Egyptian Informatics Journal 24 (2023) 277–290

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Energy-aware intelligent scheduling for deadline-constrained workflows in sustainable cloud computing

Min Cao [a], Yaoyu Li [b,*], Xupeng Wen [c], Yue Zhao [b,*], Jianghan Zhu [b]

[a] Intelligent Manufacturing College, Zhanjiang University of Science and Technology, Zhanjiang 524094, PR China
[b] Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, PR China
[c] School of Traffic and Transportation Engineering, Central South University, Changsha 410075, PR China

ARTICLE INFO

Article history:
Received 21 December 2022
Revised 19 March 2023
Accepted 3 April 2023

Keywords:
Sustainable cloud computing
Intelligent scheduling
Energy-efficiency
Workflow
Dynamic voltage/frequency scaling

ABSTRACT

It is challenging to handle the non-linear power consumption model, complex workflow structures, and diverse user-defined deadlines for energy-efficient workflow scheduling in sustainable cloud computing. Although metaheuristics are very attractive to solve this problem, most of the existing work regards the problem as a black-box and ignores the use of domain knowledge. To make up for their shortcomings, this paper tailors an energy-aware intelligent scheduling algorithm (EIS) with three new mechanisms. First, we derive the optimal execution time that minimizes energy consumption for each task on a given resource. Second, based on the optimal execution time of each workflow task, the EIS distributes the workflow slack time (difference between its completion time and deadline) to reduce the voltages and frequencies of task executions for energy saving. Third, the EIS mines the idle time gaps caused by task precedence constraints to further reduce dynamic energy consumption whilst satisfying workflows' deadline constraints. To measure the performance of the EIS, we conduct extensive comparison experiments based on actual workflow applications. The results demonstrate that the energy consumption of the EIS is much lower than that of the competitors under different deadlines, and has a faster descend rate with the evolution process.

1. Introduction

1.1. Context and issues

Cloud computing is an innovative development of distributed and utility computing [1,2]. It provides on-demand access to unlimited virtual resources such as network, storage, and computing on a pay-as-you-go basis. This flexible and scalable resource delivery paradigm is attractive to a wide range of individuals and various organizations [3–5]. According to Gartner's report, more than 60 percent of organizations resort to cloud services for required resources, and cloud computing alone accounts for nearly 15 percent of global IT expenditure [6].

To match the prosperously growing demand for cloud services, mainstream cloud service providers, e.g., Google Cloud, Amazon EC2, and Alibaba Cloud, have built a large number of super-scale datacentres around the world. A cloud datacentre often houses thousands of heterogeneous servers, network devices sensors, cooling systems, and many other facilities [7]. It is a typical high power density infrastructure, consuming enormous amounts of electrical energy [8–10]. Statistics illustrate that the energy consumption of a medium-sized datacentre reaches the energy consumption of 25,000 households [11]. Such high energy consumption of cloud datacentres subsequently increases operational costs, and leaves substantial carbon footprints that negatively pound the environmental sustainability [12]. Amazon's evaluation of its datacentres reveals that 42% of the operation and maintenance costs are caused by energy consumption [13]. In cloud datacentres, the energy consumed by computing and cooling systems together accounts for 85% of total energy consumption [8]. Meanwhile, the energy consumption of the cooling system mainly stem from dissipating the heat emitted by the computing

* Corresponding authors.
E-mail addresses: Garett_1984@hotmail.com (Y. Li), zhaoyue08a@nudt.edu.cn (Y. Zhao).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.eij.2023.04.002
1110-8665/

system. Thus, developing application oriented energy-aware optimization technologies is of upmost importance to cloud datacentres toward reducing operational cost and maintaining service sustainability [14,12,15].

Over the past decade, the widespread implementation of Internet of Things has resulted in the rapid growth of data volume and data generation speed [8,16]. To process the huge amounts of data automatically and timely, workfow has become a powerful tool for cloud platforms [17]. A workfow refers to a set of data processing tasks together with dependencies, which fulfil data communication for various scientifc and business applications. It is noteworthy that the workflow scheduling component is a middleware layer of a cloud platform, and it directly determines its energy consumption, other performance metrics, and service experience of users. In general, workflow application oriented energy-aware scheduling in clouds involves mapping tasks to resources, arranging tasks' execution order, and assigning appropriate task execution time, such optimizing energy consumption and satisfying complex constraints. It is a NP-complete problem, and has been researched extensively in the literature [18,19]. Most of the existing literature on energy-aware workflow scheduling in clouds can be roughly divided into two categories: heuristics and metaheuristics.

1.2. Literature and motivations

Heuristic rule-based workflow scheduling approaches are typically classified as: cluster-based, replication-based, and list-based scheduling heuristics. Cluster-based heuristics [20,21] are to group workflow tasks into multiple clusters according to certain criteria, and then map all tasks of the same cluster to the same resource. The replication-based algorithms [22,23] replicate the one task on multiple resources, hence decreasing data transmission overhead among workflow tasks and start time of tasks. Most heuristic approaches are based on the list strategy. They arrange all the workflow tasks based on certain priorities and then schedule tasks one by one onto suitable resources. For instance, Lee et al. [24] suggested two energy-conscious workflow scheduling methods using dynamic voltage scaling to balance makespan and energy consumption. Li et al. [25] designed a resource selection, task merging, and resouce reuse mechanism to reduce execution cost and energy consumption while meeting workflows' deadlines. Safari et al. [26] integrated the dynamic voltage/frequency scaling technique into the list-based scheduling algorithm to minimize energy consumption while considering workflow deadlines. Qureshi et al. [27] suggested a profile-based energy-efficient approach to balance power usage, CPU utilization, and monetary cost. Rani et al. [28] proposed a power and temperature-aware scheduling algorithm to minimize the computing and cooling energy of executing workflow tasks along with meeting deadlines. Fan et al. [29] designed a hybrid workflow scheduling approach to optimize energy consumption and resource utilization while meeting the deadline and data-dependency constraints of workflows. As heuristics are often tailored for specific scenarios, their generalization capability is insufficient, especially for dealing with complex structured workflows and nonlinear power consumption model.

Metaheuristics depending on stochastic search techniques are attractive for handling complex optimization problems. In recent years, many metaheuristic algorithms [30–32] have been designed to optimize the energy efficiency of workflow execution in cloud platforms. For instance, Gill et al. [12] suggested a cuckoo optimization-based cloud resource management approach to holistically cut down energy consumption and carbon footprints for cloud data centers, while satisfying service reliability. Tarafdar et al. [33] combined a heuristic search and positive feedback mechanism into the ant colony optimization to improve energy-efficiency and task schedulability. Malik et al. [34] employed particle swarm optimization to iteratively optimize energy efficiency and resource utilization for virtualized data centers. Li et al. [35] improved the nondominated-sorting Owl optimization algorithm with a chaotic local search to balance the energy consumption, makespan, and cost, while meeting the pre-specified deadline and budget constraints. Qi et al. [36] formalized virtual machine scheduling in cloud platforms as a multi-objective optimization problem, and improved NSGA-II to balance consumption, downtime, and resource utilization. Hussain et al. [37] designed an energy-aware approach including new task sequencing, variable neighborhood search, and resource searching mechanism to schedule deadline-constrained workflows. Domanal et al. [38] hybridised ant colony optimization and particle swarm optimization for task scheduling and resource management. These metaheuristics follow a completely black-box approach and fail to make effective use of domain knowledge, such as workflow structure, power consumption model, and dynamic voltage/frequency scaling technique. Modern cloud processors using a multi-core architecture commonly integrate voltage regulators for each core [39], which enables per-core dynamic voltage/frequency scaling to dynamically adjust the supply voltages and operation frequencies by taking actual workloads into account.

Meanwhile, there exists substantial work on intelligent algorithms to optimize the energy consumption of executing a set of dependent tasks in other fields [40–42]. For instance, Wang et al. [43] developed a cooperative memetic algorithm to simultaneously optimize energy consumption and delay. This algorithm includes two heuristics to initialize the population, a feedback-based cooperative search mechanism, multiple problem-specific evolutionary operators, and multiple environmental selection strategies. To solve distributed assembly flow-shop scheduling problem, Zhao et al. [44] improve the water wave optimization algorithm with a variable neighborhood search and a reinforcement learning mechanism. Zhao et al. [45] proposed a hyperheuristic with Q-learning to solve the energy-efficient distributed blocking flow shop scheduling problem. Wang et al. [46] suggested a hybrid adaptive differential evolution algorithm to optimize the completion time, delay time, and energy consumption for job-shop scheduling problems. Pan et al. [47] designed a knowledge-based two-population optimization algorithm to reduce energy consumption and tardiness for distributed energy-efficient scheduling problems. These works provide great inspiration for this study. However, the optimization problems in these fields are quite different from the energy-efficient scheduling of cloud computing workflows, such as the dynamic voltage/frequency scaling technique of computing resources and the on-demand supply of cloud resources. Thus, these intelligent algorithms cannot be directly applied.

1.3. Main contributions

The deficiencies of the existing works drive us to integrate the domain knowledge of cloud workflow execution into the evolutionary optimization framework to improve energy efficiency. Our core contributions are below.

- We derive the optimal execution time that minimizes energy consumption for each workflow task by considering the nonlinear power consumption model.
- Based on the optimal execution time of each task, we propose a mechanism to distribute workflow slack time (difference between its completion time and deadline) among tasks with complex structures for energy conservation by adjusting the voltage and frequency.

- The idle time gaps caused by task precedence constraints are excavated to further reduce dynamic energy consumption whilst satisfying workflows' deadline constraints.
- We conduct comparison experiments to demonstrate the superior energy efficiency of the proposed approach.

1.4. Paper organization

The rest of this paper is as follows: Section 2 introduces the models for power consumption and workflows, and then formulates the workflow scheduling problem as a single-objective constrained problem. Section 3 develops the energy-aware intelligent algorithm, followed by its numerical validations in Section 4. Section 5 concludes this paper and points out two research directions.

2. Preliminaries and modeling

This section formulates the constrained single-objective workflow scheduling problem by defining the power consumption model, workflow model; optimization objective; as well as precedence and deadline constraints.

2.1. Power consumption model

A cloud datacentre houses a set of virtual machines $R = \{r_1, r_2, \cdots, r_m\}$, providing computing resources to satisfy the user-defined quality of services. A virtual machine r_k can be described as quadruple $\{\acute{v}_k, \hat{v}_k, \acute{f}_k, \hat{f}_k\}$, where $\acute{v}_k$ and $\hat{v}_k$ represent its lower and upper supply voltage; $\acute{f}_k$ and $\hat{f}_k$ represent its lower and upper operation frequency.

The power consumption of r_k can be roughly divided into static part p_k^s and dynamic part p_k^d [24,48]. Static power consumption refers to the energy consumption per unit time when the virtual machine is completely idle. It is an inherent attribute of the virtual machine and has nothing to do with its working frequency. Dynamic power consumption is caused by the virtual machine executing the applications, and can be adjusted via dynamic voltage/frequency scaling technology. Then, the total power consumption of a virtual machine at time t refers to the summation of static and dynamic powers:

$$p_k(t) = p_k^s + p_k^d(t). \tag{1}$$

The dynamic power of a virtual machine is directly proportional to its supply voltage squared and frequency [24], and can be described as

$$p_k^d(t) = \alpha_k \cdot v_k(t)^2 \cdot f_k(t), \tag{2}$$

where α_k denotes the coefficient of proportionality; $v_k(t)$ and $f_k(t)$ denote the supply voltage and frequency at time t, respectively.

As the frequency is proportional to the supply voltage, the (2) can be rewritten as:

$$p_k^d(t) = \alpha_k \cdot f_k(t)^3, \tag{3}$$

Assume $\hat{p}_k$ is the maximum dynamic power consumption of r_k, its proportionality coefficient α_k can be approximated as

$$\alpha_k = \frac{\hat{p}_k}{\hat{f}_k^3}, \tag{4}$$

Based on (1), (3), and (4), the total power consumption of r_k at time t can be rewritten as below:

$$p_k(t) = p_k^s + \frac{\hat{p}_k}{\hat{f}_k^3} \cdot f_k(t)^3. \tag{5}$$

2.2. Workflow model

A workflow application often contains a set of tasks with data dependencies and is commonly abstracted to a directed acyclic graph. Formally, a workflow is depicted by a 3-tuple $G = \{T, E, D\}$, where $T = \{t_1, t_2, \cdots, t_n\}$ denotes a set of tasks, $E \subseteq T \times T$ represents a set of directed edges among data-dependent tasks, and D indicates the total deadline of this workflow.

An edge $e_{i,j} \in E$ represents the data dependency from t_i to t_j, in which t_i is the predecessor task of t_j and t_j is the successor task of t_i. The weight of a directed edge $w(e_{i,j})$ denotes the size of files transferred from t_i to t_j. The signs P_i and S_i represent the sets of t_i's immediate predecessor tasks and successor tasks, respectively. Due to data dependency, a task cannot start runing until all its predecessor tasks have completed and all the input files have arrived.

To visualize the workflow model, Fig. 1 provides an example of a workflow. For this workflow, it consists of six data-dependent tasks $T = \{t_1, t_2, \cdots, t_6\}$, and the data dependencies among tasks are defined by the directed edges as $E = \{e_{1,3}, e_{1,5}, e_{2,3}, e_{3,4}, e_{4,5}, e_{4,6}\}$. For task t_5, the set of its immediate predecessor tasks is $P_5 = \{t_1, t_4\}$. The set of the immediate successor tasks for task t_1 is $S_1 = \{t_3, t_5\}$.

2.3. Problem formulation

This work aims to optimize energy consumption while meeting the user-specified deadline constraints for workflows. Based on the models of power consumption and workflow application, this subsection formulates the constrained single-objective optimization problem.

In cloud datacentres embracing dynamic voltage/frequency scaling technology, the runtime of task t_i on virtual machine r_k depends on the computation length of t_i and the frequency of r_k, i.e.,

$$\tau_{i,k} = \frac{l_i}{f_k}, \tag{6}$$

where l_i and f_k denote the computation length of t_i and frequency of r_k, respectively.

When two immediate data-dependent tasks are executed by the same virtual machine, the data transmission time between them is negligible, whilst they are executed by different virtual machines, the data transmission time can be estimated based on the data size $w(e_{i,j})$ and bandwidth b. Assume r_g and r_k denote the virtual machines used to execute two immediate data-dependent tasks t_p and t_i. The data transmission time $\zeta_{p,i}$ from t_p to t_i can be described as:

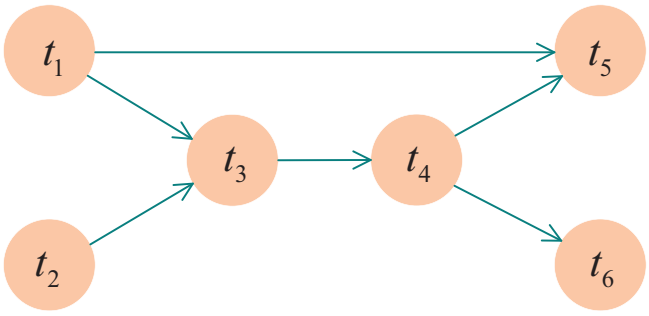

Fig. 1. Example of a DAG.

$$\zeta_{p,i} = \begin{cases} 0, & \text{if } r_g = r_k, \\ \frac{w(e_{i,j})}{b}, & \text{otherwise.} \end{cases} \tag{7}$$

The start time $\acute{\tau}_{i,k}$ and finish time $\hat{\tau}_{i,k}$ of task t_i on virtual machine r_k can be calculated as:

$$\acute{\tau}_{i,k} = \max\{a_k, \max_{t_p \in P_i}\{\hat{\tau}_{p,*} + \zeta_{p,i}\}\}, \tag{8}$$

$$\hat{\tau}_{i,k} = \acute{\tau}_{i,k} + \tau_{i,k}, \tag{9}$$

where a_k indicates the available time of virtual machine r_k to execute t_i, P_i represents the set of t_i's immediate predecessor tasks. If r_k has not been used, a_k is its set-up time; otherwise, a_k is the finish time of the last task executed on r_k.

Then, the completion time of a workflow, referring to the maximum finish time among all its tasks, can be described as:

$$\dot{\tau} = \max_{t_i \in T} \hat{\tau}_{i,*}. \tag{10}$$

Based on (8) and (9), the start time $st(r_k)$ and finish time $ft(r_k)$ of virtual machine r_k can be obtained as below:

$$\begin{aligned} st(r_k) &= \min_{t_i \in T} \acute{\tau}_{i,k}, \\ ft(r_k) &= \min_{t_i \in T} \hat{\tau}_{i,k}. \end{aligned} \tag{11}$$

Hence the energy consumption of r_k to execute workflow tasks is computed as:

$$\begin{aligned} ec_k &= \int_{st(r_k)}^{ft(r_k)} p_k(t)dt \\ &= \int_{st(r_k)}^{ft(r_k)} p_k^s + \frac{\hat{p}_k}{\hat{f}_k^3} \cdot f_k(t)^3 dt. \end{aligned} \tag{12}$$

Then, the total energy consumption for executing a workflow is

$$\begin{aligned} EC &= \sum_{k=1}^{m} ec_k \\ &= \sum_{k=1}^{m} \int_{st(r_k)}^{ft(r_k)} p_k^s + \frac{\hat{p}_k}{\hat{f}_k^3} \cdot f_k(t)^3 dt. \end{aligned} \tag{13}$$

Considering the high energy consumption in cloud datacentres, this paper attempts to allocate the workflow tasks to a suitable set of virtual machines and adjust the execution time of each task, so as to reduce the energy consumption before the user-specified overall deadline. The considered optimization problem can be summarized mathematically as follows:

$$\begin{cases} \text{Minimize} & EC, \\ \text{s.t.} & \dot{\tau} \leqslant D, \\ & \acute{\tau}_{i,k} \geqslant \max_{t_p \in P_i}\{\hat{\tau}_{p,*} + \zeta_{p,i}\}, \ \forall t_i \in T, \\ & \hat{f}_k \leqslant f_k(t) \leqslant \hat{f}_k, \ \forall r_k \in R, \\ & \hat{v}_k \leqslant v_k(t) \leqslant \hat{v}_k, \ \forall r_k \in R, \end{cases} \tag{14}$$

where the optimization objective is to minimize the energy consumption of workflow execution; the four constraints respectively come from the workflow's deadline, data dependencies among tasks, minimum and maximum CPU frequencies of virtual machines, as well as minimum and maximum supply voltages of virtual machines.

3. Algorithm design

Energy-aware workflow scheduling for cloud platforms is confronted with the non-linear power consumption model, sophisticated workflow structures, heterogeneous cloud resources, and diverse user-defined deadlines. To handle this highly challenging problem, this section designs an intelligent algorithm to reduce the energy consumption of workflow execution. The proposal first iteratively evolves the mapping from workflow tasks to cloud resources. Then, it excavates the workflow slack time (difference between its completion time and deadline) to adjust the execution time/frequency of workflow tasks for overall energy conservation. The proposal also excavates the idle time gaps caused by task precedence constraints to further reduce dynamic energy consumption.

Fig. 2 provides an example to visualize the above ideas. Fig. 2a shows the Gantt chart of the scheduling result for three tasks, i.e., t_1, t_2, and t_3. Since task t_3 needs to wait for the output files from task t_2, an idle time gap between t_3 and t_1 is left on virtual machine r_1. As shown in (3), the dynamic power consumption is proportional to the third power of the frequency. Intuitively, reducing the execution frequency of task t_1 will definitely reduce the dynamic energy consumption, as shown in Fig. 2b. As shown in Fig. 2, the completion time of this workflow is 14 s. Assuming that the deadline of this workflow is 20 s, its slack time between deadline and completion time is $20 - 14 = 6$ seconds, which will be distributed to each workflow task to reduce execution frequency, such reducing overall energy consumption.

3.1. Solution representation

The workflow scheduling in clouds involves arranging tasks' execution order, mapping tasks to resources and optimizing task execution time. To simplify the evolution process, we attempt to evolve the mappings from tasks to resources and the execution time of each task. The execution order of tasks is sorted based on their downward rank [49], which is recursively defined as follows:

$$rank(t_i) = \begin{cases} \max_{r_k \in R} \tau_{i,k}, & \text{if } P_i = \varnothing, \\ \max_{t_p \in P_i}\{rank(t_p) + \max_{r_k \in R} \tau_{i,k} + w(e_{p,i})/b\}, & \text{otherwise,} \end{cases} \tag{15}$$

where P_i denotes the set of immediate predecessor tasks of task t_i, R denotes the candidate resource pool, $\tau_{i,k}$ is the execution time of t_i on resource r_k, $w(e_{p,i})/b$ denotes the data transmission time from t_p to t_i. The downward ranks are calculated recursively by traversing the DAG downward starting from the tasks without predecessors.

Basically, the downward rank of a task stands for the longest distance from it to the tasks without predecessors. According to the definition in (15), the rank of a task is greater than that of all its predecessors. When tasks are sorted according to their downward ranks, a task must be next to all its predecessors, inevitably fulfilling the precedence constraints among tasks.

In this paper, we encode a solution as two n-dimensional vectors. The first vector corresponds to the mapping from tasks to virtual machines, where an index denotes a task, and its value denotes the virtual machine where this task will be executed. The second segment corresponds to the mapping from tasks to execution time, where an index denotes a task, and its value denotes the execution time of this task.

Fig. 3 gives the encodings for the Gantt charts in Fig. 2. As shown Fig. 3a, the values of the first vector (i.e., 1, 2, and 1) indicate that tasks t_1 and t_3 are mapped to the virtual machine r_1, and task t_2 is mapped to r_2. Then, the values of the second vector indicate that the execution time of these three tasks is 5, 8, and 4 s, respectively. Since the Gantt chart in Fig. 2b is obtained by adjusting t_1's execution time from 5 to 10 s, the corresponding encoding only changes the execution time of task t_1, as shown in Fig. 3b.

3.2. Main process

For the proposed energy-aware intelligent scheduling algorithm (EIS), its main process is summarized in Algorithm 1. Its elemental inputs contain the data about the workflow, candidate resource pool, population size of the algorithm, and the maximum function evaluations as stop condition. When the EIS meets the stop condition, it will select and output a scheduling solution with the minimum energy consumption.

Algorithm 1: Main Process of EIS

Input: A workflow G; Resource pool R; Population size N; Maximum function evaluations (MFE);
Output: A solution $\vec{p}^*$ with the minimum energy consumption;

1 $P \leftarrow$ Initialize a population arbitrarily;
2 $FEs \leftarrow N$;
3 **while** $FEs < MFE$ **do**
4 $DV \leftarrow$ Reproduce N new decision vectors;
5 $DV \leftarrow$ **Function** SlackTimeAssignment(DV, G, R);
6 $DV \leftarrow$ **Function** IdleTimeGrab(DV, G);
7 $Q \leftarrow$ Obtain an offspring population;
8 $FEs \leftarrow FEs + N$;
9 **for** $i = 1 \rightarrow N$ **do**
10 **if** $EC(P_i) > EC(Q_i)$ **then**
11 $P_i \leftarrow Q_i$;
12 **end**
13 **end**
14 **end**
15 $\vec{p}^* \leftarrow \varnothing$; $E_m \leftarrow +\infty$;
16 **for** $\vec{p} \in P$ **do**
17 **if** $E_m < EC(\vec{p})$ **then**
18 $\vec{p}^* \leftarrow \vec{p}$; $E_m \leftarrow EC(\vec{p})$;
19 **end**
20 **end**
21 Output the solution $\vec{p}^*$;

As shown in Algorithm 1, the proposal follows the popular framework of evolutionary algorithms, consisting of three stages: initialization, reproduction and selection. In the initialization stage, the proposal arbitrarily generates a population (Line 1), and employs parameter *FEs* to record the number of function evaluations that have been used (Line 2). Since we explore the energy saving of workflow execution in dynamic voltage/frequency scaling-enabled cloud datacentres, each schedule solution in population P is defined to contain two decision vectors: mappings from tasks to resources as well as mappings from tasks to execution time. Since the mappings from task to runtime involves sophisticated constraints, see (14) for details, the random generation of this vector is easy to violate these constraints. Thus, during the initialization phase, the task runtime on the mapped resource is set to the minimum value, that is, the runtime under the maximum frequency/power consumption of the mapped resource. After initialization, the EIS iterates the two processes of population reproduction and selection until the number of function evaluations used *FEs* reaches the pre-specified maximum value *MFE*.

In the population reproduction stage, N decision vectors describing the mappings from workflow tasks to cloud resources are generated using the classic reproduction operators, such as differential evolution [50] and particle swarm optimization [51] (Line 4). The above reproduction process is to evolve the mappings from tasks to resources, i.e., the first encoding segments of solutions. In this process, the task execution time on the mapped resource is set to the maximum value. Then, the Function *SlackTimeAssignment*() is called to reduce energy consumption by distributing workflow slack time to extend the runtime of each task (Line 5). Next, the Function *IdleTimeGrab*() is called to extend some tasks' runtime by mining the idle time gaps between tasks to achieve the purpose of reducing dynamic energy consumption (Line 6). The above two functions are to optimize the execution time of each task, i.e., the second encoding segments of solutions. These two functions are detailed in Algorithm 2 and Algorithm 3, respectively. Based on the mappings from workflow tasks to cloud resources and the mappings from workflow tasks to rutime, the proposed algorithm evaluates the energy consumption of each scheduling solution to generate an offspring population (Line 7). Also, the number of function evaluations used *FEs* is updated (Line 8).

During the population selection process, solutions in the offspring population Q compete with the solutions of the parent population P one by one, and the solutions with lower energy consumption will survive to the next generation (Lines 9–13). P_i represents the i-th schedule solution in population P. Similarly, Q_i corresponds to the i-th solution in Q. $EC(P_i)$ and $EC(Q_i)$ denote the energy consumption of schedule solutions P_i and Q_i, respectively. When the proposal reaches the stop condition, it checks each solution in the final population and selects a schedule solution with minimum energy consumption (Lines 15–20).

3.3. Energy-aware mechanisms

To reduce the voltages/frequencies of task executions for energy saving, the Function *SlackTimeAssignment*() distribute the workflow slack time (difference between its completion time and deadline) to each workflow task according to its optimal runtime that minimizes energy consumption. We derive the optimal runtime of a task as follows.

Assume the operating frequency of a virtual machine r_k is fixed for executing a task t_i, and t_i's computation length is l_i. The runtime of this task is a variable, expressed as $\tau_{i,k}$. Based on (6), the operating frequency of r_k is $f_k(t) = l_i/\tau_{i,k}$. Then, the functional relationship between energy consumption and task runtime can be expressed as follows:

$$\begin{aligned} ec_{k,i} &= \int_{st(r_k)}^{st(r_k)+\tau_{i,k}} p_k^s + \frac{\hat{p}_k}{\hat{f}_k^3} \cdot f_k(t)^3 dt \\ &= p_k^s \cdot \tau_{i,k} + \frac{\hat{p}_k}{\hat{f}_k^3} \cdot \left(\frac{l_i}{\tau_{i,k}}\right)^3 \cdot \tau_{i,k} \\ &= p_k^s \cdot \tau_{i,k} + \frac{\hat{p}_k}{\hat{f}_k^3} \cdot \frac{(l_i)^3}{(\tau_{i,k})^2}. \end{aligned} \tag{16}$$

Let $\frac{\partial(ec_{k,i})}{\partial(\tau_{i,k})} = 0$ for optimization, we obtain

$$p_k^s - 2 \cdot \frac{\hat{p}_k}{\hat{f}_k^3} \cdot \frac{(l_i)^3}{(\tau_{i,k})^3} = 0. \tag{17}$$

Based on (17), the optimal execution time of t_i can be computed as

$$\tau_{i,k} = \frac{l_i}{\hat{f}_k^3} \cdot \sqrt[3]{\frac{2\hat{p}_k}{p_k^s}}. \tag{18}$$

The pseudo-code of Function *SlackTimeAssignment*() is illustrated in Algorithm 2. This function first calculates the start and finish time of each workflow task based on its minimum runtime

on the corresponding virtual machine, thus obtaining the completion time of the entire workflow. Then, the workflow slack time between its deadline and completion time is available. Based on the optimal runtime in (18), the workflow slack time is distributed to appropriately extend tasks' runtime, so as to reduce the operating frequencies of virtual machines to achieve the purpose of energy saving.

Algorithm 2: **Function** SlackTimeAssignment(DV, G, R)

Input: A set of decision vectors DV; the workflow G; the resource pool R;

Output: Decision of each task's start/execution/finish time for each decision vector;

1 Sort tasks T based on their ranks, which is defined in (15);
2 **for** $\vec{x} \in DV$ **do**
3 $RT \leftarrow 0_{1\times|R|}$;
4 **for** $t_i \in T$ **do**
5 $st_i \leftarrow RT(x_i)$;
6 **for** $t_p \in P_i$ **do**
7 $at \leftarrow ft_p + I(x_p \neq x_i) \times w(t_p, t_i)/bw$;
8 **if** $at > st_i$ **then**
9 $st_i \leftarrow at$;
10 **end**
11 **end**
12 $ft_i \leftarrow st_i + l_i/f^m(x_i)$;
13 **end**
14 $ct \leftarrow \max_{t_i \in T}\{ft_i\}$;
15 $slaT \leftarrow D - ct$;
16 $OET \leftarrow 0_{1\times n}$;
17 **for** $t_i \in T$ **do**
18 $OET_i \leftarrow$ Calculate optimal runtime of task t_i by (18);
19 **end**
20 $RET \leftarrow 0_{1\times n}$;
21 **for** $t_i \in T$ **do**
22 $RET_i \leftarrow \min\{OET_i,\ l_i/f^m(x_i) + slaT \times OET_i/\sum_{h=1}^{n} OET_h\}$;
23 **end**
24 Update start/finish time for each task based on RET;
25 **end**

As shown in Algorithm 2, Function *SlackTimeAssignment*() receives a set of decision variables *DV*, each of which corresponds to a mapping from tasks to resources. According to each decision variable, denoted as $\vec{x}$, this function assigns runtime to each workflow task. It calculates the completion time of the workflow as follows (Lines 3–14). The *RT* is used to record the ready time of the *R* resources for executing the next task (Line 3). Starting from the tasks without predecessors, all the workflow tasks are traversed downward to calculate their start time st_i and finish time ft_i (Lines 4–13). The value of the parameter x_i indicates the index of the resource mapped to task t_i, and the operation in line 5 stands for that a task's start time is greater than the resource's ready time. The operation in line 7 is to calculate the latest time *at* when a task completes receiving the input files from all its predecessors. Also, the start time of a task should be larger than *at* to fulfill the precedence constraints (Lines 8–10). The operation in line 12 is to obtain the finish time for each workflow task. It is worth noting that the above calculations are based on the minimum runtime of the tasks on the mapped resources. Then, the completion time *ct* and slack time *slaT* of the workflow are obtained (Lines 14–15).

After that, Function *SlackTimeAssignment*() calculates the optimal runtime of each task (Lines 17–19), where OET_i records the optimal runtime for the *i*-th task in the workflow. Based on workflow slack time and task optimal runtime, this function assigns the real runtime for each task (Lines 21–23). From the operation in line 22, we can see that the slack time is distributed to each task in proportion to its optimal runtime, i.e., $OET_i/\sum_{h=1}^{n} OET_h$. Besides, the real runtime of a task cannot exceed its optimal value. Next, this function updates each task's start time and finish time based on its real runtime (Line 24).

The pseudo-code of Function *IdleTimeGrab*() is shown in Algorithm 3. This function excavates the idle time gaps caused by task precedence constraints to further reduce dynamic energy consumption whilst satisfying workflows' deadline constraints.

Algorithm 3: **Function** IdleTimeGrab(DV, G)

Input: A set of decision vectors DV; the workflow G;

Output: Decision of each task's start/execution/finish time for each decision vector;

1 **for** $\vec{x} \in DV$ **do**
2 $RET \leftarrow 0_{1\times n}$;
3 **for** $t_i \in T$ **do**
4 $t_a \leftarrow$ Obtain the task appended after t_i;
5 **if** $t_a \neq \emptyset$ **then**
6 $lft_i \leftarrow st_a$;
7 **for** $t_s \in S_i$ **do**
8 $lft' \leftarrow st_s - I(x_i \neq x_s) \cdot w(t_i, t_s)/bw$;
9 **if** $lft' < lft_i$ **then**
10 $lft_i \leftarrow lft'$;
11 **end**
12 **end**
13 $RET_i \leftarrow lft_i - st_i$;
14 **else**
15 $RET_i \leftarrow ft_i - st_i$;
16 **end**
17 **end**
18 Update start/finish time for each task based on RET;
19 **end**

As shown in Algorithm 3, Function *IdleTimeGrab* receives a set of decision variables *DV*, each of which contains a mapping from tasks to resources and a mapping from tasks to runtime, which has been determined by Function *SlackTimeAssignment*(). The $1 \times n$ vector *RET* is used to record the real runtime for each workflow task (Lines 2 and 14). For a decision variable $\vec{x}$ in *DV*, this function traverses each task to obtain its latest finish time lft_i, before which completing this task will not delay the start time of all its successors and the tasks executed after it. For a task t_i, its latest finish time is first initialized to the start time of the task appended immediately after it (Line 6), which can avoid affecting the start of subsequent tasks. Then, each successor task of task t_i is traversed to update its latest finish time (Lines 7–12). Next, the real runtime of task t_i is updated (Line 13). If a task is at the end of the task queue on a resource, i.e.,

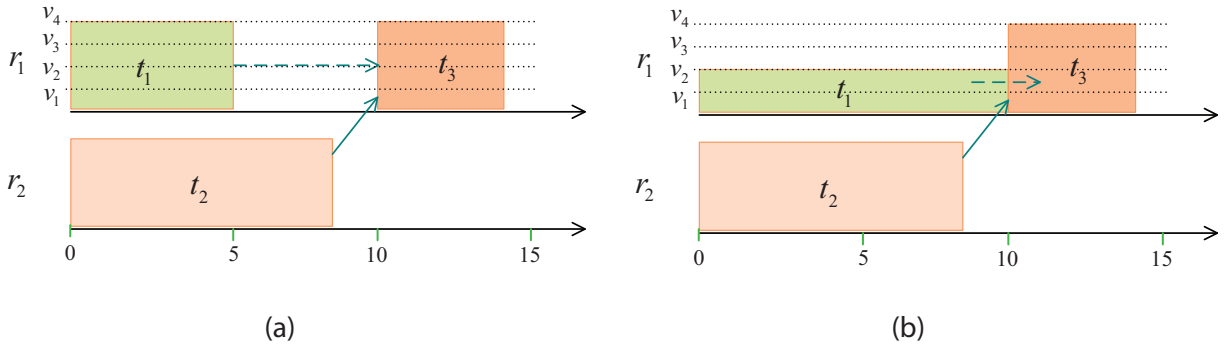

Fig. 2. Example of energy conservation by adjusting task execution time/frequency.

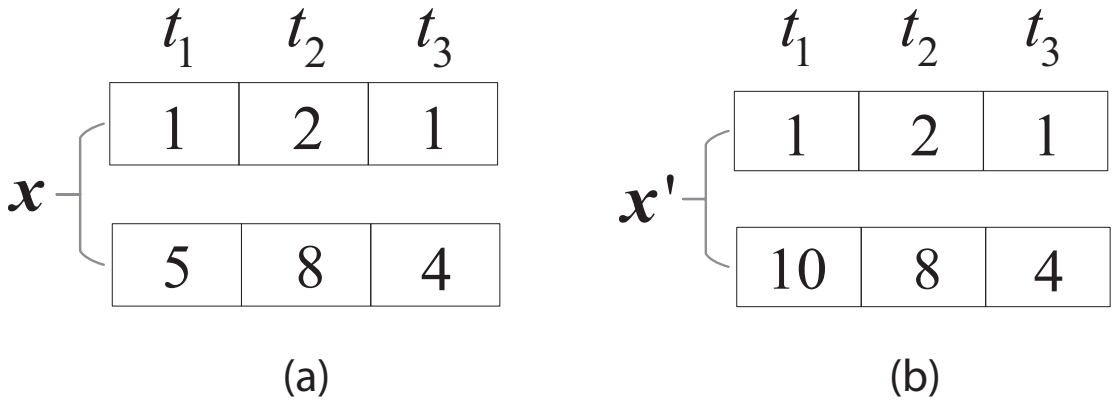

Fig. 3. Two encoding examples.

$t_a = \varnothing$, its runtime remains unchanged (Lines 14–15). After that, this function updates each task's start time and finish time based on its real runtime (Line 18).

4. Performance evaluation

This section presents the experimental verification for the proposal in detail, e.g., experiment configurations including datasets, parameters, and a performance metric, as well as experimental results and analysis.

4.1. Experimental setting

In our evaluation, we select three relevant and recent metaheuristics, i.e., DMFO-DE [52], PSO-COGENT [53], and ERTS [54], for performance comparison. The brief descriptions of these three metaheuristics are as follows.

DMFO-DE is a hybrid discrete optimization algorithm. It combines an opposition-based Moth-Flame Optimization with the Differential Evolution to map workflow tasks to cloud resources, and uses the heterogeneous earliest finish time rule to determine the task execution order.

PSO-COGENT is a particle swarm optimization-based resource allocation algorithm to reduce the energy consumption of cloud datacentres by formulating deadlines as constraints.

ERTS integrates new genetic operators and a frequency scaling strategy for resource provisioning and task scheduling in DVFS-enabled cloud workflows.

To compare the four energy-aware workflow scheduling algorithms, we employ five types of real-world workflows with different sizes [55], including Montage with 25, 50, 100, and 1000 tasks, Epigenomics with 24, 46, 100, and 997 tasks, Inspiral with 30, 50, 100, and 1000 tasks, Cybershake with 30, 50, 100, and 1000 tasks, and Sipht with 30, 60, 100, and 1000 tasks. Fig. 4 depicts the topological structures of five different real-world workflow applications with around 30 tasks.

These workflows come from different fields. For instance, Montage [56] is a flexible toolkit to assemble and process large sky images in the astronomy field. Epigenomics [57] is a pipeline data processing for automatic genome sequencing operations in bioinformatics applications. Inspiral workflow [58] is to detect gravitational radiation generated during the most violent events in astrophysics. Cybershake [59] is a powerful analysis tool to study earthquake hazards. Sipht [60] is a high-throughput technology program for kingdom-wide prediction and functional annotation of bacterial sRNA-encoding genes in the bioinformatics field.

To set the deadline constraint for a workflow, we estimate its minimum completion time by allocating each workflow task to a virtual machine with the most computationally powerful configuration. The symbol $\dot{\tau}_m$ denotes a workflow's minimum completion time. We introduce a constraint factor α to control different constraints, and the deadline of a workflow is set as follows.

$$D = \alpha \cdot \dot{\tau}_m. \tag{19}$$

From the formula (19), it can be easily derived that with the increase of factor α, the deadline of a workflow becomes more relaxed.

The proposed EIS employs the uniform crossover and bit-flip mutation operators evolve the mappings from tasks to cloud resources. Mutation rate is set as $1/n$, where n denotes the number

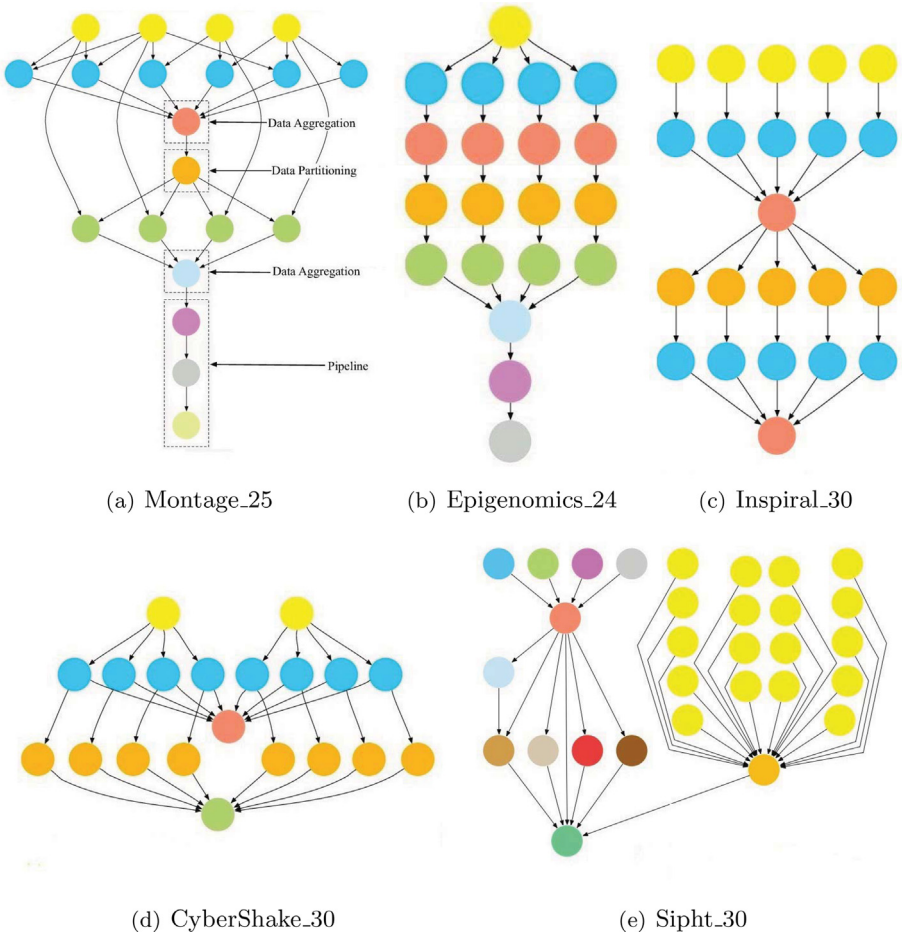

Fig. 4. Five real-world workflows [55].

of tasks. Besides, the parameter settings of the three comparison algorithms follow the recommended values of the original papers.

To ensure the fairness of performance comparison for different algorithms, the population size of the four algorithms is set to 100. The maximum number of function evaluations (*FEs*) is set as $n \times 4e3$, where n is the number of decision variables. Besides, we conduct all the experiments on a PC with 8 GB RAM, two Intel CPUs i5-6500, and 64-bit Windows 10 operating system. On this PC, we install MATLAB R2020b to run these four algorithms.

4.2. Impact of deadline

To evaluate the effect of deadline constraint factor α on the performance of the proposal and the three competitors, i.e., DMFO-DE, PSO-COGENT, and ERTS, we increase the factor α from 1.5 to 6.0 with a step of 0.5, and stabilize other parameters. Fig. 5 illustrates the change of energy consumption by prolonging the deadline on Montage_50, Epigenomics_46, Inspiral_50, CyberShake_50, and Sipht_30 workflows. The mark Montage_50 denotes the Montage workflow with 25 tasks, the other four marks can be similarly parsed.

From Fig. 5, we can see that with the increase of factor α, the energy consumed by the four energy-aware workflow scheduling algorithms decreases correspondingly. This trend can be attributed to the following facts. Cloud datacentres are heterogeneous and elastic, which means that the power-performance ratios of different virtual machines vary greatly and the number of accessible virtual machines is sufficient. Increasing factor α to prolong workflows' deadlines gives each workflow task a greater opportunity to run on a virtual machine with a higher power-performance ratio. This helps reduce the overall energy consumption of workflow executions.

In Fig. 5, one noticeable phenomenon is that when the deadline constraint factor α increases to a certain extent, the energy consumption of the four algorithms tends to be stable. The main reason is that prolonging workflow deadlines is conducive to extending the runtime of each task to reduce the dynamic energy consumption, but the static energy consumption increases accordingly. When the deadline increases to a certain extent, the reduced dynamic energy consumption will be offset by static energy consumption. Another striking phenomenon is that in different workflow instances, the factor values when energy consumption tends to be stable are different. For example, the energy consumed by the four algorithms tends to be stable when the factor α is larger than 3.0 on the Montage_50 workflow, while that is larger than 5.0 on the Inspiral_50 workflow. This is due to the significant differences in the topological structures of different types of workflows.

In comparison with the three competitors, the proposed EIS achieves the lowest energy consumption under different deadline constraint factors. Considering the Montage_50 workflow in Fig. 5a, on average the proposed EIS achieves 30.89%, 27.46%, and 18.36% improvement in comparison to DMFO-DE, PSO-COGENT, and ERT, respectively. The EIS poses similar advantages in solving the optimization probelms derived from the other four workflows, i.e., Epigenomics_46, Inspiral_50, CyberShake_50, and Sipht_30.

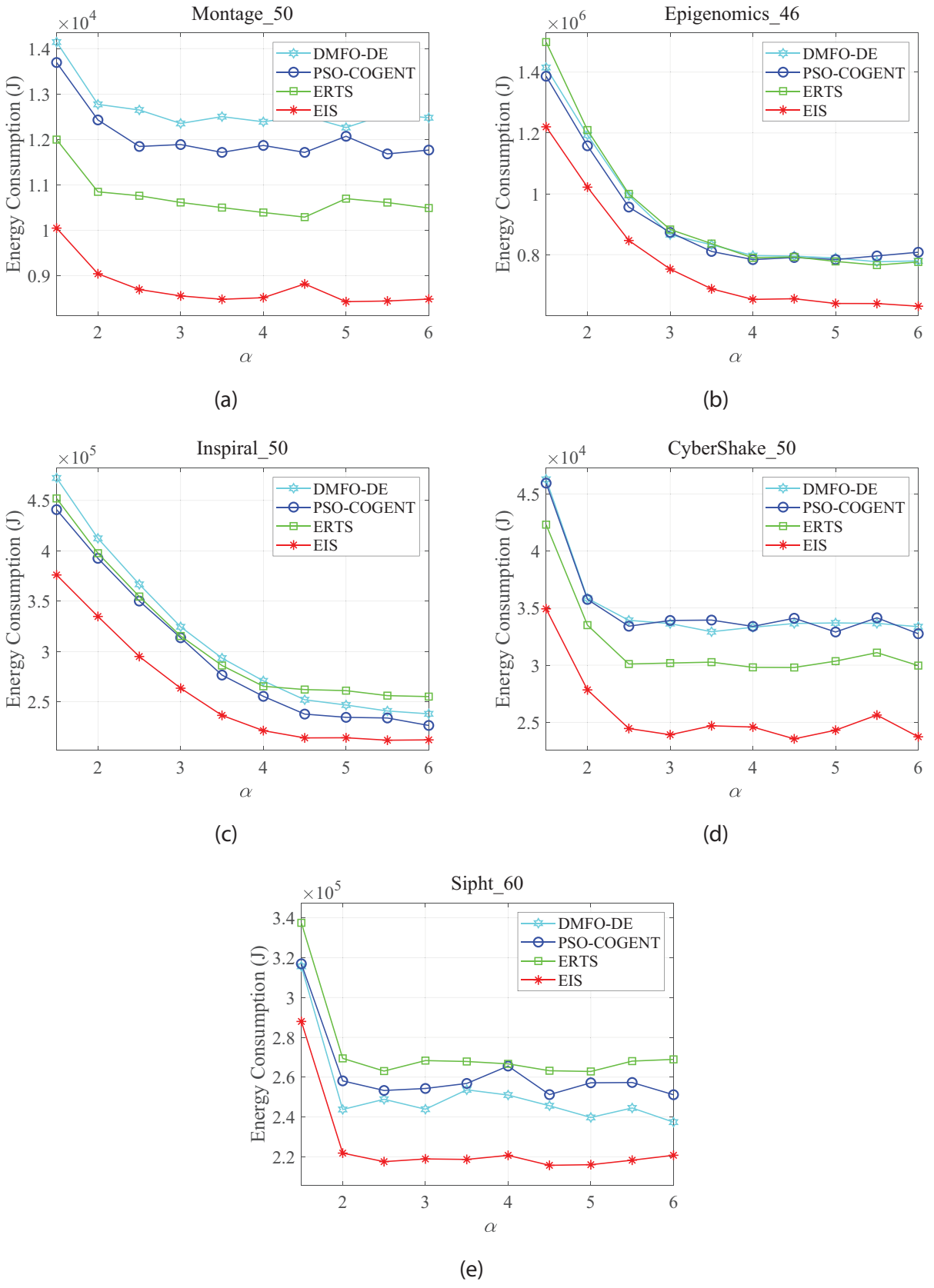

Fig. 5. Impact of workflow deadline on energy consumption.

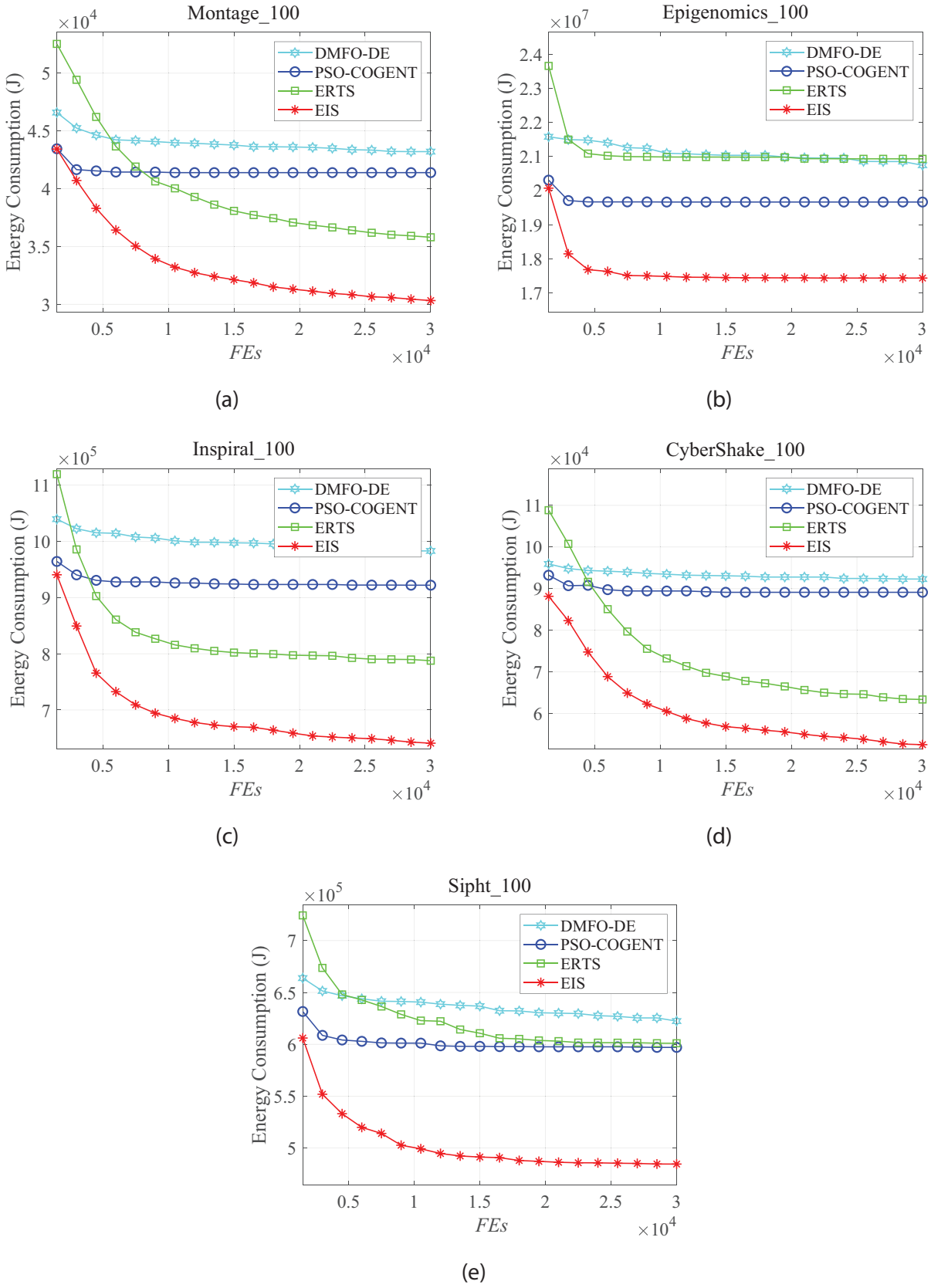

Fig. 6. Change of energy consumption with evolution process.

Although the ERT and the proposed EIS both leverage dynamic voltage/frequency scaling technology to reduce energy consumption, the proposed EIS still shows overwhelming advantages. The reason is that EIS benefits from the workflow slack time distribution mechanism. In summary, the proposed EIS achieves the lowest energy consumption while satisfying workflows' deadline constraints.

4.3. Trends of energy consumption

This set of experiments attempts to compare the convergence rates of the four algorithms. We select five workflow applications, i.e., Montage_100, Epigenomics_100, Inspiral_100, CyberShake_100, and Sipht_100, and fix their deadline constraint factor as $\alpha = 1.5$. Fig. 6 exhibits the changes in energy consumption as the evolution progresses. The parameter *FEs* corresponds to the number of function evaluations that have been used.

The intuitive impression of Fig. 6 is that the energy consumed by the four algorithms ascends as the number of function evaluations increases, especially in the initial search stage. This is because a cloud datacentre has enough virtual machines and the quality of randomly initialized solutions is low. In such a scenario, bio-inspired optimization algorithms can quickly explore better solutions. Among the three existing energy-aware workflow scheduling algorithms, the ERT has obvious advantages in convergence speed and value. It comes down to the fact that considerable energy can be reducing by slowing down the voltages/frequencies of virtual machines for executing some non-critical tasks, such that the energy consumption of workflow executions can be reduced without violating the deadline constraints. Compared with the competitor ERT, the proposed EIS poses better convergence speed and value. The primary reason is that the proposal not only mines the idle time gaps to slow down the voltages/frequencies for energy conservation, but also reasonably distributes the workflow slack time to slow down the voltages/frequencies.

4.4. Comparison results

To further compare the performance of the four algorithms, this set of experiments is to consider the schedule solution with the lowest energy consumption in the output population of each algorithm. We set the deadline constraint factor as $\alpha = 1.5$, and the *FEs* as $n \times 4e3$. Table 1 provides a comparative summary of the energy consumption of the proposed EIS and the three competitors, i.e., DMFO-DE, PSO-COGENT, and ERTS, in solving optimization problem derived from 20 frequently-used workflows. This table includes the mean and standard deviation (in brackets) of energy consumption of 30 repeated experiments.

As shown in Table 1, the proposed EIS poses a higher energy-saving performance in all workflow applications. The reason is that the competitors DMFO-DE and PSO-COGENT focus on evolving the mappings from workflow tasks to resources to reduce energy consumption, and did not consider the dynamic voltage/frequency scaling technique. Although the competitor ERT employs the dynamic voltage/frequency scaling technique, its energy consumption is much higher than the proposed EIS. Such superiority of EIS can be attributed to the following two facts. First of all, the EIS distributes workflow slack time (difference between its completion time and deadline) among tasks based on the optimal execution time of each task for energy conservation by adjusting the voltage and frequency. Secondly, the EIS excavates the idle time gaps caused by task precedence constraints to further reduce dynamic energy consumption whilst satisfying workflows' deadline constraints.

Besides, with the increase of workflow scale, the EIS improves the competitors more obviously. For instance, in comparison to ERTS, the EIS achieves 16.43% improvement in the Sipht workflow with 30 tasks, while that increases to 19.38% in the Sipht workflow with 100 tasks. In general, as the workflow scale increases, the workflows' topological structures become more complex, and the corresponding optimization problems become more difficult. Hence, the improvement of the EIS in scheduling larger-scale workflows demonstrates its advantages in handling optimization problems derived from complex workflows.

4.5. Ablation analysis

The proposed EIS mainly includes two energy-saving components. Function *SlackTimeAssignment*() is to reduce energy consumption by distributing workflow slack time to extend the runtime of each task (Line 5, Algorithm 1). Function *IdleTimeGrab*() is to extend some tasks' runtime by mining the idle

Table 1
Energy consumption (Joule) of the four algorithms on 15 workflows.

Workflows	n	DMFO-DE	PSO-COGENT	ERTS	EIS
Montage	25	5.043e+3 (9.740e+1)	4.974e+3 (2.326e+2)	4.766e+3 (1.502e+2)	3.933e+3 (2.424e+1)
	50	1.413e+4 (2.232e+2)	1.369e+4 (5.991e+2)	1.199e+4 (4.226e+2)	1.005e+4 (5.044e+2)
	100	4.321e+4 (2.321e+2)	4.139e+4 (1.084e+3)	3.579e+4 (2.004e+3)	3.032e+4 (1.863e+2)
	1000	5.759e+5 (5.827e+3)	6.056e+5 (4.921e+4)	5.490e+5 (2.061e+4)	4.583e+5 (1.764e+4)
Epigenomics	24	5.042e+5 (1.343e+3)	5.192e+5 (1.316e+3)	5.521e+5 (2.264e+4)	4.547e+5 (6.089e+3)
	46	1.415e+6 (5.999e+4)	1.387e+6 (6.477e+4)	1.499e+6 (3.844e+4)	1.220e+6 (4.145e+4)
	100	2.074e+7 (5.009e+5)	1.966e+7 (7.607e+5)	2.093e+7 (7.925e+4)	1.743e+7 (6.573e+4)
	997	2.217e+8 (8.735e+5)	2.155e+8 (2.351e+6)	2.236e+8 (1.042e+6)	1.901e+8 (7.327e+5)
Inspiral	30	2.252e+5 (6.233e+3)	2.184e+5 (6.146e+3)	2.336e+5 (3.966e+3)	1.943e+5 (4.628e+3)
	50	4.724e+5 (7.874e+3)	4.409e+5 (2.038e+3)	4.521e+5 (6.922e+3)	3.760e+5 (7.283e+3)
	100	9.829e+5 (1.543e+4)	9.219e+5 (2.847e+4)	7.876e+5 (2.543e+4)	6.409e+5 (4.276e+4)
	1000	1.187e+7 (1.460e+5)	1.134e+7 (3.204e+5)	7.714e+6 (3.211e+5)	6.445e+6 (3.830e+5)
CyberShake	30	1.979e+4 (5.564e+2)	1.959e+4 (7.552e+2)	1.945e+4 (7.628e+2)	1.615e+4 (4.901e+2)
	50	4.627e+4 (1.425e+3)	4.593e+4 (1.058e+3)	4.230e+4 (9.923e+2)	3.493e+4 (1.027e+3)
	100	9.231e+4 (1.245e+3)	8.913e+4 (1.231e+3)	6.339e+4 (6.100e+3)	5.255e+4 (3.158e+3)
	1000	1.256e+6 (3.538e+4)	1.295e+6 (5.912e+4)	1.076e+6 (8.403e+3)	9.101e+5 (2.073e+4)
Sipht	30	1.318e+5 (5.963e+2)	1.316e+5 (6.349e+2)	1.557e+5 (2.342e+2)	1.301e+5 (4.517e+2)
	60	3.156e+5 (1.642e+4)	3.169e+5 (2.292e+4)	3.375e+5 (2.292e+4)	2.880e+5 (2.710e+4)
	100	6.224e+5 (1.607e+4)	5.969e+5 (1.796e+4)	6.009e+5 (2.818e+4)	4.844e+5 (2.934e+4)
	1000	1.022e+7 (6.585e+4)	9.758e+6 (1.402e+5)	7.696e+6 (1.482e+5)	6.507e+6 (6.255e+4)

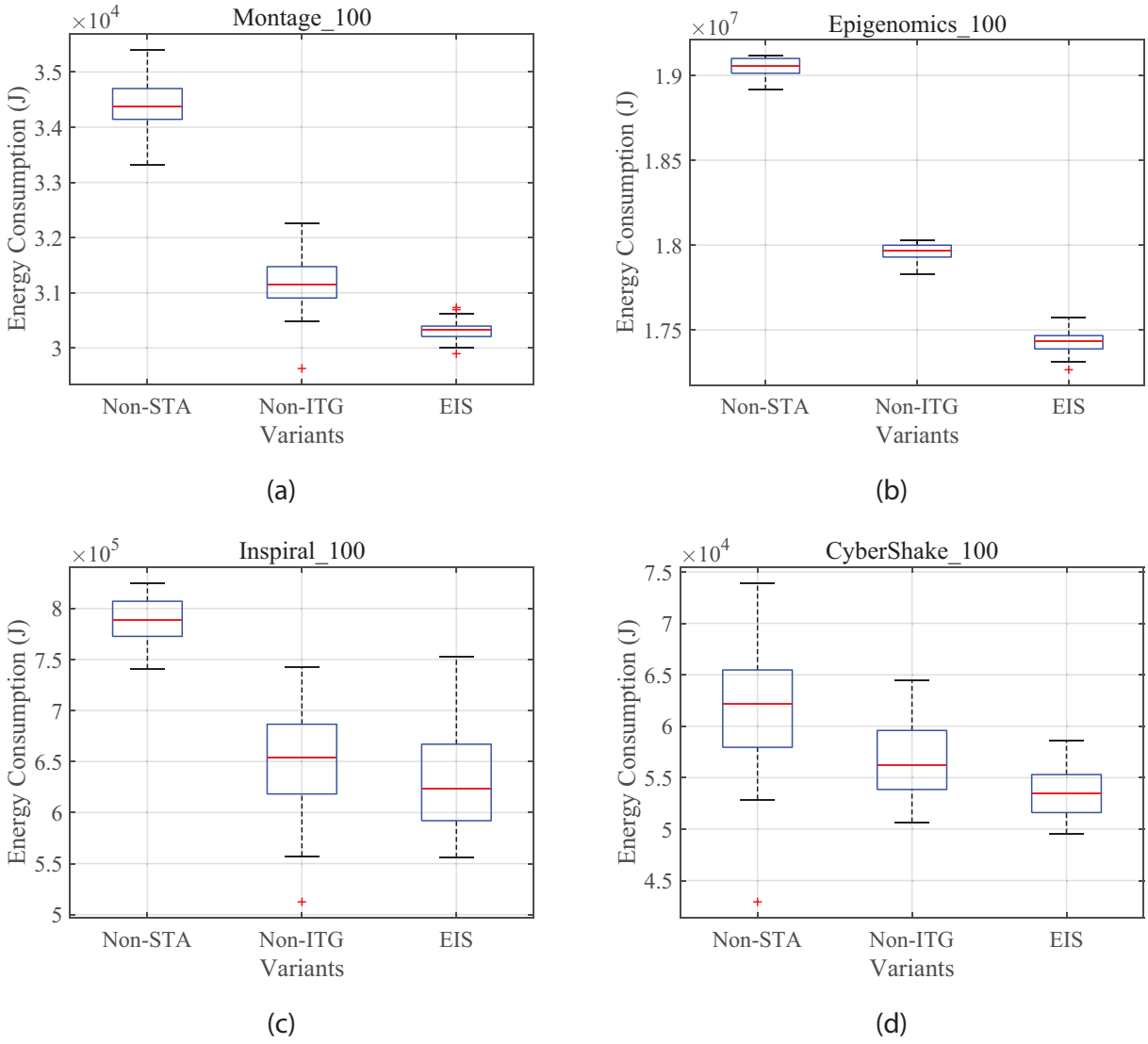

Fig. 7. Performance comparison of the proposal and its variants.

time gaps between tasks to achieve the purpose of reducing dynamic energy consumption (Line 6, Algorithm 1). To verify their performance contribution, we construct two variants of the proposed EIS, denoted as Non-STA and Non-ITG. Non-STA is constructed by removing the Function *SlackTimeAssignment*(), while Non-ITG is constructed by removing the Function *IdleTimeGrab*(). Based on four workflows, i.e., Montage_100, Epigenomics_100, Inspiral_100, and Cybershake_100, the energy consumption of Non-STA, Non-ITG, and EIS is compared in Fig. 7.

The main component difference between variant Non-STA and EIS is that Non-STA does not employ Function *SlackTimeAssignment*(). Then, the amount of energy reduced by EIS over Non-STA can be attributed to the performance contribution of Function *SlackTimeAssignment*(). Similarly, the energy reduction of EIS over Non-ITG corresponds to the performance contribution of Function *IdleTimeGrab*(). According to the comparison results on the four workflows, we can see that Function *SlackTimeAssignment*() contributes more to energy saving. For example, on workflow Montage_100, Function.

SlackTimeAssignment() reduces the energy consumption from 3.42e + 4 to 3.03e + 4, while Function *SlackTimeAssignment*() reduces the energy consumption from 3.11e + 4 to 3.03e + 4.

5. Conclusions and future work

This paper strives to optimize the energy consumed by executing workflow applications in a cloud datacentre. According to the characteristics of cloud resources and applications, this paper provides the energy power and workflow models, and formulates the optimization problem with deadline and precedence constraints. Then, an energy-aware intelligent scheduling algorithm to explore the workflow slack time and idle time gaps between tasks to reduce dynamic energy consumption whilst satisfying workflows' deadline constraints. Also, extensive comparison results demonstrate the proposal's superior performance in terms of energy saving.

Based on this research, it is promising to holistically explore the energy-efficient optimization problem for other types of subsystems including cooling, network, storage, and cache in cloud platforms. Besides, integrating the proposed techniques into real-world cloud platforms is another research direction.

CRediT authorship contribution statement

Min Cao: Conceptualization, Methodology, Software, Writing – original draft. **Yaoyu Li:** Data curation, Supervision. **Xupeng Wen:**

Visualization, Investigation. **Yue Zhao:** Supervision, Writing – review & editing. **Jianghan Zhu:** Software, Validation.

Declaration of Competing Interest

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

Acknowledgments

This research work is supported by the National Natural Science Foundation of China (71801218), the Science and Technology Innovation Program of Hunan Provincial (2022RC1241).

References

[1] Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I. A view of cloud computing. Commun ACM 2010;53(4):50–8.
[2] Elsherbiny S, Eldaydamony E, Alrahmawy M, Reyad AE. An extended intelligent water drops algorithm for workflow scheduling in cloud computing environment. Egypt Inf J 2018;19(1):33–55.
[3] Lee YC, Han H, Zomaya AY, Yousif M. Resource-efficient workflow scheduling in clouds. Knowl-Based Syst 2015;80:153–62.
[4] Chen H, Zhu X, Qiu D, Liu L, Du Z. Scheduling for workflows with security-sensitive intermediate data by selective tasks duplication in clouds. IEEE Trans Parallel Distrib Syst 2017;28(9):2674–88.
[5] Li M, Tian Z, Du X, Yuan X, Shan C, Guizani M. Power normalized cepstral robust features of deep neural networks in a cloud computing data privacy protection scheme. Neurocomputing 2023;518:165–73.
[6] Gartner, Gartner forecasts worldwide public cloud: End-user spending to grow 23% in 2021.
[7] Wang J, Palanisamy B, Xu J. Sustainability-aware resource provisioning in data centers. In: IEEE 6th International Conference on Collaboration and Internet Computing (CIC). IEEE; 2020. p. 60–9.
[8] Ilager S, Buyya R. Energy and thermal-aware resource management of cloud data centres: A taxonomy and future directions, arXiv preprint arXiv:2107.02342.
[9] Cao B, Sun Z, Zhang J, Gu Y. Resource allocation in 5G IoV architecture based on SDN and fog-cloud computing. IEEE Trans Intell Transp Syst 2021;22(6):3832–40.
[10] Wang Y, Liu Y, Xia M. Construction of a multi-source heterogeneous hybrid platform for big data. J Comput Methods Sci Eng 2021;21(3):713–22.
[11] Sun J, Chen Y, Dai M, Zhang W, Sangaiah AK, Sun G, Han H. Energy efficient deployment of a service function chain for sustainable cloud applications. Sustainability 2018;10(10):3499.
[12] Gill SS, Garraghan P, Stankovski V, Casale G, Thulasiram RK, Ghosh SK, Ramamohanarao K, Buyya R. Holistic resource management for sustainable and reliable cloud computing: An innovative solution to global challenge. J Syst Softw 2019;155:104–29.
[13] Freitag C, Berners-Lee M, Widdicks K, Knowles B, Blair G, Friday A. The climate impact of ICT: A review of estimates, trends and regulations, arXiv preprint arXiv:2102.02622.
[14] Thaman J, Singh M. Green cloud environment by using robust planning algorithm. Egypt Inf J 2017;18(3):205–14.
[15] Marahatta A, Pirbhulal S, Zhang F, Parizi RM, Choo K-KR, Liu Z. Classification-based and energy-efficient dynamic task scheduling scheme for virtualized cloud data center. IEEE Trans Cloud Comput 2021;9(4):1376–90.
[16] Lv Z, Chen D, Lv H. Smart city construction and management by digital twins and BIM big data in COVID-19 scenario. ACM Trans Multimedia Comput Commun Appl 2022;18(2s):1–21.
[17] Chen H, Zhu X, Liu G, Pedrycz W. Uncertainty-aware online scheduling for real-time workflows in cloud service environment. IEEE Trans Serv Comput 2021;14(4):1167–78.
[18] Bharany S, Badotra S, Sharma S, Rani S, Alazab M, Jhaveri RH, Gadekallu TR. Energy efficient fault tolerance techniques in green cloud computing: A systematic survey and taxonomy. Sustain Energy Technol Assessments 2022;53:102613.
[19] Medara R, Singh RS. A review on energy-aware scheduling techniques for workflows in IaaS clouds. Wireless Pers Commun 2022;125:1545–84.
[20] Choudhary A, Govil MC, Singh G, Awasthi LK, Pilli ES. Energy-aware scientific workflow scheduling in cloud environment. Cluster Comput 2022;25(6):3845–74.
[21] Ali HGEDH, Saroit IA, Kotb AM. Grouped tasks scheduling algorithm based on QoS in cloud computing network. Egypt Inf J 2017;18(1):11–9.
[22] Calheiros RN, Buyya R. Meeting deadlines of scientific workflows in public clouds with tasks replication. IEEE Trans Parallel Distrib Syst 2014;25(7):1787–96.
[23] Garg N, Singh D, Goraya MS. Energy and resource efficient workflow scheduling in a virtualized cloud environment. Cluster Comput 2021;24(2):767–97.
[24] Lee YC, Zomaya AY. Energy conscious scheduling for distributed computing systems under different operating conditions. IEEE Trans Parallel Distrib Syst 2011;22(8):1374–81.
[25] Li Z, Ge J, Hu H, Song W, Hu H, Luo B. Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds. IEEE Trans Serv Comput 2018;11(4):713–26.
[26] Safari M, Khorsand R. PL-DVFS: combining power-aware list-based scheduling algorithm with DVFS technique for real-time tasks in cloud computing. J Supercomput 2018;74(10):5578–600.
[27] Qureshi B. Profile-based power-aware workflow scheduling framework for energy-efficient data centers. Future Gener Comput Syst 2019;94:453–67.
[28] Rani R, Garg R. Power and temperature-aware workflow scheduling considering deadline constraint in cloud. Arab J Sci Eng 2020;45(12):10775–91.
[29] Fan G, Chen X, Li Z, Yu H, Zhang Y. An energy-efficient dynamic scheduling method of deadline-constrained workflows in a cloud environment. IEEE Trans Netw Serv Manage 2022 (in press).
[30] Kalra M, Singh S. A review of metaheuristic scheduling techniques in cloud computing. Egypt Inf J 2015;16(3):275–95.
[31] Houssein EH, Gad AG, Wazery YM, Suganthan PN. Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends. Swarm Evol Comput 2021;62:100841.
[32] Konjaang JK, Xu L. Meta-heuristic approaches for effective scheduling in infrastructure as a service cloud: A systematic review. J Netw Syst Manage 2021;29(2):1–57.
[33] Tarafdar A, Debnath M, Khatua S, Das RK. Energy and makespan aware scheduling of deadline sensitive tasks in the cloud environment. J Grid Comput 2021;19:1–25.
[34] Malik N, Sardaraz M, Tahir M, Shah B, Ali G, Moreira F. Energy-efficient load balancing algorithm for workflow scheduling in cloud data centers using queuing and thresholds. Appl Sci 2021;11(13):5849.
[35] Li H, Xu G, Wang D, Zhou M, Yuan Y, Alabdulwahab A. Chaotic-nondominated-sorting owl search algorithm for energy-aware multi-workflow scheduling in hybrid clouds. IEEE Trans Sustain Comput 2022;7(3):595–608.
[36] Qi L, Chen Y, Yuan Y, Fu S, Zhang X, Xu X. A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems. World Wide Web 2020;23(2):1275–97.
[37] Hussain M, Wei L-F, Rehman A, Abbas F, Hussain A, Ali M. Deadline-constrained energy-aware workflow scheduling in geographically distributed cloud data centers. Future Gener Comput Syst 2022;132:211–22.
[38] Domanal SG, Guddeti RMR, Buyya R. A hybrid bio-inspired algorithm for scheduling and resource management in cloud environment. IEEE Trans Serv Comput 2017;13(1):3–15.
[39] Cao K, Wang B, Ding H, Lv L, Tian J, Hu H, Gong F. Achieving reliable and secure communications in wireless-powered NOMA systems. IEEE Trans Veh Technol 2021;70(2):1978–83.
[40] Dai X, Xiao Z, Jiang H, Alazab M, Lui JC, Min G, Dustdar S, Liu J. Task offloading for cloud-assisted fog computing with dynamic service caching in enterprise management systems. IEEE Trans Industr Inf 2022;19(1):662–72.
[41] Xiao Z, Shu J, Jiang H, Min G, Chen H, Han Z. Perception task offloading with collaborative computation for autonomous driving. IEEE J Sel Areas Commun 2023;41(2):457–73.
[42] Zhao F, Jiang T, Wang L. A reinforcement learning driven cooperative meta-heuristic algorithm for energy-efficient distributed no-wait flow-shop scheduling with sequence-dependent setup time. IEEE Trans Industr Inf 2022 (in press).
[43] Wang J-J, Wang L. A cooperative memetic algorithm with feedback for the energy-aware distributed flow-shops with flexible assembly scheduling. Comput Ind Eng 2022;168:108126.
[44] Zhao F, Zhang L, Cao J, Tang J. A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem. Comput Ind Eng 2021;153:107082.
[45] Zhao F, Di S, Wang L. A hyperheuristic with Q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem. IEEE Trans Cybern 2022 [in press].
[46] Wang G-G, Gao D, Pedrycz W. Solving multiobjective fuzzy job-shop scheduling problem by a hybrid adaptive differential evolution algorithm. IEEE Trans Industr Inf 2022;18(12):8519–28.
[47] Pan Z, Lei D, Wang L. A knowledge-based two-population optimization algorithm for distributed energy-efficient parallel machines scheduling. IEEE Trans Cybern 2022;52(6):5051–63.
[48] Yan A, Yang K, Huang Z, Zhang J, Cui J, Fang X, Yi M, Wen X. A double-node-upset self-recoverable latch design for high performance and low power application. IEEE Trans Circuits Syst II Express Briefs 2018;66(2):287–91.
[49] Topcuoglu H, Hariri S, Wu M-Y. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst 2002;13(3):260–74.
[50] Das S, Suganthan PN. Differential evolution: A survey of the state-of-the-art. IEEE Trans Evol Comput 2010;15(1):4–31.
[51] Poli R, Kennedy J, Blackwell T. Particle swarm optimization. Swarm Intell 2007;1(1):33–57.

[52] Ahmed OH, Lu J, Xu Q, Ahmed AM, Rahmani AM, Hosseinzadeh M. Using differential evolution and Moth-Flame optimization for scientific workflow scheduling in fog computing. Appl Soft Comput 2021;112:107744.
[53] Kumar M, Sharma SC. PSO-COGENT: Cost and energy efficient scheduling in cloud environment with deadline constraint. Sustain Comput: Inf Syst 2018;19:147–64.
[54] Cao E, Musa S, Chen M, Wei T, Wei X, Fu X, Qiu M. Energy and reliability-aware task scheduling for cost optimization of DVFS-enabled cloud workflows. IEEE Trans Cloud Comput 2022 [in press].
[55] Juve G, Chervenak A, Deelman E, Bharathi S, Mehta G, Vahi K. Characterizing and profiling scientific workflows. Future Gener Comput Syst 2013;29 (3):682–92.
[56] Berriman GB, Deelman E, Good JC, Jacob JC, Katz DS, Kesselman C, Laity AC, Prince TA, Singh G, Su M-H. Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand. Optimizing scientific return for astronomy through information technologies, Vol. 5493. SPIE; 2004. p. 221–32.
[57] Laird PW. Institutional profile: The usc epigenome center. Epigenomics 2009;1 (1):29–31.
[58] Abbott B, Abbott R, Adhikari R, Ajith P, Allen B, Allen G, Amin R, Anderson S, Anderson W, Arain M. LIGO: the laser interferometer gravitational-wave observatory. Rep Prog Phys 2009;72(7):076901.
[59] Graves R, Jordan TH, Callaghan S, Deelman E, Field E, Juve G, Kesselman C, Maechling P, Mehta G, Milner K. Cybershake: A physics-based seismic hazard model for southern California. Pure Appl Geophys 2011;168(3):367–81.
[60] Livny J, Teonadi H, Livny M, Waldor MK. High-throughput, kingdom-wide prediction and annotation of bacterial non-coding RNAs. PLOS ONE 2008;3(9): e3197.