The Journal of Supercomputing (2020) 76:455–480
https://doi.org/10.1007/s11227-019-03033-y

An energy-aware scheduling algorithm for budget-constrained scientific workflows based on multi-objective reinforcement learning

Yao Qin[1] · Hua Wang[2] · Shanwen Yi[1] · Xiaole Li[3] · Linbo Zhai[4]

Published online: 23 October 2019

Abstract

Since scientific workflow scheduling becomes a major energy contributor in clouds, much attention has been paid to reduce the energy consumed by workflows. This paper considers a multi-objective workflow scheduling problem with the budget constraint. Most existing works of budget-constrained workflow scheduling cannot always satisfy the budget constraint and guarantee the feasibility of solutions. Instead, they discuss the success rate in the experiments. Only a few works can always figure out feasible solutions. These methods work, but they are too complicated. In workflow scheduling, it has been a trend to consider more than one objective. However, the weight selection is usually ignored in these works. The inappropriate weights will reduce the quality of solutions. In this paper, we propose an energy-aware multi-objective reinforcement learning (EnMORL) algorithm. We design a much simpler method to ensure the feasibility of solutions. This method is based on the remaining cheapest budget. Reinforcement learning based on the Chebyshev scalarization function is a new framework, which is effective in solving the weight selection problem. Therefore, we design EnMORL based on it. Our goal is to minimize the makespan and energy consumption of the workflow. Finally, we compare EnMORL with two state-of-the-art multi-objective meta-heuristics in the case of four different workflows. The results show that our proposed EnMORL outperforms these existing methods.

Keywords Scientific workflows · Cloud computing · Energy saving · Reinforcement learning · Multi-objective optimization · The budget constraint

✉ Hua Wang
wanghua@sdu.edu.cn

Extended author information available on the last page of the article

1 Introduction

Cloud computing has been an important computing paradigm, which can provide computing resources through the Internet [1]. For cloud consumers, cloud computing can provide an infinite amount of resources and has the advantages of reliability, scalability, and flexibility [2].

The increasing demand for cloud computing has made the energy problem be one of the major concerns in clouds [3]. It is reported that the energy consumption of cloud data centers accounted for 1% of global electricity usage in 2010, and this rate is expected to increase to 3–13% in 2030 [4]. High energy consumption of data centers also increases global greenhouse gas emissions (GHGE) [5]. Data centers have been the largest culprit in the information and communication technology (ICT), which grew from 33% in 2010 to 45% of global ICT emissions by 2020 [6]. Due to its advantages, cloud computing has been widely used to execute scientific workflow applications [7]. Over the last few years, scientific workflow scheduling has been one of the primary energy consumers in clouds [8]. As a result, we should consider energy saving when designing the workflow scheduling algorithm.

This paper considers a multi-objective workflow scheduling problem with the budget constraint. Budget-constrained workflow scheduling is one of the main research directions in the field of workflow scheduling. However, most existing works of budget-constrained workflow scheduling cannot always satisfy the budget constraint and guarantee the feasibility of solutions. Instead, they discuss the success rate of scheduling in the experiments [8–10]. Unlike these works, [11] and [12] propose methods to guarantee the feasibility. These two methods are based on a concept called the budget level. They convert the budget constraint of the whole workflow into the budget constraints of tasks. The existing methods can always figure out feasible solutions, but they are complicated. What's more, the budget constraints of tasks are just the sufficient conditions of the original budget constraint. In this paper, we propose a much simpler way to guarantee the feasibility of solutions. We directly use the original budget constraint instead of converting it. Our method is proposed based on a concept called the remaining cheapest budget (RCB) [9].

In workflow scheduling, it has been a trend to consider more than one objective [13]. However, the weight selection problem is usually ignored in these works. It is an acknowledged difficult problem to determine the appropriate weight for each objective in multi-objective optimization [14]. If the weights are inappropriate, it will reduce the quality of solutions. From the literature, we find that the combination of reinforcement learning (RL) and the Chebyshev scalarization function can efficiently solve the weight selection problem [15]. RL based on the Chebyshev scalarization function is a new algorithm framework, which provides a general solution to multi-objective optimization. As a result, we design the scheduling algorithm based on it.

In this paper, we propose an energy-aware multi-objective RL (EnMORL) algorithm. Our goal is to find a set of Pareto approximations, which can simultaneously minimize the makespan and energy consumption of the workflow under the specified budget. The energy model in this paper is based on the dynamic voltage/frequency scaling (DVFS) technique [16]. Like most list-based scheduling algorithms [17], EnMORL consists of a ranking phase and a mapping phase. In EnMORL, we first assign a priority to each task in the workflow and then select a virtual machine (VM) of the appropriate type for each task in the order of priority. Finally, we compare EnMORL with two state-of-the-art multi-objective meta-heuristics in the case of four workflows, which come from different scientific applications. The experimental results show that our proposed EnMORL outperforms these existing methods.

The main contributions of this paper can be summarized as follows:

1. RL is capable of solving difficult multi-step decision-making problems [18]. It is very suitable for solving the workflow scheduling problem in this paper.
2. We propose a new method based on RCB to satisfy the budget constraint and guarantee the feasibility of solutions. This method is much simpler than those in [11] and [12].
3. We propose the EnMORL algorithm based on the Chebyshev scalarization function, which is efficient in solving the weight selection problem.
4. The experimental results demonstrate that EnMORL can compete efficiently with two state-of-the-art multi-objective meta-heuristics in terms of makespan, energy consumption, and the hypervolume indicator [19].

The remainder of this paper is organized as follows: We introduce related works from two aspects in Sect. 2. We briefly introduce the multi-objective optimization in Sect. 3. We give the mathematical formulation of the scheduling problem in Sect. 4. We present our proposed EnMORL algorithm in Sect. 5. We evaluate the performance in Sect. 6. Finally, we conclude this paper in Sect. 7.

2 Related work

In this section, we first introduce some representative works of budget-constrained workflow scheduling, and then the related works of energy-aware workflow scheduling.

The quality of service (QoS) requirements, such as the makespan and cost, are the traditional optimization objectives in workflow scheduling. In this area, budget-constrained workflow scheduling is one of the major research hot spots.

Most existing works of budget-constrained workflow scheduling do not guarantee the feasibility of the solution. Instead, they discuss the success rate or the failure rate of the scheduling in the experiments. Arabnejad et al. [9] propose a heterogeneous budget-constrained scheduling (HBCS) algorithm. The authors aim to minimize the makespan under the specified budget. In their algorithm, they use the combination of the time and cost factors to select the processor for the current task. Garg et al. [10] propose a ϵ-fuzzy dominance sort-based discrete particle swarm optimization (PSO) algorithm called ε-FDPSO to solve the workflow scheduling in grid computing. Their goal is to optimize makespan, cost, and reliability objectives simultaneously under deadline and budget constraints. The authors use the ε-fuzzy dominance-based sorting procedure to select the particle with a lower rank.

Unlike the above works, Wu et al. [11] aim to find feasible solutions in budget-constrained workflow scheduling. Their proposed critical-greedy (CG) algorithm preassigns tasks with the budget level to meet the budget constraint. However, this algorithm only applies to homogeneous data centers. In [12], Chen et al. extend the concept of the budget level to heterogeneous environments. They propose an efficient algorithm called minimizing the schedule length using the budget level (MSLBL). The authors aim to minimize the makespan of the workflow while satisfying the budget constraint. To meet the budget constraint, the authors convert the budget constraint of the whole workflow into those of tasks. When using the budgets of tasks, MSLBL can at least assign each task assign to the processor with the minimum cost. The methods in [11] and [12] are complex. The budget constraints of tasks are just the sufficient conditions of the original budget constraint. In this paper, we will use a much simpler way to guarantee the feasibility of solutions. We design our method based on the original budget constraint instead of converting it.

An increasing number of scientific workflows are migrating into clouds. As a result, workflow scheduling has been one of the primary energy consumers in data centers. In recent years, much attention has been paid to reduce the energy consumption of workflow scheduling.

In, Li et al. [20] propose a cost and energy-aware scheduling (CEAS) algorithm to solve workflow scheduling in clouds. CEAS is composed of five sub-algorithms. Their goal is to minimize the execution cost and energy consumption under the deadline constraint. To meet the deadline constraint, they propose the concept of sub-makespan. In, Qureshi et al. [21] introduce the concept of power-aware application profiles (APs). The authors use APs to compute the execution cost of a workflow according to the energy consumption requirements. Their scheduling algorithm considers CPU, memory, I/O, and energy consumption requirements.

To reduce the makespan and energy consumption, Sofia et al. [13] propose a multi-objective algorithm based on non-dominated sorting genetic algorithm (NSGA-II) [22]. The authors also propose two single-objective genetic algorithms (GA) to minimize energy consumption and makespan individually. Verma et al. [8] propose a hybrid PSO (HPSO) algorithm to handle the workflow scheduling problem in clouds. HPSO is the combination of heterogeneous earliest finish time (HEFT) [23] and multi-objective PSO (MOPSO) [24]. Their objectives include time minimization, cost minimization, and energy saving. The authors consider both the bi-objective workflow scheduling problem as well as the tri-objective workflow scheduling problem in their work.

We notice that the energy models in the above works are all based on the DVFS technique. This technique has been widely used in energy-aware workflow scheduling problems.

3 Multi-objective optimization

A maximization multi-objective problem with m decision variables and n objectives can be defined as [25]:

$$\begin{aligned} &\max \vec{f}(\vec{x}) = [f_1(x_1, x_2, \dots, x_m), \dots, f_n(x_1, x_2, \dots, x_m)] \\ &s.t. \\ &\quad \vec{x} = (x_1, x_2, \dots, x_m) \in X \end{aligned} \tag{1}$$

$$\vec{f} = (f_1, f_2, \dots, f_n) \in Y, \tag{2}$$

where $\vec{x}$ is the decision vector, X is the search space, $\vec{f}$ is the objective vector, and Y is the objective space.

A solution $\vec{x_1}$ is said to dominate another solution $\vec{x_2}$, $(\vec{x_1} \succ \vec{x_2})$, if and only if both of the following two conditions are true [26]:

1. The solution $\vec{x_1}$ is not worse than $\vec{x_2}$ in any objective.
2. The solution $\vec{x_1}$ is strictly better than $\vec{x_2}$ in at least one objective.

The points that are not dominated by any other points are called the non-dominated points, and the corresponding decision variables are called non-dominated solutions or Pareto optimal solutions. The Pareto optimal front is defined as the set of all non-dominated points. Similarly, the Pareto optimal set is the set of all non-dominated solutions.

The concept of Pareto optimal set is used to optimize problems with multiple objectives. The multi-objective algorithm aims to find a Pareto approximate set that can approximate the true Pareto optimal set. In this paper, the set of non-dominated solutions will be updated after the construction of each feasible solution.

4 System models and mathematical formulation

Our algorithm aims to schedule a budget-constrained workflow on a set of available heterogeneous resources, which are in the form of VMs in clouds. This section consists of five parts, namely the workflow model, cloud data center model, makespan model, energy model, and mathematical formulation.

4.1 Workflow model

In general, a workflow application is represented by a directed acyclic graph (DAG). A DAG can be modeled as a tuple $G(T, D)$, where $T = \{t_1, \dots, t_m\}$ is the set of m tasks in the workflow, and D is the set of directed edges between tasks. Directed edges represent the task-dependency constraints. If there exists an edge

$(t_i, t_j) \in D$, then t_i is called an immediate predecessor of t_j, and t_j is called an immediate successor of t_i. We use pre(t_i) to denote the set of all immediate predecessors of task t_i, and succ(t_i) to denote the set of all immediate successors of task t_i. A task can only start after all of its predecessors complete their execution. The task with no predecessors is called the entry task, which can be denoted as t_{entry}. The task with no successors is called the exit task, which can be denoted as t_{exit}. The size of data processed by task t_i is denoted by in_i, which is expressed in Millions of Instructions (MI). The size of data transferred between task t_i and t_j is denoted by $\text{tran}_{i,j}$, which is expressed in MB.

4.2 Cloud data center model

In the cloud data center, the computing resources are usually provided in the form of homogeneous VMs. We assume that the cloud data center offers n types of VMs, which have different prices with different performance. We use $V = \{v_1, \dots, v_n\}$ to denote the set of VM types. For simplicity, we assume that each VM type v_j can provide an infinite number of VMs to execute all the tasks of a workflow.

In this paper, the cloud service is charged based on the number of time intervals that the VM has been used. The consumers have to pay for the whole time interval even if it is not completely used. We use c_j to denote the unit price or cost of using the VM type v_j for each time interval, and p_j to denote the processing speed of type v_j for each time interval. The processing speed p_j is expressed in million instructions per second (MIPS).

Let $x_{i,j}$ be a binary variable, which represents whether task t_i is executed on the VM type v_j ($x_{i,j} = 1$) or not ($x_{i,j} = 0$). Each task $t_i \in T$ is executed exactly once, so there exists the equation $\sum_{j=1}^{n} x_{i,j} = 1$, where n is the number of VM types in the data center.

4.3 Makespan model

Without loss of generality, we assume that each task can be executed on any VM in the data center. If task t_i is assigned to VM type v_j, the execution time $\text{ET}(t_i, v_j)$ of task t_i is expressed as follows:

$$\text{ET}(t_i, v_j) = \frac{\text{in}_i}{p_j} \tag{3}$$

When considering the decision variable $x_{i,j}$, the actual execution time $\text{ET}(t_i)$ of task t_i can be calculated as:

$$\text{ET}(t_i) = \sum_{j=1}^{n} \text{ET}(t_i, v_j) \cdot x_{i,j} = \sum_{j=1}^{n} \frac{\text{in}_i}{p_j} \cdot x_{i,j} \tag{4}$$

We have assumed that all the VMs are in the same data center, which means the storage cost and the transmission cost can be ignored. Let $\text{EC}(t_i)$ denote the cost of executing task t_i, which can be calculated as follows:

$$\mathrm{EC}(t_i) = \mathrm{ET}(t_i) \cdot c_j = \sum_{j=1}^{n} \frac{\mathrm{in}_i}{p_j} c_j \cdot x_{i,j} \tag{5}$$

We have assumed that each VM type v_j can provide a sufficient number of VMs to execute all the tasks of a workflow. We assume that all the VMs in the cloud data center have the same communication bandwidth, the value of which is set to a constant b. Then, the actual finish time $\mathrm{FT}(t_i)$ of task t_i can be calculated as:

$$\mathrm{FT}(t_i) = \begin{cases} \mathrm{ET}(t_i), & \mathrm{pre}(t_i) = \phi \\ \max\limits_{t_p \in pre(t_i)} \left\{ \mathrm{FT}(t_p) + \frac{\mathrm{tran}_{p,i}}{b} \right\} + \mathrm{ET}(t_i), & \mathrm{pre}(t_i) \neq \phi \end{cases} \tag{6}$$

The makespan *MS* of the workflow is defined as the maximum actual finish time of the exit tasks:

$$\mathrm{MS} = \max\{\mathrm{FT}(t_{\mathrm{exit}})\} \tag{7}$$

4.4 Energy model

The energy consumption in this paper is calculated based on the DVFS technique [16]. The dynamic power consumption is the most important factor, so we ignore the static energy consumption [27]. The dynamic power capability P_d is expressed as follows:

$$P_\mathrm{d} = BCV^2 f, \tag{8}$$

where B is the number of switches per clock cycle, C is the total capacitance load, V is the supply voltage, and f is the frequency. From Eq. (8), we know that the supply voltage is the dominant factor. The total energy consumed by the whole workflow can be calculated as follows [8]:

$$E = \sum_{i=1}^{m} BC(V_i)^2 f \cdot \mathrm{ET}(t_i) = \sum_{i=1}^{m} \alpha(V_i)^2 \cdot \mathrm{ET}(t_i), \tag{9}$$

where V_i denotes the supply voltage of the VM on which task t_i is executed. It is often assumed that VMs are operating at the maximum voltage level when they are busy [8]. The VM type with higher performance has a higher supply voltage. Without loss of generality, we assume that a VM with higher performance consumes more energy.

4.5 Mathematical formulation

The two objectives in this paper are to minimize makespan and energy consumption. Besides, the scheduling problem subjects to a budget constraint. Based on Eqs. (7) and (9), the scheduling problem is formulated as follows:

$$\begin{aligned} &\text{Goals:} \\ &\min\ MS \text{ and } \min\ E \\ &\text{Constraints:} \\ &\sum_{i=1}^{m} \mathrm{EC}(t_i) \le \mathrm{BG} \end{aligned} \tag{10}$$

$$\sum_{j=1}^{n} x_{i,j} = 1, \quad \forall t_i \in T \tag{11}$$

$$x_{i,j} \in \{0, 1\}, \quad \forall t_i \in T,\ v_j \in V, \tag{12}$$

where m and n is the number of all tasks and VM types respectively, and BG denotes the budget.

The constraint (10) ensures that the cost of the workflow must be smaller than the budget. The constraint (11) ensures that each task is executed only once. The constraint (12) defines the decision variables. What's more, it is quite remarkable that the task-dependency constraint is reflected by Eq. (6), and $x_{i,j}$ can be found in all objectives and constraints.

A consumer can determine the budget within the range provided by the following equation [9]:

$$\mathrm{BG} = C_c + k_{\mathrm{BG}} \cdot (C_h - C_c), \tag{13}$$

where C_c denotes the cost of the cheapest scheduling, C_h denotes the cost of the most expensive scheduling, and k_{BG} is the budget factor, which is determined by the consumer. When $k_{\mathrm{BG}} < 0$, there exists the inequation $\mathrm{BG} < C_c$. It means that the consumer cannot afford to pay the cost of the workflow. When $k_{\mathrm{BG}} \ge 1$, $\mathrm{BG} \ge C_h$. There will be no budget constraint anymore. Only when $0 \le k_{\mathrm{BG}} < 1$, the scheduling problem is budget-constrained. If $k_{\mathrm{BG}} = 0$, the consumer can only choose the cheapest scheduling. In the experiments, we set k_{BG} in the range of 0.1–0.9. The step length is set to 0.1.

Without loss of generality, we assume that a VM with higher computing performance charges more in the data center. Given $t_i \in T$ and $v_j, v_k \in V$, if processing speeds satisfy $p_j < p_k$, then there exists the inequation $\mathrm{ET}(t_i, v_j) \cdot c_j < \mathrm{ET}(t_i, v_k) \cdot c_k$. After reducing the above inequation, we obtain:

$$\frac{c_j}{p_j} < \frac{c_k}{p_k} \tag{14}$$

The cheapest cost C_c is obtained by selecting the VM with the lowest performance to execute each of the tasks. And the highest cost C_h is obtained by selecting the VM with the highest performance to execute each of the tasks.

In conclusion, the problem in this paper is about the multi-objective scientific workflow scheduling, in which the makespan and energy consumption are optimized under a budget constraint. Workflow scheduling is NP-hard [28]. In the next section, we will apply a multi-objective RL algorithm based on the Chebyshev scalarization method to the scientific workflow scheduling problems.

5 The proposed EnMORL algorithm

This section consists of four parts, namely reinforcement learning, the Chebyshev scalarization function, the learning agent of EnMORL, and the description of EnMORL.

5.1 Reinforcement learning

Single-objective RL is usually described in the form of a Markov decision process (MDP) [29]. We use $S = \{s_1, \dots, s_M\}$ to denote the state space, and $A = \{a_1, \dots, a_N\}$ to denote the set of N actions, which are available in the current state. For each combination of current state s, an available action $a \in A$ and a next state s', there always exists a transition probability $p(s'|s, a)$ and a reward signal $r(s, a)$.

RL aims to find an optimal policy π, which maximizes the expected discounted reward $\mathbb{E}[\sum_{i=0}^{\infty} \gamma^i r_i]$, where $\gamma \in (0, 1]$ is the discount factor, and r_i is the reward signal at time step i. This goal can be expressed using Q values which record the expected discounted reward for each state-action pair. We use $Q^{\pi}(s, a)$ to denote the Q value of taking action $a \in A$ in state s under policy π, and Q^* to denote the optimal Q value.

In a famous RL algorithm called Q-learning [30], $\widehat{Q}(s, a)$ was proposed to iteratively approximate Q^*. The $\widehat{Q}(s, a)$ value is updated accordingly to following update rule:

$$\widehat{Q}(s, a) = (1 - \alpha_i)\widehat{Q}(s, a) + \alpha_i \Big(r + \gamma \max_{a'} \widehat{Q}(s', a') \Big), \tag{15}$$

where $0 < \alpha_i \leq 1$ denotes the learning rate at time step i, and r is the reward of taking action a in state s. The $\widehat{Q}(s, a)$ value will converge to $Q^*(s, a)$ if visiting all state-action pairs infinitely using an appropriate learning rate [31].

In a multi-objective RL algorithm, the MDP is usually extended to a multi-objective MDP, and the reward signal is extended to a reward vector [32]. The vector of reward signals in a multi-objective RL algorithm can be denoted as $\vec{r}(s, a) = \{r_1(s, a), \dots, r_k(s, a)\}$, where k is the number of objectives.

5.2 The Chebyshev scalarization function

The linear scalarization function is widely used in multi-objective algorithms, including RL, PSO, and Ant Colony Optimization (ACO) [15]. However, it can only discover solutions in convex regions of the Pareto front [33]. Besides, it is an acknowledged difficult problem to determine the appropriate weights when using the linear scalarization function [14]. The inappropriate weights will affect the quality of solutions. Therefore, we use the Chebyshev scalarization method in our proposed algorithm.

The Chebyshev scalarization function is categorized as a nonlinear scalarization method, which overcomes the shortcomings of the linear scalarization function. It can discover Pareto optimal solutions regardless of the front shape. Besides, the Chebyshev scalarization function is not particularly dependent on the actual weights used. We can select a relative optimal weight tuple from several randomly generated ones in preliminary experiments [15].

The Chebyshev scalarization function evaluates actions using the metric L_p. This metric is defined as the distance between a point in the objective space and a utopian point z^*, which records the best value for each objective [34]. During the learning process, z^* is constantly updated. The value of z_i^* for objective i can be calculated as follows:

$$z_i^* = f_i^{\mathrm{best}}(x) + \mu, \tag{16}$$

where f_i denotes the function of objective i, $f_i^{\mathrm{best}}(x)$ denotes the best value so far for objective i of the solution x, and μ denotes a small constant. We set μ to 2.0. We use $L_p(x)$ to denote the distance between the corresponding point of solution x and the utopian point z^* and it can be expressed as follows:

$$L_p(x) = \left(\sum_{i=1}^{k} w_i |f_i(x) - z_i^*|^p \right)^{1/p}, \tag{17}$$

where $p \geq 1$. We use k to denote the number of objectives, and $0 \leq w_i \leq 1$ to denote the weight of each objective i, which satisfies the equation $\sum_{i=1}^{k} w_i = 1$. When $p = \infty$, the metric will be called the Chebyshev metric. It can be expressed as follows:

$$L_\infty(x) = \max_{i=1 \cdots k} w_i |f_i(x) - z_i^*| \tag{18}$$

In multi-objective RL, $f_i(x)$ is replaced by $\hat{Q}_i(s,a)$, which denotes the $\hat{Q}$ value for each objective i. Then, the $\widehat{SQ}$ value or the scalarized $\hat{Q}$ value of a state-action pair (s, a) can be calculated as follows:

$$\widehat{SQ}(s,a) = \max_{i=1 \cdots k} w_i |\hat{Q}_i(s,a) - z_i^*| \tag{19}$$

The above equation describes the scalarization of $\hat{Q}$ values when using the Chebyshev scalarization function in RL. The action corresponding to the smallest $\widehat{SQ}$ value will be chosen as the greedy action in state s. In other words, the greedy strategy can be expressed as follows:

$$\mathrm{greedy}_a(s) = \arg\min_{a \in A} \widehat{SQ}_i(s,a), \tag{20}$$

where A is the available action set in state s. Algorithm 1 shows the scalarized ϵ -greedy strategy for a multi-objective RL algorithm, where $\vec{w}$ is the vector of weights for multiple objectives.

Algorithm 1 scal-ϵ-greedy()

```
1: Initialize the list of $SQ$-values $List_{sq}$ to $\phi$
2: Figure out the available action set $A$ in current state $s$ %According to Eq. (23)
3: for each action $a \in A$ do
4:     $\vec{o} = \{\widehat{Q}_1(s,a), \cdots, \widehat{Q}_k(s,a)\}$
5:     $\widehat{SQ}(s,a) = scalarize(\vec{o}, \vec{w})$ %According to Eq. (19)
6:     $List_{sq} = List_{sq} \bigcup \widehat{SQ}(s,a)$
7: end for
8: return $\epsilon$-greedy($List_{sq}$)
```

5.3 The learning agent of EnMORL

In this part, we introduce the learning agent of EnMORL in terms of the state space, the available action set, and the reward vector.

5.3.1 The state space

The state space in the EnMORL algorithm describes the usage of all VM types at each time step, which is denoted by $S = \{s_1, \dots, s_q\}$. Apart from the final state, each state corresponds to a time step.

The state $s_i \in S$ denotes the usage of all VM types at time step i. The above state can be expressed as $s_i = \{s_i^1, \dots, s_i^n\}$, where n is the number of all VM types, and $s_i^j \in s_i$ is the set of all tasks which have been assigned to VM type $v_j \in V$ at time step i. In our implementation, the usage of all VM types and the task assignments in the current state s_i is figured out using the current values of decision variables. It should be noted that the constraint (11) can only be satisfied after the construction of a solution.

At each time step, we only schedule one task based on the available action set. The maximum time step is determined by the number of tasks in the workflow. There exists the equation $q = m + 1$, where m is the number of tasks. In particular, s_{m+1} is the final state, in which all tasks have been assigned to a VM.

5.3.2 The available action set

The set of available actions at time step i is denoted by A_i. Tasks are sorted and scheduled in descending order of their priorities, which are calculated according to Eq. (26). We denote the task to be scheduled at time step i as t_i.

As described above, most existing works of budget-constrained workflow scheduling cannot ensure the feasibility of solutions. Instead, they discuss the success rate or the failure rate of the scheduling in the experiments. Only a few works, like [11] and [12], can guarantee the feasibility. However, their methods are too complex.

To guarantee the feasibility of solutions, we use a concept called the RCB in our algorithm. This concept is initially proposed in [9] to select computing resources for tasks. RCB is defined as the remaining cheapest cost for unscheduled tasks excluding the current task t_i, which updates using the following equation:

$$\mathrm{RCB} = \mathrm{RCB} - \mathrm{EC}_c(t_i), \tag{21}$$

where $\mathrm{EC}_c(t_i)$ is the least cost of executing task t_i. In this paper, we define a concept called the possible remaining budget. If task t_i is assigned to VM type v_j at time step i, then the possible remaining budget $\mathrm{PRB}(t_i, v_j)$ can be calculated as:

$$\mathrm{PRB}(t_i, v_j) = \begin{cases} \mathrm{BG} - \mathrm{ET}(t_i, v_j) \cdot c_j, & i = 1 \\ \mathrm{BG} - \sum_{u=1}^{i-1} \mathrm{EC}(t_u) - \mathrm{ET}(t_i, v_j) \cdot c_j, & i > 1, \end{cases} \tag{22}$$

where $\sum_{u=1}^{i-1} \mathrm{EC}(t_u)$ denotes the total cost of tasks that have been assigned before, and $\mathrm{EC}(t_u)$ is calculated according to Eq. (5). In EnMORL, taking action means selecting an appropriate VM type to execute the current task. Then, the available action set A_i at time step i can be represented as follows:

$$A_i = \{v_j \in V | \mathrm{PRB}(t_i, v_j) \geq \mathrm{RCB}\} \tag{23}$$

Our core idea is to ensure that the actual remaining budget (ARB) is never less than the RCB during the mapping phase. In other words, always keep the remaining budget enough. This method can even be used to enable the feasibility of the solution in the worst case when the budget factor $k_{\mathrm{BG}} = 0$. At each time step i, only when task i is assigned to the cheapest VM type, the inequation $\mathrm{PRB}(t_i, v_j) \geq RCB$ holds. We do not consider $k_{\mathrm{BG}} = 0$ in the experiments, because there is only one feasible solution in that case.

5.3.3 The reward vector

In multi-objective RL algorithms, we use a vector of reward signals instead of a single reward signal. Let $\vec{r} = [r_1, r_2]^T$ denote the reward vector, where r_1 is the reward of the first objective, and r_2 is the reward of the second objective.

If the action selected at time step i is $a_i \in A_i$, the reward of the first objective can be calculated as follows:

$$r_1(s_i, a_i) = \frac{\mathrm{FT}_{\mathrm{best}}(t_i)}{\mathrm{FT}(t_i, a_i)}, \tag{24}$$

where $\mathrm{FT}_{\mathrm{best}}(t_i)$ denotes the finish time of task t_i when it is executed on a VM with the highest performance, and $\mathrm{FT}(t_i, a_i)$ denotes the finish time of task t_i when it is executed on VM type a_i. Similarly, the reward of the second objective can be calculated as follows:

$$r_2(s_i, a_i) = \frac{E_{\text{best}}(t_i)}{E(t_i, a_i)}, \tag{25}$$

where $E_{\text{best}}(t_i)$ denotes the energy consumption of task t_i when it is executed on a VM with the lowest power capability, and $E(t_i, a_i)$ denotes the energy consumption of task t_i when it is executed on VM type a_i.

5.4 The description of EnMORL

Like other list-based algorithms, EnMORL includes a ranking phase and a mapping phase.

In the ranking phase, each task t_i in the workflow is assigned a priority [23], which is denoted by $\text{pr}(t_i)$. And the value of $\text{pr}(t_i)$ can be calculated as follows:

$$\text{pr}(t_i) = \overline{ET(t_i)} + \max_{t_s \in \text{succ}(t_i)} \left\{ \frac{\text{tran}_{i,s}}{b} + \text{pr}(t_s) \right\}, \tag{26}$$

where $\overline{\text{ET}(t_i)}$ is the average execution time of task t_i over all types of VMs. The priority $\text{pr}(t_i)$ represents the length of the longest path from task t_i to the exit node. In the ranking phase, we do not know where the tasks will run. As a result, the average value is considered in the Eq. (26). In the mapping phase, tasks are sorted in descending order of their priorities and assigned according to Algorithm 2.

Algorithm 2 shows the pseudocode of the EnMORL algorithm. This algorithm can also be described in the following five steps.

Step 1: The ranking phase. In this step, we assign a priority to each task $t_i \in T$ and then sort all tasks in descending order of their priorities.

Step 2: The initialization phase. In this step, we initialize the Pareto approximate set S_{p} and the $\widehat{Q}$ value for each objective.

Step 3: The mapping phase. In this step, the learning agent constructs a feasible solution or a trajectory according to the scalarized ϵ-greedy strategy shown in Algorithm 1. After taking action, the $\widehat{Q}$ values will be updated according to Eq. (15).

Step 4: Update of the Pareto approximate set. After a solution has been constructed, the Pareto approximate set S_{p} will be updated. If the solution in the current iteration is not dominated by any other solutions in S_{p}, it will be added to the set. All solutions dominated by the newly added one will be eliminated from the set.

Step 5: The termination check. If the maximum number of iterations is reached, the algorithm terminates and returns the Pareto set S_{p}. Otherwise, go to Step 2.

Algorithm 2 Pseudocode of EnMORL

```
1: for $i = 1$ to $m$ (number of tasks) do
2:     Calculate the priority of task $t_i$ according to Eq. (26)
3: end for
4: Sort all tasks of $T$ in descending order of their priorities. And the list of sorted tasks is
   denoted by $T_s$.
5: Initialize the Pareto approximate set $S_p$ to $\phi$
6: Initialize $\widehat{Q}$-value for each objective $k$ to 0
7: repeat
8:     Initialize the current state $s$
9:     Initialize the current task $t$ to the first task of $T_s$
10:    Initialize RCB to $C_c$
11:    Initialize the available action set $A$ to $V$
12:    for $i = 1$ to $m$ (number of tasks) do
13:        $a$ =scal-$\epsilon$-greedy($s$) %According to Algorithm 1
14:        Assign the task $t$ to the VM type $a$
15:        Figure out the next state $s'$
16:        Introduce the next task $t'$ from $T_s$
17:        Update RCB according to Eq. (21)
18:        Update $A$ according to Eq. (23)
19:        $a'$ =greedy($s'$) %According to Eq. (20)
20:        for $k = 1$ to 2 (number of objectives) do
21:            if $k = 1$ then
22:                Calculate the reward $r_k$ according to Eq. (24)
23:            else
24:                Calculate the reward $r_k$ according to Eq. (25)
25:            end if
26:            Update $\widehat{Q}_k(s, a)$ according to Eq. (15)
27:        end for
28:        $s = s'$
29:        $t = t'$
30:    end for
31:    If a solution is not dominated by any other solutions in $S_p$, it will be added to $S_p$.
       All solutions dominated by the newly added one are eliminated from $S_p$.
32: until the maximal number of iterations is reached
33: return $S_p$;
```

More specifically, Step 1 corresponds to lines 1–4 in Algorithm 2, Step 2 corresponds to lines 5, 6, Step 3 corresponds to lines 8–29, Step 4 corresponds to line 31, and Step 5 corresponds to line 32.

6 Performance evaluation

In the section of experiments, we present performance comparisons of our EnMORL algorithm with some state-of-the-art scheduling algorithms in terms of makespan, energy consumption, and the quality of the Pareto approximate set. This section consists of five parts, namely workflow structure, simulation setup, weight selection, solution selection, and experimental results.

6.1 Workflow structure

In simulation experiments, we use four different synthetic workflows, which are described in [35], to evaluate the performance of our proposed algorithm. These synthetic workflows are based on realistic workflow structures from different scientific areas, which are as follows:

1. Montage: Astronomy
2. EpiGenomics: Biology
3. CyberShake: Earthquake
4. SIPHT: Biology

The characterization of the above workflows, including their structure, data, and computational requirements, is described in detail in [35]. The corresponding workflow structures are shown in Fig. 1.

6.2 Simulation setup

In simulations, all algorithms are implemented in the Java language and run on a PC with an Intel Core i5-9400F CPU at 2.90 GHz and 8 GB RAM.

In this paper, we use CloudSim [36], which is a widely used framework, to simulate the environment of a cloud data center. Cloudsim can offer a repeatable and controllable experimental environment, which enables the users to pay no attention to the hardware details.

The cloud data center provides computing resources in the form of 10 different VM types. In this paper, each VM type is dynamic voltage scaling (DVS) enabled, and each of them is generated using a method similar to those in [27] and [8]. For each VM type v_j, a set of voltage supply levels (VSLs) is random and uniformly distributed among three different sets of VSLs, which are shown in Table 1. The processing speed of each VM type is chosen at random in the range of 1000–5000 MIPS. The VM type with the highest performance is five times faster than the type of lowest performance. In this paper, a VM is operating at the maximum voltage level when there exists a task executing on it. When a VM is in the idle state, the supply voltage drops to the minimum level [8]. The bandwidth of each

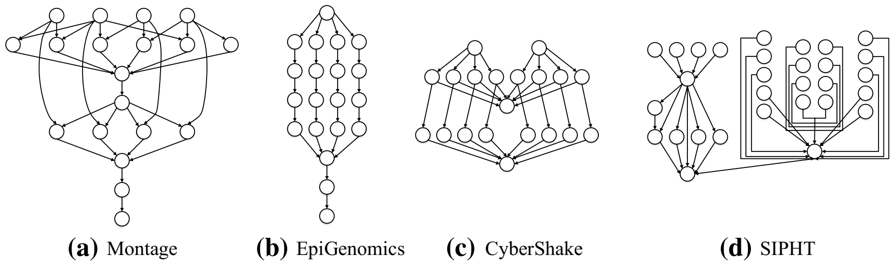


Fig. 1 Overview of four workflow structures

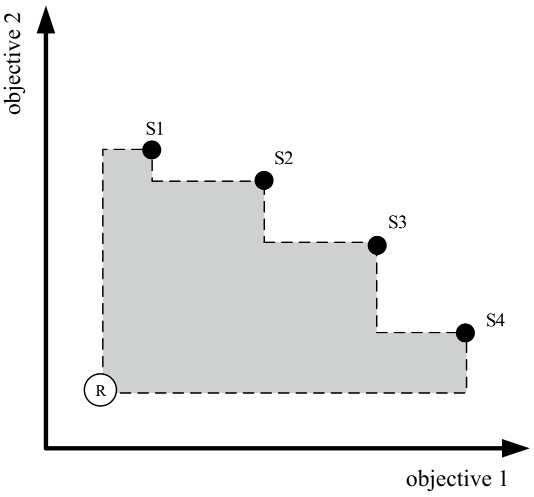


Fig. 2 Given a reference point *R*, the gray area represents the hypervolume obtained for a Pareto approximate set in a bi-objective environment

Table 1 Voltage–relative speed pairs

Level	Pair 1		Pair 2		Pair 3	
	Voltage (V_k)	Relative speed (%)	Voltage (V_k)	Relative speed (%)	Voltage (V_k)	Relative speed (%)
0	2.20	100	1.75	100	1.50	100
1	1.90	85	1.40	80	1.20	80
2	1.60	65	1.20	60	0.90	50
3	1.30	50	0.90	40		
4	1.00	35				

communication link between VMs is equal, and the value is set to 25 Mbps. What's more, the pricing model similar to Amazon EC2 is used in the experiments, and we assume that the unit price of each VM type is proportional to its performance.

In Pareto-based RL algorithms, the convergence time or the time to get the optimized solution is usually figured out according to a quality indicator called hypervolume [15, 25, 26].

As shown in Fig. 2, hypervolume is defined as the volume of the area between solution points and the reference point [19]. In the case of a maximization problem, a suitable reference point can be obtained by determining the lower limit of each objective and then subtracting a small constant. In the case of a minimization problem, the reference point can be obtained by determining the upper limit of each objective and then adding a small constant. If there is only one solution in the Pareto approximate set, the value of hypervolume will be 0. For the Pareto approximate set, a higher hypervolume means better quality.

The Q-learning algorithm will converge to the optimal policy if each action and state is sufficiently sampled [31]. For example, Fig. 3 shows the convergence

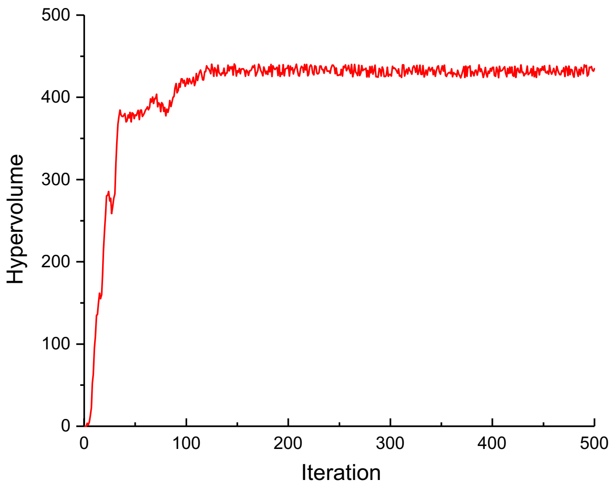


Fig. 3 The hypervolume value relative to the iterations in the case of Montage and $k_{BG} = 0.4$

of EnMORL with respect to hypervolume in the case of Montage and $k_{BG} = 0.4$. The convergence time here is expressed in the iterations. With the increase of the iterations, hypervolume tends to converge to a fixed value.

Due to the importance of parameters for RL, we conduct experiments to configure them. These parameter values are often used in RL algorithms. Table 2 shows a set of experiments in the case of Montage workflow and $k_{BG} = 0.4$. The columns "hyper" and "I_{con}" represent the hypervolume value and the convergence iteration, respectively. From the table, we find that the variation of parameters has a greater influence on the convergence time. In other cases, we have similar findings. Therefore, the parameter configurations are mainly based on convergence time. In all experiments, the parameter configurations for EnMORL are entirely identical. The parameter ϵ in the ϵ-greedy strategy is set to 0.1. The learning rate

Table 2 Parameter configuration in the case of Montage and $k_{BG} = 0.4$

Number	ϵ	α_i	γ	Hyper	I_{con}
1	0.1	0.1	0.9	441.33	119
2	0.1	0.2	0.8	448.12	131
3	0.1	0.3	0.7	439.34	127
4	0.2	0.1	0.8	454.87	134
5	0.2	0.2	0.7	460.23	145
6	0.2	0.3	0.9	469.56	149
7	0.3	0.1	0.7	481.24	157
8	0.3	0.2	0.9	477.50	165
9	0.3	0.3	0.8	489.62	163

α_i is fixed to 0.1 at any time step and the discount factor γ is set to 0.9. Results are collected and averaged over 30 trials of each 500 iterations.

6.3 Weight selection

The weight selection is very important for the multi-objective problems. The inappropriate weights have a terrible impact on the quality of the solution set. However, it is an acknowledged difficult problem to determine the weights in multi-objective algorithms [14]. Moffaert et al. [15] find that the combination of RL and the Chebyshev scalarization function is not particularly dependent on the weights used.

When applying the Chebyshev scalarization function to RL, most weight tuples can achieve good results. We can choose a relatively optimal weight tuple from randomly generated ones. The weight selection in the Chebyshev scalarization method is based on hypervolume.

In preliminary experiments, we select the relatively optimal weight tuple from randomly generated ones for each workflow according to the value of hypervolume. For instance, we present the weight selection in the case of the Montage workflow and $k_{\mathrm{BG}} = 0.4$. We randomly generate 10 scalarization weights w_1, which are in the range (0, 1), of the makespan objective. Then, the corresponding weights of the energy objective are calculated by $w_2 = 1 - w_1$. We collect and average the hypervolume of each weight tuple over 20 trials. Figure 4 shows the hypervolume value corresponding to 10 weight tuples in the case of Montage and $k_{\mathrm{BG}} = 0.4$. In particular, EnMORL returns only one solution in the case of the fourth tuple. So the corresponding hypervolume value is 0. We choose the third one, which corresponds to the largest hypervolume.

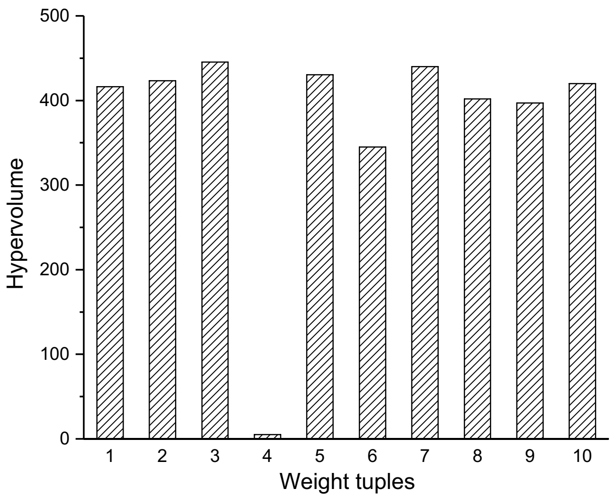


Fig. 4 The obtained hypervolume corresponding to the policies learned for each of the 10 weight tuples in the case of Montage and $k_{\mathrm{BG}} = 0.4$

6.4 Solution selection

There are multiple different solutions in the Pareto approximate set. Our goal is to minimize the makespan and energy consumption. As a result, we select the solution, which is closest to the origin, to evaluate the performance. We propose a metric to select such a solution from the approximate set. We first normalize the values of two objectives and then calculate the Euclidean distance between each normalized solution point and the origin. The Euclidean distance $D_{\mathrm{e}}(x)$ in terms of solution x can be calculated as follows:

$$D_{\mathrm{e}}(x) = \sqrt{\mathrm{NM}(x)^2 + \mathrm{NE}(x)^2}, \tag{27}$$

where $\mathrm{NM}(x) = f_1(x)/\mathrm{MS}_c$ is the normalized makespan, $\mathrm{NE}(x) = f_2(x)/E_h$ is the normalized energy consumption, f is the objective function, $f_1(x)$ is the actual makespan of solution x, MS_c is the makespan when all tasks are executed on the cheapest VM type, $f_2(x)$ is the actual energy consumption of solution x, and E_h is the energy consumption when all tasks are executed on the most expensive VM type.

6.5 Experimental results

In this section, EnMORL is compared with two popular multi-objective meta-heuristics, which are called NSGA-II [22] and MOPSO [24], in terms of makespan, energy consumption, and the hypervolume indicator. More specifically, NSGA-II used for comparison is the multi-objective algorithm in [13], MOPSO used for comparison is a new version called HSPO [8].

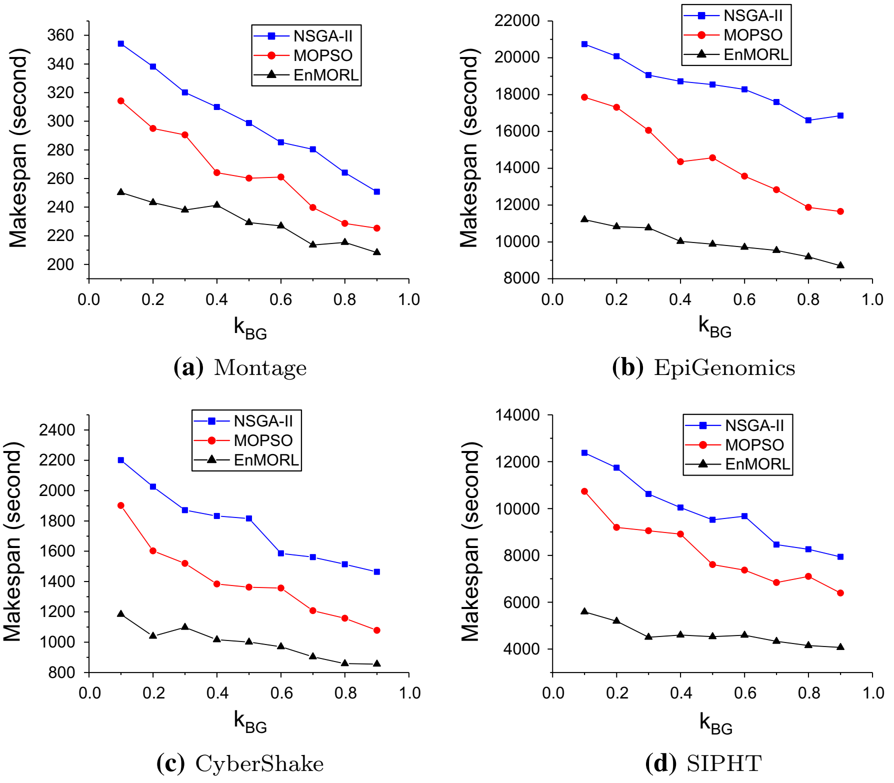
The experiments consist of four parts in the case of four different workflows, namely Montage, EpiGenomics, CyberShake, and SIPHT. They are representative and reasonable examples of workflows.

6.5.1 Montage workflow

The experimental results of the Montage workflow with different budget factors are shown in Figs. 5a, 6a, and 7a.

Figure 5a shows the results of the makespan. On the whole, the values of makespan generally decline with the rise of budget factors. When $k_{\mathrm{BG}} = 0.1$, the values of the makespan by using EnMORL are 29.33% and 20.36% lower than those obtained by using NSGA-II and MOPSO, respectively. When k_{BG} increases to 0.9, the values of the makespan obtained by using EnMORL become 16.93% and 7.55% lower than those obtained by using NSGA-II and MOPSO, respectively.

As we can see from Fig. 6a, energy consumption generally increases with the increase in budget factors. The main reason is that more tasks will be assigned to the VM types, which have higher performance when the budget increases. When a task is executed on a VM with higher performance, it consumes more energy. In the case of $k_{\mathrm{BG}} = 0.1$, the values of the energy consumption by using EnMORL are 16.02% and 5.71% lower than those obtained by using NSGA-II and MOPSO, respectively. When the value of k_{BG} increases to 0.9, the values of the energy consumption obtained



(a) Montage **(b)** EpiGenomics

(c) CyberShake **(d)** SIPHT

Fig. 5 Makespan of three multi-objective algorithms under different k_{BG} in the case of **a** Montage, **b** EpiGenomics, **c** CyberShake and **d** SIPHT

by using EnMORL become 38.24% and 29.01% lower than those obtained by using NSGA-II and MOPSO, respectively.

Hypervolume is a commonly accepted quality indicator in multi-objective RL [15]. Figure 7a shows the hypervolume values obtained by using three multi-objective algorithms when the values of budget factor k_{BG} are 0.2, 0.4, 0.6, and 0.8. We find that the hypervolume values obtained by EnMORL are higher than those obtained by NSGA-II and MOPSO. It is because that the weights used in EnMORL are selected according to hypervolume, while the weights used in NSGA-II and MOPSO are simply set to 0.5. NSGA-II and MOPSO are all scalarized based on the linear scalarization function. It is an acknowledged difficult problem to determine the weights in the linear scalarization function [14]. As a result, the weight tuple is often seen as the preference for each objective. If the hypervolume value is very low or equals to 0, the weight tuple in use is inappropriate.

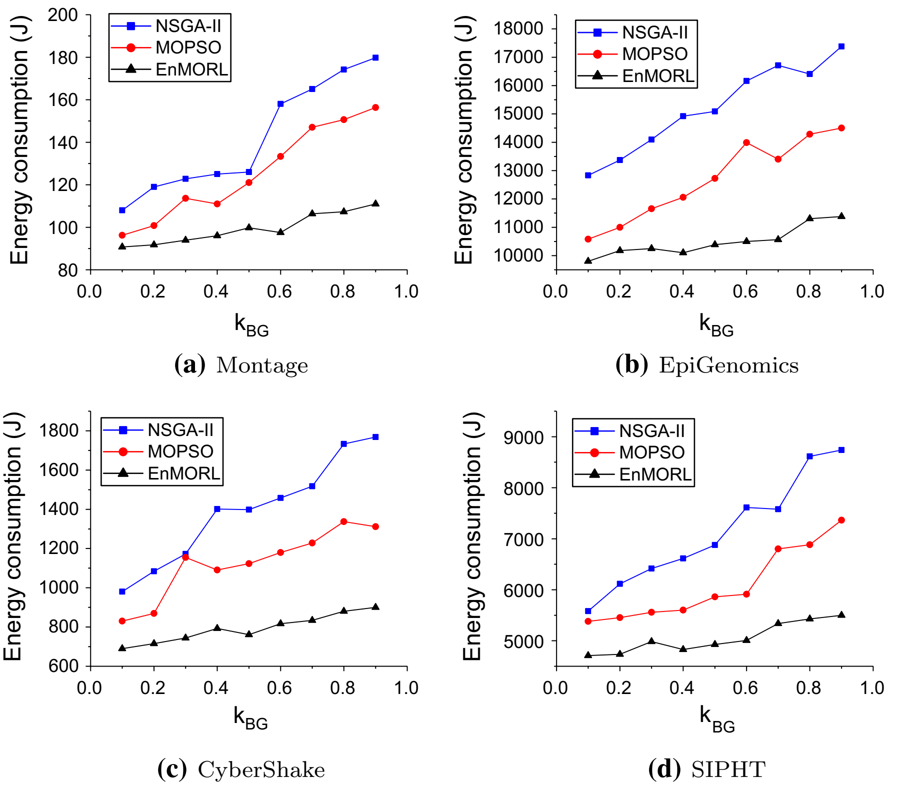


Fig. 6 Energy consumption of three multi-objective algorithms under different k_{BG} in the case of **a** Montage, **b** EpiGenomics, **c** CyberShake and **d** SIPHT

6.5.2 EpiGenomics workflow

The experimental results of the EpiGenomics workflow with different budget factors are shown in Figs. 5b, 6b and 7b.

Figure 5b shows the results of the makespan. On the whole, the values of makespan generally decline with the rise of budget factors. When $k_{BG} = 0.1$, the values of the makespan by using EnMORL are 45.95% and 37.22% lower than those obtained by using NSGA-II and MOPSO, respectively. When k_{BG} increases to 0.9, the values of the makespan obtained by using EnMORL become 48.36% and 25.29% lower than those obtained by using NSGA-II and MOPSO, respectively.

As we can see from Fig. 6b, the values of energy consumption generally increase with the increase in budget. In the case of $k_{BG} = 0.1$, the values of the energy consumption by using EnMORL are 23.63% and 7.37% lower than those obtained by using NSGA-II and MOPSO, respectively. When the value of k_{BG} increases to 0.9, the values of the energy consumption obtained by using EnMORL become 34.54% and 21.55% lower than those obtained by using NSGA-II and MOPSO, respectively.

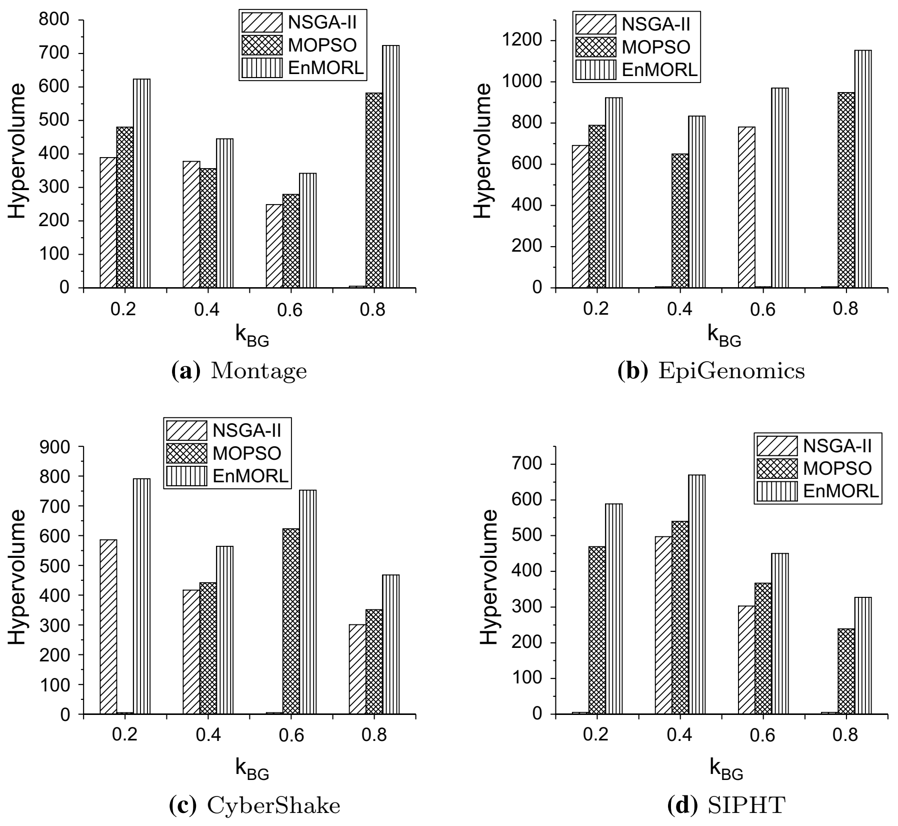


Fig. 7 Hypervolume of three multi-objective algorithms under different k_{BG} in the case of **a** Montage, **b** EpiGenomics, **c** CyberShake and **d** SIPHT

Figure 7b shows the hypervolume values obtained by using three multi-objective algorithms when the values of budget factor k_{BG} are 0.2, 0.4, 0.6, and 0.8. We can see that the hypervolume values obtained by using EnMORL are always higher than those obtained by using NSGA-II and MOPSO. If hypervolume is very low or equals to 0, it indicates that the weight tuple in use is inappropriate.

6.5.3 CyberShake workflow

The experimental results of the CyberShake workflow with different budget factors are shown in Figs. 5c, 6c and 7c.

Figure 5c shows the results of the makespan. On the whole, the values of makespan generally decline with the rise of budget factors. In the case of $k_{BG} = 0.1$, the values of makespan obtained by using EnMORL are 48.54% and 37.75% lower than those obtained by using NSGA-II and MOPSO, respectively. When k_{BG} increases to 0.9, the values of the makespan obtained by using EnMORL become 41.53% and 19.09% lower than those obtained by using NSGA-II and MOPSO, respectively.

As we can see from Fig. 6c, energy consumption generally increases with the increase in budget factors. In the case of $k_{\text{BG}} = 0.1$, the values of the energy consumption by using EnMORL are 29.60% and 16.87% lower than those obtained by using NSGA-II and MOPSO, respectively. When the value of k_{BG} increases to 0.9, the values of the energy consumption obtained by using EnMORL become 49.09% and 31.36% lower than those obtained by using NSGA-II and MOPSO, respectively.

Figure 7c shows the hypervolume values obtained by using three multi-objective algorithms when the values of budget factor k_{BG} are 0.2, 0.4, 0.6, and 0.8. We can see that the hypervolume values obtained by using EnMORL are always higher than those obtained by using NSGA-II and MOPSO. If the hypervolume value is very low or equals to 0, it indicates that the weight tuple in use is inappropriate.

6.5.4 SIPHT workflow

The experimental results of the SIPHT workflow with different budget factors are shown in Figs. 5d, 6d and 7d.

The results of the makespan are shown in Fig. 5d. On the whole, the values of makespan generally decline with the rise of budget factors. In the case of $k_{\text{BG}} = 0.1$, the values of makespan obtained by using EnMORL are 54.82% and 47.92% lower than those obtained by using NSGA-II and MOPSO, respectively. When k_{BG} increases to 0.9, the values of the makespan obtained by using EnMORL become 48.73% and 36.34% lower than those obtained by using NSGA-II and MOPSO, respectively.

As we can see from Fig. 6d, the values of energy consumption generally increase with the increase in budget factors. In the case of $k_{\text{BG}} = 0.1$, the values of the energy consumption by using EnMORL are 15.61% and 12.49% lower than those obtained by using NSGA-II and MOPSO, respectively. When the value of k_{BG} increases to 0.9, the values of the energy consumption obtained by using EnMORL become 37.07% and 25.29% lower than those obtained by using NSGA-II and MOPSO, respectively.

Figure 7d shows the hypervolume values obtained by using three multi-objective algorithms when the values of budget factor k_{BG} are 0.2, 0.4, 0.6, and 0.8. We can see that the hypervolume values obtained by using EnMORL are always higher than those obtained by using NSGA-II and MOPSO. If the hypervolume value is very low or equals to 0, it means that the weight tuple in use is inappropriate.

In conclusion, the comparison results show that the EnMORL algorithm outperforms the other two algorithms in terms of makespan, energy consumption, and the hypervolume indicator.

7 Conclusion

Workflow scheduling has been a major energy consumer in clouds. This paper considers a multi-objective workflow scheduling problem with the budget constraint. Our goal is to simultaneously minimize the makespan and energy consumption while meeting the budget constraint. This paper focuses on the weight selection problem and how to satisfy the budget constraint. Both of them were seldom considered in related works.

We propose a multi-objective RL algorithm called EnMORL. In EnMORL, we propose a simple new method to satisfy the budget constraint and guarantee the feasibility of solutions. This method is based on RCB and can be applied to other workflow scheduling problems with the budget constraint. EnMORL is based on the Chebyshev scalarization function, which can efficiently solve the weight selection problem. The experimental results show EnMORL outperforms these algorithms in the case of four workflows.

In future work, we will extend our algorithm to solve the dynamic workflow scheduling problem. We intend to use the conditional variational auto-encoder (CVAE) to predict the current situation according to the historical data.

Acknowledgements We would like to thank anonymous referees for their helpful suggestions to improve this paper. This work was supported in part by the National Natural Science Foundation of China under Grant NSFC 61672323, in part by the Fundamental Research Funds of Shandong University under Grant 2017JC043, in part by the Key Research and Development Program of Shandong Province under Grant 2017GGX10122 and Grant 2017GGX10142, and in part by the Natural Science Foundation of Shandong Province Grant ZR2019MF072.

References

1. Senyo PK, Addae E, Boateng R (2018) Cloud computing research: a review of research themes, frameworks, methods and future research directions. Int J Inf Manag 38(1):128–139
2. Khattar N, Sidhu J, Singh J (2019) Toward energy-efficient cloud computing: a survey of dynamic power management and heuristics-based optimization techniques. J Supercomput 75(8):4750–4810
3. Kintsakis AM, Psomopoulos FE, Mitkas PA (2019) Reinforcement learning based scheduling in a workflow management system. Eng Appl Artif Intell 81:94–106
4. Andrae ASG, Edler T (2015) On global electricity usage of communication technology: trends to 2030. Challenges 6(1):117–157
5. Ismayilov G, Topcuoglu HR (2020) Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing. Futur Gener Comput Syst 102:307–322
6. Belkhir L, Elmeligi A (2018) Assessing ict global emissions footprint: trends to 2040 and recommendations. J Clean Prod 177:448–463
7. Zhangjun W, Liu X, Ni Z, Yuan D, Yang Y (2013) A market-oriented hierarchical scheduling strategy in cloud workflow systems. J Supercomput 63(1):256–293
8. Verma A, Kaushal S (2017) A hybrid multi-objective particle swarm optimization for scientific workflow scheduling. Parallel Comput 62:1–19
9. Arabnejad H, Barbosa JG (2014) A budget constrained scheduling algorithm for workflow applications. J Grid Comput 12(4):665–679
10. Garg R, Singh AK (2014) Multi-objective workflow grid scheduling using ε-fuzzy dominance sort based discrete particle swarm optimization. J Supercomput 68(2):709–732
11. Wu CQ, Lin X, Yu D, Xu W, Li L (2014) End-to-end delay minimization for scientific workflows in clouds under budget constraint. IEEE Trans Cloud Comput 3(2):169–181
12. Chen W, Xie G, Li R, Bai Y, Fan C, Li K (2017) Efficient task scheduling for budget constrained parallel applications on heterogeneous cloud computing systems. Futur Gener Comput Syst 74:1–11
13. Sofia AS, GaneshKumar P (2018) Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-ii. J Netw Syst Manag 26(2):463–485
14. Das I, Dennis JE (1997) A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. Struct Optim 14(1):63–69

15. Van Moffaert K, Drugan MM, Nowé A (2013) Scalarized multi-objective reinforcement learning: Novel design techniques. In: 2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL), IEEE, pp 191–199
16. Zhu D, Melhem R, Childers BR (2003) Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems. IEEE Trans Parallel Distrib Syst 14(7):686–700
17. Zhou J, Wang T, Cong P, Lu P, Wei T, Chen M (2019) Cost and makespan-aware workflow scheduling in hybrid clouds. J Syst Arch. https://doi.org/10.1016/j.sysarc.2019.08.004
18. Gábor Z, Kalmár Z, Szepesvári C (1998) Multi-criteria reinforcement learning. In: Proceedings of the Fifteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, pp 197–205
19. Zitzler E, Thiele L, Laumanns M, Fonseca CM, Da Fonseca GV (2002) Performance assessment of multiobjective optimizers: an analysis and review. TIK-Report, vol 139
20. Li Z, Ge J, Haiyang H, Song W, Hao H, Luo B (2015) Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds. IEEE Trans Serv Comput 11(4):713–726
21. Qureshi B (2019) Profile-based power-aware workflow scheduling framework for energy-efficient data centers. Futur Gener Comput Syst 94:453–467
22. Deb K, Pratap A, Agarwal S, Meyarivan TAMT (2002) A fast and elitist multiobjective genetic algorithm: NSGA-ii. IEEE Trans Evolut Comput 6(2):182–197
23. Topcuoglu H, Hariri S, Min-you W (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst 13(3):260–274
24. Coello CAC, Pulido GT, Lechuga MS (2004) Handling multiple objectives with particle swarm optimization. IEEE Trans Evolut Comput 8(3):256–279
25. Mossalam H, Assael YM, Roijers DM, Shimon W (2016) Multi-objective deep reinforcement learning. arXiv preprint arXiv:1610.02707
26. Van Moffaert K, Nowé A (2014) Multi-objective reinforcement learning using sets of pareto dominating policies. J Mach Learn Res 15(1):3483–3512
27. Lee YC, Zomaya AY (2010) Energy conscious scheduling for distributed computing systems under different operating conditions. IEEE Trans Parallel Distrib Syst 22(8):1374–1381
28. Atkinson M, Gesing S, Montagnat J (2017) and Ian Taylor. Past, present and future, Scientific workflows
29. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction. MIT press, New York
30. Watkins CJCH (1989) Learning from delayed rewards
31. Tsitsiklis JN (1994) Asynchronous stochastic approximation and q-learning. Mach Learn 16((3):185–202
32. Wiering MA, De Jong ED (2007) Computing optimal stationary policies for multi-objective Markov decision processes. In: 2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, IEEE, pp 158–165
33. Vamplew P, Yearwood J, Dazeley R, Berry A (2008) On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts. In: Australasian Joint Conference on Artificial Intelligence, Springer, New York, pp 372–378
34. Voß T, Beume N, Rudolph G, Igel C(2008) Scalarization versus indicator-based selection in multi-objective CMA evolution strategies. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), IEEE, pp 3036–3043
35. Bharathi S, Chervenak A, Deelman E, Mehta G, Su M-H, Vahi K (2008) Characterization of scientific workflows. In: 2008 Third Workshop on Workflows in Support of Large-Scale Science, IEEE, pp 1–10
36. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R (2011) Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 41(1):23–50

Affiliations

Yao Qin[1] · Hua Wang[2] · Shanwen Yi[1] · Xiaole Li[3] · Linbo Zhai[4]

Yao Qin
sword93@mail.sdu.edu.cn

[1] School of Computer Science and Technology, Shandong University, Jinan 250101, China

[2] School of Software, Shandong University, Jinan 250101, China

[3] School of Information Science and Engineering, Linyi University, Linyi 276005, China

[4] School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China