**User-Oriented Many-Objective Cloud Workflow Scheduling**

**Based on an Improved Knee Point Driven Evolutionary Algorithm**

**Xin Yea, Sihao Liua, Yanli Yina, Yaochu Jina, b\***

aInstitute of Information and Decision Technology, Dalian University of Technology, Dalian, Liaoning Provence, P.R. China

b Department of Computer Science, University of Surrey, Guildford, GU2 7XH, UK

**Abstract:** Cloud computing is able to deliver large amount of computing resources on demand, and it has become one of the most effective ways to implement large-scale computationally intensive applications. In a cloud computing environment, applications typically involve workflows. Therefore, optimized workflow scheduling can greatly improve the overall performance of cloud computing. However, existing studies on cloud workflow scheduling usually consider at most three objectives only and effective methods to solve scheduling problems with four or more objectives still lack. To address the above issue, a new cloud workflow scheduling model is formulated that simultaneously considers four objectives, namely, minimization of makespan, minimization of the average execution time of all workflow instances, maximization of reliability, and minimization of the cost of workflow execution. To solve this four-objective scheduling problem, an improved knee point driven evolutionary algorithm is proposed. Extensive experimental results demonstrate that the improved algorithm outperforms existing popular many-objective evolutionary algorithms in most experimental scenarios studied in this work, in particular when there is sufficiently large amount of computing resource supply and the time for scheduling is limited.

**Keywords:** Cloud computing, cloud workflow scheduling, many-objective optimization problems, knee point driven evolutionary algorithm

**1. Introduction**

With the ubiquitous growth of the Internet, cloud computing has proliferated in a number of fields [1]. Most cloud computing applications are composed of multiple tasks, and there are dependency relationships among tasks. These tasks constitute a workflow, called the cloud workflow [2]. Cloud workflow scheduling aims to allocate cloud workflow tasks to the resource nodes (virtual machines) in the cloud computing environment with specific constraints to obtain the optimal scheduling result. This problem has become a hot topic in the field of cloud computing and has received increasing attention from both academia and industry.

However, most existing studies deal with cloud workflow scheduling as a single-, bi-, or three-objective optimization problems by either making unrealistic assumptions that cannot be met in many application scenarios, or neglecting some important requirements of the service provider or the users [3-5]. For example, existing studies typically minimize execution time or execution costs only, without taking into account other scheduling objectives such as reliability. In addition, time-related objectives usually consider minimizing the makespan, which is insufficient in dealing with many online application systems, in particular when the quantity of cloud workflow instances is large. In this situation, some workflow instances may start to execute at an earlier time, but their follow-up tasks may finish at a later time. This scenario does not necessarily lead to an increment in the total completion time of all workflow instances, but will result in a longer average completion time of instances, which will unfavorably affect user experience. Further, most studies do not explicitly perform scheduling for user quality of service (QoS) or scheduling for provider efficiency. Therefore, it is highly desirable to formulate cloud workflow scheduling as a many-objective optimization problem taking into account the requirements from service provider or users.

Many heuristic [3, 6-9] or meta-heuristic algorithms, such as evolutionary algorithms, have been used to solve single- or bi-objective scheduling optimization problems [4, 5]. The elitist non-dominated sorting genetic algorithm, known as NSGA-II [10,11], is one popular evolutionary algorithm for solving bi-objective cloud scheduling problems. However, little work has been done on cloud workflow scheduling considering more than three objectives, which is known as many-objective optimization problems (MaOPs). It has been recognized that most multi-objective evolutionary algorithms (MOEAs) developed for solving bi- or three-objective optimization problems are usually inefficient for solving MaOPs [12].

This paper aims to address the above-discussed issues in both problem formulation and algorithm development in cloud workflow scheduling in the following aspects. First, a new cloud workflow scheduling model is established to minimize the makespan, the average execution time and the execution cost of the cloud workflow instances, and to maximize the reliability of the cloud workflow execution. As this model takes into account users’ requirements for cloud workflow scheduling, it is known as the user-oriented approach [1] and is more practical. Second, based on knee point driven evolutionary algorithm (KnEA) [12], an improved knee point driven evolutionary algorithm (IKnEA) is proposed to more efficiently solve the four-objective cloud workflow scheduling problem. KnEA is a recently developed evolutionary algorithm for solving MaOPs, which has shown to be able to outperform several state-of-the-art MOEAs for MaOPs in terms of two widely used performance indicators, hypervolume (HV) [13] and inverted generational distance (IGD) [14]. However, it is found that in choosing non-knee points, KnEA is likely to select solutions that have a larger distance to the hyperplane while contributing less to the increment in HV. To further improve the performance of KnEA, we propose two new strategies to improve the environmental selection in KnEA by introducing entropy weight [15] and TOPSIS (Technique for Order Preference by Similarity to an Ideal Solution) [16]. Finally, extensive simulations are conducted to compare the proposed algorithm with other representative algorithms on a number of cloud scheduling scenarios. Our simulation results show that the IKnEA is able to enhance the search efficiency and solution quality in particular in scenarios when the available computing resource supply is large.

The rest of this paper is organized as follows. In Section 2, related work on cloud workflow scheduling is briefly reviewed. A user-oriented many-objective cloud workflow scheduling model is proposed in Section 3. In Section 4, the details of the IKnEA algorithm is given. Simulation results are presented in Section 5 to demonstrate the effectiveness of the IKnEA. Finally, conclusions and future work are provided in Section 6.

**2. Related work**

**2.1 Workflow types**

Two main types of workflows in cloud workflow scheduling have been investigated. The first type that has been widely investigated is scientific workflows [6-9,18] typically involve large amount of data analysis and computing, requiring high computing power and large storage capacity. The second type of workflows is business workflows in business and government information systems [5, 7, 17, 19, 20]. Unlike scientific workflows, activities in business workflows require lower computing power and less storage resources, while there are often a large number of concurrently running workflow instances and the communication time between tasks needs to be considered.

**2.2 Scheduling objectives and constraints**

As we previously discussed, most existing studies aim to minimize the execution time [21] or minimize the user cost [22]. Other objectives that have been taken into account include security [23], load balancing [24], resource utilization [4], or energy consumption [25]. Reliability is another important factor influencing the service quality in cloud workflow scheduling. Both [9] and [26] consider reliability as one objective of scheduling. For a cloud workflow scheduling, some objectives, such as makespan, user costs and reliability, are considered in scheduling for user QoS, while other objectives, such as load balancing, resource utilization and energy consumption, are considered in scheduling for provider efficiency [1].

According to the number of objectives, cloud workflow scheduling can be divided into three categories as follows. First, most studies consider a single objective, such as minimizing execution time [3, 4], or minimizing execution costs [5]. Second, a few studies have tackled cloud workflow scheduling as bi- or three-objective optimization problems, such as makespan and cost [11], makespan and energy consumption[8, 17], makespan, cost and reliability [18]. Third, a small number of studies have considered four or more objectives [9, 27]. A priori multi-objective scheduling method, namely, multi-objective list scheduling (MOLS) is proposed for cloud workflow scheduling, considering makespan, cost, reliability and energy consumption [9]. Four conflicting objectives, including minimizing task transfer time, task execution cost, energy consumption and task queue length were considered [27] to develop a comprehensive multi-objective optimization model for cloud workflow scheduling, which was optimized using two multi-objective evolutionary algorithms, a multi-objective particle swarm optimization algorithm and a multi-objective genetic algorithm. Results in [9] and [27] were optimized for energy minimization, which however, are not optimal for users, as there are conflicts between requirements from users and providers in cloud workflow scheduling [1]. As a result, such formulations lead to worse optimal solutions than those focusing only on user or provider requirements.

Two types of scheduling constraints are typically considered, i.e., QoS constraints and resource constraints. QoS constraints generally include time, cost, and reliability, whereas resource constraints include bandwidth and storage capacity [1].

**2.3 Scheduling algorithms**

Heuristic, meta-heuristic and hybrid algorithms have widely been used in cloud workflow scheduling. Heuristic algorithms are relatively simple and computationally efficient but are usually not able to obtain global optimal solutions or even satisfactory solutions [20]. Meta-heuristic algorithms mainly include genetic algorithm (GA) [24, 28], particle swarm optimization (PSO) [29], ant colony algorithm [30], among others. Since cloud workflow scheduling problems are NP-hard, meta-heuristic algorithms are often used to improve the performance. In addition, hybrid algorithms [21, 31] have shown to be more powerful as they combine heuristic rules with meta-heuristic algorithms.

Most meta-heuristic and hybrid algorithms are based on MOEAs developed for MOPs, which are very ineffective in solving cloud workflow scheduling problems having more than three objectives [12]. Recently, many MOEAs for MaOPs are proposed, such as KnEA [12], NSGA-III [32], a new variant of NSGA-II, grid-based evolutionary algorithm (GrEA) [33], and reference vector based evolutionary algorithm (RVEA) [34].

**3 A New Formulation for Cloud Workflow Scheduling**

**3.1 Problem description**

The cloud workflow scheduling problem considered in this paper is illustrated in Fig. 1.



**Fig. 1.** An illustration of a cloud workflow scheduling problem

In this problem, *n* workflow instances （*I*1, *I*2…, *In*） need to be executed on *k* virtual machines （*VM*1, *VM*2…*VMk*）, and each workflow instance consists of *m* tasks （*T*1,*T*2 ,…*Tm*） with a certain dependency relationship. Each workflow task can be executed on any virtual machine that satisfies the constraint. Cloud workflow scheduling is used to select the appropriate virtual machine for all tasks of all workflow instances and to determine the optimal execution order and start time for each task on each virtual machine. Based on the actual needs of the users, four objectives are considered in this paper, including minimizing makespan, minimizing all task execution cost, minimizing the average execution time of each workflow instance, and minimizing average failure rate of all workflow instances(maximizing reliability). In addition to the dependency relationship between tasks, this paper also considers the task security constraint. That is to say, tasks can only be assigned to a virtual machine that has a higher security level than required security level of the task.

**3.2 Workflow model**

A workflow can be modeled as a directed acyclic graph *DAG=*(*Node, Edge*), as shown in Fig. 2. *Node*={ *T1*,*T2* ,…*Tm* } represents the set of tasks in the workflow model, and *Edge* represents the set of dependency relationships between tasks. *Edgeij* indicates that *Tj* starts to execute only after receiving the required data from its preceding task *Ti*. *Tentry* represents an entry task without any predecessors, such as *T1*. *Texit* represents an exit task without any successors, such as *T8*.



**Fig. 2.** A workflow example

**3.3 Reliability of workflow execution**

The reliability of workflow execution refers to the possibility that all tasks are executed successfully, which is an important indicator for evaluating the performance of cloud workflow scheduling. Based on critical path method (CPM), if the execution time of tasks on the critical path is delayed, it will result in a delay in the execution time of the entire workflow instance. While the execution time of a task on a non-critical path is delayed within a certain range, the execution time of the whole workflow instance will not be delayed. Therefore, tasks on the critical path should be assigned to highly reliable virtual machines to avoid increasing the execution time of workflow instances resulting from rescheduling failed tasks. Thus, a slack time *slackj* of task *Tj* is defined as follows.

*Slackj=LSj-ESj = LFj-EFj* (1)

where, *ESj*,*，LSj* , *EFj*，*LFj* are the earliest start time, latest start time, the earliest finish time and the latest finish time of *Tj*, respectively. In addition, *slackj*=0 indicates that the corresponding task *Tj* is located on the critical path. Otherwise, the task with slack>0 is not located on the critical path. Combining the reliability of task proposed in [24] with the *slack*, the reliability *Reljk* of task *Tj* executed on virtual machine *VMk* can be defined as follows:

 (2)

where*TMjk* represents the execution time of task *Tj* on virtual machine *VMk*, *pk* is the failure rate of *VMk* and penalty coefficientis a real number greater than 1 and its value depends on the specific problem. Unlike the reliability definition in [9], in Equation (2), the reliability of tasks on the critical path and the non-critical path are distinguished. In addition,  is introduced in the calculation of the reliability, so that the tasks on the critical path could not be assigned to the virtual machine with low reliability. Note that assigning tasks on the critical path (*slackj*=0) to virtual machines with higher reliability (low *pk* value) can reduce the task execution failure rate, thus reducing the possibility of delay completion of the workflow instance. Finally, the reliability of the whole workflow instance *Ij* is defined by

 (3)

where *n* is the tasks’ number of workflow instance *Ij*.

**3.4 Problem formulation**

The problem considered in this paper makes the following assumptions.

(1) If a task is assigned to a virtual machine, the task should be executed immediately when the required data have transferred to the virtual machine and the virtual machine is in the idle state.

(2) Each task of each workflow instance can only be executed on one virtual machine. A virtual machine is not able to execute more than one task concurrently.

(3) If a task begins to execute, it cannot be interrupted until the end of its execution.

Symbols involved in the cloud workflow scheduling problem discussed in this paper are summarized in Table 1.

**Table 1.** Definition of symbols involved in a cloud workflow scheduling problem

|  |  |
| --- | --- |
| Symbols | Meanings |
| *IiTj* | Task *j* of workflow instance *i* |
|  | Communication time between *Ti* and *Tj*; if they are executed on different hosts and there are data transfer between them, > 0; |
|  | Execution time of task *j* executed on virtual machine *k* |
|  | Time when *VMk* is available for *IiTj* |
|  | Time when required data of *IiTj* is available at *VMk* |
|  | Start time of *IiTj* executed on *VMk* |
|  | Finish time of *IiTj* executed on *VMk* |
|  | Binary variable, if= 1, *IiTj* will be executed on *VMk* |
|  | Required security level of *Tj* |
|  | Security level of *VMk* |
|  | Cost of task *Tj* executed on *VMk* per time unit |
|  | Finish time of last taskon *VMk* |
| *CTi* | Finish time of *Ii* |
|  | Communication cost per time unit |

According to the above assumptions, the cloud workflow scheduling problem can be formulated as follows:

 (4)

 (5)

 (6)

 (7)



Where, expressions from (4) to (7) represent four objectives, namely minimizing makespan, minimizing the average execution time of each workflow instance, minimizing all task execution cost, and minimizing average failure rate of all workflow instances(maximizing reliability), respectively. *J* represents the preceding task set of task *Tj* and the constraints are defined in expressions from (8) to (14). Equation (8) defines that *DATijk* is the maximum value of the sum of the communication time and completion time of all preceding tasks of *IiTj*. When the scheduling task is the entry task of a workflow instance, *DATijk* =0. Equation (9) represents the time when the virtual machine *VMk* is available for the execution of *IiTj* , and *FTi’’j’’k*represents the finish time of task *I i’’Tj’’* that is the last task executed before *IiTj* on *VMk*. In Equations (10) and (11), *STijk and FTijk* represent the start time and finish time, respectively, of *IiTj* executed on virtual machine *VMk*. Inequality constraint in expression (12) indicates tasks can only be assigned to virtual machine, which has a higher security level than that required by the task. Equations (13) and (14) mean that a task can only be allocated to one virtual machine and can be executed only once.

**4. Improved KnEA for Many-objective Cloud Workflow Scheduling**

Since most multi-objective evolutionary algorithms developed for solving bi- or three-objective optimization problems are usually inefficient when the number of objectives is more than three, which are often known as MaOPs[12]. And KnEA is a recently developed evolutionary algorithm for solving MaOPs, which has shown to be able to outperform several state-of-the-art MOEAs for MaOPs in terms of HV and IGD. To more efficiently solve the four-objective cloud workflow scheduling problem proposed in section 3, an improved KnEA (IKnEA) is proposed.

**4.1 The framework of the proposed algorithm**

The framework of the improved KnEA (IKnEA) for many-objective cloud workflow scheduling is described as Algorithm 1.

|  |
| --- |
| **Algorithm 1** Framework of IKnEA |
| **Require:** *P* (population), *N* (population size), *K* (set of knee points), *T* (rate of knee points in population) , *e* (rate of individuals selected by SNK in population), *π*(rate of individuals selected by SBV in population), O(number of objectives)  1: *r* ← 1, *t* ← 0 /\*adaptive parameters for finding knee points\*/  2: *K* ←  3: *P* ← *Initialize*(*N*) /\*all solutions are generated randomly on the premise that each individual satisfies all  constraints \*/  4: **while** termination criterion not fulfilled **do**  5: *P’*← *Mating*\_*selection*(*P,K,N*)  6: *P* ← *P*∪*Variation*(*P’*,*N*)  7: *W*←calculate entropy weight of each objective by *P*  8: *F* ← *Nondominated*\_*sort*(*P*)/\*find the solutions in the first *i* fronts *Fj*, 1 ≤ *j* ≤ *i*, where *i* is the minimal value such that |*F*1 ∪ *. . .* ∪ *Fi*| >= *N* \*/  9: [*K, r, t*] ← *Finding*\_ *knee*\_ *point*(*F, T, r, t*)  10: *P* ← *Environmental*\_*selection*(*F,K,N,P,W,O,e, π*)  11:**end while**  12:**return** *P* |

In Algorithm 1, the underlined part is the main improvement of the proposed algorithm, which can be divided into two categories as follows. On one hand, the population initialization, the crossover and mutation operators are designed based on the characteristics of the cloud workflow scheduling problem (Algorithm 1, line 3, 5-6), which can be seen in Section 4.2. On the other hand, the environment selection is improved (Algorithm 1, line 10) by proposing two sorting and selection strategies, which is the key component of improvement and will be discussed in greater detail Section 4.3.

**4.2 Algorithm details**

**4.2.1 Population initialization**

The initial population is formed by *N* scheduling schemes generated randomly under the condition that each individual satisfies all constraints (refer to Algorithm 1, line 3). According to the characteristics of a cloud workflow scheduling problem, integer encoding is adopted. Each individual (chromosome) represents a scheduling scheme. The first half of the chrome represents the execution sequence of tasks of workflow instances (task region), and the second half represents the set of virtual machine ID that is assigned to the task in the corresponding position of first half (VM region). Fig. 3 provides an example chromosome representing a candidate solution, where numbers in the first half of the chromosome mean *I1T1*, *I1T2 ,I1T3* *I2T1 I2T2*, *I2T3*, and so on the in sequence. The second half indicates that *I1T1* is executed on *VM1*, *I1T2* is executed on *VM2*, and so on.



**Fig. 3**. An example individual

Before crossover and mutation, a binary tournament selection is required to obtain a population *P’* containing the best *N* individuals of the parents (Algorithm 1, line 5). The binary tournament selection strategy is the same as the one used in KnEA, which uses three tournament strategies, namely, dominance comparison, knee point criterion and a weighted distance. Then, crossover and mutation are performed on *P’* to generate offspring (Algorithm 1, line 6).

**4.2.2 Crossover**

Single-point crossover is performed on the chromosome. First, two chromosomes are randomly selected from *P’*. Then, a crossover point is randomly determined. Further, sequences beyond the point in the task region of each of the selected chromosome are swapped. Similarly, sequences beyond the point in VM region of either chromosome are swapped. Thus, it can ensure that crossover only changes the order of task execution without changing task allocation of the virtual machine. After the crossover, the total number of tasks of each workflow instance may be incorrect, and therefore repair operations are carried out to those infeasible solutions to make sure that each workflow instance has correct total number of tasks in each chromosome.

**4.2.3 Mutation**

The mutation operator randomly changes genes in the VM region at a certain probability. The task region remains unchanged. That is, the task assignment of the virtual machine is changed while maintaining the order of task execution. Moreover, in mutation, the task should be only assigned to the virtual machine that satisfying constrains such as security constraint.

**4.3 Improvement in environmental selection of KnEA**

After crossover and mutation, environmental selection is performed on a combination of offspring and parent population to generate the next population in KnEA. The major improvement of the proposed algorithm concerns the environmental selection.

KnEA’s environmental selection contains the following main operations. First, non-dominated sorting is performed on the combination of the parent and offspring population and a number of non-dominated fronts are obtained. Then, a percentage of the combined population are identified as knee points according to the distance of the solution to the hyperplane constructed based on the end solutions and adaptive neighborhood strategy (Algorithm 1, line 8-9). Further, KnEA selects solutions on the basis of the following three criteria: (1) Solutions in the front with smaller number are preferred. (2) Knee points are more preferred than non-knee points. (3) Solutions with a larger distance to the hyperplane are preferred. Among the three criteria, criterion 1 has the highest priority and criterion 3 has the lowest. When the number of selected candidate solutions according to the the first and second criteria is smaller than population size, KnEA use criterion 3 to select other solutions.

However, it is found that among solutions selected according to criterion 3, those having a larger distance to the hyperplane do not necessarily contribute more to the HV. Therefore, a key to further improving the quality of the selected solution set is make sure that solutions that can contribute more to HV are selected even if their distance to the hyperplane is small. To this end, the following two ideas are proposed. First, by combining the entropy weight and TOPSIS in the field of multi-criteria decision-making [15, 35-37], non-knee point solutions are sorted and selected. Second, a certain number of solutions with better values in each objective are selected. The above two improvements can be used as the complementary selection mechanisms in KnEA’s environmental selection to improve the quality of the solution set. In the following, we describe the details of the two ideas.

**4.3.1** **Sorting and selection strategy of non-knee point solutions based on Entropy Weight and TOPSIS**

In this section, a sorting and selection strategy of non-knee point solutions based on entropy weight and TOPSIS, termed SNK, is proposed to improve the environmental selection of KnEA. Entropy weight and TOPSIS have been used widely and successfully in many scenarios of multi-criteria decision making [15, 35-37]. Entropy is a measure of system chaos, and entropy weight can be used to determine the weight of an objective by calculating the entropy function. When the solution set (the evaluation object) is more discrete on objective *j* (evaluation indicator), the entropy of objective *j* is smaller and the corresponding entropy weight is larger. It is shown that for this solution set, objective *j* provides more useful information to decision makers and should be prioritized. TOPSIS evaluates an evaluation object by its distance to the optimal solution and the worst solution. The closer to the optimal solution and the farther away from the worst solution at the same time, the better the evaluation object is. Compared with the method of evaluating the solution by distance from either the optimal solution or the worst solution, entropy weight and TOPSIS have been shown effective in multi-criteria decision-making problems without decision-maker preference [15, 35-37]. For cloud workflow scheduling problem addressed in this paper, if the decision maker has no preference, these two methods can be employed to obtain satisfactory solutions. Fig. 4 (a) show example solutions that can be chosen by the entropy weight and the TOPSIS method that will not be chosen based on the distance to the hyperplane.

Since criteria (1) and (2) in KnEA’s environmental selection are effective, we employ SNK only to select a few non-knee points based on the entropy weight and the TOPSIS method. Firstly, the entropy weight of each objective is calculated according to Expressions (15)-(17).

 (15)

 (16)

 (17)

Where, *zij*is the normalized *i-th* objective value of solution *j*, and *N* is the number of solutions. *Hi* and *ωi* denotes the entropy value and the entropy weight of the *i-th* objective, respectively.

Then, SNK uses the following evaluation indicator *D* to evaluate solutions, which combines the advantages of these two methods.

 (18)

where is the *D* value of solution *j*, and *M* is the number of objectives. refers to the ideal point and  represents the normalized negative ideal point. , and the greater is, the better solution *j* is.

In Fig. 4, *f*1 and *f*2 are two objectives, and P is reference point. The filled black dots denote knee points identified by KnEA, and the boxes in dashed lines are the neighborhoods of the knee points defined by KnEA. Other points in each neighborhood are non-knee points, e.g., both points A and B. L1 is the hyperplane. L2 and L3 are lines parallel to the hyperplane. Let *dA* be the distance from point A to the hyperplane L1, and *dA* equals the distance from L2 to L1. Similarly, let *dB* be the distance from point B to the hyperplane L1, and *dB* equals the distance from L3 to L1. When point A or B is selected, the HV value of the population in Fig. 4 is denoted as *HVA* and *HVB*, and corresponding shaded boxes *SA* and *SB* indicate the increase of HV. From Fig. 4, we can see that *SA* > *SB*.



**Fig. 4.** Illustrations that SNK may select solutions with larger HV than original KnEA

According to Equation (18) and the definition of HV, we can get *DA*=0.579651, *DB*=0.554296, *HVA*=0.55744 and *HVB*=0.552863. In this scenario, KnEA selects point B because *dA* < *dB*. By contrast, point A is selected by using the strategy proposed in this work because of *DA* >*DB*. It can be easily found that selecting point A is more beneficial than selecting point B in terms of HV. Thus, the example shown in Fig. 4 shows that KnEA selects the non-knee point B with a larger distance to the hyperplane, which results in a smaller increase of HV. Based on SNK, we can select the non- knee point with a larger HV increase even if its distance to hyperplane is smaller.

**4.3.2** **Selection solutions with better values in each objective based on the entropy weight**

To further improve the selection strategy in KnEA, we propose to select solutions with better values on each objective based on the entropy weight (SBV). The reasons behind this strategy are as follows.

(1) The current better values of each objective may be further optimized using crossover and mutation operations in the subsequent evolutionary process, thus increasing diversity.

(2) Solutions with a better value on one objective are preserved because the value on other objectives of those solutions are more likely to be further optimized, assuming that the better value on this objective does not deteriorate. Selecting those solutions is expected to increase convergence to the Pareto front.

(3) In some scenarios, solutions with better values on all objectives have a larger HV increase than those with a larger distance from the hyperplane, as illustrated in Fig. 5, where the symbols are the same as those in Fig. 4, and both points A and B are non-knee points. Solution A has a better value on objective *f*1 and L is the hyperplane. Selecting solution A will result in a larger HV increase (*SA*) than selecting B (*SB*), although the distance from solution A to the hyperplane is smaller than solution B.



**Fig. 5**. Illustrations that SBV may select solutions with larger HV than original KnEA

In addition, in environmental selection, SBV prefers selecting more solutions with a better value on an objective with a smaller entropy weight. The main reason for this operation is described as follows. Since the objective values are densely distributed along the objective with a smaller entropy weight and sparsely distributed along the objective with a larger entropy weight, solutions with a better value on an objective with a smaller entropy weight have a greater probability to obtain solutions with a slightly worse value on this objective while considerably improving other objectives in a subsequent evolutionary process. Fig 6 provides an illustrative example, where, *f*1 and *f*2 are two objectives, and the entropy weight of *f*1 is larger than that of *f*2. Solutions *A*, *B*, *X*1 and *X*2 are non-dominated solutions, among which A and B have the best value on objective *f*1 and *f*2, respectively, and *X*1 and *X*2 are solutions have the second best value on objective *f*1 and *f*2, respectively. Let *f*1(*X*2) be the value on *f*1 of *X*2, *f*2(*X*1) be the value on *f*2 of *X*1,. min *f*1 and min *f*2 be the best value on *f*1 and *f*2, respectively. Then, ∆1= *f*1 (*X*2)-min*f*1, ∆2= *f*2(*X*1)-min*f*2. From the figure, the area of *S*1 represents the probability that *X*1 deteriorates by up to *d* on *f*1 while improves by up to ∆2 on *f*2. Similarly, the area of *S*2 represents that *X*2 deteriorates by up to *d* on *f*2 while improves by up to ∆1 on *f*1 . Where, *S*1=∆2\**d* and *S*2=∆1\**d* . Since *f*1 has a larger entropy weight then *f*2, the objective value on *f*1 is more sparsely distributed than *f*2, causing ∆1>∆2. Therefore, *S*1<*S*2. So comparing with *X*1, *X*2 has a larger possibility to be further optimized on *f*1 on the premise that better value on *f*2 deteriorate for a given degree *d.*



**Fig. 6** An illustrative example showing that SBV select more solutions with a better value in smaller entropy weight objective

In summary, SBV retains a total number of *Nm*=*N* \**π* solutions with a better value in each objective according to a certain percentage π in the environmental selection, where *N* denotes the population size. The number of solutions with better value on the *i-th* objective  is determined by the entropy weight of the objective. The larger the entropy weight of an objective is, the smaller the number of solutions with a better value on this objective.  can be calculated as Equation (19).

 (19)

where *ωi* is the entropy weight of *i-th* objective andis the total number of objectives.

**4.3.3 Improved environmental selection process**

The following operations need to be performed before the improved environmental selection strategy can be implemented. First, the entropy weight *ω* of each objective is calculated according to Equations (15)-(17) in Section 4.3.1. Second, a non-dominated sorting is conducted, and *NF*non-dominated fronts areformed, where *Fi* represents *i*-th front and 1 ≤ *i* ≤ *NF*. Then, the adaptive strategy proposed in KnEA is used to identify knee points in each non-dominated front. All above operations are performed on the combination of parent and offspring population obtained after crossover and mutation.

Based on the strategies proposed in Section 4.4.1 (SNK, described in Algorithm 4), 4.4.2 (SBV, described in Algorithm 3) and KnEA’s environmental selection, the environmental selection process of the IKnEA is presented in Algorithm 2.

**Algorithm 2** Environmental selection process of the IKnEA

|  |
| --- |
| **Require:** *F* (sorted population), *K* (set of knee points), *N* (population size) ,*W*(entropy weight set of objectives), *P* (population),*O*(number of objectives), *e* (rate of individuals selected by SNK in population), *π*(rate of individuals selected by SBV in population)  1: *Q* ←  /\*next population\*/  2:*M*←*SBV (P,W,N,O,π,)*  3: **if** *F*1 > *N* **then** /\* when *F*1 > *N* , *N*\**e non-knee points who have maximum D value calculated by Equation (15) will be chosen* *according to the strategy* SNK *proposed in section 4.3.1*\*/  4: *N’*←*N\*(1-e)*  5: **else**  6: *N’*←*N*  7: **end if**  8: *Q* ← *M* *F*1 *. . .* *Fj*−1 // where *j* is the minimal value such that | *M* *F*1*. . .*  *Fj*|> *N’*  9: *Q* ← *Q*( *K**Fj* )  10: **if** |*Q*| *> N’* **then**  11: delete |*Q*|− *N’* solutions from *Q* which belong to *K**Fi* and have the minimum distances to the hyperplane  12: **else if** |*Q*| *< N’* **then**  13: add *N’* −|*Q*| solutions from *Fi****＼****(K* *Fj)* to *Q* which have the maximum distances to the hyperplane  14: **end if**  15: **if**|*Q*|*< N* **then**  16: *E*←*SNK(F, Q ,W,N);*  17: *Q*←*E**Q*  18: **end if**  29: **return** *Q* |

At the beginning of the environmental selection,  solutions with a better value in each object are selected by SBV. Algorithm 3 describes the details of SBV. First, based on entropy weight, the number of solutions with a better value in the *i-th* objective  is calculated using (19) (Algorithm 3, line 4). Then, the population, which combines the parent population with the population after mating selection and variation (Algorithm 1, line 5, 6) is sorted according to each objective value, and  with a better value solutions in *i-th* objective are selected to be added to the next generation *Q* (Algorithm 3, line 5-7).

Next, the IKnEA continues to use KnEA's knee points-driven environmental selection to select additional solutions. The number of combinations of solutions selected by SBV and KnEA’s environmental selection is denoted as *N’*. It is very likely that already in the early generations of MaOPs the number of solutions in the first non-dominated front *F*1 is larger than population size *N* [12]. Let = (Algorithm 2, line 3-4), which means that the IKnEA adopts the SNK to select  non-knee points in the final stage of environmental selection. Otherwise, all solutions in next generation are selected by SBV and KnEA’s environmental selection. Since the threshold *T* that controls the ratio of knee points in each front *Fi* is set to T = 0.5, the number of knee points identified by KnEA will not exceed *N/2.* Letthe value of *e* be far less than 0.5, which ensures that the SNK strategy is only used to select a certain number of non-knee points. Lines 8-14 in Algorithm 2 describe the process of adopting KnEA’s environmental selection to select solutions, which is the same as KnEA.

After the above operation, if |*Q*| < *N*, SNK is used to select of the remaining *N*－|*Q*| solutions（Algorithm 2, line 16-17）; otherwise, *Q* is the next generation population. Algorithm 4 describes the operation of SNK in detail.

|  |
| --- |
| **Algorithm3** *SBV (P,W,N,O,π,)* |
| **Require:** *P*(population),*N*(population size) ,*W*(entropy weight of each objective),*O*(number of objectives), *π(rate of individuals selected by SBV in population)*  1:*M* ← /\*solutions selected by SBV strategy \*/  2:*i*=0;  3: **while** *i* <= *O* **do**  4: calculate  by formula 16  5: *P*’←Sort *P* by *i*-th Object value in an ascending order  6:  ←Select  solutions which have minimal *i*-th Object value in *P*’  7: *M*←*M*  *M*i  8: *i*++  9: **end while**  10: **Return** *M* |

|  |
| --- |
| **Algorithm 4** *SNK (F,Q,W,N)* |
| **Require:** *F* (sorted population), *N* (population size) ,W(entropy weight set of objectives), *Q* (population selected before algorithm 4 ),  1:**for***s**F1****＼****(Q**F1)* **do**  2: calculate of *s* by formula 15 /\* denotes the D value of solution *s* \*/  3:**end for**  4:add *N* −|*Q*| solutions from *F1****＼***(*Q* *F1*)to *Q* which have the maximum *D* value /\* since|*F*1|> *N* ,all solutions in next population are selected from *F*1 \*/ |

**5. Experiments and Analysis**

**5.1 Design of experiments**

Experiments are designed to examine the performance of the proposed algorithm by comparing it with three popular MOEAs for MaOPs, namely, KnEA [12], NSGA-III [32] and GrEA [33].

The workflow model in this experiment is described in Fig. 2, and each workflow instance consists of 8 tasks. Table 2 lists the time needed for each task to be executed on a standard virtual machine and the required security level of each task. It is noted that the task can only be scheduled to a virtual machine whose security level is not lower than that required by a task. If two tasks *Ti* and *Tj* with communication are assigned to virtual machines on different hosts, the inter-task communication time *αij*between *Ti* and *Tj* is greater than 0. Without loss of generality, let *αij*= 1 and the communication cost per time unit =0.5. Furthermore, let parameter λ = 2 in Equation (2).

**Table 2** Parameters of tasks

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Execution time on a standard virtual machine | 4 | 2 | 8 | 6 | 2 | 4 | 4 | 2 |
| Security level requirements | 1 | 2 | 3 | 3 | 2 | 2 | 3 | 1 |

There are three kinds of virtual machines, and their specifications are given in Table 3. The unit cost of each virtual machine is determined by the execution speed, execution failure rate and the security level. The major factor in determining the virtual machine cost is its execution speed.

Parameter settings for all experimental algorithms are defined in Table 4, where *pc* is crossover rate, *Pm* is mutation rate, *N* is population size, and *iter* is the number of iterations (generations). For fair comparisons, the parameters mentioned above are set to the same for all compared algorithms. In addition, *T* represents the threshold that controls the ratio of knee points in each front *Fi* in both the IKnEA and KnEA. For IKnEA, *e* and *π* are the ratio of solutions selected by SNK and SBV, respectively, which are based on a number of pilot studies conducted on different values of *e* and *π* in all experiment scenarios. For GrEA, the parameter setting for *div* is taken from [26], which stands for the number of divisions in each dimension. *p1* and *p2* are parameters controlling the numbers of reference points along and inside the boundary of the Pareto front in NAGA-III, respectively.

**Table 3.** Parameters of three types of virtual machines

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type of virtual machine | Execution speed | Execution failure rate | Security level | Cost |
| 1 | 1 | 0.008 | 2 | 1.4 |
| 2 | 2 | 0.005 | 3 | 3.2 |
| 3 | 4 | 0.002 | 4 | 6.8 |

**Table 4**. Parameter setup for MOEAs

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| parameters | *pc* | *Pm* | *N* | *iter* | *T* |
| value | 0.8 | 0.02 | 300 | 1000 | 0.5 |
| parameters | *e* | *π* | *div* | (*p1*, *p2*) | |
| value | 0.05 | 0.09 | 9 | (7,0) | |

In real-world cloud workflow scheduling, the number of cloud workflow instances as well as the number of computing resources provided in cloud computing often varies. Thus, the experiments are divided into nine scenarios with respect to the number of workflow instances and the number of resources (virtual machines), and each scenario is denoted by (*n, k*), where (*n, k*) represents that *n* workflow instances can be executed on *k* virtual machines. In this paper, 10, 30 and 50 workflow instances are considered to be executed on 3, 6, 9 virtual machines, and all scenarios are scheduled by IKnEA, KnEA, GrEA, NSGA-III. Then, the HV and IGD, which are two widely used performance indicators, are used to evaluate the performance of the compared algorithms. It is believed that these two performance indicators account not only for convergence but also for the distribution of the achieved non-dominated solutions. Note that the larger the HV value is, the better the performance of the algorithm. By contrast, a smaller IGD value indicates better performance of the MOEA. In this paper, (1, 1, 1, 1) is chosen as the reference point in HV calculation. For the objective values to have the same scale, each of the objective values has been normalized between the interval (0, 1) before calculating the HV. Note also that, IGD is applicable even if the theoretical Pareto optimal solutions are unknown for the real-world cloud workflow scheduling problem proposed in this work. To calculate IGD, for a specific scenario, the combination of population obtained in IKnEA, KnEA, GrEA, NSGA-III in this scenario is sorted, and solutions of the first non-dominated front of the combined population are used as the reference set of Pareto optimal solutions.

Each algorithm is performed 30 times for each scenario, and the average of HV and IGD in 500 and 1000 iterations of these 30 experiments are calculated to compare the performance of the algorithms. Then, t-test was performed to verify if there is statistically significant difference in the average HV and IGD between the IKnEA and compared algorithms.

**5.2 Experimental results**

The average values of HV and IGD obtained from the experiment and the improvement rate of the IKnEA relative to each algorithm are shown in Tables 5-8. The scenarios in which there is not statistically significant difference in the average HV and IGD between the IKnEA and compared algorithms are highlighted.

**Table 5** HV results of four compared algorithms in each scenario

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithm | Iterations | (10,3) | (10,6) | (10,9) | (30,3) | (30,6) | (30,9) | (50,3) | (50,6) | (50,9) |
| KnEA | 500 | 0.0334 | 0.0485 | 0.0558 | 0.0135 | 0.0184 | 0.0198 | 0.0086 | 0.0104 | 0.0111 |
|  | 1000 | 0.0340 | 0.0508 | 0.0604 | 0.0149 | 0.0202 | 0.0222 | 0.0096 | 0.0115 | 0.0124 |
| NSGA-III | 500 | 0.0342 | 0.0436 | 0.0493 | 0.0140 | 0.0170 | 0.0174 | 0.0087 | 0.0093 | 0.0095 |
|  | 1000 | 0.0344 | 0.0412 | 0.0472 | 0.0151 | 0.0180 | 0.0180 | 0.0097 | 0.0100 | 0.0103 |
| GrEA | 500 | 0.0353 | 0.0488 | 0.0539 | 0.0144 | 0.0185 | 0.0193 | 0.0090 | 0.0105 | 0.0108 |
|  | 1000 | 0.0360 | 0.0524 | 0.0587 | 0.0153 | 0.0208 | 0.0221 | 0.0097 | 0.0120 | 0.0126 |
| IKnEA | 500 | 0.0353 | 0.0512 | 0.0564 | 0.0145 | 0.0193 | 0.0202 | 0.0090 | 0.0111 | 0.0113 |
|  | 1000 | 0.0362 | 0.0537 | 0.0603 | 0.0158 | 0.0217 | 0.0226 | 0.0099 | 0.0122 | 0.0129 |

**Table 6** Improvement ratio of IKnEAon HV（%）

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithm | Iterations | (10,3) | (10,6) | (10,9) | (30,3) | (30,6) | (30,9) | (50,3) | (50,6) | (50,9) |
| KnEA | 500 | 5.74 | 5.43 | 1.07 | 7.22 | 4.89 | 2.04 | 5.64 | 6.22 | 2.17 |
|  | 1000 | 6.55 | 5.60 | -0.12 | 5.88 | 7.32 | 1.83 | 3.75 | 6.04 | 4.06 |
| NSGA-III | 500 | 3.01 | 17.28 | 14.42 | 3.96 | 14.05 | 16.14 | 3.61 | 18.81 | 19.54 |
|  | 1000 | 5.35 | 30.44 | 27.74 | 4.98 | 20.24 | 25.27 | 2.36 | 21.43 | 25.75 |
| GrEA | 500 | -0.10 | 4.81 | 4.78 | 0.56 | 4.42 | 4.28 | 0.29 | 5.08 | 4.89 |
|  | 1000 | 0.56 | 2.47 | 2.80 | 3.37 | 4.24 | 2.45 | 2.12 | 1.09 | 2.52 |

**Table 7** IGD results of four compared algorithms in each scenario

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithm | Iterations | (10,3) | (10,6) | (10,9) | (30,3) | (30,6) | (30,9) | (50,3) | (50,6) | (50,9) |
| KnEA | 500 | 7.3808 | 9.4974 | 8.3560 | 12.8180 | 12.9545 | 16.2855 | 18.3092 | 22.3221 | 20.1269 |
|  | 1000 | 7.2222 | 8.7873 | 6.9068 | 10.3940 | 12.0250 | 14.7900 | 14.3890 | 21.0720 | 19.4330 |
| NSGA-III | 500 | 5.6285 | 6.0126 | 4.9557 | 11.6919 | 11.3751 | 11.5608 | 16.0083 | 18.8635 | 17.9297 |
|  | 1000 | 5.7024 | 5.7998 | 4.7109 | 11.2670 | 10.8910 | 9.8371 | 15.0540 | 17.1250 | 17.1180 |
| GrEA | 500 | 5.5579 | 11.2853 | 10.0447 | 11.4465 | 12.5820 | 15.5784 | 16.0072 | 21.6699 | 19.5459 |
|  | 1000 | 5.1809 | 11.2150 | 9.6578 | 11.7210 | 12.4110 | 14.2000 | 15.2500 | 19.7280 | 18.3740 |
| IKnEA | 500 | 5.7458 | 6.3875 | 5.3260 | 10.2807 | 10.5921 | 12.1350 | 16.5382 | 16.8882 | 13.9368 |
|  | 1000 | 5.3637 | 4.9127 | 3.7281 | 8.3173 | 8.7044 | 9.1386 | 12.8910 | 14.2480 | 11.1890 |

**Table 8** Improvement ratio of IKnEAon IGD（%）

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithm | Iterations | (10,3) | (10,6) | (10,9) | (30,3) | (30,6) | (30,9) | (50,3) | (50,6) | (50,9) |
| KnEA | 500 | 22.15 | 32.75 | 36.26 | 19.80 | 18.24 | 25.49 | 9.67 | 24.34 | 30.76 |
|  | 1000 | 25.73 | 44.09 | 46.02 | 19.98 | 27.61 | 38.21 | 10.41 | 32.39 | 42.43 |
| NSGA-III | 500 | -2.08 | -6.24 | -7.47 | 12.07 | 6.88 | -4.97 | -3.32 | 10.47 | 22.27 |
|  | 1000 | 5.94 | 15.29 | 20.86 | 26.18 | 20.08 | 7.10 | 14.36 | 16.80 | 34.64 |
| GrEA | 500 | -3.38 | 43.40 | 46.98 | 10.19 | 15.82 | 22.10 | -3.31 | 22.07 | 28.70 |
|  | 1000 | -3.53 | 128.28 | 61.40 | 29.04 | 42.58 | 35.64 | 15.46 | 27.78 | 39.11 |

**5.3 Analysis of experiment results**

**5.3.1 Comparison and analysis of experiment results without constraints in optimization time**

Experiment results show that all compared algorithms in all scenarios basically exhibit convergence when 1000 iterations are conducted. Thus, when constraints of optimization time are not considered, the mean of the HV and IGD of each algorithm after 1000 iterations in different scenarios can be used to evaluate the performance of compared algorithms. The improvement rates of the mean HV and IGD of the IKnEA in different scenarios are compared to those of KnEA, NSGA-III and GrEA, as shown in Figs. 7-8, respectively. The abscissa represents the experimental scenarios, and the ordinate represents the improvement rate of the IKnEA with each comparison algorithm on HV or IGD. Note that the improvement rate are set to 0 in scenarios there is not statistically significant difference in the average HV and IGD between the IKnEA and compared algorithms.

**(1) Comparison and analysis of improvement rate of IKnEA with other algorithms in term of the mean HV**

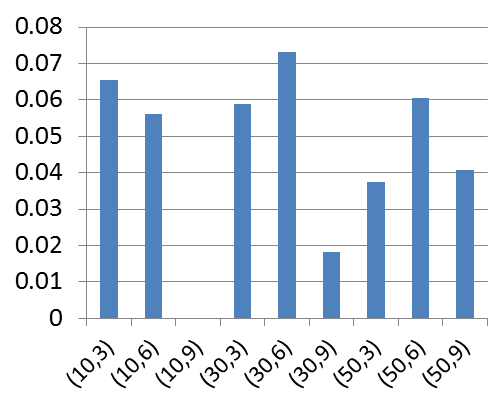
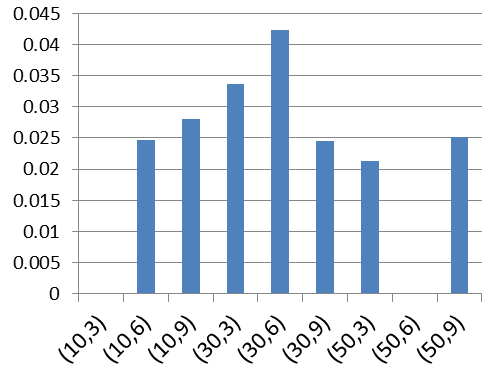
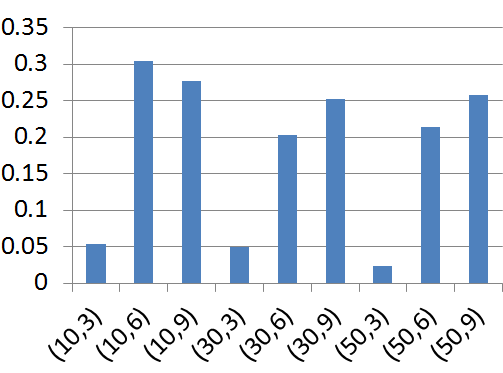
As can be seen from Fig. 7 (a)-(c), compared with the contrast algorithm, the mean HV of the IKnEA statistically enhances in most of the scenarios. It can be concluded that the IKnEA is comparable to or better than the compared to algorithm in most of the scenarios.

Compared with KnEA, the mean HV of the IKnEA shows enhancement of about 2%-7% and over 5% in most scenarios. Compared with GrEA, the improvement rate of mean HV is about 2%-4%. Compared with NSGA-III, the improvement rate of mean HV is around 5%-30%, and the range of the improvement ratio is about 20%-30% in most of the scenarios.

**(2) Comparison and analysis of improvement rate of IKnEA with other algorithms in terms of the mean IGD**

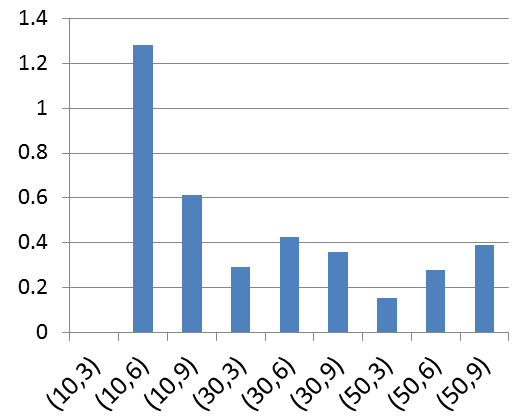
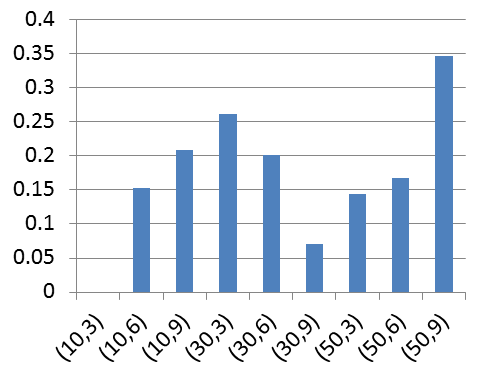
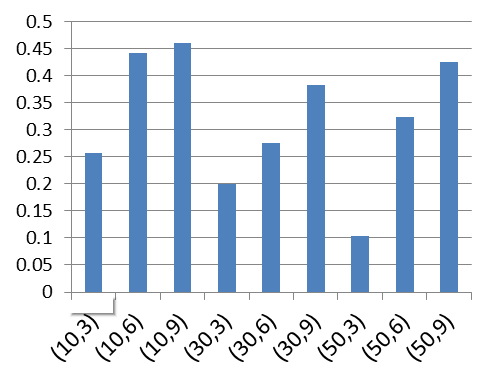
Fig. 8(a)-(c) show that there is no significant difference between the IKnEA and GrEA or NSGA-3 in terms of the mean IGD for scenario (10, 3). However, in the other scenarios, the mean IGD of the IKnEA is greatly improved.

The mean IGD of the IKnEA is lower (better) than KnEA by about 10% in the scenario (50, 3), while the means IGD of the IKnEA are reduced by 20%-45% in the remaining scenarios. In particular, the rate of reduction reaches a maximum of 46.02% in scenario (10,6). Compared with GrEA, the mean IGD of the IKnEA is about 20%-40% lower (better) in most scenarios, and the rate of reduction reaches a maximum of 128.28% in scenario (10,6). Compared with NSGA-III, the IGD improvement rate of the IKnEA is about 10%-25% in most scenarios and 34.64% in scene (50,9).

(a) KnEA (b) NSGA-III (c) GrEA

**Fig. 7** Comparison of IKnEA and KnEA, NSGA-III and GrEA, on the mean HV with respect to the improvement ratio in each scenario without optimization time constraints



(a) KnEA (b) NSGA-III (c)GrEA

**Fig. 8** Comparison of IKnEA and KnEA, NSGA-III and GrEA, on the mean IGD with respect to the improvement ratio in each scenario without optimization time constraints

**(3) Application scenario analysis of IKnEA**

Three types of algorithms are compared on nine different scenarios in terms of two evaluation indicators. To evaluate the performance of the IKnEA in a more comprehensive way, the multi-criteria decision analysis method is used. First, we take the improvement rate of HV and IGD from 1000 iterations as the evaluation criterion, and different experiment scenarios are used as decision alternatives. Then, the entropy weight method is used to evaluate the optimization effect of the IKnEA in each experimental scenario. A larger sum of the evaluation indicators’ entropy weight indicates a better optimization result. Finally, the evaluation results sorted from best to worst are listed in Table 9. The reason for using the entropy weight method is that it can quantify and synthesize the information of the evaluation object. Compared with the decision methods of the weight of objectives of people, the entropy weight method has the advantages of high accuracy and objectivity, and it is well suited for multi-criteria decision-making problems without explicit preferences of the decision maker. Column 2 in Table 9 represents the number of instances and virtual machines in each scenario. Columns 3-8 are the improvement rate of the IKnEA in terms of HV and IGD from the 1000 iterations compared to the compared algorithm. An improvement rate of 0 indicates that there is no statistically significant difference between the IKnEA and the compared algorithm in the corresponding indicators.

**Table 9** Result of comprehensive evaluation for analysis of IKnEA’s applicable Scenario

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Scenario ID | (*n, k*) | HV(%) | | | IGD(%) | | | | | Variance | Evaluation  value | Rank |
| KnEA | NSGA-III | GrEA | KNEA | NSGA-III | | GrEA | |
| Scenario 1 | （10,6） | 5.60 | 30.44 | 2.47 | 44.09 | | 15.29 | | 128.28 | 22.143 | 0.438 | 1 |
| Scenario 2 | （10,9） | 0.00 | 27.74 | 2.80 | 46.02 | | 20.86 | | 61.40 | 5.793 | 0.267 | 2 |
| Scenario 3 | （50,9） | 4.06 | 25.75 | 2.52 | 42.43 | | 34.64 | | 39.11 | 3.079 | 0.232 | 3 |
| Scenario 4 | （30,6） | 7.32 | 20.24 | 4.24 | 27.61 | | 20.08 | | 42.58 | 1.953 | 0.209 | 4 |
| Scenario 5 | （30,9） | 1.83 | 25.27 | 2.45 | 38.21 | | 7.10 | | 35.64 | 2.787 | 0.173 | 5 |
| Scenario 6 | （50,6） | 6.04 | 21.43 | 0.00 | 32.39 | | 16.80 | | 27.78 | 1.561 | 0.159 | 6 |
| Scenario 7 | （30,3） | 5.88 | 4.98 | 3.37 | 19.98 | | 26.18 | | 29.04 | 1.332 | 0.151 | 7 |
| Scenario 8 | （50,3） | 3.75 | 2.36 | 2.12 | 10.41 | | 14.36 | | 15.46 | 0.373 | 0.082 | 8 |
| Scenario 9 | （10,3） | 6.55 | 5.35 | 0.00 | 25.73 | | 0.00 | | 0.00 | 0.995 | 0.033 | 9 |

The following conclusions can be drawn from Table 9. First, the comprehensive evaluation of the IKnEA is smaller when the number of virtual machines is 3, which means that the improvement of the IKnEA is smaller when the resource supply is less. Second, the comprehensive evaluation is better in scenario 1 and scenario 2, which indicates that the improvement of the algorithm is larger when there is more resource supply. The corresponding variance for scenario 1 is largest, which indicates that the improvement rate of the IKnEA in each indicator differs to some degree with the comparison algorithm. In particular, the mean IGD of the IKnEA is 128.28% higher than GrEA. Overall, the IKnEA is more suited for the cloud workflow scheduling problems with a large number of virtual machines related to computing requirements.

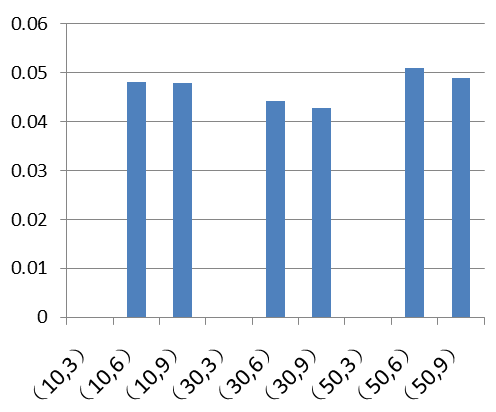
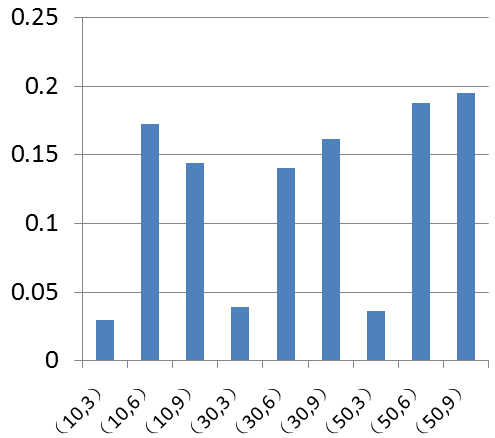
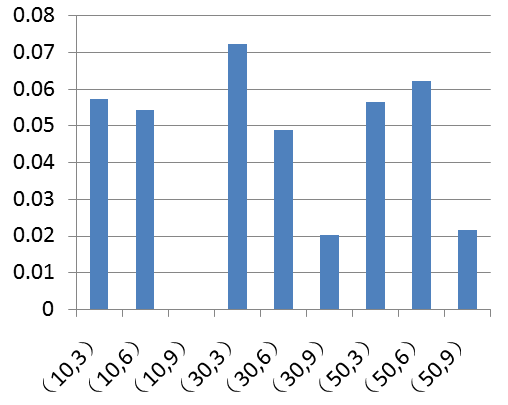
**5.3.2 Comparison and analysis of experiment results with constraints on optimization time**

The computing time of the scheduling optimization algorithm is often an important factor affecting the user experience. In realistic cloud workflow scheduling, there are often scenarios where a large number of workflow instances are submitted to be executed. These workflow instances are often scheduled by a batch strategy. Each batch of tasks need to be optimized within a short time period, thus requiring high real-time responsiveness, and the optimization time should be as short as possible. In order to analyze the optimization result of the IKnEA under computing time constraint, the improvement ratio of the IKnEA compared with those of other MOEAs in terms of HV and IGD are provided in Figs. 9-10.

**(1) Comparison and analysis of the improvement ratio of the IKnEA with respect to other algorithms on HV**

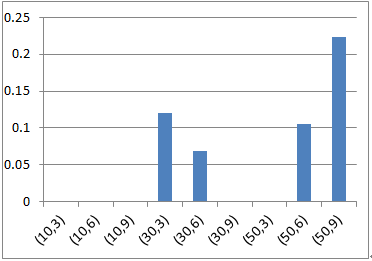
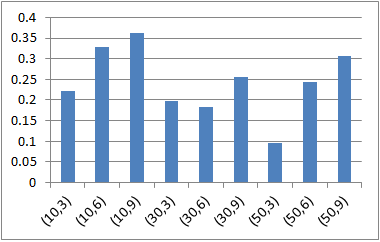
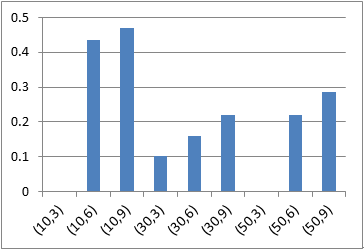
It can be concluded from Fig. 9 (a)-(c) that except for a few scenarios, the mean HV of the IKnEA are significantly better than the compared algorithm, which indicates that the IKnEA has higher search efficiency and can obtain better solutions within a shorter optimization time.

Compared with KnEA, the improvement ratio of the mean HV is about 2%-7% and over 5% in most scenarios. Compared with NSGA-III, except for the scenarios with 3 virtual machines, IKnEA enhances the mean HV at a ratio of 14-20%. Even when the number of virtual machines is set to 3, the mean HV of the IKnEA is also about 4% higher than the others. Compared with GrEA, the mean HV of the IKnEA exhibits no significant difference with GrEA's in scenarios with three virtual machines, and the improvement ratio of mean HV in other scenarios is greater than 4%. In the case of 3 virtual machines, compared with NSGA-III and GrEA, the improvement ratio of the mean HV is smaller or insignificant, which is expected.



(a) KnEA (b) NSGA-III (c) GrEA

**Fig. 9** Compared with KNEA, NSGA-III and GrEA, HV improvement ratio of IKnEA in each scenario with optimization time constraints

(a) KnEA (b) NSGA-III (c)GrEA

**Fig. 10** Compared with KNEA, NSGA-III and GrEA, IGD improvement ratio of IKnEA in each scenario with optimization time constraints

**(2) Comparison and analysis of improvement ratio of IKnEA with respect to other algorithms on IGD**

Fig. 10 (a)-(c) shows the improvement ratio of the mean IGD of the IKnEA over the compared algorithms when 500 iterations are run for each scenario. In most scenarios, the IKnEA is higher than others. The results show that the IKnEA has a significant advantage over the compared algorithms when the allowed optimization time is short.

Compared with KnEA, the improvement ratio of the IKnEA is more than 10%, and the maximum ratio reaches 36.26%. Compared with NSGA-III, the improvement ratio of IGD are greater than 5% with a maximum value of 22.26% in scenarios (30,3), (30,6), (50,6), and (50,9). Compared with GrEA, the improvement ratio of the IGD of the IKnEA can reach approximately 10% or more except for the scenarios with 3 virtual machines, and the maximum is 46.98%.

Overall, compared with other MOEAs, we can see that the IKnEA can obtain better solutions within fewer iterations, and its search efficiency is higher than the compared algorithms. Therefore, the IKnEA is suited for scheduling scenarios that require high real-time performance.

**6. Conclusions and future work**

In this paper, a new cloud workflow scheduling model is proposed based on practical demands from users. This model considers four objectives, namely, minimization of makespan and the average execution time of all workflow instances, maximization of reliability, and minimization of the cost of the workflow execution. At the same time, security is also considered as a constraint. Since MOEAs developed for solving bi- or three-objective optimization problems are usually inefficient for optimization problems with four objectives (known as MaOPs), and KnEA has shown to be able to outperform several state-of-the-art MOEAs for solving this kind of problem. To efficiently solve the four-objective cloud workflow scheduling problem, KnEA is improved by enhancing it environmental selection strategies. Experimental results demonstrate that the IKnEA exhibits better performance in most scenarios compared with KnEA, GrEA and NSGA-III, three popular algorithms for MaOPs.

Though IKnEA is more effective and efficient in most scenarios compared with other algorithms, there are several scenarios in which the enhancement achieved by IKnEA is not obvious. In future work, the performance of IKnEA could be further improved. In addition, we will investigate issues to deal with the cloud workflow scheduling problem with large number of variables.

**Acknowledgments**

This work was supported in part by the Fundamental Research Funds for the Central universities of China (No.DUT13JS11), and in part by the National Natural Science Foundation of China under Grant 71533001.

**References**

[1] Z. H. Zhan, X. F. Liu, Y. J Gong et al, Cloud computing resource scheduling and a survey of its evolutionary approaches, Acm Comput Surv 47(4) (2015) 63.

[2] W. Li, Q. Zhang, J. Wu et al. Trust-based and QoS demand clustering analysis customizable cloud workflow scheduling strategies, in: Proceedings of the 2012 IEEE International Conference on Cluster Computing Workshops (Cluster Workshops), 2012, pp. 111-119.

[3] K. C. Huang, Y. L. Tsai, H. C. Liu. Task ranking and allocation in list-based workflow scheduling on parallel computing platform, J Supercomput. 71(1) (2015) 217-240.

[4] K. Nishant, P. Sharma, V. Krishna et al, Load balancing of nodes in cloud using ant colony optimization, in: Proceedings of the 14th IEEE International Conference on Computer Modelling and Simulation (UKSim), 2012, pp. 3-8.

[5] L. Singh and S. Singh, A genetic algorithm for scheduling workflow applications in unreliable cloud environment, in: Proceedings of International Conference on Security in Computer Networks and Distributed Systems 2014, pp. 139-150.

[6] Y. C. Lee, H. Han, A. Y. Zomaya et al, Resource-efficient workflow scheduling in clouds, Knowl-Based Syst, 80 (2015) 153-162.

[7] G. Yao, Y. Ding, L. Ren et al, An immune system-inspired rescheduling algorithm for workflow in Cloud systems, Knowl-Based Syst, 99 (2016) 39-50.

[8] J. J. Durillo, V. Nae, R. Prodan, Multi-objective energy-efficient workflow scheduling using list-based heuristics, Future Gener Comp Syst, 36 (2014) 221-236.

[9] H. M. Fard, R. Prodan, T. Fahringer, Multi-objective list scheduling of workflow applications in distributed computing infrastructures, J Parallel Distr Com 74(3) (2014) 2152-2165

[10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, A fast and elitist multi-objective genetic algorithm: NSGA-II, IEEE Trans Evolut Comput, 6(2) (2002) 182–197.

[11] Szabo C, Kroeger T. Evolving multi-objective strategies for task allocation of scientific workflows on public clouds, in: Proceedings of the 2012 IEEE Congress on Evolutionary Computation (CEC). 2012, pp. 1-8.

[12] X. Zhang, Y. Tian, Y. Jin, A knee point-driven evolutionary algorithm for many-objective optimization, IEEE Trans. Evolut Comput , 19(6) (2015) 761-776.

[13] L. While, P. Hingston, L. Barone, and S. Huband, A faster algorithm for calculating hypervolume, IEEE Trans on Evolut. Comput, 10(1) (2006), 29–38.

[14] M. Li and J. Zheng, Spread assessment for evolutionary multiobjective optimization, in: Proceedings of 5th International Conference on Evolutionary Multi-Criterion Optimization., Nantes, France, 2009, pp. 216–230

[15] Liu P, Zhang X, Research on the supplier selection of a supply chain based on entropy weight and improved ELECTRE-III method, Int J Prod Res, 49(3) (2011), 637-646.

[16] Chen S J J, Hwang C L / M J, Krelle W, Fuzzy Multiple Attribute Decision Making: Methods and Applications, Springer-Verlag, New York,. 1992, pp.302-310.

[17] M. Guzek, J. E. Pecero, B. Dorronsoro et al, Multi-objective evolutionary algorithms for energy-aware scheduling on distributed computing systems, Appl Soft Comput, 24 (2014) 432-446.

[18].T. Wenan, S. Yong, L. Ling Xia, L. GuangZhen, and W. Tong, A Trust Service-Oriented Scheduling Model for Workflow Applications in Cloud Computing, IEEE Syst J, 8(3) (2014) 868-878.

[19] Ding Y, Yao G, Hao K, Fault-Tolerant Elastic Scheduling Algorithm for Workflow in Cloud Systems, Inform Sciences, 393(2017) 47-65.

[20] Ye X, Li J, Liu S, et al. A hybrid instance-intensive workflow scheduling method in private cloud environment, Natural Computing, (2017) 1-12. doi:10.1007/s11047-016-9600-3

[21] D. Nasonov and N. Butakov, Hybrid Scheduling Algorithm in Early Warning Systems, Procedia Computer Science (29) (2014) 1677-1687.

[22] J. Pecero and P. Bouvry, Workflow Scheduling on Virtualized Servers, in: Advanced Approaches to Intelligent Information and Database Systems, Springer International Publishing, Switzerland, 2014, pp. 247-254.

[23] T. Xiaoyong, K. Li, Z. Zeng et al. A novel security-driven scheduling algorithm for precedence-constrained tasks in heterogeneous distributed systems, IEEE Trans Comput. 60(7) (2011) 1017-1029.

[24] K. Zhu, H. Song, L. Liu, et al, Hybrid genetic algorithm for cloud computing, in: Proceedings of the 2011 IEEE Asia-Pacific on applications, Services Computing Conference, 2011, pp. 182-187

[25] B. Dorronsoro, S. Nesmachnow, J. Taheri et al, A hierarchical approach for energy-efficient scheduling of large workloads in multicore distributed systems, Sustain Comput-Info, 4(4) (2014) 252-261.

[26] Y. C. Lee, A. Y. Zomaya, M. Yousif, Reliable workflow execution in distributed systems for cost efficiency, in: Proceedings of the 11th IEEE/ACM International Conference on Grid Computing (GRID) 2010, pp. 89-96.

[27] F. Ramezani, J. Lu, J. Taheri et al, Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments, World Wide Web-internet & Web Information Systems, , 18(6) (2015) 1737-1757.

[28] G. Shen, Y Q. Zhang, A shadow price guided genetic algorithm for energy aware task scheduling on cloud computers, Advances in Swarm Intelligence. Springer Berlin Heidelberg, 2011, 522-529.

[29] M. A. Rodriguez, R. Buyya, Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds, IEEE Trans Cloud Comput, 2(2) (2014) 222-235

[30] L. Zhu, Q. Li, L. He. Study on cloud computing resource scheduling strategy based on the ant colony optimization algorithm, Int J Comput Sci Issues, 9(5) (2012) 54.

[31] S. S. K. Kumar and P. Balasubramanie, Dynamic scheduling for cloud reliability using transportation problem, J Comput Sci 8(10) (2012) 1615.

[32] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints, IEEE Trans Evolut Comput, 18(4) (2014) 577-601.

[33] S. Yang, M. Li, X. Liu et al. A grid-based evolutionary algorithm for many-objective optimization, IEEE Trans on Evolut Comput, 17(5) (2013) 721-736

[34] Cheng R, Olhofer M, Jin Y. Reference vector based a posteriori preference articulation for evolutionary multiobjective optimization, IEEE Trans Evolut Comput. (2015) 939-946.

[35] Zou Z H, Yi Y, Sun J N, Entropy method for determination of weight of evaluating indicators in fuzzy synthetic evaluation for water quality assessment. J Environ Sci, 18(5) (2006) 1020-1023.

[36] Kelemenis A, Askounis D. A new TOPSIS-based multi-criteria approach to personnel selection, Expert Syst Appl, 37(7) (2010) 4999-5008.

[37] Li D F. TOPSIS-based nonlinear-programming methodology for multiattribute decision making with interval-valued intuitionistic fuzzy sets. IEEE Trans Fuzzy Syst, 18(2) (2010) 299-311.