Distrib Parallel Databases (2018) 36:339–368
https://doi.org/10.1007/s10619-017-7215-z

An adaptive multi-objective evolutionary algorithm for constrained workflow scheduling in Clouds

Miao Zhang[1] · Huiqi Li[1] · Li Liu[2] · Rajkumar Buyya[3]

Published online: 2 December 2017

Abstract The Cloud workflow scheduling is to find proper Cloud resources for the execution of workflow tasks to efficiently utilize resources and meet different user's quality of service requirements. Cloud workflow scheduling is a constrained and NP-complete problem and multi-objective evolutionary algorithms have shown their excellent ability to solve such problem. But most existing works simply use static penalty function to handle constraints which usually result in premature when the constraints become strict. On the other hand, with the search space being more tremendous and chaotic, how to balance the ability of exploring the entire search space and exploiting the important regions during the evolutionary process is increasingly important. In this paper, an adaptive individual-assessment scheme based on evolutionary states is proposed to handle the constraints in multi-objective optimization problems. In addition, the evolutionary parameters are also adjusted accordingly to balance the exploration and exploitation ability. These are distinguishable from most previous studies that directly incorporate multi-objective evolutionary algorithm to search excellent solutions for Cloud workflow scheduling. Experimental results demonstrate the proposed algorithm outperforms other state-of-the-art methods in convergence and diversity, and it also achieves better optimization ability when it is applied to solve Cloud workflow scheduling problem.

✉ Huiqi Li
huiqili@bit.edu.cn

1 School of Information and Electronics, Beijing Institute of Technology, Beijing, China

2 School of Automation and Electrical Engineering, University of Science and Technology Beijing, Beijing, China

3 The University of Melbourne, Parkville, Australia

Keywords Cloud computing · Workflow scheduling · Evolutionary algorithm · Pareto entropy

1 Introduction

Workflow is a common model to present scientific experiments [1] and it is formed by a number of tasks, data flows, and computing dependencies [2]. A process that maps workflow tasks with computational resources (VMs) for execution (preserving dependencies between tasks) is termed workflow scheduling. The scientific workflows are large-scale [3] which include tremendous data and computing requirements and need a high-performance computing environment for execution, and Cloud computing has been found as an applicable environment to run scientific workflows [2]. Cloud computing is the latest development of distributed computing, grid computing and parallel computing [4,5],which is able to deliver massive scaling computing resource as a utility, similarly as the way water and electricity are delivered to households these days. The service models provided by Cloud computing include: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) [6], where we focus on IaaS Cloud in this paper. IaaS Cloud provides computational and storage resources in the form of virtual machines (VMs). The difference between Cloud computing and distributed computing is that Clouds charge users by the running time of the leased instances (Virtual Machines), rather than instance-based.

In this paper, a proper task-VM mapping schedule to minimize the total financial cost and degree of imbalance is investigated while makespan satisfy the deadline constraint. It is a constrained multi-objective optimization and NP-complete problem. The multi-objective evolutionary algorithms (MOEAs), such as NSGA-II (Non-dominated Sorting Genetic Algorithm-II) [7,8] and MOPSO (Multi-Objective Particle Swarm Optimization) [11], have shown their excellent ability to solve multi-objective optimization problems. They could find a set of well converged and diversified non-dominated solutions within a short time for they reduce the computational complexity tremendously [9,10]. However, these evolutionary algorithms are designed for solving unconstrained problems, and how to handle the constraints is an urgent problem in Cloud workflow scheduling. Most researchers use penalty function to transform constrained problems into unconstrained ones, or simply eliminate infeasible individuals in evolutionary process [12,13], which usually leads the population to premature or infeasible search space.

To address constraint issue in Cloud workflow scheduling, a Pareto Entropy based NSGA-II with adaptive individual-assessment scheme is proposed, called ai-NSGA-II-PE. In our approach, an adaptive individual-assessment scheme is applied to the evolutionary process of EAs, which will give priority to different types of individuals for entering the next generation in different evolutionary states. In the initial and converging state, new regions are searched and excellent genes are explored in the global search space. In such evolutionary process, not only feasible solutions but also those solutions with outstanding objective values, even though they do not satisfy the constraints are searched for populations. After the population goes into diversifying state, all individuals are Pareto solutions (we assume the number of Pareto individual

is much larger than population size), so the algorithm should avoid the infeasible individual going into population, and enhance its local exploitation ability. In this way, the search for the feasible space and outstanding objective space is balanced. In addition, this paper uses the population information (the number of Pareto solutions and the Pareto Entropy) to detect population states, and adjusts the crossover and mutation probability accordingly, which are helpful for the convergence.

Our main contributions can be summarized as follows: (1) We utilize a Multi-Objective Evolutionary Algorithm to schedule Cloud scientific workflows which is based on the pay-per-runtime IaaS model. (2) The amount of Pareto solutions and Pareto Entropy of Pareto set in a population are combined to detect the population states accurately. To the best of our knowledge, it is the first approach to use the number of Pareto individuals and the Pareto Entropy to distinguish the evolutionary states of NSGA-II, and guide the evolutionary process to balance the global exploration and local exploitation. (3) An adaptive individual-assessment scheme is adopted in NSGA-II for solving constrained Cloud workflow scheduling, which utilizes different constraint-violation handling schemes in different evolutionary states. Experiments illustrate it could prevent premature efficiently and improve deadline meeting rate for workflow scheduling using Multi-Objective Evolutionary Algorithms (MOEAs). (4) The evolutionary parameters p_c and p_m are very significant for the convergence and diversity performance of NSGA-II. In this paper, we adaptively adjust these parameters based on the population feedback information and balance MOEA's exploration and exploitation ability accordingly.

The remainder of this paper is organized as follows: the following section introduces the related works on Cloud workflow scheduling, with particular emphasis to evolutionary-based methods. In the Sect. 3, an overview of our workflow scheduling architecture and optimization objective and constraints models as well as the definitions of problem are detailed. Section 4 introduces Multi-Objective Optimization and the Pareto Entropy, as well as the population state detection in MOEAs. In Sect. 5, the proposed method (ai-NSGA-II-PE) is applied to solve scientific workflow scheduling problem in Cloud. An adaptive individual-assessment scheme and adaptive evolutionary coefficients based on the population states are presented. Section 6 shows the results of this new adaptive evolutionary approach for 12 benchmark functions. The proposed approach is evaluated under several famous Scientific Workflows in Sect. 7, and comparison with classical MOEAs is performed. We conclude this paper with a discussion and a description of future work in Sect. 8.

2 Relate works

There are many works on the scientific workflow scheduling in the cloud environment, and two most popular list-based heuristic algorithms are Heterogeneous Earliest Finish Time (HEFT) and Critical-Path-on-a-Processor (CPOP) [32]. The Heterogeneous Earliest Finish Time (HEFT) is a scheduling algorithm that gives higher priority to the workflow task which has higher upper rank value from the beginning to end. This upper rank value is calculated by utilizing average execution time for each task and average communication time between resources of two successive tasks, where the

tasks in the critical path (CP) have comparatively higher rank values. On the other hand, CPOP uses the summation of the upper rank value and downward rank value, which is calculated same as the upper rank value but from the beginning task to end task, for prioritizing tasks. The two classical algorithms only aim to minimize finish time regardless of QoS constraints and total monetary cost. Recent studies on the list-based heuristics begin to take constraints into consideration and schedule workflow as a multi-objective problem. Paper [33] proposed a Budget-constrained Heterogeneous Earliest Finish Time (BHEFT) for workflow scheduling. Different with HEFT, this algorithm reserves the best budget in each assignment. Heterogeneous Budget Constrained Scheduling (HBCS) [34] is a recent budget constrained scheduling algorithm which define a cost coefficient to adjust the ratio between budget and cost to schedule workflow tasks. Multi-Objective Heterogeneous Earliest Finish Time (MOHEFT) [35] is a Pareto-based list heuristic algorithm. It is an extension of HEFT which optimizes time and cost simultaneously to generate trade-off schedules according to the two objectives.

Evolutionary algorithm is another widely-used approach to address the problem of constrained cloud workflow scheduling. Rodriguez and Buyya [12] presented a Particle Swarm Optimization (PSO) based approach for Cloud scientific workflow scheduling, which aimed to minimize execution cost of a workflow and make the makespan satisfy user-defined deadline-constraint. A static penalty function was used in this algorithm which considers the solutions that violate constraints are inferior to other feasible solutions. A GA-based method for virtual machines configuration in Cloud workflow scheduling is present in [36]. It considers constraints as objective, so as to transform the constrained single-objective optimization problem to the unconstrained multi-objective problem, and use multi-objective evolutionary algorithm to solve it. Paper [37] and [38] use multi-objective particle swarm optimization (MOPSO) algorithm to obtain Pareto-optimal tradeoffs between makespan and cost for workflow scheduling problem. Two improved MOEAs, NSGA-II* and SPEA2*, were proposed in [39] to generate a set of trade-off scheduling solutions according to the two users QoS requirements including service prices and execution time. Paper [2] proposed a multi-objective evolutionary algorithm -based approach to solve workflow scheduling problem on IaaS platform. In paper [40], a multi-objective evolutionary algorithm, SPEA2+, is employed to solve the multi-objective workflow scheduling problems with various QoS constraints. Although Multi-objective evolutionary algorithms have been widely applied in multi-objective workflow scheduling problems for its superiority that could optimize multiple objectives simultaneously and take the constraints into consideration at the same time, there are few works on adaptively handling constraints when applying MOEAs in constrained workflow scheduling. In this paper, we also focus on investigating different MOEAs with different constraints-handling methods for workflow scheduling.

3 General modeling of Cloud scientific workflow scheduling

Workflows are described as DAGs (Directed Acyclic Graph), $G = (V, E)$, where $V = \{t_1, t_2, \ldots, t_n\}$ is a set of tasks and E is the set of all data dependencies between

tasks. A vertex in the graph represents a task t and the edges maintain execution order and data transfer between tasks [14].The IaaS Cloud providers offer a range of VM types, and different VM types provide different computing resources. We define VM type in terms of its processing capacity P_{VM_i}, cost per unit of time C_{VM_i}, and the bandwidth β_{VM_i}, $VM_i = (P_{VM_i}, C_{VM_i}, \beta_{VM_i})$. Cloud Providers charge users based on the time unites when they leased VMs for executing tasks. In our experiments, we assume that each task is executed by a single VM, and a VM can execute several tasks.

The actual running time for task t_i executed by VM_{t_i} is calculated as:

$$RT_{t_i}^{VM_{t_i}} = S_{t_i} / P_{VM_{t_i}} \tag{1}$$

where S_{t_i} is the size of taskt$_i$, and $P_{VM_{t_i}}$ is the processing capacity of VM_{t_i}. The transfer time between a parent taskt_i and its child task t_j is calculated as:

$$TT_{e_{i,j}} = d_{t_i}^{out} / min\left\{\beta_{VM_i}, \beta_{VM_j}\right\} \tag{2}$$

where$d_{t_i}^{out}$ is the output data size produced by taskt_i. If two tasks are executed in the same VM, the transfer time is 0.

In this paper, we focus on finding a scheme that minimizes the total execution cost (*TEC*) and achieves a well-balanced load across all VMs in Cloud, which needs to minimize the imbalance among VMs, and the total execution time is supposed to satisfy the deadline constraint as well. We define a scheme $S = (M, TEC, DI, TET)$ in terms of task-VM matching (M), the total execution cost (*TEC*), the degree of imbalance (*DI*), and the total execution time (*TET*). M is comprised of VM types, start time and end time, $M = (m_{t_1}^{VM_{n_1}}, m_{t_2}^{VM_{n_2}}, \ldots, m_{t_M}^{VM_{n_M}})$, $m_{t_i}^{VM_{t_i}} = (t_i, VM_{t_i}, ST_{t_i}, ET_{t_i})$.The start time and end time for task t_i are calculated as:

$$ST_{t_i} = \begin{cases} LET_{VM_{n_i}}, & if\ t_i\ is\ an\ entry\ task \\ max\left(\max_{t_a\ \in\ parent\ (t_i)} \left(ET_{t_a} + TT_{e_{a,i}}\right), LET_{VM_{n_i}}\right) & otherwise \end{cases} \tag{3}$$

$$ET_{t_i} = ST_{t_i} + RT_{t_i}^{VM_{n_i}} \tag{4}$$

$$T_{VM_i} = \sum ET_{t_j} \quad for\ all\ tasks\ t_j\ assigned\ to\ VM_i \tag{5}$$

where $LET_{VM_{t_i}}$ is the lease end time of VM_{t_i}, which is also the time that VM_{t_i} becomes idle [12,15].

There will not be charge for data transfers within the same data center, so this fee is not considered when calculating the total cost of workflow. The total execution cost *TEC*, the total execution time *TET* and the degree of imbalance *DI* are calculated as:

$$TEC = \sum_{i=1}^{|VM|} C_{VM_{t_i}} * \left[\frac{RT_{t_i}^{VM_{t_i}}}{\tau}\right] + \sum_{i \in T, j \in T} TT_{e_{i,j}} {}^{*} TC_{VM_{t_i}} \tag{6}$$

$$DI = \frac{T_{max} - T_{min}}{T_{avg}} \tag{7}$$

$$\{R_1, R_j, \ldots, R_K\}\ S_j V_j$$
$$E(I) = \sum_{j=1}^{K} \frac{S_j}{S_I} E(R_j) \tag{8}$$

where $TC_{VM_{t_i}}$ is the data transfer cost for VM_{t_i}.T_{max} and T_{min} are the maximum and minimum running time among all VMs respectively; T_{avg} is the average running time of VMs. Each workflow has a deadline d_W associated with it, which determines the allowed longest time to complete its execution.

In this article, the problem can be described as: finding a scheme S with minimum total execution cost TEC and degree of imbalance (DI), and the total execution time TET satisfies the workflow's deadline constraint d_W. Equation (9) has depicted this problem.

$$\begin{aligned} &Minimize: TEC \\ &\qquad\quad DI \\ &Subject\ to: TET \leq d_W \end{aligned} \tag{9}$$

4 Background of multi-objective optimization and Pareto entropy

This section describes the background about Multi-Objective Evolutionary Algorithms (MOEAs), Pareto Entropy and population states detection in NSGA-II.

4.1 MOEAs

MOEAs are effective methods to handle multi-objective problems with two or more conflicting objectives. They could find a set of well converged and well diversified non-dominated solutions within short time due to the nature that they could reduce the computational complexity tremendously. There are several mature MOEAs, including MOPSO [11], SPEA, SPEA2 [10], NSGA, NSGA-II [7,8], MOEA/D [9], Hyper-MOEA [41], and so on.

In evolutionary processes, the evolutionary parameters have a big impact on balancing the global exploration of the entire search and local exploitation of important regions [17]. For example, in NSGA-II, the bigger crossover probability p_c and mutation probability p_m are, the more new individuals will be generated along with the diversity of population. But too big p_c and p_m will destroy good gens easily, while too small p_c and p_m, is supposed to slow down search speed is not conducive to generate new individuals [16].The selection of p_c and p_m is very critical to the performance of GAs. The crossover probability p_c controls the capability of GAs in exploiting a located hill to reach the local optima, and the higher p_c means the quicker local exploitation proceeds. The mutation probability p_m controls the speed of GAs in exploring a new area, which represents the global exploration of GAs. In order to balance the exploitation and exploration ability of GAs, the crossover probability p_c and mutation probability p_m should be adjusted differently based on population states.

According to Zhang et al. [17,18], the evolutionary process in NSGA-II can be depicted as four states, including initial state, converging state, diversifying state and matured state. For example, in the initial and converging state, the algorithm is supposed to increase its global exploration ability, to guide the population to search new areas and approximate the Pareto Front. However, in the diversifying state, the local optima should be exploited along with the diversity. In the matured states, the population needs to decrease its global exploration and local exploitation ability simultaneously.

So, how to detect the evolutionary states for NSGA-II is significant for their performance. In this paper, we use the number of Pareto individuals and the Pareto Entropy to detect the population states of NSAG-II, and adaptively adjust p_c and p_m accordingly.

4.2 Pareto entropy in MOEAs

Entropy indicates the chaos of a system in thermodynamics, and it is also a good way to describe the diversity of a population in MOEAs. To calculate the entropy, we have to find out the distribution of all solutions in the population, and Parallel coordinates [19] is a widely-used visualization method for multi-dimension data analysis, which presents multi-dimension data in a two-dimension plane. Paper [18,20] proposed a Parallel Cell Coordinate System (PCCS) to assess the density of each Pareto solution in an archive for selecting leaders and updating the archive in MOPSO.

In PCCS, the *m-th* objective value for *k-th* Pareto solution, f_k^m, $k = 1, 2, 3, \ldots, K$, $m = 1, 2, \ldots, M$, is mapped to an integer label within a 2-dimension grid with $K \times M$ cells according to (10):

$$L_k^m = \left\lceil K \frac{f_k^m - f_m^{min}}{f_m^{max} - f_m^{min}} \right\rceil \tag{10}$$

where K is the current size of archive, $\lceil x \rceil$ is the ceiling operator that returns the smallest integer which is not less than x. f_m^{min} is the minimum and f_m^{max} is the maximum of the *m-th* objective value for all solutions in the archive. $L_k^m \in \{1, 2, \ldots, K\}$ and it is set as 1 if $f_k^m = f_m^{\min}$. In this way, each solution in the archive is expected to share along a cell in each dimension which indicates that all individuals in the Pareto front are well-distributed.

Figure 1 illustrates the transforming process from Cartesian Coordinate System into PCCS. It is clear that PCCS is better to show the distribution of solutions in every dimension.

When the Pareto solutions are mapped to PCCS from Cartesian Coordinate System, the Entropy could be used to measure the diversity of Pareto Front. Evolutionary algorithms update Pareto Front every iteration that some new individuals are added into the Pareto front or some new ones substitute old ones. Those changes will also incur the variation of Entropy $\Delta Entropy$. Larger $\Delta Entropy$ indicates that there are many old Pareto solutions replaced by new ones, or a plenty of solutions are added into the Pareto set, or the new added solutions change the f_m^{min} and f_m^{max} dramatically and incurs the redistribution of coordinates in PCCS. In such situations, the population is likely in the initial or converging state. Conversely, if the $\Delta Entropy$ is very small,

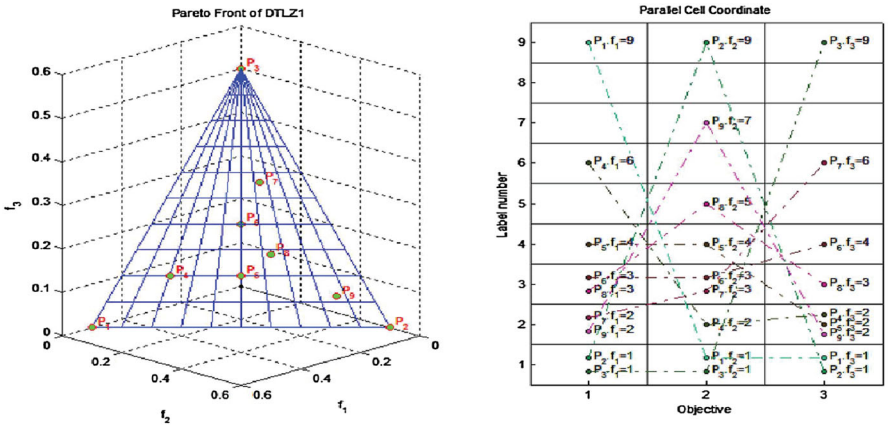

Fig. 1 A mapping example for Pareto front from Cartesian coordinate system to parallel cell coordinate system [20]

there are just several Pareto solutions that have been replaced by individuals with better Crowding distance, and the population more likely situates in the diversifying or matured state.

According to [8,20], the Pareto Entropy in the *t-th* iteration is calculated based on (11):

$$Entropy(t) = -\sum_{k=1}^{K}\sum_{m=1}^{M}\frac{Cell_{k,m}(t)}{KM}\log\frac{Cell_{k,m}(t)}{KM} \tag{11}$$

where $Cell_{k,m}(t)$ is the number of member with the label L_k^m. The variation of Pareto Entropy between generation t and t-1 can be obtained through the following equation:

$$\Delta Entropy(t) = Entropy(t) - Entropy(t-1) \tag{12}$$

We assume an extreme example as depicted in the Fig. 2. In this case, the population is in the diversifying state, and several new solutions with better Crowding distance are diversified-replaced into the Pareto Front (which means the new enter solution has better Crowding distance than old one, and do not dominate the old solution; a new solution cloud only diversified-replace an old solution).This evolution incurs the maximum $\Delta Entropy$ from the best distributed population to the worst distributed, for these new added solutions lead to a dramatic change in f_m^{min} and f_m^{max}, and makes the well distributed population with biggest entropy (where each members in terms of each objectives takes up a cell along in PCCS) to the worst one (where K-1 members in terms of each objectives crowd in a single cell, and the remaining one take up another cell). And the biggest $\Delta Entropy$ in this situation $\Delta Entropy_{max-diver}$ is calculated:

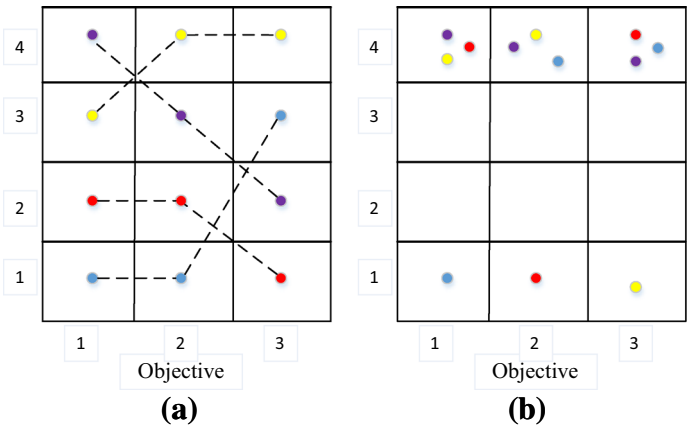

Fig. 2 Examples for individual distribution in the PCCS. **a** is with Crowding distance and **b** is with the worst Crowding distance

$$
\begin{aligned}
\Delta Entropy_{max-diver} &= Entropy_{max} - Entropy_{min} \\
&= -\sum_{k=1}^{K}\sum_{m=1}^{M}\frac{1}{KM}\log\frac{1}{KM} \\
&\quad + \sum_{k=1}^{K-1}\sum_{m=1}^{M}\frac{Cell_{k\neq l,m}(t)}{KM}\log\frac{Cell_{k\neq l,m}(t)}{KM} \\
&\quad + \sum_{m=1}^{M}\frac{Cell_{k=l,m}(t)}{KM}\log\frac{Cell_{k=l,m}(t)}{KM} \\
&= \log KM - \left(\frac{K-1}{K}\log\frac{KM}{K-1} + \frac{1}{K}\log KM\right) \\
&= \frac{K-1}{K}\log KM - \frac{K-1}{K}\log\frac{KM}{K-1} \\
&= \frac{K-1}{K}\log(K-1)
\end{aligned} \tag{13}
$$

Figure 3 illustrates the $Entropy(t)$ and $\Delta Entropy(t)$ of the population when we use NSGA-II to solve a test problem ZDT1 [29]. We could find that the entropy has an increasing trend during the evolution as a whole. At the beginning of the evolution, the Entropy has several abrupt declines where the population is in the initial state or converging state. That may be due to that an excellent solution goes into the Pareto set, dominates some previous Pareto solutions and excludes them out of Pareto set. Another reason might be that the new added solutions change the f_m^{min} and f_m^{max} dramatically, which incurs the redistribution of coordinates in PCCS and makes the old Pareto solution huddle together. After the population goes to the later generations, the Pareto entropy increases gradually. That is because that the well-distributed solutions with bigger Crowding Distance will replace those ones with smaller Crowding distance,

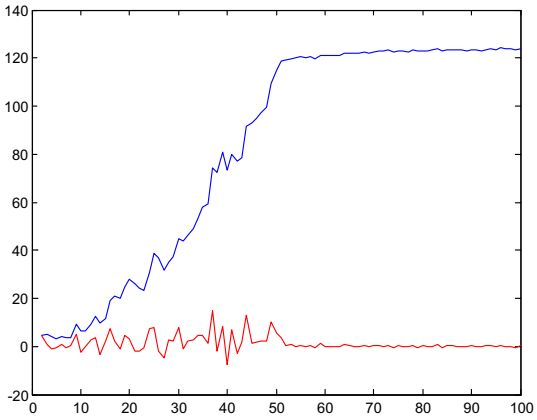

Fig. 3 Curves of $Entropy(t)$ and $\Delta Entropy(t)$ detected from ZDT1 with NSGA-II. The blue curve is the $Entropy(t)$ and the red one is the $\Delta Entropy(t)$

which results in the growth of Pareto Entropy. And the entropy remains the same after the population goes to the matured state, where the $\Delta Entropy(t)$ still remains 0.

4.3 Population states detection in NSGA-II

Convergence speed and optimization accuracy are conflict goals in the evolutionary algorithm: putting too much emphasis on convergence speed may lead to premature and local optimum; on the contrary, it will consume too much time if only the optimization accuracy is considered [21,22]. So, devising an adaptive scheme for global exploration and local exploitation is the key to balance the convergence and diversity ability. In the multi-objective genetic algorithm, adopting an adaptive crossover and mutation scheme based on the feedback information from the population could control the evolution efficiently. For example, in NSGA-II, we want to increase the mutation probability p_m and decrease the crossover probability p_c in the converging state, to guide the population to search new areas and approximate to the Pareto Front, and use the smaller p_m and larger p_c to exploit the local optima along with the diversity in the diversifying state. So how to distinguish the evolutionary states is a key issue for adaptively adjusting global exploration and local exploitation ability in NSGA-II.

In our proposed NSGA-II (detailed in the next section), we use the $\Delta Entropy$ and the number of individuals of Pareto set to detect the evolutionary states. There are several scenarios we need to notice.

Firstly, if the number of Pareto solutions $np(t)$ in generation t is less than population size (generally, the amount of Pareto solutions for all search space is much larger than the population size), the population is identified as initial or converging state, and the algorithm needs to find new non-dominated solutions to dominated-replace the old ones.

Secondly, if the number of Pareto solutions in generation t is not equal to that of generation $t+1$, which means p Pareto solutions of generation t are replaced by q new Pareto individuals, $\Delta np(t+1) = q - p \neq 0$, the population is in the converging state, because only the dominated-replace operator (the new enter solution dominates

several old Pareto solution and exclude them out of the archive) may change the number of Pareto solutions, while the diversified-replace operator would not (a new solution cloud only diversified-replace a old solution).

Thirdly, it is possible that the number of Pareto solutions in generation t is equal to population size S, and there are p Pareto solutions are replaced by p new Pareto individuals in generation $t+1$, $\Delta np(t+1) = 0$, and it is very hard to identify the evolutionary state. In this situation, our algorithm assumes that the population is in the converging state when $\Delta Entropy(t+1) > \frac{1}{M*2}\Delta Entropy_{max-diver}$ (M is the number of objectives), because the f_m^{min} and f_m^{max} of some optimization objectives may be changed by the new entered solutions and incur the redistribution of coordinates in PCCS, leading to large $\Delta Entropy$. In this scenario, we want to enhance the population's exploration ability to search new regions. When $\Delta Entropy(t+1) \leq \frac{1}{M*2}\Delta Entropy_{max-diver}$, $np(t) = S$, $\Delta np(t+1) = 0$, the population is asserted as diversifying state.

So the rules of evolutionary states detection for generation $t+1$ in our algorithm can be depicted as following:

Rule 1 The population is in the initial state in the generation $t+1$ if $np(t) = 0$;

Rule 2 The population is in the converging state in the generation $t+1$ if $0 < np(t) < S$;

Rule 3 The population is in the converging state in the generation $t+1$ if $np(t) = S$, $\Delta np(t+1) \neq 0$;

Rule 4 The population is in the converging state in the generation $t+1$ if $np(t) = S$, $\Delta np(t+1) = 0$, and $\Delta Entropy(t+1) > \frac{1}{M*2}\Delta Entropy_{max-diver}$;

Rule 5 The population is in the diversity state in the generation $t+1$ if $np(t) = S$, $\Delta np(t+1) = 0$, and $\Delta Entropy(t+1) \leq \frac{1}{M*2}\Delta Entropy_{max-diver}$.

Rule 6 The population is in the matured state in the generation $t+1$ if $np(t) = S$, $\Delta np(t+1) = 0$, and $\Delta Entropy(t+1) = 0$.

5 Proposed constrained Cloud workflow scheduling

Scientific workflow scheduling in Cloud environment is a NP-hard and constrained optimization problem, and it is also very hard for the existing general evolutionary approaches to find a suitable solution. Considering the problem's properties, we apply an adaptive individual-assessment scheme in NSGA-II to handle the constraints, and adaptively adjust the evolutionary coefficients based on evolutionary states, and we call it ai-NGSA-II-PE.

5.1 General modeling of Cloud scientific workflow scheduling

In this paper, the chromosome in ai-NGSA-II-PE for scientific workflow scheduling in Clouds is shown in Fig. 4; $chromosome_i$ represents a decision solution. Unlike other previous works' encoding strategies which stipulate the execution order of the tasks in a single VM in a chromosome, we only encode the mapping between tasks and VMs, and the execution order of the tasks in a single VM is predefined according

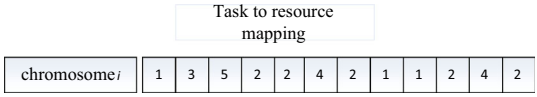

Fig. 4 A chromosome representation of workflow task-VM mapping

to a set of deterministic developmental rules: the task with the smallest ready time (the time that all parent tasks are executed and all required data have been transferred) in a single VM will be executed first. If there are several tasks with same ready time, the task with earliest finish time is expected to execute firstly. This strategy is also in line with the real workflow execution environment [23,24].

For the scheduling scenario here, a $chromosome_i$ represents a task-resource matching scheme of a workflow, where the index represents a task and its value represents the number of VM associated with this task.

ALGORITHM 1
TEC AND TET ESTIMATION

Input: *a set of workflow tasks T, a set of VMs* $\overline{VM}$, *and a* $chromosome_{k,j}$ $inpopulationion_{1,j}$
Output: *TEC and TET*

1. *Initial* VMs state matrix *VS* and task state matrix *TS*.

2. *Calculate* execution time $RT[|T|\times|\overline{VM}|]$;

Calculate transfer time $TT[|T|\times|T|]$;

3. For i=1: $|T|$

If $TS(T(i))$ is unscheduled

3.1. $t_i = T(i), VM_{t_i} = vm_{chromosome_{k,j}(i)}$

3.2. If t_i *has no parents*

$$ST_{t_i} = LET_{VM_{t_i}};$$

Else

$$ST_{t_i} = max\left(\max_{t_a \in parent(t_i)}(ET_{t_a} + TT_{e_{a,i}}), LET_{VM_{t_i}}\right);$$

End

3.3. For *each child task* t_c *of* t_i

If t_c *is mapped to a VM different to* VM_{t_i}

$$TT\ i = TT\ i + TT\ i,c$$

End

End

3.4. $RT_{t_i}^{VM_{t_i}} = RT(t_i, VM_{t_i})$;

3.5. $ET_{t_i} = RT_{t_i}^{VM_{t_i}} + TT(i)$

3.6. *update VS , and TS , set the time period* $[ST_{t_i}, ET_{t_i}]$ *for* VM_{t_i} *is busy, set* $TS(T(i))$ *as scheduled.*

End

End

4. *Calculate TEC according to (6);*
5. *Calculate DI according to (7);*
6. *Calculate TET according to (8);*

Figure 4 represents a workflow with 12 tasks and 5 different types of VMs available. The fitness function is used to determine the performance of a solution, which is calculated by two objectives: total execution cost TEC, and degree of imbalance DI,

and a constraint: total execution time TET. The calculations of *TEC, DI* and *TET* for a decision solution are explained in Algorithm 1.

All the calculation of *TEC*, *TET* and *DI* are based on the execution time of VMs, so identifying the running time of VMs is the core to determine how good a decision solution is.

Firstly, we initial VMs state matrix *VS*, task schedule state matrix *TS* and a set of VMs; a set of tasks of workflow are also given. Then we estimate the execution time $RT_{t_i}^{VM_{t_i}}$ of each workflow task t_i on every type of VM according (1), and the transfer time $TT_{e_{i,j}}$ between tasks according to (2).

There are two scenarios for the start time value ST_{t_i}.If there is no parent tasks, the task can start as soon as the VM assigned to the task is idle. Otherwise, the task starts after the parent tasks have been finished and the output data are transferred. Furthermore, if the VM is still busy, the start time has to be delayed until the VM is available. The end time value ET_{t_i} is calculated according to (4). After a task has been scheduled, we update VMs state matrix *VS* and task state matrix *TS*, set the task t_i as scheduled, and the time period between ST_{t_i} and ET_{t_i} as busy for VM_{t_i}. The process continues until all tasks have been scheduled. Finally, the *TEC*, *DI* and *TET* are calculated based on (6)–(8) respectively.

5.2 Adaptive individual-assessment scheme for fitness function

We adopt an adaptive individual-assessment scheme for fitness function in our proposed algorithm in different evolutionary states.

Inspired by Nanakorn et al. [25,26], we find that the constraint violation should play a dominate role to push the population to feasible areas when there are few feasible individuals, and where the population is supposed to move towards the feasible areas first. When most individuals are feasible, those individuals with excellent objective values are supposed to be retained, and the penalty function needs to be weakened. The final fitness value for each optimization objective under different evolutionary states is formulated in the follow.

(1) In the initial state, the fitness function is calculated as:

$$F^a(x_i) = \tilde{E}(x_i) \tag{14}$$

where $\tilde{E}(x_i$ is the normalized constraint violation.
We use the constraint violation to sort the individuals rather than Non-dominate rank and Crowding distance. Obviously, by using such approach, the individuals with smaller amount of constraints violation are considered better. Consequently, the search will look for the region where the sum of constraints violation is small (i.e. the boundary of the feasible region) when there is no feasible solution in the population.

(2) When the population goes into the converging state where there is at least one feasible individual in the current population, the final fitness of *i-th* individual for *k-th* objective $F_k^{\alpha} x_i)$ can be depicted as:

$$F_k^{\alpha}(x_i) = \begin{cases} \tilde{F}_k(x_i) & for\ feasible\ individual \\ \sqrt{\tilde{F}_k(x_i)^2 + \tilde{E}(x_i)^2} + \left[\left(1 - r_f\right)\tilde{E}(x_i) + r_f \tilde{F}_k(x_i)\right] & otherwise \end{cases} \tag{15}$$

where $\tilde{F}_k(x_i)$ is the normalized fitness value of $i - th$ individual for $k - th$ objective, $\tilde{E}(x_i)$ is the normalized constraint violation, $r_f = \frac{number\ of\ feasible\ individuals}{population\ size}$. Individuals with both better objective values and low constraint violation are considered better than individuals with worse fitness value or high constraint violation or both. And if the feasibility ratio (r_f) in the population is small, then the individuals closer to the feasible space are considered better. Otherwise, the individual with lower normalized fitness value is better. We calculate the Non-dominate rank and Crowding distance for each individual based on their final fitness values, and then sort them accordingly.

(3) When the population enters into the diversifying state, our algorithm employs another kind of individual-assessment scheme to sort individuals. In the diversifying state, the population should enhance its local exploitation ability and avoid infeasible solution entering into the population. The constraint-domination principle in [27] is very suitable for this situation:

Definition 1 A solution S_1 is considered better than another solution $S_{2,}$ if any of the following conditions is true:

1. If S_1 is feasible and S_2 is infeasible;
2. Both S_1 and S_2 are infeasible, and S_1 has lower constraint violation;
3. S_1 and S_2 are feasible, and S_1 have a higher non-domination rank than S_2 are feasible;
4. S_1 and S_2 are feasible, and have the same non-domination rank, and S_1 has better Crowding distance than S_2.

Our approach guarantees that it will search the feasible individuals with better diversity and avoid infeasible individuals entering the population in the diversifying state.

5.3 Adaptive parameter adjustment in ai-NSGA-II-PE

In this section, we explicitly describe the adaptive parameters adjustment of the proposed approach applied into NSGA-II for constrained Cloud workflow scheduling, which is called ai-NSGA-II-PE (Pareto Entropy based NGSA-II with adaptive individual-assessment scheme).

The key difference between the proposed algorithm and the original NSGA-II is that there is an archive (also called as Pareto set) in our algorithm to retain those feasible Pareto solutions. And the number of individuals and Pareto Entropy of the archive are used to detect population state and adjust parameters accordingly.

As discussed in the previous sections, adaptively adjusting crossover probability p_c and mutation probability p_m is very essential for balancing the global exploration and local exploitation ability. In the GAs, larger p_m and smaller p_c facilitate global exploration, while larger p_c and smaller p_m promote local exploitation [21,22]. Therefore, we decrease p_c and increase the p_m in the initial and converging states, to push the population to the Pareto front. Conversely; during the diversifying state, we increase the p_c and decrease the p_m which increases the diversity of the population. The parameters, p_c and p_m, are kept decreasing when the population is in the matured state.

In our proposed ai-NSGA-II-PE, the crossover probability and mutation probability are adjusted based on population state, number of Pareto solutions and Pareto Entropy. The adjustment schemes are depicted as follow:

$$p_c(t) = \begin{cases} \langle p_c(t-1) + c_1 \rangle \ in\ the\ initial\ state \\ \langle p_c(t-1) - c_2{}^*\Delta np(t-1) \rangle \ in\ the\ converaging\ state \\ \langle p_c(t-1) + c_3{}^*\Delta Entropy(t-1) \rangle \ in\ the\ diversifing\ state \\ \langle p_c(t-1) - c_4{}^*\Delta Entropy(t-1) \rangle \ in\ the\ matured\ state \end{cases} \tag{16}$$

$$p_m(t) = \begin{cases} \langle p_m(t-1) - m_1 \rangle \ in\ the\ initial\ state \\ \langle p_m(t-1) - m_2{}^*\Delta np(t-1) \rangle \ in\ the\ converaging\ state \\ \langle p_m(t-1) - m_3{}^*\Delta Entropy(t-1) \rangle \ in\ the\ diversifing\ state \\ \langle p_m(t-1) - \mathrm{m}_4{}^*\Delta Entropy(t-1) \rangle \ in\ the\ matured\ state \end{cases} \tag{17}$$

$\langle \bullet \rangle$ is a function that keep the p_c and p_m in the boundaries. When p_c and p_m are less than the lower boundaries, they are set to their minima; when they exceed the upper boundary, they are set to their maxima. $\Delta np(t)$ is the variation of Pareto set's size in generation t. $\Delta Entropy(t)$ is the variation of Pareto Entropy in generation t.

In this way, the crossover probability p_c decreases quickly and p_m increases slowly to enhance the algorithm's global exploration ability to push the population towards the Pareto front in the initial and converging state. In the diversifying state, $p_c$increases slowly and p_m decreases quickly to enhance the algorithm's local exploitation ability. The both parameters decrease quickly in the matured state. This adjustment strategy is termed fast-down and slow-up [18].

5.4 Ai-NSGA-II-PE for scientific workflow scheduling in Cloud

The pseudo-code of the proposed ai-NSGA-II-PE is depicted in Algorithm 2.

In NSGA-II, a diversified initial population could accelerate the search procedure greatly, so we generate the initial population by different methods:

ALGORITHM 2 EVOLUTION PROCESS OF AI-NSGA-II-PE

Input: g_{max} p_c, p_m, N,M

Output: *best population* pop_b

1. ***Set*** g_{max} *//set the maximum generation*

2.*Initializing* *a population* pop_g *randomly*

3. FOR $g=1$: g_{max} {

3.1. NDS_g = *non-domination-sort (* pop_g *)//rank individual by non-dominant sort.*

3.2. CD_g = *crowding-distance-calculation(* pop_g *)// calculate crowding distance of individuals with same* NDS_g.

3.3.*offsprings= selection (* pop_g, NDS_g, CD_g *) // generate a new population according to non-dominant rank and crowding distance.*

3.4. *offspringc= crossover (offsprings,* $p_c(t-1)$, $s(t-1)$ *)//execute crossover operation to population based on population state*

3.5. *offspringm= mutation (offsprings,* $p_m(t-1)$, $s(t-1)$ *)//apply c mutation operation to population based on population state*

3.6.*pop-comb=* pop_g *+ offspringm //combine the former population and the generated population by evolutionary operation*

3.7. $NDS_{pop-comb}$ = *non-domination-sort (pop-comb) // rank each individual according to non-dominant sort for the*

3.8. $CD_{pop-comb}$ *=crowding-distance-calculation (pop-comb) // calculate crowding distance of individuals with same* NDS_g *for the combined population*

3.9. *pop-combs=replace (pop-comb) // get a new population according to non-dominant rank and crowding distance.*

3.10.*get the Pareto set, calculate the Pareto entropy and detect the population state s(t)*

3.10. pop_{g+1} *=pop-combs//get the next generation population.*

3.11.*g=g+1}*

END

- N/10 individuals generated through HEFT method [32] which assigns the unscheduled tasks to VM that could execute them with shortest time, and it is treated as the fastest schedule.
- N/10 "cheapest" individuals are also produced which always assign tasks to VM with least cost
- N*8/10 random individuals are initialized through RANDOM algorithm.

In our ai-NSGA-II-PE, the Non-dominated sorting and Crowding Distance calculation operations in each generation are the same as that in NSGA-II, and the time complexity for these two operations is $O(M(2N)^2 + M^*2N^*log(2N))$. The time complexity for both crossover and mutation operation are $O(N)$, the fitness calculation for a solution is $O(N^*n^*n)$, and the evolutionary states detection has $O(N^*M)$ complexity. Above all, the overall complexity of our proposed ai-NSGA-II-PE is $O(g^*(M(2N)2 + M^*2N^*log(2N) + N + N^*n^2 + N^*M))$.

6 Experiments on benchmark functions

To examine the optimization performance of our proposed evolutionary approach, experiments have been conducted on multi-objective optimization benchmark functions, including unconstrained problems and constrained problems. All the experiments were performed on computers with Inter Core i5-4570S CPU (2.9 GHz and 8 G RAM).

Table 1 Descriptions of test benchmark functions

Code	Nature	Dimensions of		Feature of PF
		Variables	Objectives	
ZDT1	Unconstrained	30	2	Convex
ZDT2	Unconstrained	30	2	Concave
ZDT3	Unconstrained	30	2	Disconnected multi-modal
UP1	Unconstrained	10	2	Convex
UP2	Unconstrained	10	2	Convex
UP4	Unconstrained	10	2	Concave
Binh2	Constrained	30	2	Convex
Srinivas	Constrained	30	2	Convex, multi-modal
Tanaka	Constrained	30	2	Disconnected multi-modal
CTP	Constrained	30	2	Linear, multi-modal
CP1	Constrained	10	2	Disconnected, Linear
CP2	Constrained	10	2	Disconnected Convex

6.1 Benchmark functions

Six widely used unconstrained test instances (ZDT1-ZDT3, UP1, UP2, UP4) and six constrained problems (Binh(2), Srinivas, TANAKA, CTP, CP1, CP2)[28,29] are employed here for comparing the proposed algorithm with other MOEAs. Among all these twelve problems, UP1, UP2, UP4 and CTP, CP1, CP2 have much more complicated search space. The description of each optimization problem is depicted in Table 1.

6.2 Compared algorithms

In order to validate our proposed algorithm, we compare ai-NSGA-II-PE with five MOEAs, including NSGA-II [7,8], NSGA-II-PE (This algorithm adopts our adaptive crossover and mutation probability adjusting scheme while using the normal penalty function to handle the constraints), MOEA/D [9], SPEA2 [10], MOPSO [11]. The population sizes for all algorithms are set as 200, the generation is 100, and the niche size (the number of neighboring sub-problems) in MOEA/D is set to20. In the GA-based algorithms (NSGA-II, NSGA-II-PE, MOEA/D, SPEA2), the initial p_c and p_m are set as 0.8 and 0.2 respectively. As to the MOPSO, the initial learning factor $c1 = c2 = 1$, and inertial weight $\omega = 0.4$.

6.3 Performance metrics

Two comprehensive performance metrics, inverted generational distance (IGD) and Spread [8,30], are used to quantify the convergence and diversity of an approximate

Pareto front obtained by an algorithm. The performances of the above algorithms are compared in terms of IGD (inverse generational distance) in (18) and Spread in (19).

IGD calculated by (18) determines the convergence of an algorithm, which is the average distance of the obtained solution points to the real Pareto fronts.

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{18}$$

where d_i is the Euclidean distance between the obtained solution points and the closest point of the real Pareto front and n is the number of the obtained solutions. Hence,$IGD = 0$ indicates that the obtained solutions are in the real Pareto front.

Spread in (19) illustrates the diversity of an algorithm [24], and it is a metric to calculate the broadness using (19).

$$Spread = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \overline{d}|}{d_f + d_l + (N-1) * \overline{d}} \tag{19}$$

whered_i is the Euclidean distance between two neighboring points, $\overline{d}$ is the average of $d_i.d_f$ and d_l are the Euclidean distance of the two boundary points in the obtained Pareto front set. Spread $= 0$ means the obtained Pareto front performs well in diversity.

6.4 Experimental results

Figure 5 presents the Pareto fronts obtained by compared algorithms and proposed approach for test benchmark functions. These pictures intuitively show that our approach could find better Pareto fronts that are closer to the true Pareto fronts in both unconstrained problems and constrained problems robustly. Even though these comparing algorithms could get satisfactory results in some test instances, they can hardly find the right Pareto front when the search space becomes complicated, such as UP1, UP2, UP4, CTP, CP1, CP2. Conversely, our approach is still able to get the approximate Pareto front in such cases.

The summary of statistical experimental results, including mean IGD and Spread generated by the above algorithms over 12 test instances, are listed in Table 1. The best value (minimum) of IGD and Spread among those test algorithms is highlighted in boldface in each test instance.

It can be seen from Table 2 that NSGA-II-PE obtains 3 best values of IGD on ZDT3, UP2 and UP4 and SPEA2 obtains 3 best values of IGD on ZDT2, UP1. MOEA/D gets the best result on ZDT1. In constrained problems, our algorithm ai-NSGA-II-PE performs more excellent which obtains 4 best results over 6 test instances. Especially, the convergence results obtained by ai-NSGA-II-PE with adaptive penalty function are better than that of NSGA-II-PE which uses a fixed penalty function during the whole evolutionary process.

As to the diversity indicated by Spread in Table 2, the proposed NSGA-II-PE approach for solving unconstrained problems and ai-NSGA-II-PE for solving con-

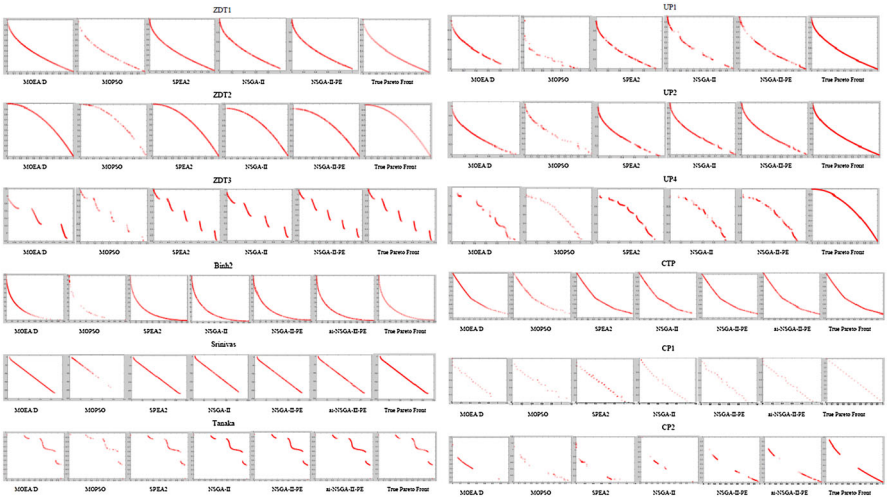

Fig. 5 Pareto fronts obtained by related algorithms and the proposed algorithm. (The last ones are the true Pareto fronts)

Table 2 Mean IGD and spread generated by 6 algorithms (Color table online)

		Unconstrained problem						Constrained problem					
		Zdt1	Zdt2	Zdt3	Up1	Up2	Up4	Binh2	Tanaka	Srinivas	CTP	CP1	CP2
MOEAD	IGD	**4.37e−3**	4.40e−3	1.76e−2	3.50e−3	3.22e−3	1.44e−2	**4.55e−2**	4.99e−3	**7.42e−2**	2.50e−3	2.07e−2	1.18e−2
	Spread	2.85e−1	**1.40e−1**	8.99e−1	**6.54e−1**	5.90e−1	1.13e−0	9.79e−1	1.41e−0	1.85e−1	6.60e−1	1.62e−0	1.25e−0
MOPSO	IGD	5.76e−3	2.34e−2	1.69e−2	4.34e−2	1.06e−2	2.12e−2	4.39e−1	1.51e−2	1.90e−1	4.86e−3	7.96e−2	4.54e−2
	Spread	9.48e−1	9.60e−1	8.79e−1	1.00e−0	9.79e−1	8.39e−1	1.43e−0	**9.68e−1**	9.32e−1	7.12e−1	**8.52e−1**	**8.62e−1**
SPEA2	IGD	5.10e−3	**4.21e−3**	8.78e−3	**3.32e−3**	3.05e−3	2.20e−2	5.70e−2	3.97e−3	8.38e−2	2.66e−3	2.48e−2	1.33e−2
	Spread	**1.52e−1**	1.48e−1	**4.61e−1**	6.03e−1	4.56e−1	6.92e−1	**1.45e−1**	1.38e−0	**1.69e−1**	**2.74e−1**	1.43e−0	1.45e−0
NSGA-II	IGD	1.41e−2	6.76e−3	2.17e−2	7.38e−3	3.46e−3	1.51e−2	5.30e−2	3.72e−3	7.79e−2	2.73e−3	3.62e−2	1.72e−2
	Spread	4.01e−1	4.81e−1	6.78e−1	1.01e−0	4.50e−1	1.13e−0	4.89e−1	1.13e−0	3.66e−1	6.42e−1	1.43e−0	1.67e−0
NSGA-II-PE	IGD	1.28e−2	4.47e−3	**1.11e−2**	4.68e−3	**3.16e−3**	**1.26e−2**	5.40e−2	3.65e−3	8.51e−2	1.25e−2	3.17e−2	1.72e−2
	Spread	3.43e−1	4.76e−1	6.34e−1	1.08e−0	**3.84e−1**	**6.53e−1**	5.12e−1	1.14e−0	1.06e−1	1.15e−0	1.36e−0	1.40e−0
ai-NSGA-II-PE	IGD	—	—	—	—	—	—	5.15e−2	**3.55e−3**	8.07e−2	**2.63e−3**	**2.01e−2**	**1.02e−2**
	Spread	—	—	—	—	—	—	4.71e−1	1.13e−0	4.58e−1	5.65e−1	1.11e−0	1.31e−0

The performance of these benchmark functions using different algorithms are displayed in the above table. In each column, the algorithm with red bold get the best IGD performance under this benchmark function, and the green bold indicate the best Spread performance

strained problems could get 2 best Spread results in UP2 and UP4, which are all with complicated search space. Although the SPEA2 get better diversified results in 5 test instance, its optimizing ability is worse than our approach under most test instances.

The ability to balance the global exploration and local exploitation to enhance the optimization ability in our algorithm can be observed from the results of all test problems, which could get the most number of best results. In addition, adaptively adjusting the penalty function based on evolutionary states could effectively prevent the premature comparing with those algorithms using simple penalty functions, especially in solving constrained optimization problems.

Table 3 Types of VMs used in the experiments

Name	Processing capacity (MFLOPS)	Bandwidth (bytes/s)	Cost per hour
m1.small	44	39,321,600	$0.03
m1.large	176	85,196,800	$0.12
m1. xlarge	352	131,072,000	$0.24
c1. medium	220	85,196,800	$0.15
c1. xlarge	880	131,072,000	$0.60

7 Experiments on practical problems

After solving several typical test problems, we apply the proposed algorithm (ai-NSGA-II-PE) to the Constrained Workflow Scheduling in Cloud environment. This problem aims at finding a task-VM mapping scheme S with minimal total execution cost TEC and degree of imbalance (DI), and the total execution time TET shouldnot exceed the workflow's deadline constraint d_W.

7.1 Simulation settings

In our experiments, an IaaS provider which offers a single data center and 8 types of VMs is modeled. The VM configurations are based on current Amazon EC2 offerings and are presented in Table 3. We set processing capacity of each type of VMs based on the work by Ostermann et al.[24].

We have defined 4 different deadlines in our experiments. These deadlines lie between the slowest and the fastest runtimes, where the slowest runtime is obtained via using a single VM to execute all tasks, and the fastest runtimes is obtained by using HEFT to scheduling workflows. The deadlines gradually become stricter, and Deadline 4 is the strictest one.

All the experiments were performed on computers with Inter Core i5-4570S CPU(2.9GHz and 8G RAM).

7.2 Workflows

The simulated workflows are 5 famous scientific workflows: Epigenomics, Montage, CyberShake, Sipht, and Inspiral [3,24]. Each of these workflows has different structures as shown in Fig. 6.

7.3 Optimizing objectives

In these experiments, we want to minimize the total execution cost (TEC) and degree of imbalance (DI), while the total execution time (TET) meets the deadline constraints. To compare the results, we considered the average, and 95% CI for TEC, DI and

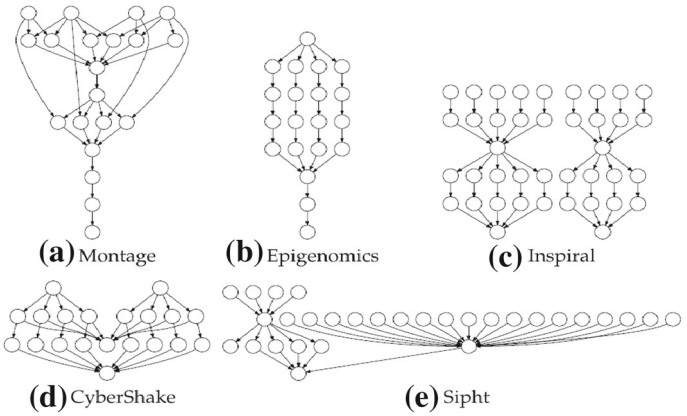

Fig. 6 Structures of five famous scientific workflows [2]

Pareto Fronts of feasible solutions, obtained through each algorithm after running each experiment for 30 times.

7.4 Compared algorithms

In order to validate the proposed algorithm, we compare ai-NSGA-II-PE with five MOEAs, including NSGA-II [7,8], NSGA-II-PE(This algorithm adopt our adaptive crossover and mutation probability adjusting scheme while using the normal penalty function to handle the constraints), MOEA/D [9], SPEA2 [10], MOPSO [11].The parameters for all algorithms are set as same as those in the benchmark function experiments. In the following experiments, MOEA/D and SPEA2 both use a excluding strategy which will harshly exclude those infeasible solution out of population in the evolutionary process. On the other hand, the remaining compared algorithms adopt a static penalty function.

7.5 Experimental results

In this section, we analyze the algorithms in terms of total execution cost *TEC*, and degree of imbalance DI under different user's defined deadlines.

7.5.1 The total execution cost (TEC)

The 95% CI for total execution costs obtained by all algorithms for each of the workflows under different deadlines are displayed in Table 4, and the average of TEC could also be observed through 95% CI. Our algorithm acquires the most number of best values, and it performs much better than other peer algorithms when constraints become strict.

Table 4 TEC obtained by MOEAs for different workflows ubder different constraints

Algorithm	Epigenomics	Montage	CyberShake	Sipht	Inspiral
	Deadline 1				
MOEA/D	[5.2082, 5.3402]	[5.7191,5.8104]	[3.1689, 3.4260]	[3.274, 3.320]	[3.9091, 4.1172]
MOPSO	**[4.1118, 4.1565]**	[5.5776, 5.8995]	[3.2453, 3.6781]	**[2.881, 3.027]**	[5.2589, 5.6470]
SPEA2	[4.3991, 5.2091]	[2.3568, 12.4846]	**[2.3050, 3.6969]**	[2.982, 4.767]	**[1.4322, 2.1323]**
NSGA-II	[4.6582, 4.6919]	[5.3994, 5.7927]	[4.5027, 4.9173]	[3.601, 3.650]	[5.9850, 6.4263]
NSGA-II-PE	[4.5572, 4.7919]	[5.2546, 5.6078]	[4.5027, 4.9173]	[3.113, 3.153]	[5.9850, 6.4263]
ai-NSGA-II-PE	[4.4352, 4.5860]	**[5.1571, 5.4523]**	[4.2227, 4.8673]	[3.023, 3.049]	[6.1395, 6.5865]
	Deadline 2				
MOEA/D	[5.2706, 5.4568]	[5.6378,5.7246]	[3.2784, 3.4207]	[3.4293, 3.4519]	**[4.4281, 4.5823]**
SPEA2	**[4.2272, 4.3057]**	[6.2863, 6.5375]	**[2.7213, 3.1470]**	[3.9330,3.9472]	[4.8438, 5.1588]
MOPSO	[5.2833, 5.3409]	[5.9689, 7.6913]	[3.3075, 4.8445]	[3.3651, 4.1949]	[2.3028, 3.4138]
NSGA-II	[5.2407, 5.3562]	[5.2951, 5.3993]	[4.8117, 5.2714]	[3.6496, 3.7898]	[5.3935, 5.8619]
NSGA-II-PE	[5.2407, 5.3562]	[6.4329, 6.7245]	[4.6117, 5.0714]	[3.1603, 3.2009]	[5.3935, 5.8619]
ai-NSGA-II-PE	[4.8568, 5.7157]	**[5.3193, 5.3698]**	[4.4811, 4.8951]	**[3.1593, 3.1884]**	[5.5901, 6.0526]
	Deadline 3				
MOEA/D	[5.2277, 5.3221]	–	**[3.0901, 3.2878]**	[4.118, 4.133]	[6.2028, 6.2540]
SPEA2	**[4.5341, 4.5853]**	–	[3.2800, 3.6833]	[2.010, 2.010]	[6.2265, 6.2835]
MOPSO	[6.0012, 7.7508]	[8.9958, 10.562]	[2.7719, 4.9253]	[4.239, 7.265]	[5.3429,5.3471]
NSGA-II	[5.1764, 5.2381]	[7.2572, 7.4080]	[4.3751, 4.7581]	[3.643, 3.723]	[5.8891,6.2563]
NSGA-II-PE	[5.1764, 5.2381]	[7.2572, 7.4080]	[4.5672, 4.9886]	[3.580, 3.690]	[5.4951, 5.9463]
ai-NSGA-II-PE	[5.2014, 5.2892]	**[6.3933, 6.4848]**	[4.5054, 4.8411]	**[3.812, 3.825]**	**[5.4056, 5.8294]**

Table 4 continued

Algorithm	Epigenomics	Montage	CyberShake	Sipht	Inspiral
	Deadline 4				
MOEA/D	[5.4326, 5.5231]	–	[5.1070, 5.2191]	–	–
SPEA2	[5.2249, 5.3005]	–	[5.4534, 5.8452]	–	–
MOPSO	[5.7357, 6.0162]	–	[4.5025, 4.7539]	[5.5169, 5.7746]	**[5.2458, 5.6223]**
NSGA-II	[5.2726, 5.3621]	[8.2829, 8.3912]	[4.6065, 4.9410]	[4.5324, 4.6670]	[5.7009, 6.1464]
NSGA-II-PE	[5.2726, 5.3621]	[8.2829, 8.3912]	[4.6065, 4.9410]	[4.6578, 4.8181]	[5.7009, 6.1464]
ai-NSGA-II-PE	**[4.8953, 4.9375]**	**[7.9305, 8.1332]**	**[4.3823, 4.7296]**	**[4.4538, 4.5486]**	[6.0647, 6.5938]

The best performances under different deadlines for each workflow are highlighted in bold

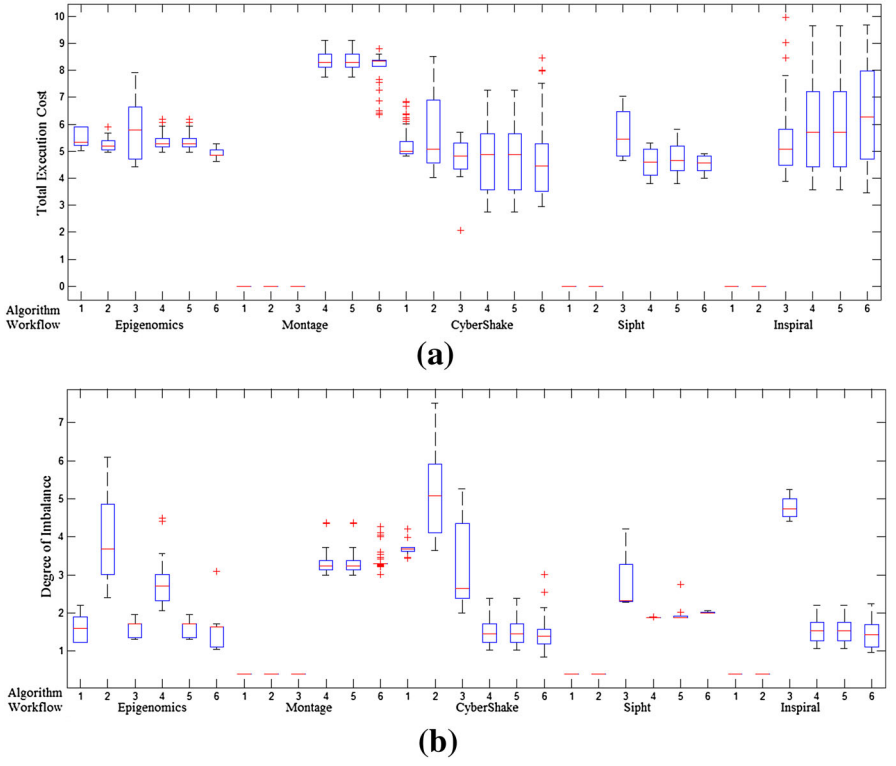

Fig. 7 **a** Box plots for TEC values obtained by 6 MOEAs for 5 workflows under Deadline4. ('–' means this MOAE could not get any feasible solution for workflow scheduling under this constraint. Number 1–6 are MOEA/D, MOPSO, SPEA2, NSGA-II, NSGA-II-PE, and ai-NSGA-II-PE, respectively). **b** Box plots for DI values obtained by 6 MOEAs for 5 workflows under Deadline4. ('–' means this MOAE could not get any feasible solution for workflow scheduling under this constraint, Number 1–6 are MOEA/D, MOPSO, SPEA2, NSGA-II, NSGA-II-PE, and ai-NSGA-II-PE, respectively)

Generally, the TEC of solutions under strict constraints will be worse than that of loose ones, because the constraints restrict the search space. We can find that, in the first two deadlines, the total execution costs obtained by all algorithms are nearly the same. However, when the constraints become stricter, the peering algorithms could not find satisfactory solutions. For example, in the Montage workflow, the comparing algorithms MOPSO and MOEA/D are able to get satisfactory results in terms of TEC in the first 2 constraints, but in the last 2 constraints, they could not get feasible solutions in several workflows. Conversely, our proposed approach is still capable of getting excellent solutions in the strict constraints, even though with a small increase. Obviously, when the constraints become stricter, especially in the Deadline 4, the peering algorithms which harshly exclude unfeasible solution out of the evolution or simply use a static penalty function are unable to find any correct solutions because the populations search towards a wrong direction and are trapped in a local optimum. Figure 7a gives the box plots of TET values obtained

by all algorithms under Deadline4 for five scientific workflows. The figure shows that ai-NSGA-II-PE clearly outperforms these compared multi-objective evolutionary algorithms under the strictest deadline. Although some peer algorithms with static penalty function could still get several feasible solutions under strict constraints, their qualities are much worse than the proposed algorithm. The advantage of ai-NSGA-II-PE is that it adaptive adjusts its local exploitation and global exploration ability, and adopts different individual-assessment schedule in different evolutionary states, both of which have prevented the premature and being trapped in local optimum effectively.

7.5.2 Degree of imbalance (DI)

Turning to another optimizing objective DI, the results are much like that of TEC, and we also illustrate the 95% CI of DI in Table 5. The data indicate that, although in some cases (e.g. Montage with deadline 1, 2) the obtained results of compared algorithms are similar to or even better than ai-NSGA-II-PE, our algorithm is able to find satisfactory solutions under strict constraints while these compared algorithms could not. Figure 7b also gives the box plots of DI values obtained by all algorithms under the strictest Deadline4 for five scientific workflows, and our algorithm gets 4 best results from 5 workflows. This observation confirms the analysis in the above section. However, it should be noted that the DI obtained by each algorithms shows a downward trend when the constraints become stricter, while TEC is opposite. That is mainly because the TET constraint and DI optimizing objective are not mutually independent. When the TET constraints make the algorithms to use more VMs to reduce the execution time, this will also reduce DI at the same time.

7.5.3 Tradeoff between the two optimizing objectives

We plot the two optimizing objectives TEC-DI trade-off on the Epigenomics and Inspiral workflow under different users' defined constraints as examples in Fig. 8. The Pareto fronts just contain those feasible solutions, as the algorithms should generate a cost-efficient and balancing schedule but not at the expense of a long execution time. There is no use for an algorithm to generate cheap and balancing schedules without meeting the deadlines. Obviously, Fig. 8 is a summary of the above results. We can find that, under the first two constraints, SPEA2 and MOEA/D could even get the best Pareto fronts. However, when it turns to the strictest constraints, they are not able to generate any feasible solution, where the meeting rates are 0. In these peering algorithms with static and improper penalty function, the Pareto fronts are worse than that of our algorithm, especially when users strictly restrict the TEC. On the contrary, ai-NSGA-II-PE always performs well in different deadlines, and could get the best Pareto fronts in the strict deadlines.

Table 5 DI obtained by MOEAs for different workflows ubder different constraints

Algorithm	Epigenomics	Montage	CyberShake	Sipht	Inspiral
	Deadline 1				
MOEA/D	[1.3383, 1.4832]	[1.4039, 1.4359]	[1.3541, 1.4356]	[1.680, 1.742]	[1.0804, 1.1549]
SPEA2	[1.4372, 1.4459]	**[0.9411,0.9949]**	[1.0507, 1.3150]	[1.994, 2.017]	**[0.7673, 0.8501]**
MOPSO	[2.7198, 3.7450]	[1.8338, 5.1380]	[1.8470, 3.4927]	[1.737,2.309]	[4.0846,4.5523]
NSGA-II	[1.3953, 1.4379]	[1.0084,1.0723]	[1.1335, 1.2513]	[1.643, 1.703]	[0.9694, 1.0681]
NSGA-II-PE	**[1.3953, 1.4379]**	[1.0616, 1.1403]	[1.1335, 1.2513]	[1.328, 1.577]	[0.9694, 1.0681]
ai-NSGA-II-PE	[2.6980, 3.0498]	[1.1035, 1.1978]	**[1.1070, 1.2401]**	**[1.222, 1.395]**	[0.9567, 1.0512]
	Deadline 2				
MOEA/D	[1.7449, 1.8285]	[1.4306, 1.4694]	**[1.0718, 1.1851]**	[1.4927, 1.5558]	[1.2805, 1.3404]
SPEA2	[1.5034, 1.6029]	**[1.3968, 1.4372]**	[1.5014, 1.7939]	[1.9799, 2.0516]	**[0.8756, 0.9634]**
MOPSO	[2.4009, 3.3781]	[3.7792,6.0811]	[1.4999,2.2666]	[1.6613,1.6644]	[2.9396,3.6926]
NSGA-II	[1.3714, 1.4340]	[1.5161, 1.5655]	[1.1156, 1.2505]	[1.7003,1.8236]	[1.1988, 1.3127]
NSGA-II-PE	**[1.3714, 1.4340]**	[1.4516, 1.6272]	[1.1156, 1.2505]	[1.9593,2.0138]	[1.1988, 1.3127]
ai-NSGA-II-PE	[2.3768, 2.6958]	[1.6514, 1.7280]	[1.1743, 1.2644]	**[1.4946, 1.5675]**	[1.1597, 1.2548]
	Deadline 3				
MOEA/D	**[1.2099, 1.2876]**	–	[1.4636, 1.5421]	[1.320, 1.321]	[1.7559,1.7973]
SPEA2	[2.3443, 2.7008]	–	[1.0674, 1.2866]	[1.708, 1.708]	[1.9440,1.9467]
MOPSO	[1.4871, 2.9208]	[3.5799, 3.670]	[1.4674, 2.4111]	[1.537, 2.335]	[2.9239, 3.6838]
NSGA-II	[1.7686, 1.9687]	[1.9230, 2.0611]	[1.4461, 1.5246]	[1.538, 1.595]	[1.1661, 1.2653]
NSGA-II-PE	[1.5686, 1.7287]	[1.8307, 1.9956]	[1.2136, 1.3201]	**[1.483, 1.503]**	[1.1661, 1.2653]
ai-NSGA-II-PE	[1.7742, 1.8734]	**[1.8007, 1.9866]**	**[1.1241, 1.2097]**	[1.579, 1.579]	**[1.1636, 1.2462]**

Table 5 continued

Algorithm	Epigenomics	Montage	CyberShake	Sipht	Inspiral
	Deadline 4				
MOEA/D	[1.5892, 1.6887]	–	[1.7134, 1.7721]	–	–
SPEA2	[3.7755, 4.0945]	–	[2.0033, 2.3207]	–	–
MOPSO	[1.6275, 1.6914]	–	[2.1812, 2.4871]	[2.6678, 2.8994]	[4.7382, 4.8160]
NSGA-II	[2.7272, 2.9304]	[3.3376, 3.4527]	[1.4503, 1.5284]	**[1.8836, 1.8856]**	[1.4981, 1.5817]
NSGA-II-PE	[1.6275, 1.6914]	[3.3376, 3.4527]	[1.4503, 1.5284]	[1.9223, 1.9674	[1.4981, 1.5817]
ai-NSGA-II-PE	**[1.4723, 1.6334]**	**[3.3255, 3.3870]**	**[1.3596, 1.4807]**	[2.0201, 2.0260]	**[1.4064, 1.5102]**

The best performances under different deadlines for each workflow are highlighted in bold

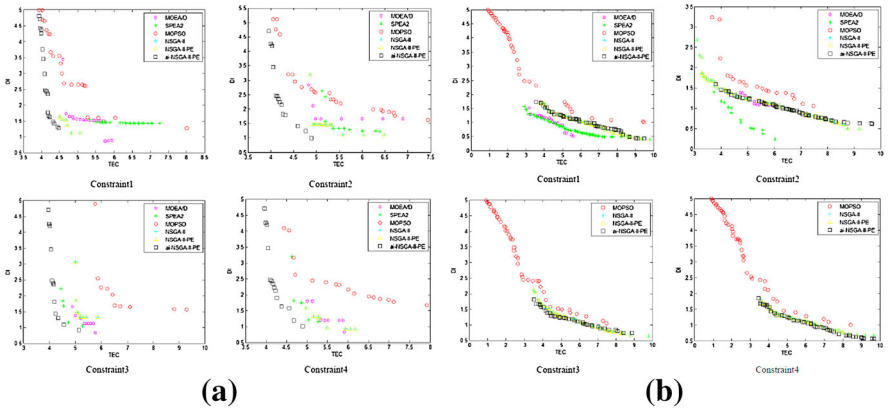

Fig. 8 Pareto Fronts obtained by different algorithms under several constraints for two workflows. **a** Workflow = Epigenomics, **b** Workflow = Inspiral

8 Conclusion and future work

In this paper, an adaptive Multi-Objective evolutionary approach for constrained scientific workflow scheduling in Clouds is proposed. For most previous works on Cloud workflow scheduling, the most common drawback is that they use a static penalty function to handle the constraints, which leads to premature convergence easily when the users define a strict constraint. In addition, as the Cloud environment is very chaotic, the workflow scheduling in Cloud has a high demand for optimizing ability to prevent being trapped in local optimum. As a solution to these problems, the proposed algorithm designs an adaptive penalty function based on population states and adjusts crossover and mutation probability accordingly. Benchmark function results indicate that our approach could effectively exploit the information hidden in different states and guides the evolutionary direction for algorithms to escape from local optimum, particularly in solving constrained problems. Experimental results on practical problems also show that our approach has outperformed several state-of-the-art algorithms, and could find better solutions especially under strict constraints.

Many-Objective Evolutionary Algorithm (MaOEA) to solve the cloud resource scheduling problem will be investigated in the future, as sometimes the number of optimizing objectives in Cloud workflow scheduling is far more than 2 or 3. However, the complexity of EAs rises sharply and the algorithms have a higher demand for local exploitation and global exploration ability with the increase of the number of optimizing objectives and search space becoming complex. How to reduce the complexity of MaOEAs without the cost of degradation is our research emphasis. Another future work is to construct a Multi-Cloud environment with multiple data center for simulating Workflows, which will consider the data transfer cost between data centers so that VMs can be deployed on different geographic-regions.

References

1. Vöckler, J.S., Juve, G., Deelman, E., et al.: Experiences using cloud computing for a scientific workflow application, pp. 15–24. International Workshop on Scientific Cloud, Computing (2011)
2. Zhu, Z., Zhang, G., Li, M., et al.: Evolutionary multi-objective workflow scheduling in cloud. IEEE Trans. Parallel Distrib. Syst. **27**(5), 1344–1357 (2016)
3. Juve, G., Chervenak, A., Deelman, E., et al.: Characterizing and profiling scientific workflows. Future Gener. Comput. Syst. **29**(3), 682–692 (2013)
4. Buyya, R., Yeo, C.S., Venugopal, S., et al.: Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. **25**(6), 599–616 (2009)
5. Foster, I., Zhao, Y., Raicu, I., et al.: Cloud computing and grid computing 360-degree compared. Grid Computing Environments Workshop, 2008. GCE '08, pp. 1–10. IEEE, New York (2009)
6. Wang, X., Yeo, C.S., Buyya, R., et al.: Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm. Future Gener. Comput. Syst. **27**(8), 1124–1134 (2011)
7. Deb, K., Pratap, A., Agarwal, S., et al.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
8. Liu, L., Zhang, M.: Multi-objective optimization model with AHP decision-making for Cloud service composition. Ksii Trans. Internet Inform. Syst. **9**(9), 3293–3311 (2015)
9. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **11**(6), 712–731 (2008)
10. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Springer, Berlin (2001)
11. Coello, C.A.C., Pulido, G.T., Lechuga, M.S.: Handling multiple objectives with particle swarm optimization. IEEE Trans. Evol. Comput. **8**(3), 256–279 (2004)
12. Rodriguez, M.A., Buyya, R.: Deadline based resource provisioning and scheduling algorithm for scientific workflows on Clouds. IEEE Trans. Cloud Comput. **2**(2), 222–235 (2014)
13. Malawski, M., Juve, G., Deelman, E., et al.: Cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds. In: SC '12 Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, pp. 1–18 (2012)
14. Liu, L., Zhang, M., Lin, Y., et al.: A survey on workflow management and scheduling in cloud computing. IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 837–846. ACM, New York (2014)
15. Deng, K., Ren, K., Zhu, M., et al.: A data and task co-scheduling algorithm for scientific Cloud workflows. IEEE Trans. Cloud Comput. **7161**, 1 (2015)
16. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co. Inc., Boston (1989)
17. Zhang, J., Chung, S.H., Lo, W.L.: Clustering-based adaptive crossover and mutation probabilities for genetic algorithms. IEEE Trans. Evol. Comput. **11**(3), 326–335 (2007)
18. Hu, W., Yen, G.G.: Adaptive multiobjective particle swarm optimization based on parallel cell coordinate system. IEEE Trans. Evol. Comput. **19**(1), 1–18 (2015)
19. Inselberg, A.: The plane with parallel coordinates. Visual Comput. **1**(2), 69–91 (1985)
20. Hu, W., Yen, G.G., Zhang, X.: Multiobjective particle swarm optimization based on Pareto entropy. J. Softw. **25**(5), 1025–1050 (2014). (in Chinese)
21. Toombs, R., Reed, J., Barricelli, N.A.: Simulation of biological evolution and machine learning. J. Theor. Biol. **17**(3), 319 (1967)
22. Zhi-Hui, Z., Jun, Z., Yun, L., et al.: Adaptive particle swarm optimization. IEEE Trans. Syst. Man Cybern. Part B **39**(6), 1362–81 (2009)
23. Calheiros, R.N., Ranjan, R., Beloglazov, A., et al.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp. **41**(1), 23–50 (2011)
24. Chen, W., Deelman, E.: WorkflowSim: a toolkit for simulating scientific workflows in distributed environments. In: IEEE International Conference on E-Science. IEEE, pp. 1–8 (2012)
25. Nanakorn, P., Meesomklin, K.: An adaptive penalty function in genetic algorithms for structural design optimization. Comput. Struct. **79**(29–30), 2527–2539 (2001)

26. Tessema, B., Yen, G.G.: An adaptive penalty formulation for constrained evolutionary optimization. IEEE Trans. Syst. Man Cybern. Part A **39**(3), 565–578 (2009)
27. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. IEEE Trans. Evol. Comput. **18**(4), 577–601 (2014)
28. Zhang, Q., Zhou, A., Zhao, S., et al.: Multiobjective optimization test instances for the CEC 2009 special session and competition. Technical Report, University of Essex (2008)
29. Deb, K., Sinha, A., Kukkonen, S.: Multi-objective test problems, linkages, and evolutionary methodologies. In: Genetic and Evolutionary Computation Conference, GECCO 2006, Proceedings, Seattle, Washington, USA, pp. 1141–1148 (2006)
30. Deb, K., Kalyanmoy, D.: Multi-Objective Optimization Using Evolutionary Algorithms. Wiley, New York (2001)
31. He, Q., Wang, L.: An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng. Appl. Artif. Intell. **20**(1), 89–99 (2007)
32. Topcuoglu, H., Hariri, S., Wu, M.Y.: Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. **13**(3), 260–274 (2002)
33. Zheng, W., Sakellariou, R.: Budget-deadline constrained workflow planning for admission control. J. Grid Comput. **11**(4), 105–119 (2013)
34. Arabnejad, H., Barbosa, J.G.: A budget constrained scheduling algorithm for workflow applications. J. Grid Comput. **12**(4), 15 (2014)
35. Durillo, J.J., Prodan, R.: Multi-objective workflow scheduling in amazon ec2. Cluster Comput. **17**(2), 169–189 (2014)
36. Sawant, S.: A genetic algorithm scheduling approach for virtual machine resources in a Cloud computing environment. Master thesis, San Jose State University (2011)
37. Garg, R., Singh, A.K.: multi-objective workflow grid scheduling based on discrete particle swarm optimization. In: International Conference, Semcco, Visakhapatnam, Andhra Pradesh, India, December, pp. 183–190 (2011)
38. Garg, R., Singh, A.K.: Multi-objective workflow grid scheduling using ϵ-fuzzy dominance sort based discrete particle swarm optimization. J. Supercomput. **68**(2), 709–732 (2014)
39. Yu, J., Kirley, M., Buyya, R.: Multi-objective planning for workflow execution on Grids. In: IEEE/ACM International Conference on Grid Computing. IEEE Computer Society, pp. 10–17 (2007)
40. Wang, X., Xia, JW., Li, JZ., et al.: Grid workflow scheduling with various QoS constraints using SPEA2+. In: Fourth International Conference on Genetic and Evolutionary Computing. IEEE Xplore, pp. 829–832 (2011)
41. Bader, J., Zitzler, E.: HypE: an algorithm for fast hypervolume-based many-objective optimization. Evol. Comput. **19**(1), 45–76 (2011)