, the percentage of communication overheads ratio is computed when data placement is used to the scheduling without data placement. In these experiments, we compare the proposed scheme's results with other optimization algorithms such as ACO, PSO, DE, and two variants of the DE presented in [53] and [54].

Figures 9, 10, and 11 indicate the percentage of communication overheads ratio in the scientific workflows such as LIGO, CyberShake, and Montage with 50, 100, and 1000 tasks. Figure 12 indicates the percentage of communication overheads ratio in the SIPTH scientific workflows with 60, 100, and 1000 tasks. Finally, Figure 13 indicates the percentage of communication overheads ratio in the Epigenomics scientific workflows with 46, 100, and 997 tasks. As shown in this figure, data placement with the proposed DBOA discrete optimization algorithm incurs much less data access overheads than other optimization algorithms such as BOA, ALO, DE, and no data placement policy on MECs. The

reasons for these improvements are the discretization of the BOA algorithm, application of Levy flight-based search in the DBOA algorithm, and chaotic variables applied to produce better random numbers.

In these scenarios, two places for the data are considered: cloud data center storages and MEC storages, in which MEC storage is limited. This scheme tries to put specific data fragments in these storages to mitigate the data access overhead of the scheduling process in MECs. However, for data that are not in the MEC storage, cloud storage should be accessed inevitably. Since the DBOA discrete optimization algorithm can place data replicas in better storage locations, it incurs much less data access overheads than other optimization algorithms such as ACO, PSO, DE, and DE variants.

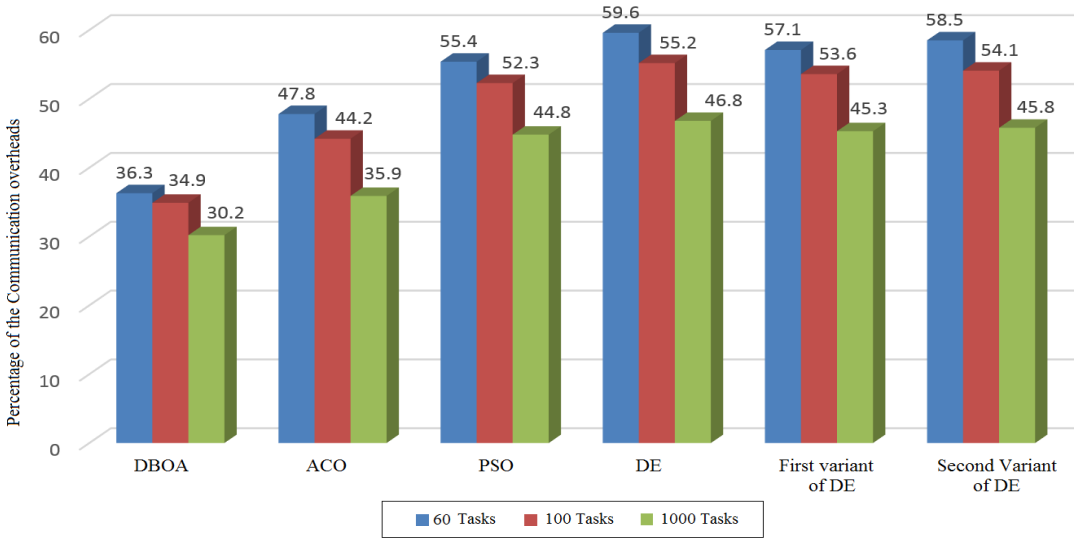


Figure 9: Percentage of the communication overheads ratio in the LIGO workflows

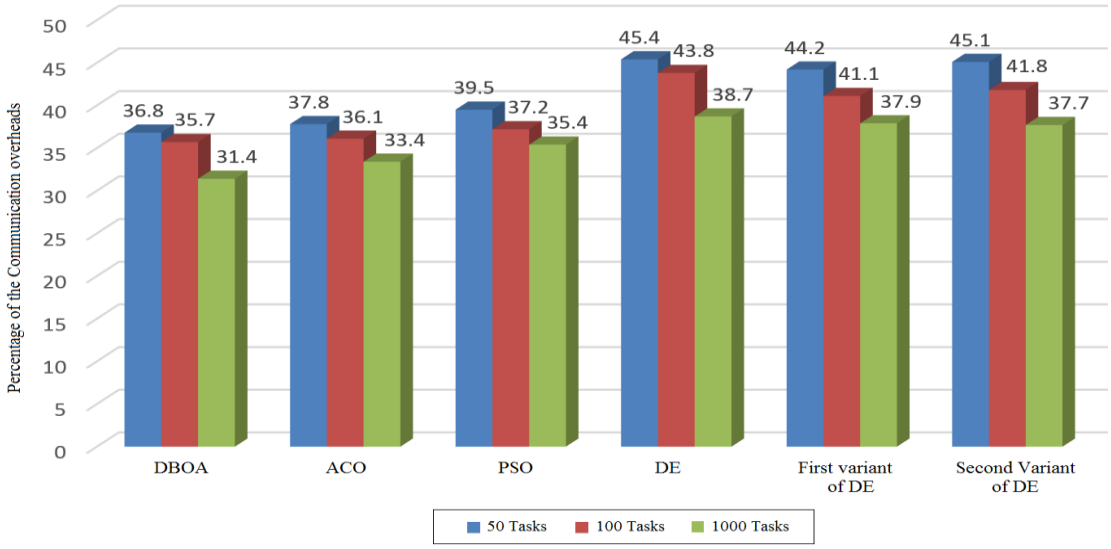


Figure 10: Percentage of the communication overheads ratio in the CyberShake workflows

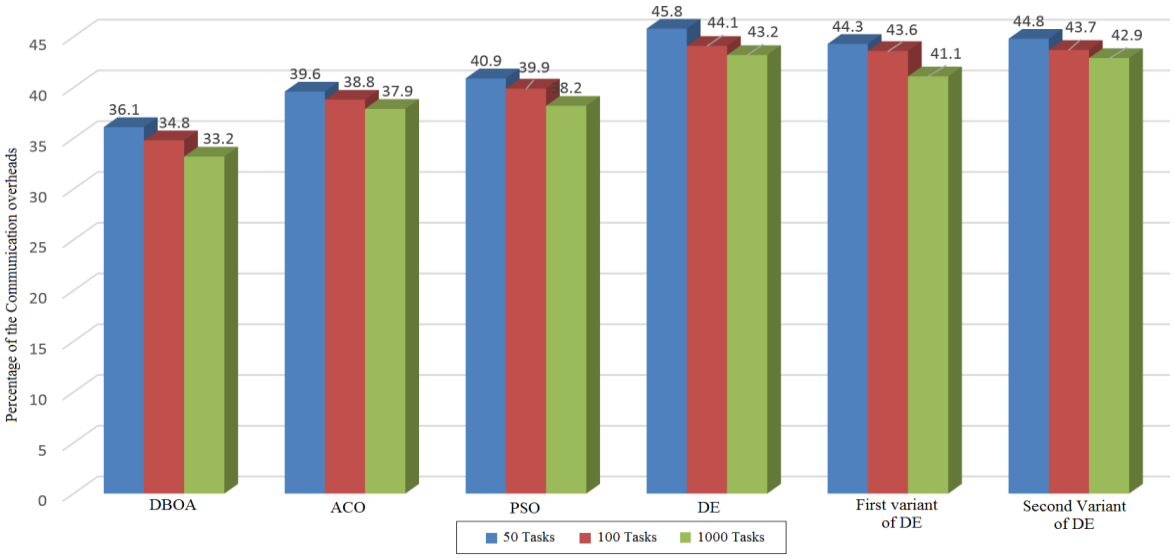


Figure 11: Percentage of the communication overheads ratio in the Montage workflows

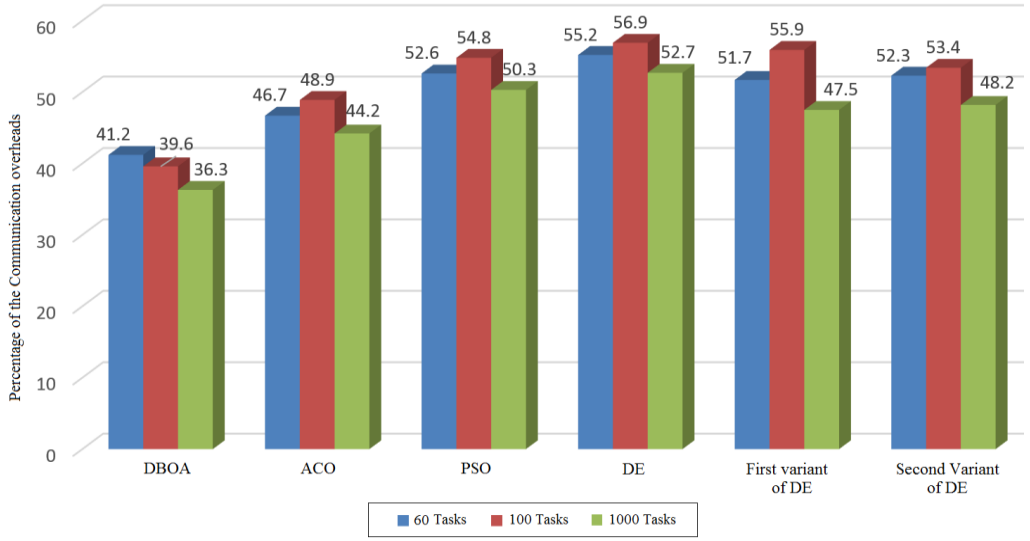


Figure 12: Percentage of the communication overheads ratio in the SIPTH workflows

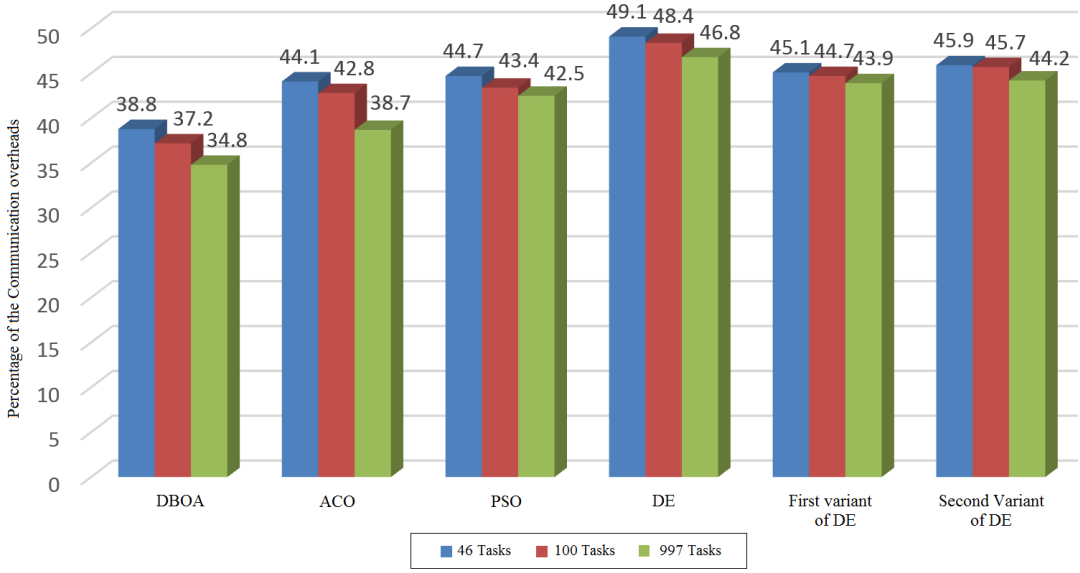


Figure 13: Percentage of the communication overheads ratio in the Epigenomics workflows

F. Makespan

Almost all of the scheduling schemes minimize the makespan somehow. This subsection indicates how much our scheme successfully reduced the scheduling makespan compared to other optimization algorithms. Figures 14 to 18 show the ratio of makespan to the deadline in our proposed algorithm (DBOA), ACO, PSO, DE, and two variants of the DE presented in [53] and [54] when applied for scheduling of five scientific workflows discussed in the previous subsections. These experiments are conducted on the Epigenomics workflows with 997 tasks and LIGO, Montage, SIPHT, CyberShake workflows with 1000 tasks. As specified in Table 6, for all algorithms, 60 solutions are considered as their population, and the exhibited results are the average of 40 different runs of the studied optimization algorithms. Since all applied optimization algorithms use the same initial population, their fitness is the same at first. Then, the algorithms are run for 900 rounds, and as shown in these figures, our proposed scheduling scheme can quickly converge to a near-optimal solution. Generally, the achieved results indicate that some of the algorithms cannot reach the proposed DBOA algorithm's result.

In contrast, some others achieve the same result with our scheme, but our algorithm can converge to the final result quicker. The reason for these improvements is the modifications that have been made to the basic BOA algorithm. By changing the BOA algorithm, the proposed DBOA algorithm can better handle discrete solutions and explores the problem space. Using the Levy flight method can prevent the local optima problem and find near-optimal solutions more quickly.

Table 6: The communication overheads ratio in the second scenario

Workflow	Size	DBOA	ACO	PSO	DE	[53]	[54]
LIGO	1000	29.2	40.9	47.5	50.4	48.5	48.6
LIGO	100	31.7	41.4	49.3	51.2	50.7	50.8
LIGO	50	33.2	45.3	51.5	57.6	55.2	54.3
Epigenomics	997	28.1	29.1	30.3	35.4	31.9	32.1
Epigenomics	100	29.6	30.5	31.1	36.1	32.8	33.2
Epigenomics	46	30.2	31.3	32.8	36.5	33.4	33.8
SIPHT	1000	30.3	35.4	36.7	39.9	38.4	38.9
SIPHT	100	33.6	38.3	39.5	43.8	41.6	42.3
SIPHT	60	35.1	40.7	41.3	48.9	46.7	47.3
CyberShake	1000	28.3	28.8	29.4	32.5	30.9	31.2
CyberShake	100	29.1	29.6	31.2	34.6	32.7	33.1
CyberShake	50	31.8	32.4	35.3	38.5	36.7	37.1
Montage	1000	28.1	31.3	35.5	39.7	37.8	38.2
Montage	100	29.4	32.9	37.1	41.6	40.2	41.1
Montage	50	32.1	38.7	39.9	43.8	41.6	41.9

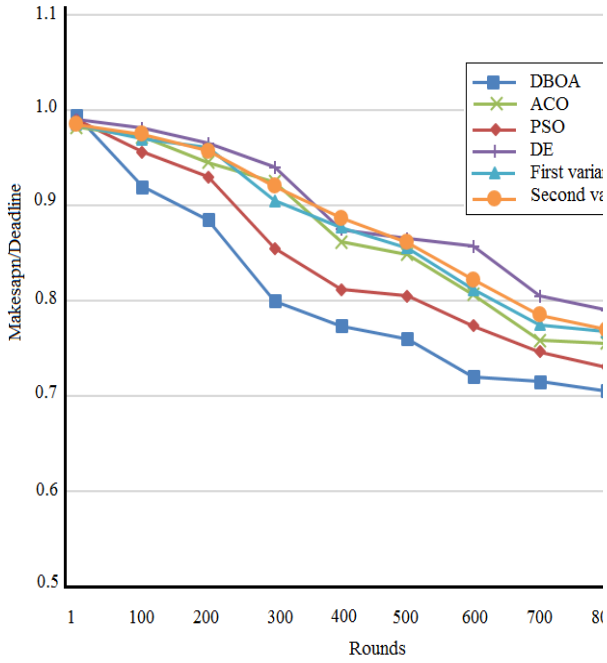


Figure 14: Ratio of the makespan/deadline in the scheduling of Montage workflow with 1000 tasks

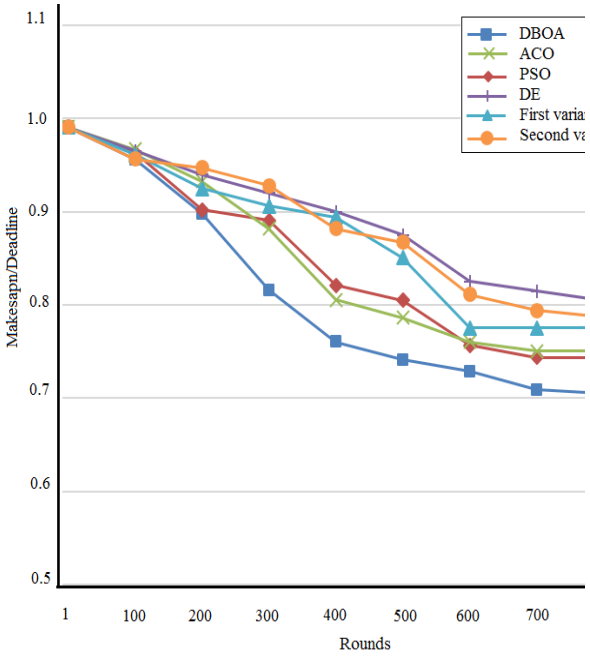


Figure 15: Ratio of the makespan/deadline in the scheduling of LIGO workflow with 1000 tasks

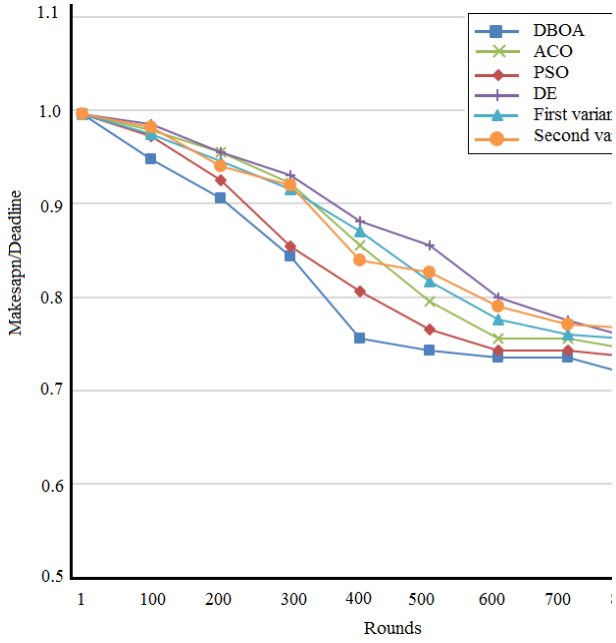


Figure 16: Ratio of the makespan/deadline in the scheduling of SIPHT workflow with 1000 tasks

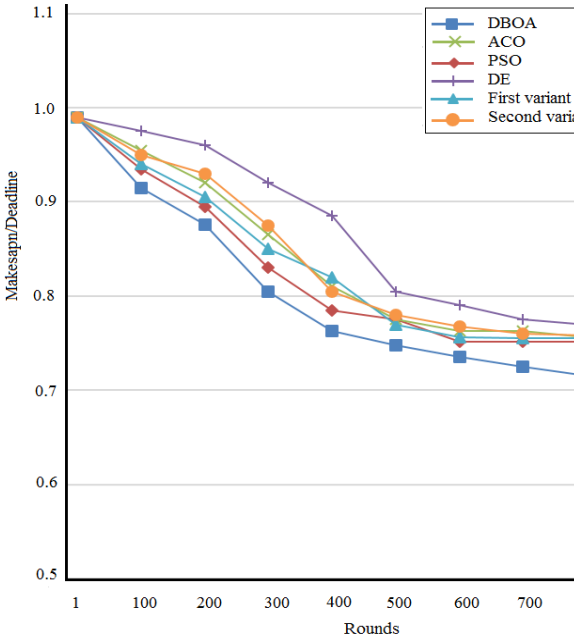


Figure 17: Ratio of the makespan/deadline in the scheduling of CyberShake workflow with 1000 tasks

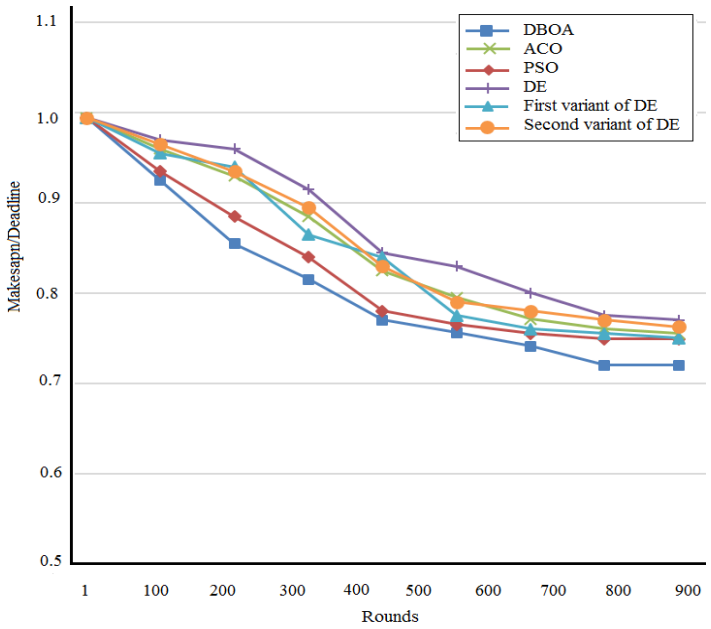


Figure 18: Ratio of the makespan/deadline in the scheduling of Epigenomics workflow with 997 tasks

V. Conclusion

Mobile Edge Computing (MEC) is an intermediate layer between the IoT and cloud computing data centers. Its resource is located at the edge of the IoT network to be used by them. Data-intensive workflows need to access several large datasets found in different cloud DCs. MEC can provide a cost-effective and low-latency computing model to deploy and run data-intensive workflows. The MECs also support storing dataset replicas, but optimizing data and replica placement and the execution of scientific workflows to mitigate data transmission delays and the energy consumption is challenging. DVFS or Dynamic voltage and frequency scaling is an exciting and useful energy management method that can benefit the MEC virtual resources to mitigate the processors' frequency and voltage to reduce their energy consumption. Designing an efficient data placement method to decrease the data access costs in data-intensive scientific workflows is highly important in the MEC environment.

This article presented DBOA, an improved and discrete version of the butterfly optimization algorithm (BOA) to solve the BOA's local optima problem and improve its convergence speed. The DBOA optimization algorithm is then used for DVFS-based data placement and data-intensive workflow scheduling in MECs to reduce VMs' energy consumption while meeting the deadlines. By allocating proper VMs for the workflow tasks and placing data fragments in the proper MEC environments, this scheme tries to minimize the data access overheads regarding the storage constraints of the MECs. For verifying the effectiveness of the proposed scheduling approach, extensive simulations are carried out on the five well-known scientific workflows with four different sizes. The results exhibited that the proposed approach can outperform scheduling solutions created using other optimization algorithms such as ACO, PSO, DE, and two variants of the DE, regarding metrics such as energy consumption, communication overheads, and makespan.

In future studies, we try to deal with the MEC environment's scheduling problem using the multi-objective DBOA algorithm. Furthermore, enhancing MEC environments with multi-cloud environments can be further investigated in future studies. Also, considering factors such as reliability and the effect of DDoS attacks on the MEC resources should be examined in the subsequent analyses.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks,* vol. 54, pp. 2787-2805, 2010.

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

[3] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review,* vol. 44, pp. 27-32, 2014.

[4] M. Yannuzzi, R. Milito, R. Serral-Gracià, D. Montero, and M. Nemirovsky, "Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing," in *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2014, pp. 325-329.

[5] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of Network and Computer Applications,* vol. 98, pp. 27-42, 2017.

[6] L. Yin, J. Luo, and H. Luo, "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing," *IEEE Transactions on Industrial Informatics,* vol. 14, pp. 4712-4721, 2018.

[7] D. Rahbari and M. Nickray, "Low-latency and energy-efficient scheduling in fog-based IoT applications," *Turkish Journal of Electrical Engineering & Computer Sciences,* vol. 27, pp. 1406-1427, 2019.

[8] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, "Fog computing for energy-aware load balancing and scheduling in smart factory," *IEEE Transactions on Industrial Informatics,* vol. 14, pp. 4548-4556, 2018.

[9] S. Arora and S. Singh, "Butterfly optimization algorithm: a novel approach for global optimization," *Soft Computing,* vol. 23, pp. 715-734, 2019.

[10] S. Arora and P. Anand, "Binary butterfly optimization approaches for feature selection," *Expert Systems with Applications,* vol. 116, pp. 147-160, 2019.

[11] M. Sharma and P. Kaur, "A Comprehensive Analysis of Nature-Inspired Meta-Heuristic Techniques for Feature Selection Problem," *Archives of Computational Methods in Engineering,* pp. 1-25, 2020.

[12] S. Arora, H. Singh, M. Sharma, S. Sharma, and P. Anand, "A new hybrid algorithm based on grey wolf optimization and crow search algorithm for unconstrained function optimization and feature selection," *Ieee Access,* vol. 7, pp. 26343-26361, 2019.

[13] M. M. Fouad, A. I. El-Desouky, R. Al-Hajj, and E.-S. M. El-Kenawy, "Dynamic group-based cooperative optimization algorithm," *IEEE Access,* vol. 8, pp. 148378-148403, 2020.

[14] S. Sharma, A. K. Saha, and S. Nama, "An Enhanced Butterfly Optimization Algorithm for Function Optimization," in *Soft Computing: Theories and Applications*, ed: Springer, 2020, pp. 593-603.

[15] M. Lei, Q. Luo, Y. Zhou, C. Tang, and Y. Gao, "BFPA: Butterfly strategy flower pollination algorithm," in *International Conference on Intelligent Computing*, 2019, pp. 739-748.

[16] S. Sharma, A. K. Saha, V. Ramasamy, J. L. Sarkar, and C. R. Panigrahi, "hBOSOS: an ensemble of butterfly optimization algorithm and symbiosis organisms search for global optimization," in *Advanced Computing and Intelligent Engineering*, ed: Springer, 2020, pp. 579-588.

[17] L. Wen and Y. Cao, "A hybrid intelligent predicting model for exploring household CO2 emissions mitigation strategies derived from butterfly optimization algorithm," *Science of The Total Environment,* vol. 727, p. 138572, 2020.

[18] A. Fathy, "Butterfly optimization algorithm based methodology for enhancing the shaded photovoltaic array extracted power via reconfiguration process," *Energy Conversion and Management,* vol. 220, p. 113115, 2020.

[19] K. Aygül, M. Cikan, T. Demirdelen, and M. Tumay, "Butterfly optimization algorithm based maximum power point tracking of photovoltaic systems under partial shading condition," *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects,* pp. 1-19, 2019.

[20] G. Li, F. Shuang, P. Zhao, and C. Le, "An improved butterfly optimization algorithm for engineering design problems using the cross-entropy method," *Symmetry,* vol. 11, p. 1049, 2019.
[21] P. P. Dey, D. C. Das, A. Latif, S. Hussain, and T. S. Ustun, "Active power Management of Virtual Power Plant under penetration of central receiver solar thermal-wind using butterfly optimization technique," *Sustainability,* vol. 12, p. 6979, 2020.
[22] M. Faheem, R. A. Butt, B. Raza, M. W. Ashraf, S. Begum, M. A. Ngadi*, et al.*, "Bio-inspired routing protocol for WSN-based smart grid applications in the context of Industry 4.0," *Transactions on Emerging Telecommunications Technologies,* vol. 30, p. e3503, 2019.
[23] S. Arora and S. Singh, "Node localization in wireless sensor networks using butterfly optimization algorithm," *Arabian Journal for Science and Engineering,* vol. 42, pp. 3325-3335, 2017.
[24] L. S. Tan, Z. Zainuddin, and P. Ong, "Wavelet neural networks based solutions for elliptic partial differential equations with improved butterfly optimization algorithm training," *Applied Soft Computing,* vol. 95, p. 106518, 2020.
[25] S. M. J. Jalali, S. Ahmadian, P. M. Kebria, A. Khosravi, C. P. Lim, and S. Nahavandi, "Evolving artificial neural networks using butterfly optimization algorithm for data classification," in *International Conference on Neural Information Processing*, 2019, pp. 596-607.
[26] G. Suciu, V. Suciu, A. Martian, R. Craciunescu, A. Vulpe, I. Marcu*, et al.*, "Big data, internet of things and cloud convergence–an architecture for secure e-health applications," *Journal of medical systems,* vol. 39, p. 141, 2015.
[27] X. Li, L. Wang, J. H. Abawajy, X. Qin, G. Pau, and I. You, "Data-Intensive Task Scheduling for Heterogeneous Big Data Analytics in IoT System," *Energies,* vol. 13, p. 4508, 2020.
[28] Z. Lv and W. Xiu, "Interaction of edge-cloud computing based on SDN and NFV for next generation IoT," *IEEE Internet of Things Journal,* 2019.
[29] B. Yang, W. K. Chai, Z. Xu, K. V. Katsaros, and G. Pavlou, "Cost-efficient NFV-enabled mobile edge-cloud for low latency mobile applications," *IEEE Transactions on Network and Service Management,* vol. 15, pp. 475-488, 2018.
[30] T.-M. Pham, "Optimization of Resource Management for NFV-Enabled IoT Systems in Edge Cloud Computing," *IEEE Access,* vol. 8, pp. 178217-178229, 2020.
[31] E. Schiller, N. Nikaein, E. Kalogeiton, M. Gasparyan, and T. Braun, "CDS-MEC: NFV/SDN-based application management for MEC in 5G systems," *Computer Networks,* vol. 135, pp. 96-107, 2018.
[32] K. Han, S. Li, S. Tang, H. Huang, S. Zhao, G. Fu*, et al.*, "Application-driven end-to-end slicing: When wireless network virtualization orchestrates with NFV-based mobile edge computing," *IEEE Access,* vol. 6, pp. 26567-26577, 2018.
[33] L. Tianze, W. Muqing, Z. Min, and L. Wenxing, "An overhead-optimizing task scheduling strategy for ad-hoc based mobile edge computing," *IEEE Access,* vol. 5, pp. 5609-5622, 2017.
[34] A. A. Al-Habob, O. A. Dobre, and A. G. Armada, "Sequential task scheduling for mobile edge computing using genetic algorithm," in *2019 IEEE Globecom Workshops (GC Wkshps)*, 2019, pp. 1-6.
[35] J. Huang, S. Li, and Y. Chen, "Revenue-optimal task scheduling and resource management for IoT batch jobs in mobile edge computing," *Peer-to-Peer Networking and Applications,* pp. 1-12, 2020.
[36] T. Zhu, T. Shi, J. Li, Z. Cai, and X. Zhou, "Task scheduling in deadline-aware mobile edge computing systems," *IEEE Internet of Things Journal,* vol. 6, pp. 4854-4866, 2018.
[37] J. Sun, L. Yin, M. Zou, Y. Zhang, T. Zhang, and J. Zhou, "Makespan-Minimization Workflow Scheduling for Complex Networks with Social Groups in Edge Computing," *Journal of Systems Architecture,* p. 101799, 2020.
[38] H. Cao, X. Xu, Q. Liu, Y. Xue, and L. Qi, "Uncertainty-aware resource provisioning for workflow scheduling in edge computing environment," in *2019 18th IEEE International Conference On*

Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), 2019, pp. 734-739.

[39] N.-N. Dao, D.-N. Vu, Y. Lee, S. Cho, C. Cho, and H. Kim, "Pattern-identified online task scheduling in multitier edge computing for industrial IoT services," *Mobile Information Systems,* vol. 2018, 2018.

[40] Y. Li, Y. Ma, and Z. Zeng, "A Novel Approach to Location-Aware Scheduling of Workflows Over Edge Computing Resources," *International Journal of Web Services Research (IJWSR),* vol. 17, pp. 56-68, 2020.

[41] Y. Shao, C. Li, and H. Tang, "A data replica placement strategy for IoT workflows in collaborative edge and cloud environments," *Computer networks,* vol. 148, pp. 46-59, 2019.

[42] Y. Shao, C. Li, Z. Fu, L. Jia, and Y. Luo, "Cost-effective replication management and scheduling in edge computing," *Journal of Network and Computer Applications,* vol. 129, pp. 46-61, 2019.

[43] M. Breitbach, D. Schäfer, J. Edinger, and C. Becker, "Context-aware data and task placement in edge computing environments," in *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom*, 2019, pp. 1-10.

[44] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *2016 IEEE International Symposium on Information Theory (ISIT)*, 2016, pp. 1451-1455.

[45] Z. Wang, Z. Zhao, G. Min, X. Huang, Q. Ni, and R. Wang, "User mobility aware task assignment for mobile edge computing," *Future Generation Computer Systems,* vol. 85, pp. 1-8, 2018.

[46] C. Li, J. Bai, and J. Tang, "Joint optimization of data placement and scheduling for improving user experience in edge computing," *Journal of Parallel and Distributed Computing,* vol. 125, pp. 93-105, 2019.

[47] B. Lin, F. Zhu, J. Zhang, J. Chen, X. Chen, N. N. Xiong*, et al.*, "A time-driven data placement strategy for a scientific workflow combining edge computing and cloud computing," *IEEE Transactions on Industrial Informatics,* vol. 15, pp. 4254-4265, 2019.

[48] M. François, T. Grosges, D. Barchiesi, and R. Erra, "Pseudo-random number generator based on mixing of three chaotic maps," *Communications in Nonlinear Science and Numerical Simulation,* vol. 19, pp. 887-895, 2014.

[49] J. A. González and R. Pino, "A random number generator based on unpredictable chaotic functions," *Computer Physics Communications,* vol. 120, pp. 109-114, 1999.

[50] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, and K. Li, "An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment," *Journal of Grid Computing,* vol. 14, pp. 55-74, 2016.

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks,* vol. 54, pp. 2787-2805, 2010.

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

[3] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review,* vol. 44, pp. 27-32, 2014.

[4] M. Yannuzzi, R. Milito, R. Serral-Gracià, D. Montero, and M. Nemirovsky, "Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing," in *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2014, pp. 325-329.

[5] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of Network and Computer Applications,* vol. 98, pp. 27-42, 2017.

[6] L. Yin, J. Luo, and H. Luo, "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing," *IEEE Transactions on Industrial Informatics,* vol. 14, pp. 4712-4721, 2018.

[7] D. Rahbari and M. Nickray, "Low-latency and energy-efficient scheduling in fog-based IoT applications," *Turkish Journal of Electrical Engineering & Computer Sciences,* vol. 27, pp. 1406-1427, 2019.

[8] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, "Fog computing for energy-aware load balancing and scheduling in smart factory," *IEEE Transactions on Industrial Informatics,* vol. 14, pp. 4548-4556, 2018.

[9] S. Arora and S. Singh, "Butterfly optimization algorithm: a novel approach for global optimization," *Soft Computing,* vol. 23, pp. 715-734, 2019.

[10] S. Arora and P. Anand, "Binary butterfly optimization approaches for feature selection," *Expert Systems with Applications,* vol. 116, pp. 147-160, 2019.

[11] M. Sharma and P. Kaur, "A Comprehensive Analysis of Nature-Inspired Meta-Heuristic Techniques for Feature Selection Problem," *Archives of Computational Methods in Engineering,* pp. 1-25, 2020.

[12] S. Arora, H. Singh, M. Sharma, S. Sharma, and P. Anand, "A new hybrid algorithm based on grey wolf optimization and crow search algorithm for unconstrained function optimization and feature selection," *Ieee Access,* vol. 7, pp. 26343-26361, 2019.

[13] M. M. Fouad, A. I. El-Desouky, R. Al-Hajj, and E.-S. M. El-Kenawy, "Dynamic group-based cooperative optimization algorithm," *IEEE Access,* vol. 8, pp. 148378-148403, 2020.

[14] S. Sharma, A. K. Saha, and S. Nama, "An Enhanced Butterfly Optimization Algorithm for Function Optimization," in *Soft Computing: Theories and Applications*, ed: Springer, 2020, pp. 593-603.

[15] M. Lei, Q. Luo, Y. Zhou, C. Tang, and Y. Gao, "BFPA: Butterfly strategy flower pollination algorithm," in *International Conference on Intelligent Computing*, 2019, pp. 739-748.

[16] S. Sharma, A. K. Saha, V. Ramasamy, J. L. Sarkar, and C. R. Panigrahi, "hBOSOS: an ensemble of butterfly optimization algorithm and symbiosis organisms search for global optimization," in *Advanced Computing and Intelligent Engineering*, ed: Springer, 2020, pp. 579-588.

[17] L. Wen and Y. Cao, "A hybrid intelligent predicting model for exploring household CO2 emissions mitigation strategies derived from butterfly optimization algorithm," *Science of The Total Environment,* vol. 727, p. 138572, 2020.

[18] A. Fathy, "Butterfly optimization algorithm based methodology for enhancing the shaded photovoltaic array extracted power via reconfiguration process," *Energy Conversion and Management,* vol. 220, p. 113115, 2020.

[19] K. Aygül, M. Cikan, T. Demirdelen, and M. Tumay, "Butterfly optimization algorithm based maximum power point tracking of photovoltaic systems under partial shading condition," *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects,* pp. 1-19, 2019.

[20] G. Li, F. Shuang, P. Zhao, and C. Le, "An improved butterfly optimization algorithm for engineering design problems using the cross-entropy method," *Symmetry,* vol. 11, p. 1049, 2019.

[21] P. P. Dey, D. C. Das, A. Latif, S. Hussain, and T. S. Ustun, "Active power Management of Virtual Power Plant under penetration of central receiver solar thermal-wind using butterfly optimization technique," *Sustainability,* vol. 12, p. 6979, 2020.

[22] M. Faheem, R. A. Butt, B. Raza, M. W. Ashraf, S. Begum, M. A. Ngadi, and V. C. Gungor, "Bio-inspired routing protocol for WSN-based smart grid applications in the context of Industry 4.0," *Transactions on Emerging Telecommunications Technologies,* vol. 30, p. e3503, 2019.

[23] S. Arora and S. Singh, "Node localization in wireless sensor networks using butterfly optimization algorithm," *Arabian Journal for Science and Engineering,* vol. 42, pp. 3325-3335, 2017.

[24] L. S. Tan, Z. Zainuddin, and P. Ong, "Wavelet neural networks based solutions for elliptic partial differential equations with improved butterfly optimization algorithm training," *Applied Soft Computing,* vol. 95, p. 106518, 2020.

[25] S. M. J. Jalali, S. Ahmadian, P. M. Kebria, A. Khosravi, C. P. Lim, and S. Nahavandi, "Evolving artificial neural networks using butterfly optimization algorithm for data classification," in *International Conference on Neural Information Processing*, 2019, pp. 596-607.

[26] X. Li, L. Wang, J. H. Abawajy, X. Qin, G. Pau, and I. You, "Data-Intensive Task Scheduling for Heterogeneous Big Data Analytics in IoT System," *Energies,* vol. 13, p. 4508, 2020.

[27] Z. Lv and W. Xiu, "Interaction of edge-cloud computing based on SDN and NFV for next generation IoT," *IEEE Internet of Things Journal,* 2019.

[28] B. Yang, W. K. Chai, Z. Xu, K. V. Katsaros, and G. Pavlou, "Cost-efficient NFV-enabled mobile edge-cloud for low latency mobile applications," *IEEE Transactions on Network and Service Management,* vol. 15, pp. 475-488, 2018.

[29] T.-M. Pham, "Optimization of Resource Management for NFV-Enabled IoT Systems in Edge Cloud Computing," *IEEE Access,* vol. 8, pp. 178217-178229, 2020.

[30] E. Schiller, N. Nikaein, E. Kalogeiton, M. Gasparyan, and T. Braun, "CDS-MEC: NFV/SDN-based application management for MEC in 5G systems," *Computer Networks,* vol. 135, pp. 96-107, 2018.

[31] K. Han, S. Li, S. Tang, H. Huang, S. Zhao, G. Fu, and Z. Zhu, "Application-driven end-to-end slicing: When wireless network virtualization orchestrates with NFV-based mobile edge computing," *IEEE Access,* vol. 6, pp. 26567-26577, 2018.

[32] G. Suciu, V. Suciu, A. Martian, R. Craciunescu, A. Vulpe, I. Marcu, S. Halunga, and O. Fratu, "Big data, internet of things and cloud convergence–an architecture for secure e-health applications," *Journal of medical systems,* vol. 39, p. 141, 2015.

[33] L. Tianze, W. Muqing, Z. Min, and L. Wenxing, "An overhead-optimizing task scheduling strategy for ad-hoc based mobile edge computing," *IEEE Access,* vol. 5, pp. 5609-5622, 2017.

[34] A. A. Al-Habob, O. A. Dobre, and A. G. Armada, "Sequential task scheduling for mobile edge computing using genetic algorithm," in *2019 IEEE Globecom Workshops (GC Wkshps)*, 2019, pp. 1-6.

[35] J. Huang, S. Li, and Y. Chen, "Revenue-optimal task scheduling and resource management for IoT batch jobs in mobile edge computing," *Peer-to-Peer Networking and Applications,* pp. 1-12, 2020.

[36] Y. Shao, C. Li, and H. Tang, "A data replica placement strategy for IoT workflows in collaborative edge and cloud environments," *Computer networks,* vol. 148, pp. 46-59, 2019.

[37] Y. Shao, C. Li, Z. Fu, L. Jia, and Y. Luo, "Cost-effective replication management and scheduling in edge computing," *Journal of Network and Computer Applications,* vol. 129, pp. 46-61, 2019.

[38] M. Breitbach, D. Schäfer, J. Edinger, and C. Becker, "Context-aware data and task placement in edge computing environments," in *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom*, 2019, pp. 1-10.

[39] T. Zhu, T. Shi, J. Li, Z. Cai, and X. Zhou, "Task scheduling in deadline-aware mobile edge computing systems," *IEEE Internet of Things Journal,* vol. 6, pp. 4854-4866, 2018.

[40] J. Sun, L. Yin, M. Zou, Y. Zhang, T. Zhang, and J. Zhou, "Makespan-Minimization Workflow Scheduling for Complex Networks with Social Groups in Edge Computing," *Journal of Systems Architecture,* p. 101799, 2020.

[41] H. Cao, X. Xu, Q. Liu, Y. Xue, and L. Qi, "Uncertainty-aware resource provisioning for workflow scheduling in edge computing environment," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2019, pp. 734-739.

[42] N.-N. Dao, D.-N. Vu, Y. Lee, S. Cho, C. Cho, and H. Kim, "Pattern-identified online task scheduling in multitier edge computing for industrial IoT services," *Mobile Information Systems,* vol. 2018, 2018.

[43] Y. Li, Y. Ma, and Z. Zeng, "A Novel Approach to Location-Aware Scheduling of Workflows Over Edge Computing Resources," *International Journal of Web Services Research (IJWSR),* vol. 17, pp. 56-68, 2020.

[44] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *2016 IEEE International Symposium on Information Theory (ISIT)*, 2016, pp. 1451-1455.

[45] Z. Wang, Z. Zhao, G. Min, X. Huang, Q. Ni, and R. Wang, "User mobility aware task assignment for mobile edge computing," *Future Generation Computer Systems,* vol. 85, pp. 1-8, 2018.

[46] C. Li, J. Bai, and J. Tang, "Joint optimization of data placement and scheduling for improving user experience in edge computing," *Journal of Parallel and Distributed Computing,* vol. 125, pp. 93-105, 2019.

[47] B. Lin, F. Zhu, J. Zhang, J. Chen, X. Chen, N. N. Xiong, and J. L. Mauri, "A time-driven data placement strategy for a scientific workflow combining edge computing and cloud computing," *IEEE Transactions on Industrial Informatics,* vol. 15, pp. 4254-4265, 2019.

[48] M. François, T. Grosges, D. Barchiesi, and R. Erra, "Pseudo-random number generator based on mixing of three chaotic maps," *Communications in Nonlinear Science and Numerical Simulation,* vol. 19, pp. 887-895, 2014.

[49] J. A. González and R. Pino, "A random number generator based on unpredictable chaotic functions," *Computer Physics Communications,* vol. 120, pp. 109-114, 1999.

[50] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, and K. Li, "An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment," *Journal of Grid Computing,* vol. 14, pp. 55-74, 2016.

[51] C.-M. Wu, R.-S. Chang, and H.-Y. Chan, "A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters," *Future Generation Computer Systems,* vol. 37, pp. 141-147, 2014.

[52] H. Zhao and R. Sakellariou, "An experimental investigation into the rank function of the heterogeneous earliest finish time scheduling algorithm," in *European Conference on Parallel Processing*, 2003, pp. 189-194.

[53] G. Sun, B. Yang, Z. Yang, and G. Xu, "An adaptive differential evolution with combined strategy for global numerical optimization," *Soft Computing,* pp. 1-20, 2019.

[54] J. Liu, C. Wu, G. Wu, and X. Wang, "A novel differential search algorithm and applications for structure design," *Applied Mathematics and Computation,* vol. 268, pp. 246-269, 2015.

Figures

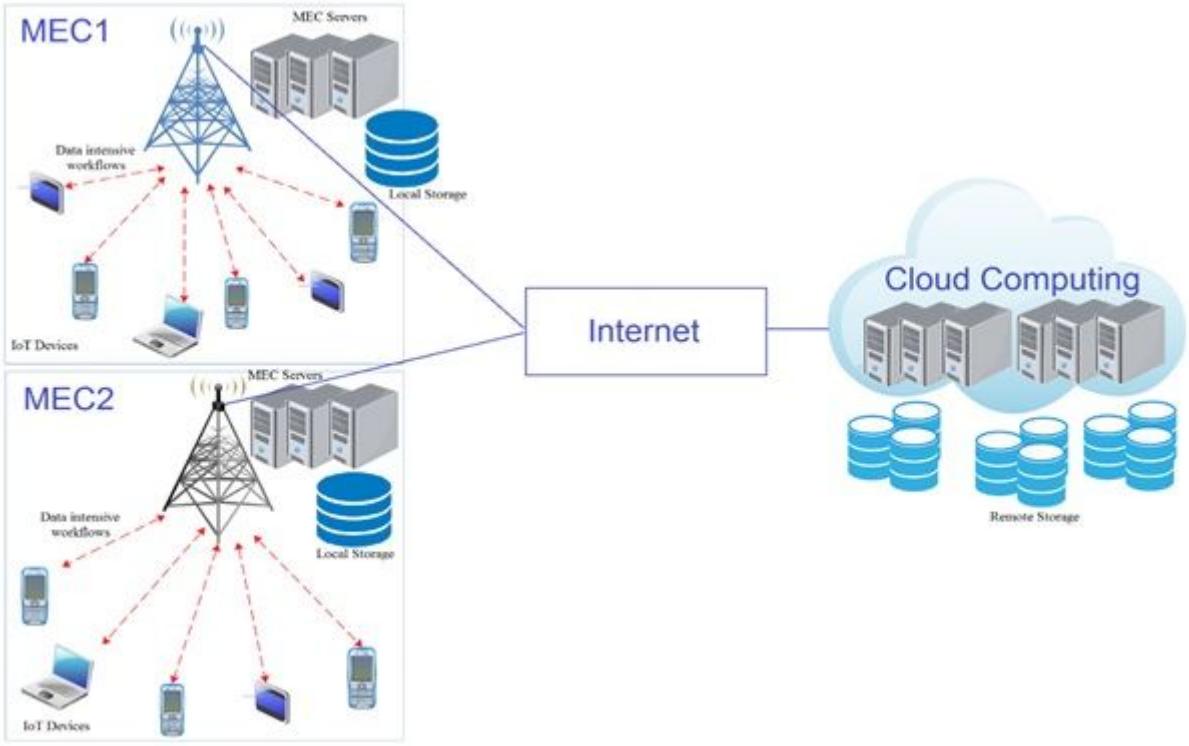


Figure 1

MEC architecture

```
Produce initial solutions containing n butterflies
Compute their Intensity
Define c, a, p, sensor modality, power exponent, and modality.
For i=1 to Max_iteration
    Compute the fragrance of the population
    Find the best butterfly
    For each butterfly do
        Produce a random number called r
        If r < p Then
            Go towards the best butterfly using Equation 2
        Else
            Make a random move using Equation 3
        End
    Next
    Update the power exponent a
Next i
Return the best solution
```

Figure 2

BOA pseudo-code

Function Discrete BOA (DBOA)

Input: Number of solutions
Maximum number of iterations
Output: The best solution

For i= 1 : *Npopulation*
For j=1 to *Ntask* step 2
$X_j^i = Floor(Nvm * Chaotic\_MAP())$
$X_j^i = Floor(Ndvfs * Chaotic\_MAP())$
End
For j=1 to *Nreplica*
$X_j^i = Floor(Nmec * Chaotic\_MAP())$
End
End

Compute their Intensity
Define *c, a, p,* sensor modality, power exponent, and modality.

For *i*=1 to *Max_iteration*
Compute the fragrance of the population
Find the best butterfly
For each butterfly **do**
Produce a random number called *r*
If $r < p$ **Then**
Go towards the best butterfly using
Equation 11 to 15
Else
If *randLevy* > *0.5*
Make a random move using Equation
16 to 19
Else
Make a random move based on the
Levy Flight and using Equation 26
End
End
Next
Update the power exponent *a*
Next *i*
Return the best solution

Figure 3

DBOA pseudo-code

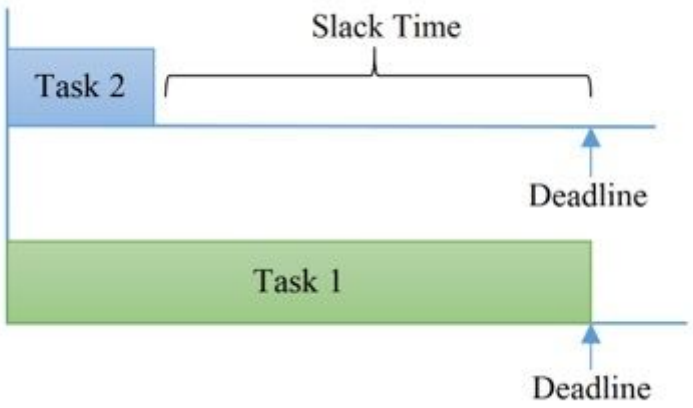


Figure 4

Slack time

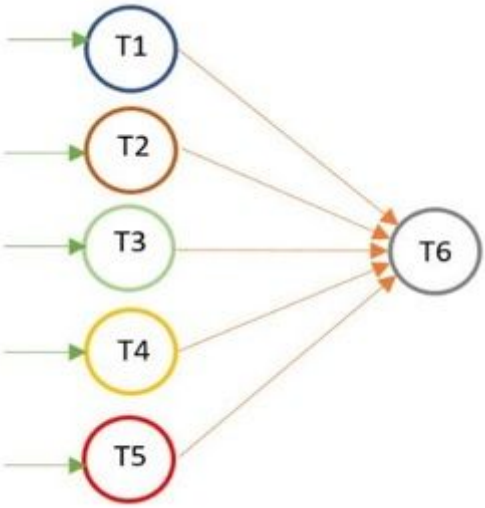


Figure 5

Data aggregation point in a workflow

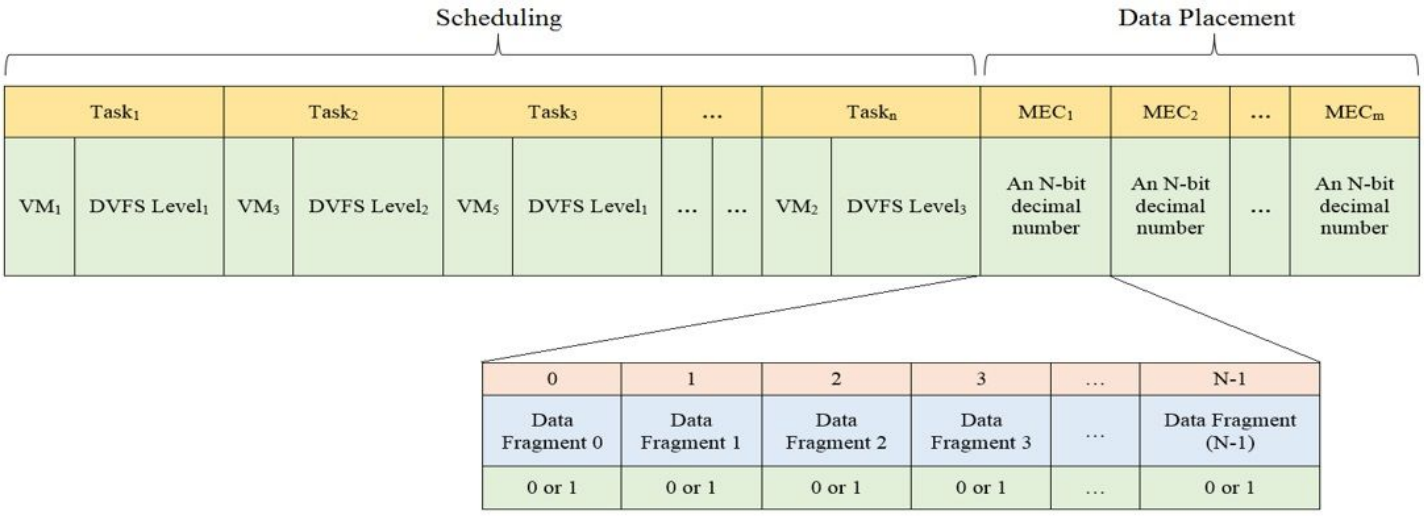


Figure 6

A sample encoding in the proposed scheduling scheme

```
Procedure MEC_Scheduling()

Input: Workflow wi
       DVFS levels
       Deadline
       MEC environments specifications

Determine the number of VMs in the MEC environment
Determine the number of DVFS levels
Determine the workflow deadline

Read workflow data from its DAX file
Compute the rank of workflow tasks using the HEFT ranking method
Sort tasks in increasing order of their rank
Compute the workflow slack time
Allocate the slack time to each workflow levels
Set the objective function
Use the DBOA optimization algorithm to obtain the best solution
For each  task in workflow
    Find the VMi and DVFS_Leveli from the best solution
    Find the predecessor set of Ti
    If  Ti's predecessor is empty  Then
        While ( VMi is not idle )
            Wait
        End
    Else
        While ( VMi is not idle  or all Ti's predecessor are not executed )
            Wait
        End
    End
    Allocate the DVFS levels to the VMs
    Execute task Ti on the VMi

End
```

Figure 7

DVFS-based workflow scheduling using DBOA

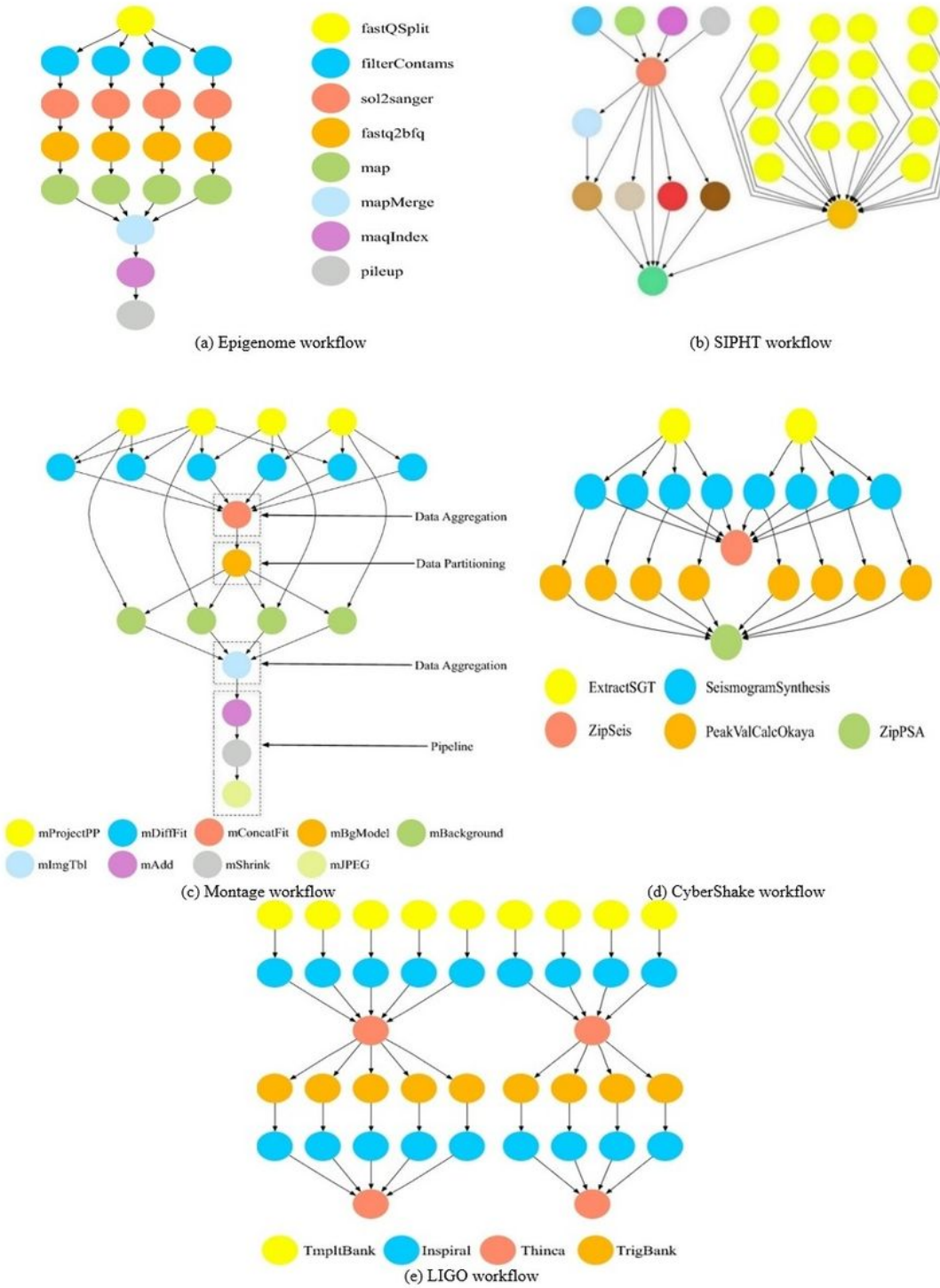


Figure 8

Five well-known scientific workflows

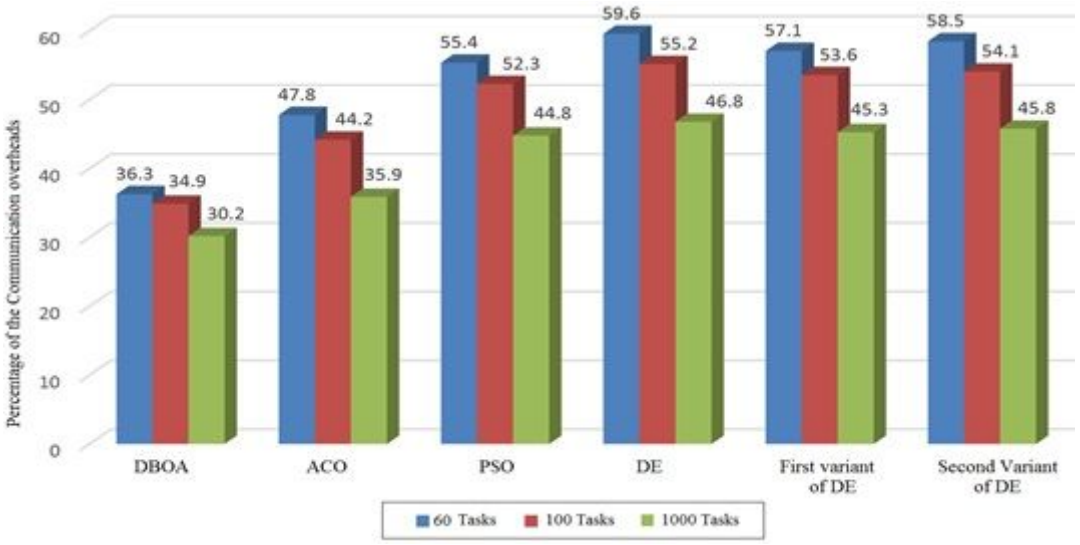


Figure 9

Percentage of the communication overheads ratio in the LIGO workflows

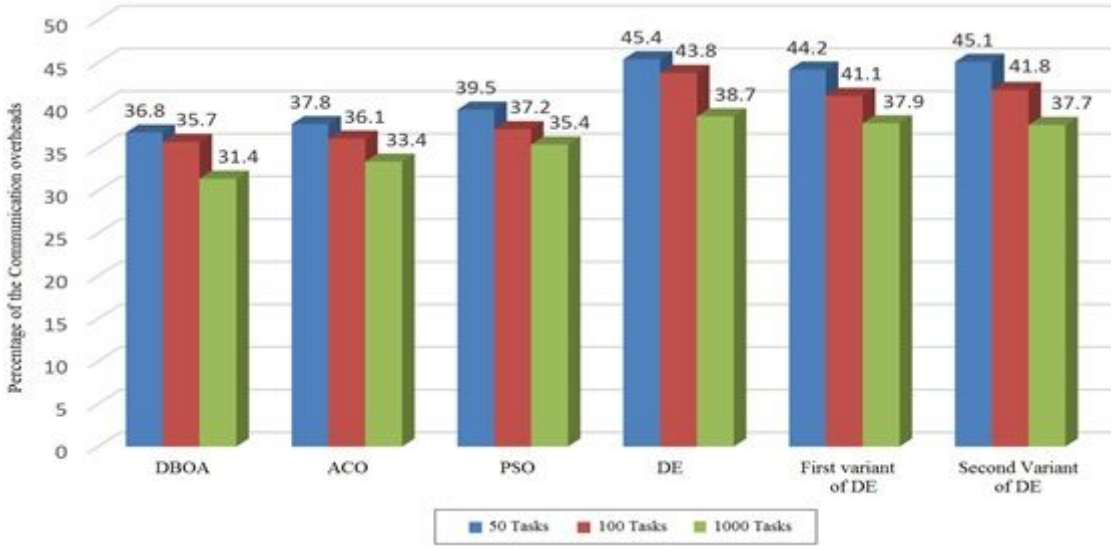


Figure 10

Percentage of the communication overheads ratio in the CyberShake workflows

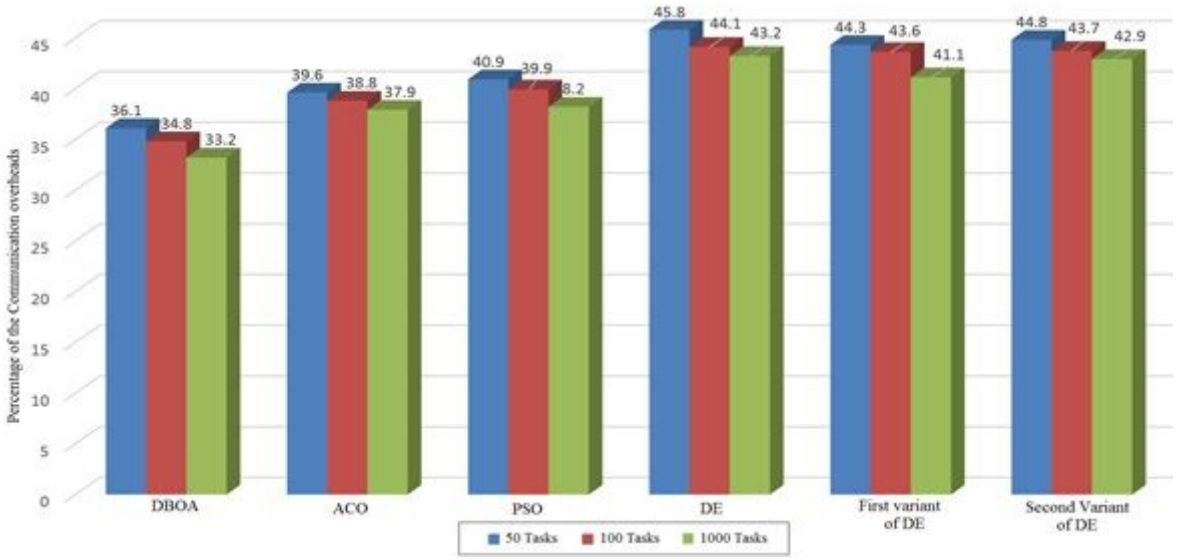


Figure 11

Percentage of the communication overheads ratio in the Montage workflows

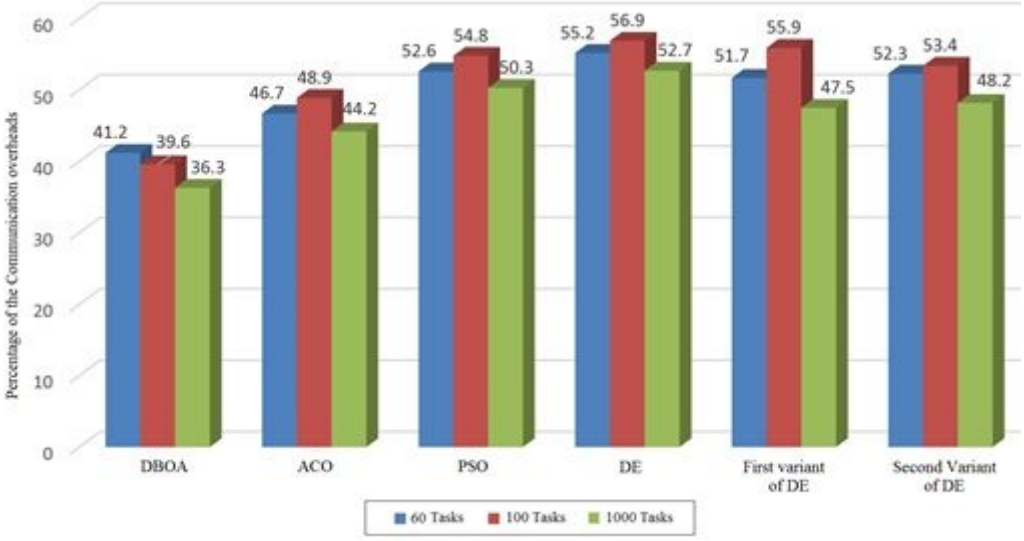


Figure 12

Percentage of the communication overheads ratio in the SIPTH workflows

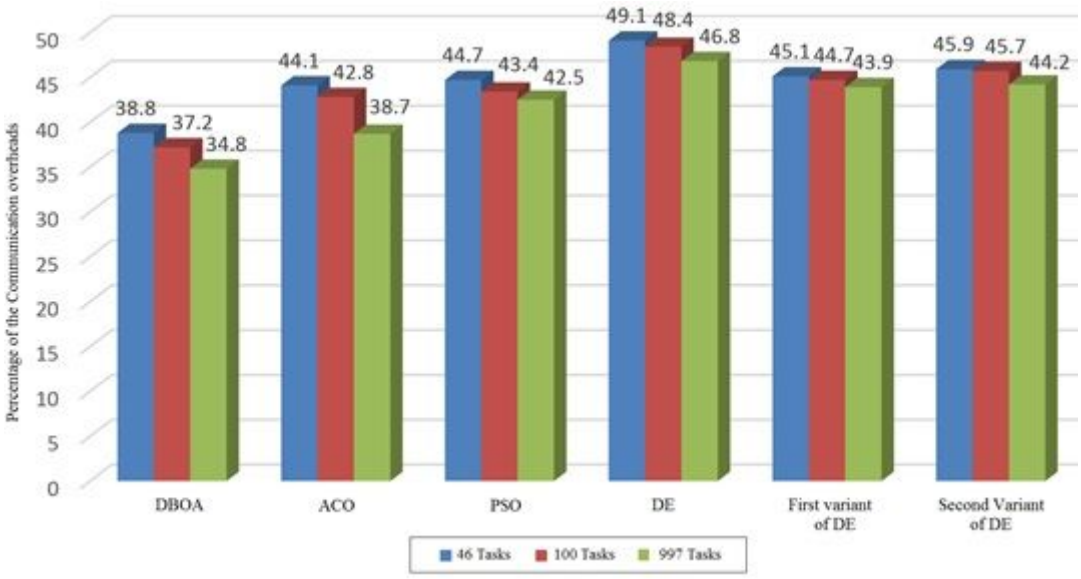


Figure 13

Percentage of the communication overheads ratio in the Epigenomics workflows

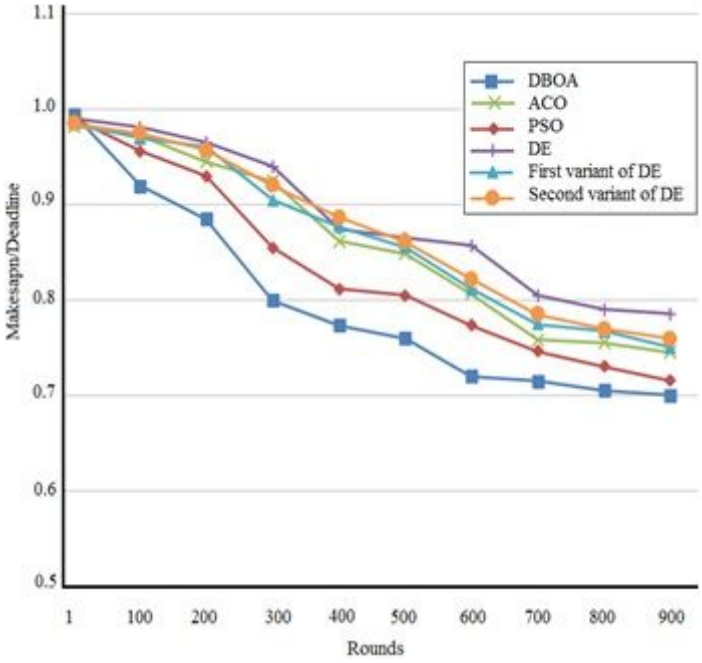


Figure 14

Ratio of the makespan/deadline in the scheduling of Montage workflow with 1000 tasks

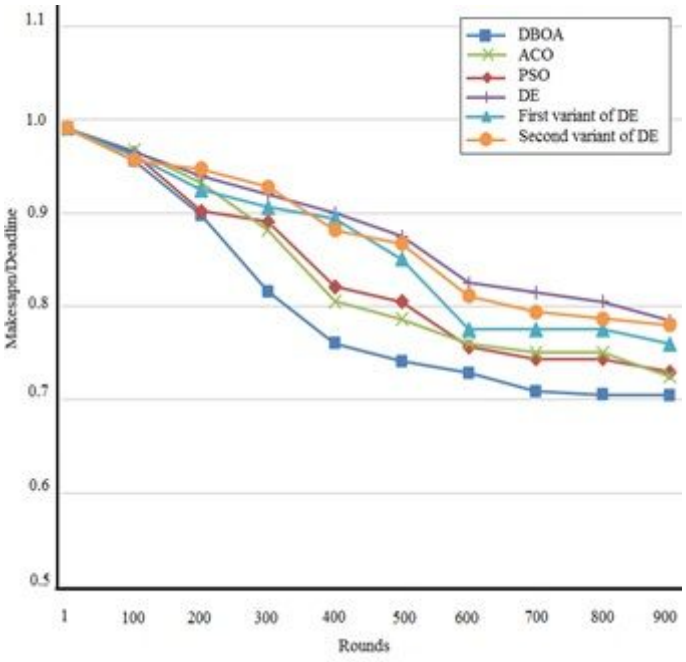


Figure 15

Ratio of the makespan/deadline in the scheduling of LIGO workflow with 1000 tasks

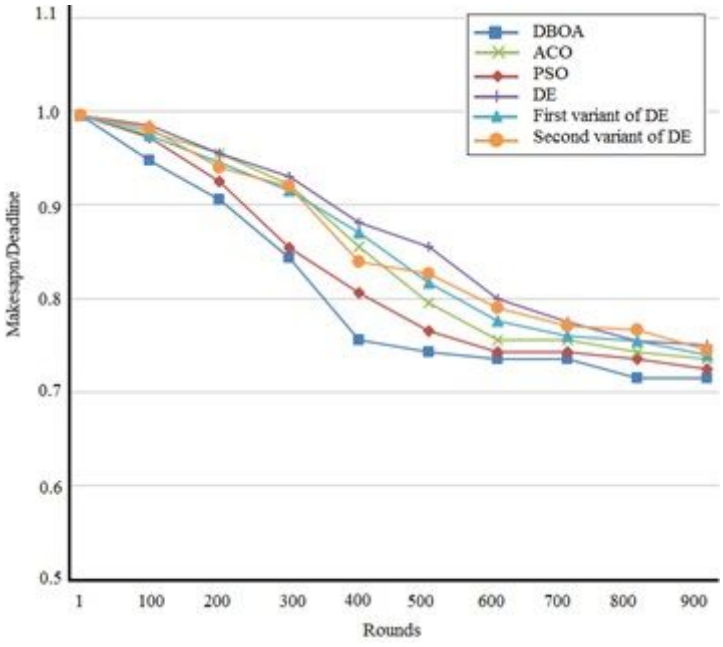


Figure 16

Ratio of the makespan/deadline in the scheduling of SIPHT workflow with 1000 tasks

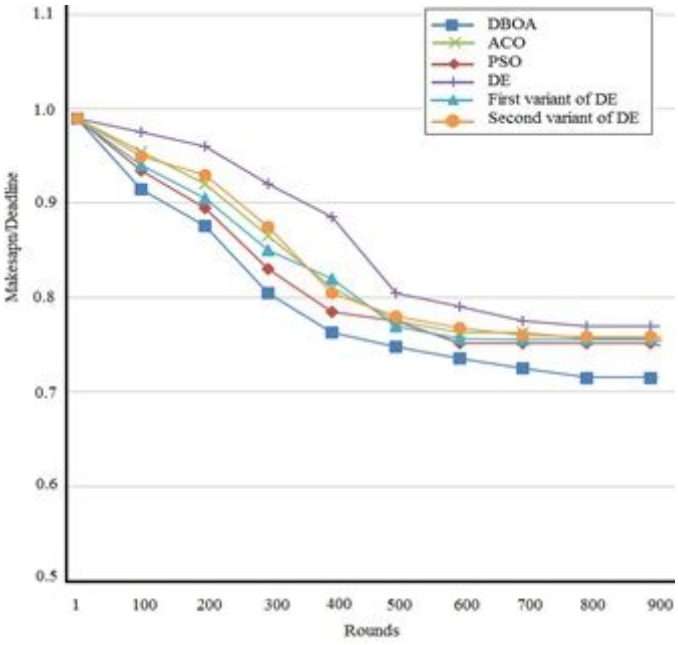


Figure 17

Ratio of the makespan/deadline in the scheduling of CyberShake workflow with 1000 tasks

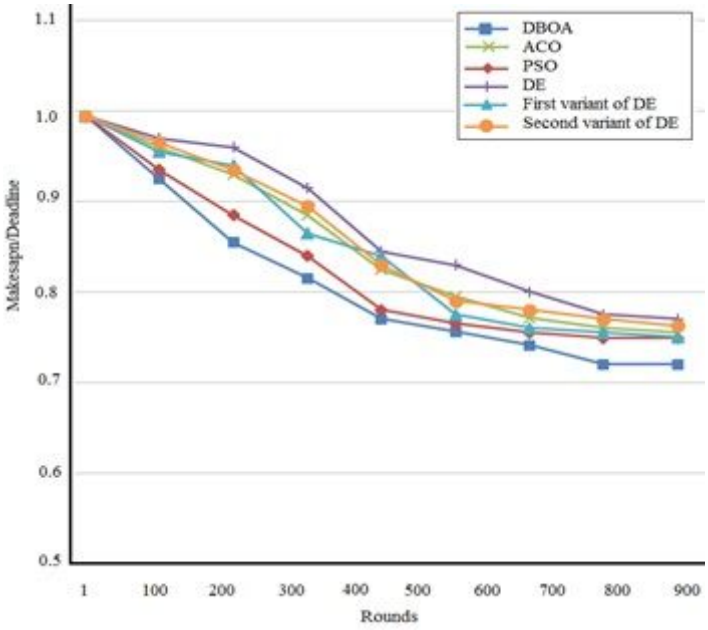


Figure 18

Ratio of the makespan/deadline in the scheduling of Epigenomics workflow with 997 tasks