Journal of Systems Architecture 142 (2023) 102938

Contents lists available at ScienceDirect

Journal of Systems Architecture

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

Scheduling energy consumption-constrained workflows in heterogeneous multi-processor embedded systems

Jinchao Chen [a,*], Pengcheng Han [a,b], Ying Zhang [a], Tao You [a], Pengyi Zheng [c]

[a] School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China
[b] Xi'an Aeronautics Computing Technique Research Institute, AVIC, Xi'an 710075, China
[c] College of Information and Engineering, Shaanxi Institute of International Trade and Commerce, Xi'an 712046, China

ARTICLE INFO

Keywords:
Workflow scheduling
Heterogeneous multi-processor
Embedded system
Energy consumption

ABSTRACT

Heterogeneous multi-processor architecture which achieves rich functionalities with different types of processors, is widely used to provide powerful calculating capability while keeping energy consumption under control. Although this architecture can improve system flexibility for adapting to future requirement variations, it results in a complex multi-task scheduling problem for system designers to produce a reasonable schedule that satisfies all deadline, data dependency, and energy limitation constraints. In this paper, we concentrate on the energy consumption-constrained scheduling problem of workflows in heterogeneous multi-processor embedded systems. First, we model the workflows and energy consumption of processors, and formulate the energy consumption-constrained scheduling problem as an optimization one whose objective is to shorten the schedule length of workflows as much as possible. Then, with an improved energy per-assignment strategy, we propose a novel energy difference coefficient-based scheduling algorithm to produce an approximately optimal allocation of processors, frequencies, and start times for each task while guaranteeing that the data dependency and energy limitation constraints are satisfied. Finally, experiments on both randomly-generated and real-world workflows are conducted to verify the reliability and efficiency of the proposed approach.

1. Introduction

The great advancement in multi-processor architectures facilitates the implementation of various heterogeneous embedded systems, and there is an increasing trend toward achieving rich functionalities upon a multi-processor platform with different types of processors to offer powerful calculating capability while keeping energy consumption under control [1]. In these systems, each type of processor is designed specifically for efficiently dealing with a special functionality. Heterogeneous multi-processor architecture can adapt to various kinds of computing requirements [2,3] and provides significant advantages in computing capability, energy consumption, and application range, with regard to the homogeneous one.

Although heterogeneous multi-processor architectures can effectively accelerate the execution of real-time applications and improve system flexibility for adapting to future requirement variation, they result in a complex energy consumption-constrained scheduling problem for system designers to produce a reasonable schedule scheme for tasks while guaranteeing all constraints are completely satisfied. Especially in large-scale practical embedded applications, the energy consumption-constrained scheduling problem is one of the main bottlenecks to enhancing execution efficiency and control security [4]. Excessive energy consumption will not only reduce the lifetime of the processors but also damage the system's reliability. Meanwhile, a large amount of carbon dioxide emissions resulting from the excessive consumption of resources can exacerbate the greenhouse effect and severely harm human survival environment. Aiming to alleviate the effects of energy dissipation on economical, ecological, and social aspects, various energy-efficient scheduling techniques are designed to reduce energy consumption while meeting the deadline constraints [5]. A well-known technique named *d*ynamic *v*oltage and *f*requency *s*caling (DVFS) [6], has been widely used in energy-efficient scheduling and efficiently lowers the system energy consumption by simultaneously scaling down the supply voltages and frequencies of processors. Currently, DVFS-enabled processors are provided by most chip manufacturers (e.g., Intel, AMD, ARM, etc.) [7], and the corresponding scheduling algorithms are devised to fully realize the potential of DVFS for energy-efficient scheduling.

* Corresponding author.
E-mail addresses: cjc@nwpu.edu.cn (J. Chen), hanpengcheng_1990@nwpu.edu.cn (P. Han), zhang_ying@nwpu.edu.cn (Y. Zhang), youtao@nwpu.edu.cn (T. You), zhengpengyi@csiic.edu.cn (P. Zheng).

https://doi.org/10.1016/j.sysarc.2023.102938
Received 25 February 2023; Received in revised form 6 June 2023; Accepted 4 July 2023
Available online 7 July 2023

Along with the development of computers, microelectronics, and information processing, applications in embedded systems are increasingly parallel [8], and tasks in each application exhibit obvious data dependencies and are eventually described as a workflow. Workflow is usually modelled as a directed acyclic graph, in which nodes and edges respectively represent the tasks and dependencies between tasks [9]. The energy consumption-constrained scheduling problem of workflows studied in DVFS-enabled heterogeneous multi-processor systems is to select reasonable processors and frequencies for tasks, so as to minimize the energy consumption and makespan (i.e., schedule length of the workflow) while satisfying the data dependency and energy consumption constraints. This problem is quite difficult to solve since not only the multi-processor property gives it an NP-hard complexity [10], but also the energy consumption constraints add a further complication in obtaining feasible schedules.

In recent years, many significant approaches have been proposed to solve the energy consumption-constrained scheduling problem with different objectives and requirements [11]. Generally speaking, the existing approaches have two major steps: First, an energy pre-assignment strategy is adopted to divide the total energy and distribute a reasonable energy consumption limitation for every task. Then, the best processor, frequency, and time window are selected for each task while satisfying the pre-assigned energy consumption limitation. Since an appropriate energy consumption limitation is an essential prerequisite for guaranteeing the efficiently executing of workflows, energy pre-assignment strategies are regarded as the key part of improving the performance of these scheduling algorithms. However, most of the existing algorithms reserve the minimum energy for every task and divide the slack energy to tasks equally [12] or based on the minimum and maximum energy consumption [13], without fully and comprehensively evaluating the unique energy requirements of tasks, resulting in unsatisfactory or lower accuracy solutions. Meanwhile, due to the huge complexity of the energy consumption-constrained scheduling problem, most existing approaches neglect to consider the data dependencies of tasks and the heterogeneity feature of processors, prohibiting the widespread use of these results.

In this paper, we focus on solving the energy consumption-constrained scheduling problem of workflows in heterogeneous multi-processor systems, and try to propose a three-phase scheduling algorithm to ensure the correct and efficient running of tasks. The main contributions are summarized as follows:

1. Workflows and energy consumption in heterogeneous multi-processor systems are modelled, and the energy consumption-constrained scheduling problem of workflows is formulated as a single-objective optimization one that seeks the best processor, frequency, and start time for every task to shorten the schedule length of tasks under the data dependency and energy consumption constraints.

2. With an improved energy pre-assignment strategy that takes the variation tendency of task execution times on different processors into account, a novel energy difference coefficient-based scheduling algorithm is proposed to prioritize the tasks, assign energy limitations to tasks, and produce a reasonable schedule for tasks with a view to minimizing the schedule length.

3. Experiments on randomly-generated and real-world workflows are conducted to verify the performance of the proposed approach. Experimental results confirm that the schedule lengths obtained by our approach are shorter than those obtained by other approaches in different scales.

The remainder of this paper is organized as follows. Section 2 presents a brief overview of related works. Section 3 builds the related models and formulates the workflow scheduling problem. Section 4 presents the detailed design of the proposed energy difference coefficient-based algorithm. Section 5 gives the experimental results. Section 6 concludes this paper.

2. Related works

Scheduling techniques for workflows consisting of dependent tasks have been extensively researched and developed in the literature [14]. The goal of scheduling is to allocate tasks to appropriate processors so as to meet pre-defined requirements and optimize specific objectives. Before energy consumption is widely concerned, researchers were likely to pay attention to heuristic algorithms that improved the computing, storage, and communicating performance of workflow applications in various large-scale systems such as UAV swarms [15–17], IoT devices [18], and cloud environments [19,20]. The representative works include the *h*eterogeneous *e*arliest *f*inish *t*ime algorithm (HEFT) [21], *p*redict *e*arliest *f*inish *t*ime algorithm (PEFT) [22], and *t*ask *d*uplication-based *c*lustering *a*lgorithm (TDCA) [23], all of which can effectively shorten the schedule lengths of workflows while maintaining a low time complexity. However, all these algorithms cannot be directly adopted in energy consumption-constrained workflow scheduling problems, and the serious energy consumption becomes one of the most crucial elements that limit the extensive application of these results. In general, the existing scheduling algorithms for energy-constrained workflows are classified into three categories according to their optimization objectives and scheduling constraints: energy-efficient approaches, energy-conscious approaches, and energy-aware approaches [7].

Energy-efficient workflow scheduling approaches aim to achieve the minimum energy consumption while meeting pre-defined time requirements. These approaches usually begin with a feasible schedule generated by a makespan-minimization algorithm, and then employ a slack time reclaim technique to cut down the processor frequencies and lower the energy consumption [24]. For example, Huang et al. [25] generated a feasible schedule for workflows with HEFT, and allocated tasks to processors with the lowest frequencies, based on the latest finish time of tasks. Hu et al. [26] sought a group containing the maximum number of independent tasks, and tried to allocate slack energy to the group that could effectively reduce the total energy consumption. Xie et al. [27,28] employed a processor merging method to achieve the energy-saving workflow scheduling, by shutting down processors with the least number of tasks under the deadline constraints. Li [29] combined the slack time reclaim technique and the processor merging method to comprehensively lower the dynamic and static system energy costs. In [30], the authors partitioned tasks into different levels, evenly distributed the slack deadline to different levels, and minimized the energy consumption by locally reclaiming the free time units. Aiming to reduce the negative impact of frequency switching, Huang et al. [31] presented a DVFS-weakly scheduling approach for dependent tasks to seek the optimal frequencies for processors and DVFS was only allowed when tasks in applications were switched. Xie et al. [32] minimized the energy consumption of parallel applications by combining the non-DVFS and DVFS-enabled techniques together. All approaches mentioned above help in analysing the constraints and objectives of workflow scheduling problems, and can provide feasible solutions for cutting down the energy consumption upon a heterogeneous multi-processor platform.

Energy-conscious workflow scheduling approaches try to simultaneously optimize the energy consumption and other performance objectives such as reliability, development cost, and response time [33], and can greatly enhance the operating efficiency of real-time systems. Durillo et al. [34] introduced the Pareto dominance theory into the HEFT algorithm, and devised a *m*ulti-*o*bjective *HEFT* algorithm (MOHEFT) to shorten the response time and cut down the energy consumption at the same time. Zhang et al. [35] aggregated the schedule length and energy consumption reduction into a single optimization objective with a weighted average method, and employed the genetic algorithm to maximize the aggregated goal. Lee and Zomaya [6] proposed two DVFS-enabled energy-conscious scheduling heuristics to

address the joint optimization problem of schedule quality and energy consumption of workflows in a heterogeneous distributed system. Meanwhile, a relative advantage indicator was used to seek the best processor and frequency pair for every task, achieving a reasonable balance between energy consumption and schedule quality. Then the total energy consumption was further reduced by attempting to use the newly generated schedule to replace the original one.

Energy-aware workflow scheduling approaches focus on improving the system performance while meeting the energy requirement. As we pointed out previously, energy pre-assignment strategies, which allocate an energy limitation to each task, are the key part of these approaches and largely determine the scheduling performance. Song et al. [12] designed a fair energy pre-assignment policy by splitting the energy slack evenly to each task, and presented an *e*fficient *s*cheduling algorithm with *e*nergy *c*onsumption *c*onstraint (ESECC) to solve the problem. Xiao et al. [13] reserved a minimum energy for each task, and proposed a *m*inimum *s*chedule *l*ength with *e*nergy *c*onsumption *c*onstraint (MSLECC) approach to allocate tasks to the best processors on which tasks would finish their executions in the shortest time. This energy pre-assignment policy was overly pessimistic and forced tasks with low priorities to be executed on relatively poor processors with low energy consumption but a long execution time. Considering the different demands of tasks on energy consumption, Quan et al. [36] devised a weight-based energy pre-assignment policy and introduced an *i*mproved *s*cheduling *a*pproach for *e*nergy *c*onsumption *c*onstrained workflows (ISAECC) to distribute energy slack to tasks according to their minimum and maximum energy costs. Similarly, Xie et al. [37] proposed an improved energy pre-assignment policy and designed a *r*atio-based *e*nergy *p*re-assignment algorithm (REP) to split the slack energy into several aspects according to the minimum energy span of tasks.

In recent two years, lots of energy-aware scheduling algorithms have been proposed by pre-assigning energy limitations to tasks and scheduling the tasks while satisfying the pre-assigned energy limitations. Peng et al. [38] first divided the allocatable energy into two parts, i.e., the static pre-allocate energy that was proportionally distributed to each task and the dynamic pre-allocate energy that would be shared by all tasks during their executions, and then proposed a two-stage hybrid energy allocation strategy to solve the energy-aware workflow scheduling problem. Chen et al. [24] assigned an energy limitation to each task according to its weight computed by average energy consumption, and presented a novel list-based heuristic to find nearly optimal processors and start times for tasks while meeting the energy consumption constraints. Zhu et al. [39] allocated the free energy to tasks according to their average execution time among different processors, and designed a structure-aware task scheduling strategy for the parallel applications to enhance the scheduling performance. Although all the above approaches can find feasible solutions for energy-aware workflow scheduling problems, they did not fully evaluate the unique energy costs of tasks and got an unsatisfactory scheduling result due to the unfair energy pre-assignment policies.

3. System models and problem formulation

In this section, we first model the workflows and the energy consumption of processors in a heterogeneous embedded system, and then formulate the energy consumption-constrained scheduling problem researched in this paper.

3.1. Workflow model

We consider an embedded system where several fully connected heterogeneous processors are used to execute the workflows. Let $U = \{u_1, u_2, \dots, u_{|U|}\}$ denote the processor set, and processors support the DVFS technologies, i.e., each processor can work at different and discrete voltages and frequencies. In this work, we assume that computation can overlap with communication, that is to say, each processor is capable of sending (or receiving) data to (or from) others while processing a task.

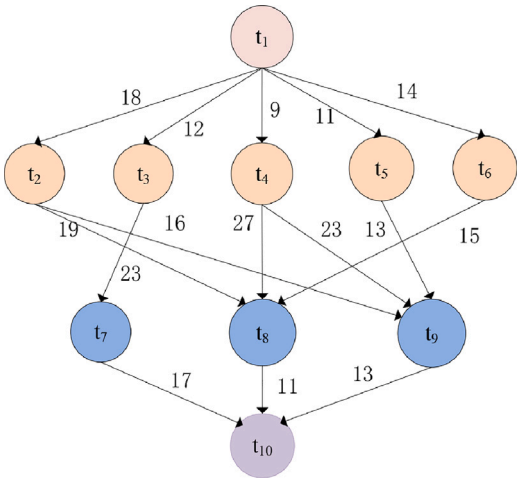


Fig. 1. A DAG-based workflow model with ten tasks [24].

A workflow application is represented by a directed acyclic graph (DAG) $G = (T, E, W, C)$. $T = \{t_1, t_2, \dots, t_{|T|}\}$ is the node set, and each node t_i represents a task. $E \subseteq T \times T$ is the edge set, describing the data dependencies between tasks. Each element $e_{i,j}$ implies that task t_j could start to run only after having received the necessary data from t_i. In this case, t_i and t_j are referred to as predecessor and successor, respectively. Two notations $succ(t_i)$ and $pred(t_i)$ are used to respectively denote all immediate successors and predecessors of t_i, then $succ(t_i) = \{t_j \mid \forall j, e_{i,j} \in E\}$, $pred(t_i) = \{t_j \mid \forall j, e_{j,i} \in E\}$. W is a two-dimension matrix representing the execution times of tasks executed on each processor. Due to the heterogeneous property of processors, tasks require different execution times on different processors or the same processor with different frequencies. Each element $w_{i,k} \in W$ denotes the execution time of t_i when it runs on u_k with the maximum frequency. C is a two-dimension matrix denoting the communication costs between tasks, and each element $c_{i,j}$ represents the time cost required by data transformed from t_i to t_j. Fig. 1 gives an example of a DAG-based workflow model with ten tasks.

In this work, we do not consider the deadlines of tasks in workflows, and assume that each workflow has one and only one entry and exit task. If several entry (or exit) tasks are used in a workflow, a dummy entry (or exit) task with zero execution time is linked to the original entry (or exit) tasks by edges with zero communication time.

3.2. Energy consumption model

Following the prior works [12,13,36,37], we use a system-level power model to compute the energy consumption of a workflow. Specifically, the power of a given processor u_k, denoted by $P_k(f)$, is modelled as a function of the frequency f and calculated via:

$$P_k(f) = P_{k,s} + h(P_{k,ind} + P_{k,d}) = P_{k,s} + h(P_{k,ind} + C_{k,ef} f^{m_k}) \quad (1)$$

where $P_{k,s}$, $P_{k,ind}$, and $P_{k,d}$ represent the static power, the frequency-independent power, and the frequency-dependent dynamic power of processor u_k, respectively. h is a boolean factor denoting the processor state. If $h = 1$, the processor is active and dynamic power is consumed; otherwise, the processor is inactive and no dynamic power is consumed. $C_{k,ef}$ and m_k denote the effective switching capacitance and the dynamic power exponent of u_k, respectively.

Since the static power $P_{k,s}$ is always consumed and unmanageable, we ignore the static power and focus on the dynamic power

Table 1
Power parameters of three processors.

	$P_{k,ind}$	$C_{k,ef}$	m_k	$f_{k,min}$	$f_{k,max}$
u_1	0.03	0.9	2.8	0.24	1.0
u_2	0.04	0.8	2.5	0.24	1.0
u_3	0.06	1.0	2.5	0.27	1.0

in the energy optimization process. We use $f_{k,min}$ and $f_{k,max}$ to respectively represent the minimum and the maximum frequency of a given processor u_k, then the actual frequencies used by tasks when they are executed on processor u_k should be selected from the set $f_k = \{f_{k,min}, f_{k,\alpha}, f_{k,\beta}, \ldots, f_{k,max}\}$. Table 1 gives an example of the power parameters of three heterogeneous processors.

The execution time of task t_i on processor u_k is nearly inversely proportional to the currently used frequency $f_{k,c}$, and can be computed via:

$$ET(t_i, u_k, f_{k,c}) = \frac{f_{k,max}}{f_{k,c}} w_{i,k}$$

Then the energy consumption of t_i upon processor u_k with frequency $f_{k,c}$, denoted by $E(t_i, u_k, f_{k,c})$, is calculated via:

$$E(t_i, u_k, f_{k,c}) = (P_{k,ind} + C_{k,ef}(f_{k,c})^{m_k}) ET(t_i, u_k, f_{k,c}) \quad (2)$$

Owing the existence of $P_{k,ind}$, the minimum frequency does not mean the minimum dynamic energy consumption. It is very necessary and meaningful to find the energy-efficient frequency that the considered task would have the least energy consumption. Here, we assume the frequency $f_{k,c}$ is a continuous variable, then the derivative of Eq. (2) could be obtained via:

$$\frac{d(E(t_i, u_k, f_{k,c}))}{d(f_{k,c})} = w_{i,k} f_{k,max} \left(C_{k,ef}(m_k - 1)\left(f_{k,c}\right)^{m_k - 2} - P_{k,ind}\left(f_{k,c}\right)^{-2} \right) \quad (3)$$

The energy-efficient frequency of processor u_k, denoted by $f_{k,ee}$, can be computed by setting the value of Eq. (3) to zero, then

$$f_{k,ee} = \sqrt[m_k]{\frac{P_{k,ind}}{(m_k - 1)C_{k,ef}}} \quad (4)$$

Accordingly, the minimum frequency of processor u_k in practical use should be updated to $f_{k,low} = max\{f_{k,min}, f_{k,ee}\}$. The minimum and maximum energy consumption of task t_i upon all processors is respectively calculated via the following equations:

$$E_{min}(t_i) = \min_{u_k \in U} \{E(t_i, u_k, f_{k,low})\}$$

$$E_{max}(t_i) = \max_{u_k \in U} \{E(t_i, u_k, f_{k,max})\}$$

Then, with the above equations, the minimum and maximum energy consumption of a workflow G could be computed via:

$$E_{min}(G) = \sum_{i=1}^{|T|} E_{min}(t_i) \quad (5)$$

$$E_{max}(G) = \sum_{i=1}^{|T|} E_{max}(t_i) \quad (6)$$

We use $E_{limit}(G)$ to represent the energy limitation of a workflow G. If $E_{limit}(G)$ is less than $E_{min}(G)$, no scheduling solution satisfies the energy constraint. Conversely, if $E_{limit}(G)$ is larger than $E_{max}(G)$, the energy constraint of any scheduling solution is always satisfied. Therefore, in this paper, we assume that $E_{limit}(G)$ is seated in the interval $[E_{min}(G), E_{max}(G)]$, i.e.,

$$E_{min}(G) \le E_{limit}(G) \le E_{max}(G)$$

3.3. Problem formulation

The energy consumption-constrained scheduling problem researched in this paper is defined as seeking an optimal processor, frequency, and start time allocation for every task in a workflow, such that the schedule length of the workflow would be minimized while meeting the data dependency and energy limitation constraints.

We adopt three vectors $\vec{p} = (u_{p_i})$, $\vec{f} = (f_i)$, and $\vec{s} = (s_i)$ to respectively represent the processors, frequencies, and start times selected for tasks in a workflow G. Then the finish time of task t_i (denoted by $FT(t_i)$) and the schedule length of the workflow G (denoted by $M(G)$) are computed via:

$$\begin{aligned} FT(t_i) &= ET(t_i, u_{p_i}, f_i) + s_i \\ M(G) &= \max_{t_i \in T} FT(t_i) \end{aligned} \quad (7)$$

The energy consumption-constrained workflow scheduling problem researched in this paper is formally stated as follows:

minimize $M(G)$

subject to:

$$u_{p_i} \in U, s_i \ge 0, f_{p_i,mim} \le f_i \le f_{p_i,max} \quad (8)$$

$$\sum_{t_i \in T} E(t_i, u_{p_i}, f_i) \le E_{limit}(G) \quad (9)$$

$$\forall t_i \in T, s_i \ge \max_{t_j \in Pred(t_i)} \{FT(t_j) + c_{j,i}\} \quad (10)$$

$$\forall t_i, t_j \in T, \{p_i \ne p_j\} \vee \{s_i \ge FT(t_j)\} \vee \{FT(t_i) \le s_j\} \quad (11)$$

Constraint (8) shows the value ranges of parameters. Constraint (9) requires that the energy consumption of a workflow could not exceed its limitation. Constraint (10) expresses the dependency constraints of tasks that each task starts to run only after all data has been received from its predecessors. Constraint (11) requires that tasks assigned to the same processor have no overlapping time units.

4. Energy difference coefficient-based scheduling algorithm

In this section, an energy difference coefficient-based scheduling algorithm (SEDC) is designed to obtain approximately optimal processors, frequencies, and start times for tasks with a view to reducing the schedule lengths of workflows. The proposed SEDC algorithm is constituted of three major phases as follows: a *task prioritization phase* for distributing priorities to tasks and constructing a task sequence according to which tasks would be scheduled successively, an *energy pre-assignment phase* for assigning an appropriate energy limitation to each task while keeping the system energy consumption under control, and a *task assignment phase* for choosing the most appropriate processors, frequencies, and start times for tasks to minimize the schedule lengths. Please note that although the proposed SEDC algorithm spends energy in the three major phases, its energy consumption is not included in the energy consumption of the workflows. The proposed SEDC algorithm is a typical static and off-line scheduling one, and tries to minimize the schedule length while ensuring that the energy limitation of workflows is not exceeded.

4.1. Task prioritization phase

As we pointed out previously, the dependency constraint requires that a successor task could not begin to run until all data from its predecessors has been received. Therefore, task prioritization is required to ensure that the successor task is scheduled after its predecessors. Previous works such as [12,13,36], and [37], always employ the upward rank value ($rank_u$) or the downward rank value ($rank_d$) presented in [21] as the task prioritization standard. Although both $rank_u$ and

$rank_d$ are very efficient, they are computed via the average computation time of each task among all processors, and cannot reflect the variation trend of the execution times on heterogeneous processors. In this subsection, we adopt an improved ranking function based on *time difference coefficient* to compute the priorities of tasks and create a more reasonable task sequence.

The time difference coefficient of a task t_i, denoted by $TD(t_i)$, is defined as the coefficient of variation of the execution time of t_i, and computed through dividing the standard deviation by the mean value of its execution times among all processors, i.e.,

$$TD(t_i) = \frac{\sqrt{\sum_{k=1}^{|U|}(w_{i,k} - \overline{w_i})^2/(|U|-1)}}{\overline{w_i}} \tag{12}$$

where $\overline{w_i} = \frac{\sum_{k=1}^{|U|} w_{i,k}}{|U|}$ represents the average execution time of t_i upon processors with the maximum frequencies. From Eq. (12) we can see that, the value of the time difference coefficient increases with the differences among the execution times on processors. Therefore, time difference coefficient would be a recognized sign of the heterogeneity of processors, and reflects the variation trend of the execution time of a given task. We introduce the time difference coefficient into the computation process of priorities and assign a more effective priority to each task.

With the time difference coefficient, the modified upward rank value (denoted by $rank'_u$) and the modified downward rank value (denoted by $rank'_d$) are recursively defined and calculated via:

$$rank'_u(t_i) = (1 + TD(t_i)) \times \overline{w_i} + \max_{t_j \in succ(t_i)} \{rank'_u(t_j) + c_{i,j}\} \tag{13}$$

$$rank'_d(t_i) = \max_{t_j \in pred(t_i)} \{rank'_d(t_j) + \overline{w_i} \times (1 + TD(t_i)) + c_{i,j}\} \tag{14}$$

As shown in Eqs. (13) and (14), the modified upward and downward ranks represent the longest weighted lengths of the critical paths to the exit node and to the entry node, respectively. Since the variation tread of task execution times is taken into consideration, task priorities calculated with the modified upward and downward ranks are more reasonable and effective.

The priority of a given task t_i, denoted by $P(t_i)$, is recursively calculated by traversing the DAG from both the entry and the exit tasks as follows:

$$P(t_i) = rank'_u(t_i) + rank'_d(t_{exit}) - rank'_d(t_i) \tag{15}$$

where $rank'_d(t_{exit})$ is used to ensure that all priorities are positive. Using this improved ranking function, we build a task sequence by sorting the tasks in decreasing order according to their priorities, and tasks in this sequence would be scheduled one by one.

4.2. Energy pre-assignment phase

In this subsection, a weight-based energy pre-assignment policy is proposed to divide the slack energy and distribute an energy limitation to each task according to its weight. In order to fully consider the effects of the heterogeneity of processors on the energy consumption of tasks, we propose a novel ranking function based on *energy difference coefficient* to compute the energy weights of tasks.

The energy difference coefficient of a task is defined as the ratio between the standard deviation and the average value of its energy consumption on processors. Since tasks could be executed on any processor with any given frequency, the energy difference coefficient of a task on a processor should be calculated with all frequencies. However, as we pointed out in Section 3.2, the frequency of each processor in this work is a continuous variable. Therefore, only the maximum and the minimum energy consumption of tasks on each processor is considered when the energy difference coefficients of tasks are calculated. We adopt $ED_{min}(t_i)$ and $ED_{max}(t_i)$ to respectively denote the energy difference coefficient of task t_i on the minimum and the maximum energy consumption, then

$$\begin{aligned} ED_{min}(t_i) &= \frac{\sqrt{\sum_{k=1}^{|U|}(E(t_i,u_k,f_{k,min}) - \overline{E_{min}(t_i)})^2/(|U|-1)}}{\overline{E_{min}(t_i)}} \\ ED_{max}(t_i) &= \frac{\sqrt{\sum_{k=1}^{|U|}(E(t_i,u_k,f_{k,max}) - \overline{E_{min}(t_i)})^2/(|U|-1)}}{\overline{E_{max}(t_i)}} \end{aligned} \tag{16}$$

where $\overline{E_{min}(t_i)}$ and $\overline{E_{max}(t_i)}$ respectively represent the average values of the minimum and the maximum energy consumption of t_i on all processors, i.e.,

$$\begin{aligned} \overline{E_{min}(t_i)} &= \frac{\sum_{k=1}^{|U|} E(t_i,u_k,f_{k,min})}{|U|} \\ \overline{E_{max}(t_i)} &= \frac{\sum_{k=1}^{|U|} E(t_i,u_k,f_{k,max})}{|U|} \end{aligned}$$

The energy difference coefficient reflects how far the energy consumption on different processors is spread out from the average value. A larger energy difference coefficient means the differences between energy consumption on processors are greater, and implies that the considered task may need more energy to get a better execution. We apply the energy difference coefficient into the calculation process of energy weights, and the energy weight of a given task t_i (denoted by $E_{weight}(t_i)$) is defined and computed via:

$$\begin{aligned} E_{weight}(t_i) = &\left(1 + ED_{max}(t_i)\right) \times \overline{E_{max}(t_i)} \\ &+ \left(1 + ED_{min}(t_i)\right) \times \overline{E_{min}(t_i)} \end{aligned} \tag{17}$$

Assume the task to be scheduled is $t_{s(j)}$, where $s(j)$ denotes the jth task in the task sequence built according to priorities calculated in Section 4.1. Then, tasks in workflow G can be classified into two sets: an already allocated task set $\{t_{s(1)}, t_{s(2)}, \ldots, t_{s(j-1)}\}$ and an unallocated task set $\{t_j, t_{s(j+1)}, \ldots, t_{s(|T|)}\}$. The current slack energy for all unallocated tasks, denoted by $E_{slack,j}(G)$, could be calculated via:

$$E_{slack,j}(G) = E_{limit}(G) - \sum_{l=1}^{j-1} E(t_{s(l)}, u_{s(l)}, f_{s(l)}) - \sum_{l=j}^{|T|} E_{min}(t_{s(l)}) \tag{18}$$

where $u_{s(l)}$ and $f_{s(l)}$ denote the assigned processor and frequency for an already allocated task $t_{s(l)}$, respectively. Then the pre-assigned energy limitation for task $t_{s(j)}$ can be calculated via:

$$E_{pre}(t_{s(j)}) = E_{min}(t_{s(j)}) + \frac{E_{weight}(t_{s(j)})}{\sum_{k=j}^{|T|} E_{weight}(t_{s(k)})} \times E_{slack,j}(G) \tag{19}$$

4.3. Task assignment phase

In this section, tasks are sequentially scheduled according to their priorities and an appropriate processor, frequency, and start time would be selected for each task with an objective of reducing the makespan while satisfying the energy limitation computed in Section 4.2. Since earliest finish time (EFT) is a classic task-processor mapping strategy with a low complexity and high performance, we also adopt the EFT strategy to assign the considered task to the best processor upon which this task can finish its execution in the shortest time while meeting the pre-assigned energy limitation.

For a given task t_i, its earliest start time (EST) on processor u_k is the maximum value between the available time of processor u_k and the ready time of all data from its predecessors, i.e.,

$$EST(t_i, u_k) = \max\left\{avail[u_k], \max_{t_j \in pred(t_i)} \{AFT(t_j) + c'_{j,i}\}\right\} \tag{20}$$

where $avail[u_k]$ represents the earliest available time of processor u_k, and $AFT(t_j)$ is the actual finish time of t_j. $c'_{j,i}$ represents the actual

data transmission time from task t_j to t_i. If t_j and t_i are executed upon the same processor, $c'_{j,i} = 0$; otherwise, $c'_{j,i} = c_{j,i}$, i.e.,

$$c'_{j,i} = \begin{cases} 0, & \text{if } t_j \text{ and } t_i \text{ are executed upon the same processor;} \\ c_{j,i}, & \text{otherwise} \end{cases}$$

We use $f_{best}(t_i, u_k)$ and $EFT(t_i, u_k)$ to respectively denote the best frequency of processor u_k and the earliest finish time of t_i upon u_k while meeting the pre-assigned energy limitation, then

$$\begin{aligned} f_{best}(t_i, u_k) &= \arg\min_{f_{k,c} \in f_k} ET(t_i, u_k, f_{k,c}) \ s.t. \ E(t_i, u_k, f_{k,c}) \le E_{pre}(t_i) \\ EFT(t_i, u_k) &= EST(t_i, u_k) + ET(t_i, u_k, f_{best}(t_i, u_k)) \end{aligned} \tag{21}$$

After the earliest finish times on processors are computed, the best processor, frequency, and start time (denoted by $u_{best}(t_i)$, $f_{best}(t_i)$, and $s_{best}(t_i)$, respectively) of task t_i is selected as follows:

$$\begin{aligned} u_{best}(t_i) &= \arg\min_{u_k \in U} EFT(t_i, u_k) \\ f_{best}(t_i) &= f_{best}\left(t_i, u_{best}(t_i)\right) \\ s_{best}(t_i) &= EST\left(t_i, u_{best}(t_i)\right) \end{aligned} \tag{22}$$

4.4. Algorithm description

In this subsection, we briefly introduce the details of the proposed SEDC algorithm, and its pseudo-code is outlined as Algorithm 1.

Algorithm 1: Scheduling algorithm based on the energy difference coefficient (SEDC)

Input: Workflow G, energy limitation $E_{limit}(G)$, processor set U
Output: Schedule length of the workflow G

1 **for** $t_i \in T$ **do**
2 Compute the time difference coefficient of t_i with Eq. (12);
3 Compute the priority of t_i with Eq. (15);
4 **end**
5 Sort all tasks in decreasing order according to their priorities;
6 Calculate the energy difference coefficients of all tasks with Eq. (16);
7 **while** *there are unallocated tasks* **do**
8 Choose the first task, t_i, from the ordered task sequence;
9 Calculate the energy weight of t_i with Eq. (17);
10 Calculate the current slack energy with Eq. (18);
11 Calculate the pre-assigned energy limitation of t_i with Eq. (19);
12 **for** $u_k \in U$ **do**
13 Calculate the EST and EFT of t_i on processor u_k according to Eq. (20) and Eq. (21);
14 **end**
15 Select the best processor, frequency and start time for t_i according to Eq. (22);
16 Schedule t_i according to the selected configuration;
17 Remove t_i from the ordered task sequence;
18 **end**
19 **return** *the actual finish time of the exit task*;

We start by calculating the time difference coefficients (Line 2) and the priorities (Line 3) of tasks, and building a task sequence by sorting all tasks in decreasing order according to their priorities (Line 5). Then we compute the energy difference coefficient of each task (Line 6), and sequentially schedule the tasks according to their priorities (Lines 7 to 18). For each unallocated task, we calculate its energy weight (Line 9), current slack energy (Line 10), pre-assigned energy limitation (Line 11), earliest start time, earliest finish time on each processor (Lines 12 to 14), and allocate it to the best processor where it can have the smallest EFT while under the pre-assigned energy limitation (Lines 15 to 16). Thereafter, we remove the considered task out from the ordered task sequence (Line 17). Since only the current slack energy is divided among the unallocated tasks and each task is scheduled under the energy limitation, the proposed algorithm provides a guarantee that the energy consumption of tasks does not exceed the pre-defined energy budget. The SEDC algorithm runs until all tasks have been assigned to processors, and the schedule length, i.e., the actual finish time of the exit task would be returned.

Now we analyse the complexity of the proposed SEDC algorithm. From Algorithm 1 we can see that the most time-consuming part of this algorithm is from lines 7 to 18, which has a cycle structure and would be repeated $|T|$ times where $|T|$ represents the task number in the workflow. In each iteration, the pre-assigned energy limitation of the considered task is finished in $O(|T| \times |U|)$ where $|U|$ is the number of processors. Meanwhile, all processors, frequencies and predecessor tasks should be traversed to find the optimal processor, frequency, and start time for every task, which runs in $O(|T| \times |U| \times |F|)$, where $|F|$ is the number of optional frequencies in an interval $[f_{min}, f_{max}]$. Then the running-time complexity of each iteration is $O(|T| \times |U| \times |F|)$. Since the iteration repeats $|T|$ times, the total complexity of the SEDC algorithm is $O(|T|^2 \times |U| \times |F|)$, which is the same as those of the widely-used state-of-the-art approaches such as ESECC [12], MSLECC [13], ISAECC [36], and REP [37].

5. Experimental results

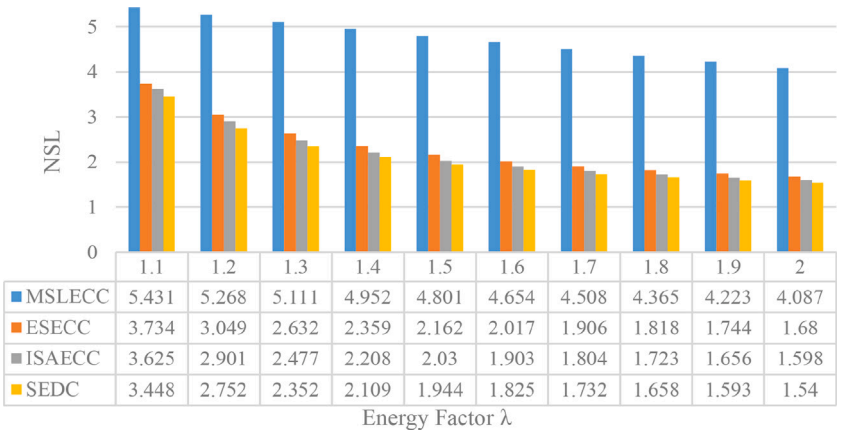
In this section, experiments on randomly-generated and real-world workflows are conducted to evaluate the performance of the proposed approach. To achieve a comprehensive and fair evaluation, the proposed SEDC algorithm is compared with three existing algorithms including MSLECC [13], ESECC [12], and ISAECC [36]. The reasons for choosing these three algorithms are that they are widely adopted in practical applications and solve the same workflow scheduling problem as ours with different energy pre-assignment strategies. Specifically, MSLECC only reserves the minimum energy consumption for every task, ESECC equally splits the slack energy into tasks, while ISAECC adopts a weight-based and customizable way to distribute the slack energy by considering the minimum and maximum energy values of tasks. In the following, we firstly introduce the experiment settings and performance metrics, then show the experiment results and discuss the performance on the randomly-generated and real-world workflows sequentially.

5.1. Experiment settings and performance metrics

Since simulation techniques can provide a configuration controllable and repeatable manner for designers to quantitatively verify the performance of workflow scheduling approaches, we build a simulated heterogeneous multi-processor platform to execute the workflows and compare their performance. The simulated platform is built on the 64-bit Windows 10 operating systems version 21H1 with 64 processors and each processor has an 8-core 3.6 GHz CPU and 16 GB of system memory. The parameters of the workflows and processors are configured as follows [8]: $10 \text{ ms} \le w_{i,k} \le 100 \text{ ms}$, $10 \text{ ms} \le c_{i,j} \le 100 \text{ ms}$, $0.03 \le P_{k,ind} \le 0.07$, $0.8 \le C_{k,ef} \le 1.2$, $2.5 \le m_k \le 3.0$, and $f_{k,max} = 1.0$ GHz. Meanwhile, the frequency precision of processors is 0.01, and the minimum frequency of processors is set to $f_{k,ee}$ calculated by Eq. (4).

When evaluating the performance of different scheduling approaches, a straightforward manner is to compare the scheduling lengths of solutions obtained by these approaches. However, the scale and structure of workflows, as well as the performance and power of the processors, make the schedule lengths have different orders of magnitude. Therefore, it is necessary to normalize the schedule lengths of workflows and compare the performance of approaches with the normalized values. In this section, schedule lengths obtained by the HEFT algorithm without any energy limitations are applied as a standard for comparisons, and the normalized schedule length (NSL) of a specific approach is defined as the ratio between the schedule lengths of solutions obtained by this approach and by the HEFT algorithm, i.e.,

$$NSL(G) = \frac{SL_{Approach}(G)}{SL_{HEFT}(G)} \tag{23}$$



	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2
MSLECC	5.431	5.268	5.111	4.952	4.801	4.654	4.508	4.365	4.223	4.087
ESECC	3.734	3.049	2.632	2.359	2.162	2.017	1.906	1.818	1.744	1.68
ISAECC	3.625	2.901	2.477	2.208	2.03	1.903	1.804	1.723	1.656	1.598
SEDC	3.448	2.752	2.352	2.109	1.944	1.825	1.732	1.658	1.593	1.54

Fig. 2. Average normalized schedule lengths of four algorithms when the energy factor λ varies from 1.1 to 2.0.

5.2. Experimental results on randomly-generated workflows

In this subsection, workflows are randomly generated by a synthetic DAG generator [22] to fully test the performance of scheduling algorithms. The six parameters for shaping the structure of the workflows are described as follows: the number of tasks n, sharp parameter fat that determines the overall structure of a workflow and the task number of each level would be produced by a uniform distribution with a mean value $fat \times \sqrt{n}$, out degree of task $density$ that provides the average edge number between tasks in different levels, regularity coefficient $regularity$ that determines the uniformity of the task numbers among different levels, jump coefficient $jump$ that allows tasks to establish connections with others in subsequent $jump$ levels at most, and energy factor λ that controls the energy limitations of workflows and the energy limitation of a given workflow G is computed via $E_{limit}(G) = \lambda E_{min}(G)$. As we pointed out in Section 3.2, if the energy limitation $E_{limit}(G)$ is less than $E_{min}(G)$, no scheduling solution exists, indicating that the energy factor λ needs to be greater than or equal to 1.0. However, when $\lambda = 1.0$, the only scheduling solution is to allocate each task to the processor which ensures the task can have the minimum energy consumption. In order to make experiments more valuable, we set the minimum value of the parameter λ to 1.1. Meanwhile, we find that when $\lambda > 5.0$, the difference between the schedule lengths obtained by different approaches is very small. Therefore, in our experiments, the energy factor λ is set to a discrete value range of 1.1 to 5.0. The values of the above parameters are randomly chosen from sets as follows:

- $SET_n = \{20, 40, 60, 80, 100, 200, 300, 400\}$;
- $SET_{fat} = \{0.2, 0.4, 0.8\}$;
- $SET_{density} = \{0.1, 0.2, 0.4, 0.8\}$;
- $SET_{regularity} = \{0.2, 0.4, 0.8\}$;
- $SET_{jump} = \{1, 2, 3\}$;
- $SET_{\lambda} = \{1.1, 1.2, \ldots, 5.0\}$.

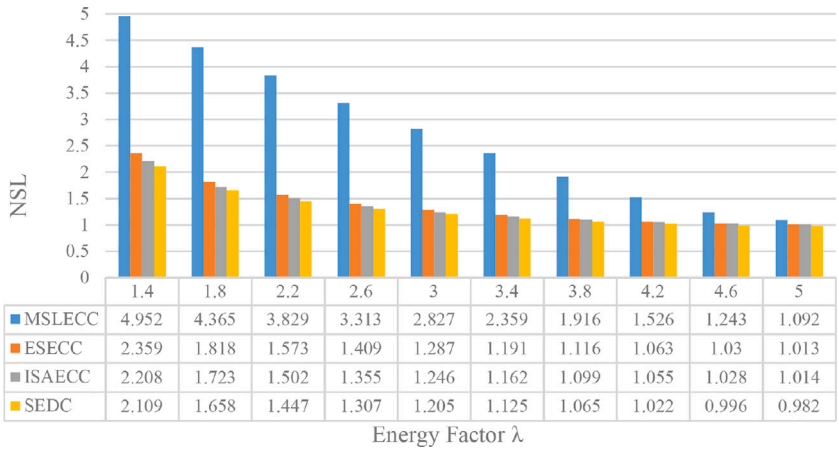
Experiment 1. We start by comparing the normalized schedule lengths of different scheduling approaches by varying the energy limitations. First, tight energy limitations are set to generated workflows and λ is increasing from 1.1 to 2.0 with a precision equal to 0.1. Fig. 2 shows the average normalized schedule lengths of different approaches. Please note that each point in this figure (also in other figures used in this section) represents the average value of experimental results of 100 instances. As can be seen from Fig. 2, the normalized schedule lengths of approaches go down with values of the energy factor λ. The reason is that, a larger energy factor λ provides each workflow a larger energy limitation, and faster processors with higher frequencies will be selected to execute the tasks and greatly shorten the schedule lengths. The average performance of MSLECC in terms of schedule length is far worse than that of ESECC, ISAECC, and SEDC, indicating the importance of a reasonable energy pre-assignment strategy. Going one step further, our approach named SEDC performs better than the other three algorithms under all energy limitation settings, demonstrating that our energy pre-assignment strategy based on the energy difference coefficient is effective. When the energy factor λ is equal to 1.5, the average normalized schedule length of our approach SEDC is 1.944, which is 59.51%, 10.08%, and 4.24% smaller than those of MSLECC, ESECC, and ISAECC, respectively.

Table 2
Numbers of better, equal, and worse solutions obtained by different approaches.

		MSLECC	ESECC	ISAECC	SEDC	Combined
MSLECC	Better		1	1	1	0.003%
	Equal	*	0	0	0	0.0%
	Worse		34559	34559	34559	99.997%
ESECC	Better	34559		4715	2830	40.610
	Equal	0	*	0	0	0.0%
	Worse	1		29845	31730	59.390%
ISAECC	Better	34559	29845		8242	69.983%
	Equal	0	0	*	4	0.004%
	Worse	1	4715		26314	30.013
SEDC	Better	34559	31730	26314		89.316%
	Equal	0	0	4	*	0.004%
	Worse	1	2830	8242		10.680%

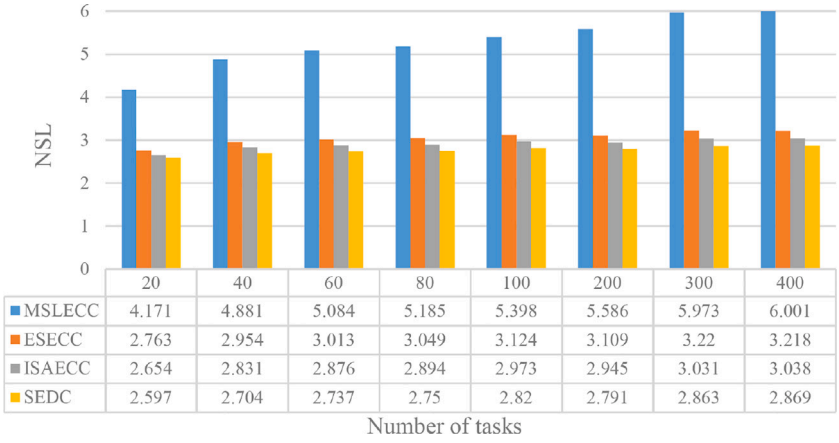
Subsequently, we collect the number of better, equal, and worse solutions obtained by every approach, and the results are shown in Table 2. Since all approaches are tested on the same randomly-generated workflows, a higher number of better solutions indicates a more accurate scheduling approach. As can be found in Table 2, the number of better solutions found by our approach SEDC is much larger than those found by other approaches. Compared to MSLECC, ESECC and ISAECC, the probabilities that our approach produces a better solution are 99.99%, 90.76%, and 76.14%, respectively, demonstrating that our approach SEDC can find feasible solutions with shorter schedule lengths for the energy consumption-constrained workflow scheduling problems.

For the purpose of comprehensively evaluating the performance of the four scheduling approaches, a larger value of the energy factor λ is used in the following experiments. We set the minimum and the maximum values of λ to 1.4 and 5.0, respectively. Fig. 3 shows the average normalized schedule lengths of solutions produced by different approaches. An interesting phenomenon revealed by Fig. 3 is that, along with the increasing of the energy factor λ, the schedule lengths achieved by all approaches are coming close to those obtained by the HEFT algorithm, and our approach SEDC can even find a better solution than HEFT. For example, when $\lambda = 5.0$, the average normalized



	1.4	1.8	2.2	2.6	3	3.4	3.8	4.2	4.6	5
MSLECC	4.952	4.365	3.829	3.313	2.827	2.359	1.916	1.526	1.243	1.092
ESECC	2.359	1.818	1.573	1.409	1.287	1.191	1.116	1.063	1.03	1.013
ISAECC	2.208	1.723	1.502	1.355	1.246	1.162	1.099	1.055	1.028	1.014
SEDC	2.109	1.658	1.447	1.307	1.205	1.125	1.065	1.022	0.996	0.982

Fig. 3. Average normalized schedule lengths of solutions when the energy factor increases from 1.4 to 5.0.



	20	40	60	80	100	200	300	400
MSLECC	4.171	4.881	5.084	5.185	5.398	5.586	5.973	6.001
ESECC	2.763	2.954	3.013	3.049	3.124	3.109	3.22	3.218
ISAECC	2.654	2.831	2.876	2.894	2.973	2.945	3.031	3.038
SEDC	2.597	2.704	2.737	2.75	2.82	2.791	2.863	2.869

Fig. 4. Average normalized schedule lengths of approaches on workflows with different numbers of tasks when $\lambda = 1.2$.

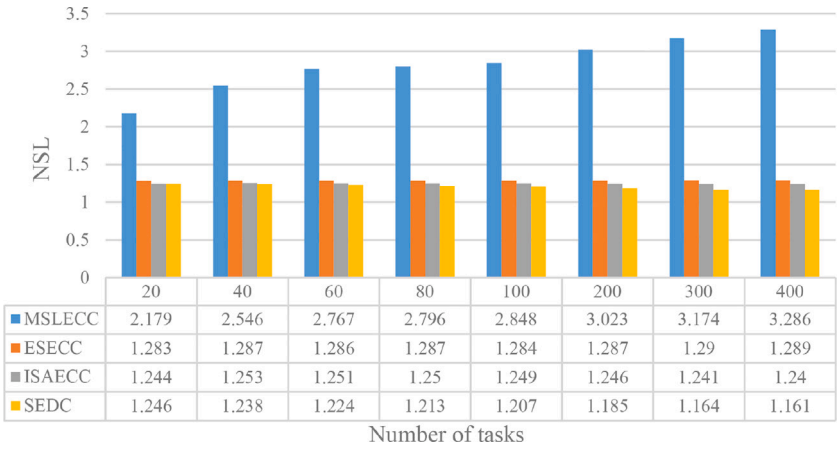
schedule length of SEDC is 0.982, illustrating the schedule lengths of our approach are shorter than those of HEFT. We can also find in Fig. 3, our approach SEDC still performs better than the others to reduce the schedule lengths of randomly-generated workflows. When $\lambda = 2.6$, the percentages of performance improvement achieved by our approach over MSLECC, ESECC, and ISAECC are 60.55%, 7.24%, and 3.53%, respectively. Based on the analysis above, we can achieve the conclusion that our approach has the ability of finding solutions with shorter schedule lengths for the randomly-generated workflows with various energy limitation settings.

Experiment 2. In addition to varying the energy limitations, we also test the performance of these approaches by changing the number of tasks in workflows. In these experiments, the energy factor λ is equal to 1.2, and the task number is increasing from 20 to 400. Fig. 4 displays the average normalized schedule lengths of approaches on workflows generated with different numbers of tasks. In Fig. 4 we can find that the average normalized schedule length of each approach grows with the task number. This is because that a larger task number results in a sharp rise in the amount of time and dependencies, and a larger schedule length is required to execute the tasks while satisfying the energy consumption constraint. However, when the value of λ is fixed, our approach would have the smallest normalized schedule length.

Fig. 5 displays the scheduling results of the four approaches for workflows with different numbers of tasks when the value of λ is set to 3.0. As we can see in this figure, the average normalized schedule lengths of approaches increase slightly along with the task number. Meanwhile, similar to results shown in Fig. 4, our approach achieves the best performance in terms of normalized schedule length, followed by ISAECC, ESECC and MSLECC. When the task number is 400, the average normalized schedule lengths of MSLECC, ESECC, and ISAECC are 2.83, 1.11 and 1.07 times more than that of our approach, respectively. This experiment demonstrates that our approach has better scalability and greater potential in solving the large-scale workflow scheduling problems. An interesting phenomenon shown in Fig. 5 is that, when the number of tasks is 20, the average normalized schedule length of our approach is slightly larger than that of ISAECC. This is because that all workflows were randomly generated with the pre-defined parameters, and the randomness of workflows had a great impact on the scheduling solutions, especially when a small number of tasks were used. From Fig. 5 we also can find that, along with the increase of the number of tasks, the advantage of our approach in achieving solutions with short schedule lengths is becoming more obvious.

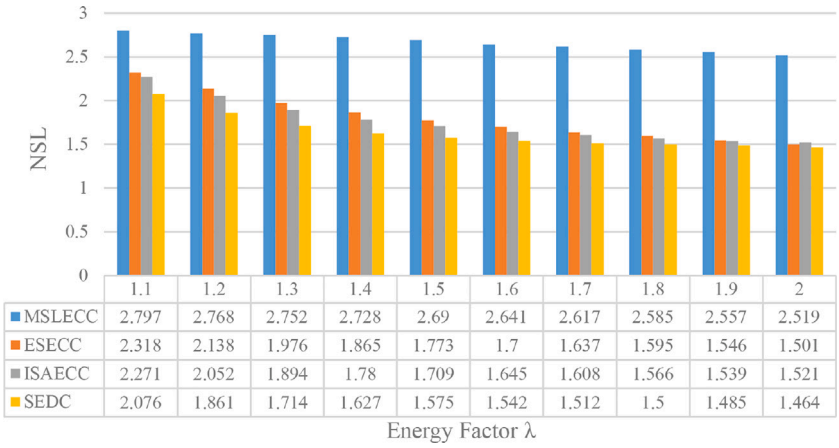
5.3. Experimental results on real-world workflows

Besides the randomly-generated workflows, we also test the performance of our proposed approach by using two types of real-world workflows named Montage [40] and CyberShake [41]. Montage workflow comes from the astronomy field, and is adopted to construct the sophisticated mosaic of the sky with all input images. CyberShake workflow is an earthquake science application to characterize the earthquake hazards and evaluate the order of severity of the earthquake hazards through analysing and interrogating huge datasets. All two types of workflows have been published in the Pegasus workflow project [42] and are frequently employed to verify the performances of newly proposed scheduling algorithms. In our experiments, the number



	20	40	60	80	100	200	300	400
MSLECC	2.179	2.546	2.767	2.796	2.848	3.023	3.174	3.286
ESECC	1.283	1.287	1.286	1.287	1.284	1.287	1.29	1.289
ISAECC	1.244	1.253	1.251	1.25	1.249	1.246	1.241	1.24
SEDC	1.246	1.238	1.224	1.213	1.207	1.185	1.164	1.161

Fig. 5. Average normalized schedule lengths of the approaches on randomly-generated workflows with different numbers of tasks when $\lambda = 3.0$.



	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2
MSLECC	2.797	2.768	2.752	2.728	2.69	2.641	2.617	2.585	2.557	2.519
ESECC	2.318	2.138	1.976	1.865	1.773	1.7	1.637	1.595	1.546	1.501
ISAECC	2.271	2.052	1.894	1.78	1.709	1.645	1.608	1.566	1.539	1.521
SEDC	2.076	1.861	1.714	1.627	1.575	1.542	1.512	1.5	1.485	1.464

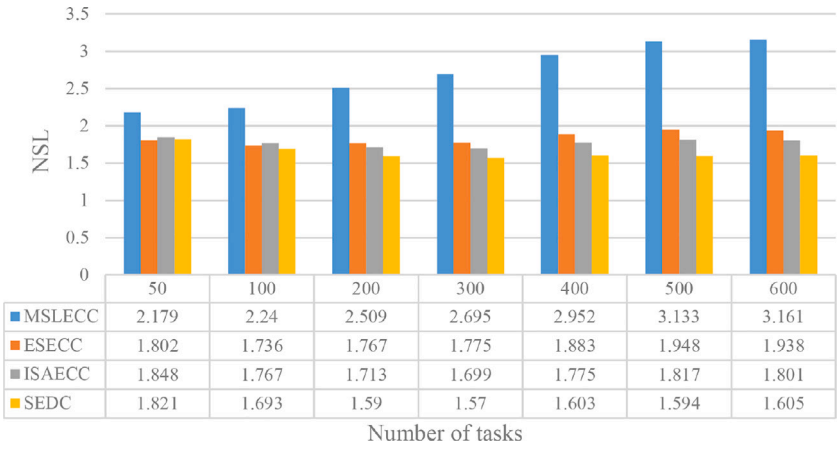
Fig. 6. Average normalized schedule lengths of the Montage workflows with different energy factor λ.

of tasks in a workflow and the number of processors are chosen from sets $\{50, 100, 200, 300, 400, 500, 600\}$ and $\{4, 8, 12\}$, respectively.

Experiment 3. We start by comparing the performance of the four approaches on the Montage workflows under different energy limitations. In this experiment, the number of tasks in a workflow is fixed at 300 and the energy factor λ is varied from 1.1 to 2.0. Fig. 6 presents the average normalized schedule lengths of different approaches. As it is expected, the average normalized schedule length of each approach decreases with the values of λ. The reason may be that a larger energy factor allows more possible processor and frequency allocations for every task and further reduces the schedule lengths of workflows by selecting the best allocations. Meanwhile, when the energy factor λ is fixed, our approach has an obvious advantage in producing solutions with shorter schedule lengths, and the average performance improvement percentages of our approach against MSLECC, ESECC, and ISAECC are 38.79%, 8.89%, and 6.76%, respectively.

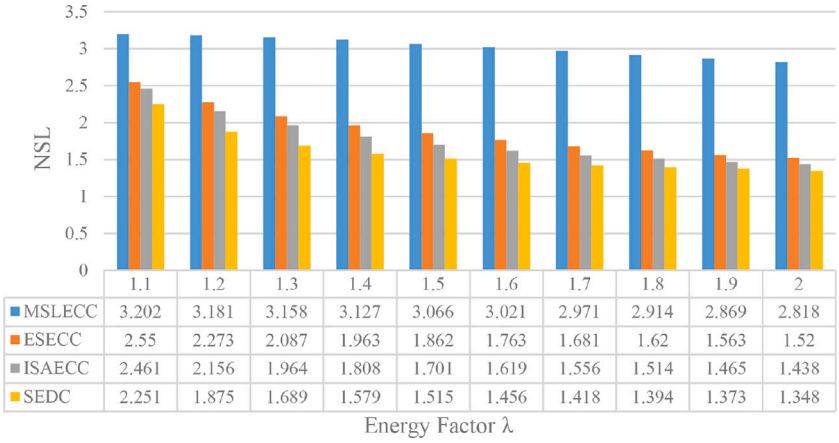
Experiment 4. In this experiment, the performances of different approaches are evaluated by varying the number of tasks in the Montage workflows. The energy factor λ is set to 1.5, i.e. $E_{limit}(G) = 1.5 \times E_{min}(G)$, and the task number in each Montage workflow gradually increases from 50 to 600. Fig. 7 shows the scheduling results of the four approaches. At the first glance of Fig. 7, we can see that our approach SEDC produces solutions with shorter schedule lengths than the others, and the gaps between the average normalized schedule lengths achieved by SEDC and by the other algorithms increase with the number of tasks. An interesting phenomenon revealed by Fig. 7 is that, the change trends of the normalized schedule lengths of different approaches are not completely similar. When the task number varies from 50 to 600, the normalized schedule length of the MSLECC algorithm grows gradually from 2.18 to 3.16. However, the other approaches have the same change trend that their normalized schedule lengths firstly decrease to the minimum values and then slightly increase. One of the most possible reasons is that, the increasing number of tasks results in a more complex workflow scheduling problem, and more reasonable energy pre-assignment strategies are required to allocate energy limitations to tasks to achieve solutions with shorter schedule lengths. As also can be seen in Fig. 7, when the number of tasks in the Montage workflows is 50, the average normalized schedule length of the ESECC algorithm is 1.802, which is 0.019 smaller than that of our approach SEDC. The reason is that, the ESECC algorithm adopts an average and fair energy consumption assignment for tasks to avoid a long executing time for low-priority tasks with relatively small energy limitations assigned by the unfair energy distribution strategies. Although the fair energy distribution strategy adopted by the ESECC algorithm loses its strength when the number of tasks is more than 50, it may be very valuable in guiding the performance enhancements of our approach.

Experiment 5. In this experiment, we try to compare the scheduling results of approaches on the CyberShake workflows by varying the energy factor λ. The task number is fixed at 300 and the value of the energy factor λ is gradually increasing. Since the curves representing the normalized schedule lengths of four approaches have the same change trend when the energy factor λ is larger than 2.0, we remove the experiment results when $\lambda > 2.0$, and show the average normalized schedule lengths of different approaches in more detail by varying the value of λ from 1.1 to 2.0. As can be seen in Fig. 8, similar



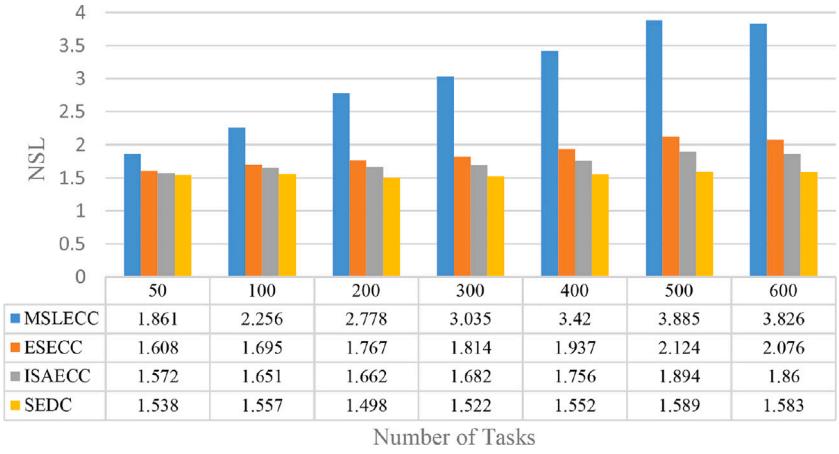
	50	100	200	300	400	500	600
MSLECC	2.179	2.24	2.509	2.695	2.952	3.133	3.161
ESECC	1.802	1.736	1.767	1.775	1.883	1.948	1.938
ISAECC	1.848	1.767	1.713	1.699	1.775	1.817	1.801
SEDC	1.821	1.693	1.59	1.57	1.603	1.594	1.605

Fig. 7. Average normalized schedule lengths of different approaches on the Montage workflows with different task numbers.



	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2
MSLECC	3.202	3.181	3.158	3.127	3.066	3.021	2.971	2.914	2.869	2.818
ESECC	2.55	2.273	2.087	1.963	1.862	1.763	1.681	1.62	1.563	1.52
ISAECC	2.461	2.156	1.964	1.808	1.701	1.619	1.556	1.514	1.465	1.438
SEDC	2.251	1.875	1.689	1.579	1.515	1.456	1.418	1.394	1.373	1.348

Fig. 8. Average normalized schedule lengths of approaches on the Montage workflows when the value of the energy factor λ is varying from 1.1 to 2.0.



	50	100	200	300	400	500	600
MSLECC	1.861	2.256	2.778	3.035	3.42	3.885	3.826
ESECC	1.608	1.695	1.767	1.814	1.937	2.124	2.076
ISAECC	1.572	1.651	1.662	1.682	1.756	1.894	1.86
SEDC	1.538	1.557	1.498	1.522	1.552	1.589	1.583

Fig. 9. Average normalized schedule lengths of approaches on the CyberShake workflows with different numbers of tasks.

to the previous experimental results, the performance of MSLECC is much weaker than the other approaches. Even though the energy factor doubles, the normalized schedule lengths of MSLECC are just reduced by 12.0%. As expected, our approach SEDC performs better than the others in terms of normalized schedule length. When $\lambda = 1.5$, the normalized schedule length of our approach SEDC is 1.515, which is 50.59%, 18.64%, and 10.93% smaller than those of MSLECC, ESECC, and ISAECC, respectively.

Experiment 6. Experiments are conducted to verify the performance of the approaches by varying the number of tasks in the CyberShake workflows. The energy factor λ is fixed at 1.5, and the number of tasks is growing from 50 to 600. The experiment results are displayed in Fig. 9. Similar to the above experiments, our approach produces a better solution with a smaller schedule length than others. Meanwhile, the advantage of our approach becomes more obvious along with the increasing of the task number. For example, when 50 tasks are tested, the normalized schedule lengths of MSLECC, ESECC, and ISAECC are

respectively 1.19, 1.05 and 1.02 times larger than that of our approach. However, these values grow to 2.42, 1.31 and 1.17 when the number of tasks is up to 600. Combining the experiment results on the Montage and CyberShake workflows, we can find that our approach has a higher effectiveness in reducing the schedule lengths for real-world workflows.

6. Conclusions and future work

In this paper, a three-phase approach is proposed to address the energy-constrained scheduling problem of workflows in heterogeneous multi-processor systems. The proposed approach takes into account both the data dependencies and energy limitations of tasks, and provides an optimal processor, frequency, and start time allocation for every task with a view to minimizing the schedule lengths of workflows while meeting the constraints. Experiments on both the real-world and randomly-generated workflows are conducted to verify the performances of the proposed approach, and demonstrate that our approach can consistently produce solutions with shorter schedule lengths. The main results of this work not only provide excellent scheduling strategies for workflow applications, but also help in guiding the development of large-scale heterogeneous multi-processor systems. In the future, we plan to enhance the problem-solving performances of the proposed approach, and to see whether the results obtained in the work adapt to energy-aware scheduling problems of workflows with constraints besides the data dependencies and energy limitations.

Declaration of competing interest

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

Acknowledgements

This paper is supported in part by the National Natural Science Foundation of China No. 62106202 and No. 62102316.

References

[1] J. Chen, C. Du, F. Xie, B. Lin, Scheduling non-preemptive tasks with strict periods in multi-core real-time systems, J. Syst. Archit. 90 (2018) 72–84.
[2] G. Xie, G. Zeng, X. Xiao, R. Li, K. Li, Energy-efficient scheduling algorithms for real-time parallel applications on heterogeneous distributed embedded systems, IEEE Trans. Parallel Distrib. Syst. 28 (12) (2017) 3426–3442.
[3] P. Han, C. Du, J. Chen, F. Ling, X. Du, Cost and makespan scheduling of workflows in clouds using list multiobjective optimization technique, J. Syst. Archit. 112 (2021) 101837.
[4] J. Zhou, L. Li, A. Vajdi, X. Zhou, Z. Wu, Temperature-constrained reliability optimization of industrial cyber-physical systems using machine learning and feedback control, IEEE Trans. Autom. Sci. Eng. 20 (1) (2023) 20–31.
[5] X. Liu, Z. Zhan, J.D. Deng, Y. Li, T. Gu, J. Zhang, An energy efficient ant colony system for virtual machine placement in cloud computing, IEEE Trans. Evol. Comput. 22 (1) (2018) 113–128.
[6] Y.C. Lee, A.Y. Zomaya, Energy conscious scheduling for distributed computing systems under different operating conditions, IEEE Trans. Parallel Distrib. Syst. 22 (8) (2011) 1374–1381.
[7] G. Xie, X. Xiao, H. Peng, R. Li, K. Li, A survey of low-energy parallel scheduling algorithms, IEEE Trans. Sustain. Comput. 7 (1) (2022) 27–46.
[8] G. Xie, Y. Chen, X. Xiao, C. Xu, R. Li, K. Li, Energy-efficient fault-tolerant scheduling of reliable parallel applications on heterogeneous distributed embedded systems, IEEE Trans. Sustain. Comput. 3 (3) (2018) 167–181.
[9] J. Zhou, J. Sun, M. Zhang, Y. Ma, Dependable scheduling for real-time workflows on cyber–physical cloud systems, IEEE Trans. Ind. Inform. 17 (11) (2021) 7820–7829.
[10] W.Y. Lee, Energy-efficient scheduling of periodic real-time tasks on lightly loaded multicore processors, IEEE Trans. Parallel Distrib. Syst. 23 (3) (2012) 530–537.
[11] J. Huang, H. Sun, F. Yang, S. Gao, R. Li, Energy optimization for deadline-constrained parallel applications on multi-ECU embedded systems, J. Syst. Archit. 132 (2022) 102739.
[12] J. Song, G. Xie, R. Li, X. Chen, An efficient scheduling algorithm for energy consumption constrained parallel applications on heterogeneous distributed systems, in: 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications, 2017, pp. 32–39.
[13] X. Xiao, G. Xie, R. Li, K. Li, Minimizing schedule length of energy consumption constrained parallel applications on heterogeneous distributed systems, in: 2016 IEEE Trustcom/BigDataSE/ISPA, 2016, pp. 1471–1476.
[14] A. Arunarani, D. Manjula, V. Sugumaran, Task scheduling techniques in cloud computing: A literature survey, Future Gener. Comput. Syst. 91 (2019) 407–415.
[15] J. Chen, C. Du, Y. Zhang, P. Han, W. Wei, A clustering-based coverage path planning method for autonomous heterogeneous UAVs, IEEE Trans. Intell. Transp. Syst. 23 (12) (2022) 25546–25556.
[16] J. Chen, Y. Zhang, L. Wu, T. You, X. Ning, An adaptive clustering-based algorithm for automatic path planning of heterogeneous UAVs, IEEE Trans. Intell. Transp. Syst. 23 (9) (2022) 16842–16853.
[17] J. Chen, F. Ling, Y. Zhang, T. You, Y. Liu, X. Du, Coverage path planning of heterogeneous unmanned aerial vehicles based on ant colony system, Swarm Evol. Comput. 69 (2022) 101005.
[18] J. Zhou, Y. Shen, L. Li, C. Zhuo, M. Chen, Swarm intelligence based task scheduling for enhancing security for IoT devices, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (2022) 1–14.
[19] K.K. Chakravarthi, L. Shyamala, TOPSIS inspired budget and deadline aware multi-workflow scheduling for cloud computing, J. Syst. Archit. 114 (2021) 101916.
[20] B.P. Rimal, M. Maier, Workflow scheduling in multi-tenant cloud computing environments, IEEE Trans. Parallel Distrib. Syst. 28 (1) (2017) 290–304.
[21] H. Topcuoglu, S. Hariri, M. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE Trans. Parallel Distrib. Syst. 13 (3) (2002) 260–274.
[22] H. Arabnejad, J.G. Barbosa, List scheduling algorithm for heterogeneous systems by an optimistic cost table, IEEE Trans. Parallel Distrib. Syst. 25 (3) (2014) 682–694.
[23] K. He, X. Meng, Z. Pan, L. Yuan, P. Zhou, A novel task-duplication based clustering algorithm for heterogeneous computing environments, IEEE Trans. Parallel Distrib. Syst. 30 (1) (2019) 2–14.
[24] J. Chen, Y. He, Y. Zhang, P. Han, C. Du, Energy-aware scheduling for dependent tasks in heterogeneous multiprocessor systems, J. Syst. Archit. 129 (2022) 102598.
[25] Q. Huang, S. Su, J. Li, P. Xu, K. Shuang, X. Huang, Enhanced energy-efficient scheduling for parallel applications in cloud, in: 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Ccgrid 2012, 2012, pp. 781–786.
[26] Y. Hu, C. Liu, K. Li, X. Chen, K. Li, Slack allocation algorithm for energy minimization in cluster systems, Future Gener. Comput. Syst. 74 (2017) 119–131.
[27] G. Xie, G. Zeng, R. Li, K. Li, Energy-aware processor merging algorithms for deadline constrained parallel applications in heterogeneous cloud computing, IEEE Trans. Sustain. Comput. 2 (2) (2017) 62–75.
[28] G. Xie, G. Zeng, J. Jiang, C. Fan, R. Li, K. Li, Energy management for multiple real-time workflows on cyber–physical cloud systems, Future Gener. Comput. Syst. 105 (2020) 916–931.
[29] K. Li, Optimal task execution speed setting and lower bound for delay and energy minimization, J. Parallel Distrib. Comput. 123 (2019) 13–25.
[30] G. Xie, J. Jiang, Y. Liu, R. Li, K. Li, Minimizing energy consumption of real-time parallel applications using downward and upward approaches on heterogeneous systems, IEEE Trans. Ind. Inform. 13 (3) (2017) 1068–1078.
[31] J. Huang, R. Li, J. An, H. Zeng, W. Chang, A DVFS-weakly dependent energy-efficient scheduling approach for deadline-constrained parallel applications on heterogeneous systems, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 40 (12) (2021) 2481–2494.
[32] G. Xie, G. Zeng, X. Xiao, R. Li, K. Li, Energy-efficient scheduling algorithms for real-time parallel applications on heterogeneous distributed embedded systems, IEEE Trans. Parallel Distrib. Syst. 28 (12) (2017) 3426–3442.
[33] P. Han, C. Du, J. Chen, X. Du, Minimizing monetary costs for deadline constrained workflows in cloud environments, IEEE Access 8 (2020) 25060–25074.
[34] J.J. Durillo, V. Nae, R. Prodan, Multi-objective energy-efficient workflow scheduling using list-based heuristics, Future Gener. Comput. Syst. 36 (2014) 221–236.
[35] L. Zhang, K. Li, C. Li, K. Li, Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems, Inform. Sci. 379 (2017) 241–256.
[36] Z. Quan, Z.-J. Wang, T. Ye, S. Guo, Task scheduling for energy consumption constrained parallel applications on heterogeneous computing systems, IEEE Trans. Parallel Distrib. Syst. 31 (5) (2020) 1165–1182.
[37] G. Xie, J. Huang, Y.L.R. Li, K. Li, System-level energy-aware design methodology towards end-to-end response time optimization, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (2019) 1–14.
[38] J. Peng, K. Li, J. Chen, K. Li, HEA-PAS: A hybrid energy allocation strategy for parallel applications scheduling on heterogeneous computing systems, J. Syst. Archit. 122 (2022) 102329.

[39] W. Zhu, W. Wu, X. Yang, G. Zeng, TSSA: Task structure-aware scheduling of energy-constrained parallel applications on heterogeneous distributed embedded platforms, J. Syst. Archit. 132 (2022) 102741.

[40] G.B. Berriman, E. Deelman, J.C. Good, J.C. Jacob, D.S. Katz, C. Kesselman, A.C. Laity, T.A. Prince, G. Singh, M.-H. Su, Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand, in: Optimizing Scientific Return for Astronomy Through Information Technologies, 5493, SPIE, 2004, pp. 221–232.

[41] P. Maechling, E. Deelman, L. Zhao, R. Graves, G. Mehta, N. Gupta, J. Mehringer, C. Kesselman, S. Callaghan, D. Okaya, H. Francoeur, V. Gupta, Y. Cui, K. Vahi, T. Jordan, E. Field, SCEC CyberShake workflows-automating probabilistic seismic hazard analysis calculations, in: I.J. Taylor, E. Deelman, D.B. Gannon, M. Shields (Eds.), Workflows for E-Science: Scientific Workflows for Grids, Springer London, 2007, pp. 143–163.

[42] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, K. Vahi, Characterizing and profiling scientific workflows, Future Gener. Comput. Syst. 29 (3) (2013) 682–692.