Computers & Operations Research 40 (2013) 3045–3055

Contents lists available at ScienceDirect

Computers & Operations Research

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

Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm

Jinn-Tsong Tsai [a,*], Jia-Cen Fang [a], Jyh-Horng Chou [b,c,d]

[a] Department of Computer Science, National Pingtung University of Education, 4-18 Min-Sheng Road, Pingtung 900, Taiwan, ROC
[b] Institute of System Information and Control, National Kaohsiung First University of Science and Technology, 1 University Road, Yenchao, Kaohsiung 824, Taiwan, ROC
[c] Department of Electrical Engineering, National Kaohsiung University of Applied Sciences, 415 Chien-Kung Road, Kaohsiung 807, Taiwan, ROC
[d] Department of Healthcare Administration and Medical Informatics, Kaohsiung Medical University, 100 Shi-Chuan 1st Road, Kaohsiung 807, Taiwan, ROC

ARTICLE INFO

Available online 16 July 2013

Keywords:
Cloud computing
Differential evolution algorithm
Task scheduling
Cost and time models
Multi-objective approach

ABSTRACT

Purpose: The objective of this study is to optimize task scheduling and resource allocation using an improved differential evolution algorithm (IDEA) based on the proposed cost and time models on cloud computing environment.
Methods: The proposed IDEA combines the Taguchi method and a differential evolution algorithm (DEA). The DEA has a powerful global exploration capability on macro-space and uses fewer control parameters. The systematic reasoning ability of the Taguchi method is used to exploit the better individuals on micro-space to be potential offspring. Therefore, the proposed IDEA is well enhanced and balanced on exploration and exploitation. The proposed cost model includes the processing and receiving cost. In addition, the time model incorporates receiving, processing, and waiting time. The multi-objective optimization approach, which is the non-dominated sorting technique, not with normalized single-objective method, is applied to find the Pareto front of total cost and makespan.
Results: In the five-task five-resource problem, the mean coverage ratios C(IDEA, DEA) of 0.368 and C(IDEA, NSGA-II) of 0.3 are superior to the ratios C(DEA, IDEA) of 0.249 and C(NSGA-II, IDEA) of 0.288, respectively. In the ten-task ten-resource problem, the mean coverage ratios C(IDEA, DEA) of 0.506 and C(IDEA, NSGA-II) of 0.701 are superior to the ratios C(DEA, IDEA) of 0.286 and C(NSGA-II, IDEA) of 0.052, respectively. Wilcoxon matched-pairs signed-rank test confirms there is a significant difference between IDEA and the other methods. In summary, the above experimental results confirm that the IDEA outperforms both the DEA and NSGA-II in finding the better Pareto-optimal solutions.
Conclusions: In the study, the IDEA shows its effectiveness to optimize task scheduling and resource allocation compared with both the DEA and the NSGA-II. Moreover, for decision makers, the Gantt charts of task scheduling in terms of having smaller makespan, cost, and both can be selected to make their decision when conflicting objectives are present.

1. Introduction

Cloud applications are very popular in recent years. Especially, cloud computing has emerged as a promising approach to rent IT infrastructures on a short-term pay-per-usage basis. With cloud computing, companies can scale up to massive capacities in an instant without having to invest in new infrastructure, train new personnel, or license new software. Cloud computing is of particular benefit to small and medium-sized businesses who wish to completely outsource their data-center infrastructure, or large companies who wish to get peak load capacity without incurring the higher cost of building larger data centers internally. In both instances, service consumers use what they need on the Internet and pay only for what they use. Thus, operators of so-called Infrastructure-as-a-Service (IaaS) clouds, like Amazon EC2 [3], let their customers allocate, access, and control a set of virtual machines which run inside their data centers and only charge them for the period of time the machines are allocated. Therefore, workflow management on cloud computing becomes more important, when many tasks are sent to cloud environment at the same time.

Researches on specification and scheduling of workflows have concentrated on temporal and causality constraints, which specify existence and order dependencies among tasks. However, another set of constraints that specify resource allocation is also equally important. Execution of a task has a cost and this may vary depending on the resources allocated. Resource allocation constraints define restrictions on how to allocate resources, and scheduling under resource allocation constraints provide proper

* Corresponding author. Tel.: +886 8 7226141; fax: +886 8 7215034.
E-mail addresses: jttsai@mail.npue.edu.tw, jttsainpue@gmail.com (J.-T. Tsai).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.06.012

resource allocation to tasks [27,5,12,21,10,19,38,14,18]. In cloud computing environment, machines are located in different regions and have disparate processing abilities, characteristics (number of CPU cores, amount of main memory, etc.), and cost. In these situations, the cost and makespan associated with the task schedule and the resources allocated should be taken into account. Therefore, task scheduling and resource allocation should be carefully coordinated and optimized jointly in order to achieve an overall cost and time-effective schedule. That is, to find optimal task schedules by minimizing cost and makespan.

This study developed cost and time models for computing the cost and time in a task schedule. The cost model includes the processing and receiving cost and the time model incorporates receiving, processing, and waiting time. A parallel-machine scheduling involving both task processing and resource allocation was studied by using an improved differential evolution algorithm (IDEA). The proposed IDEA combines Taguchi method [23,22,32] and a differential evolution algorithm (DEA) [30,31,17]. The DEA had a powerful global exploration capability on macro-space and uses fewer control parameters. The systematic reasoning ability of the Taguchi method was used to exploit the better individuals on micro-space to be potential offspring. The objective is to find optimal task schedules by minimizing total cost and makespan. Therefore, the multi-objective optimization approach, which was the non-dominated sorting technique [28,8,9], not with normalized single-objective method, was applied to find the Pareto front of total cost and makespan. Finally, based on suitable schedules from the Pareto-optimal solution sets, efficiency of the task distribution and the resource utilization were discussed and analyzed.

The paper is organized as follows. Section 2 defines the research issue. The related works are briefly described in Section 3. The proposed approach is illustrated in Section 4. Section 5 shows case study, results, and discussions. Finally, Section 6 concludes the study.

2. Problem description

To generalize the discussion, the assumption is that there is a set of cloud customer tasks and each task has many subtasks with precedence constraints. Each subtask is allowed to be processed on any given available resource. A cloud resource has a given level of capacity (e.g., CPU, memory, network, storage). A subtask is processed on one resource at a time and the given resources are available continuously. The cloud computing discussion in this study highlighted the fact that either cloud resource allocation or cloud task scheduling scenarios are actual NP-Complete problems [2]. Task scheduling of cloud computing can be stated as follows.

Inputs: It is given a set $J=(J_1, J_2, \ldots, J_i, \ldots, J_n)$, where $i\in[1, n]$ and n is the number of independent tasks. A task J_i is formed by a sequence J_{i1}, J_{i2}, …, J_{ij},…, J_{iq} ($j\in[1, q]$ and q is the number of subtasks) to be performed one after the other according to the given sequence. A given available resource set is $R=(R_1, R_2, \ldots, R_k, \ldots, R_m)$, in which $k\in[1, m]$ and m is the number of available resources. Each subtask J_{ij} can be executed on any among a subset $R_k \subseteq R$ of available resources.

Outputs: An efficient Gantt chart of scheduling, including the assignment of tasks on available resources and makespan. For a three-task four-resource problem as an example, if its schedule is $\{(J_{11}, R_2), (J_{31}, R_4), (J_{21}, R_2), (J_{12}, R_4), (J_{22}, R_1), (J_{32}, R_2), (J_{23}, R_4), (J_{33}, R_3), (J_{13}, R_1)\}$, its Gantt chart and makespan are drawn in Fig. 1.

Constraints: The processing time of each subtask is resource-dependent. Pre-emption is not allowed, i.e., each subtask must be completed without interruption once started. Resources cannot perform more than one subtask at a time.

Objectives: The problem is to assign each subtask to an appropriate resource (routing problem), and to sequence the subtasks on the resources (sequencing problem) in order to minimize the total cost and makespan. The makespan is the total length of the schedule (that is, when all the jobs have finished processing). The challenge is that task scheduling and resource allocation should be carefully coordinated and optimized jointly in order to achieve an overall cost and time-effective schedule.

3. Related works

This section gives a brief review about differential evolution algorithms, Taguchi method, and a multi-objective approach used in this study.

3.1. Differential evolution algorithms

Storn and Price [30,31] proposed a DEA to solve the Chebychev polynomial fitting problem. DEA, which is an evolutionary algorithm, includes four operations: initialization, mutation, recombination, and selection, as shown below. The advantages of using a DEA for solving design problems are its global solution finding property, simple but powerful search capability, easy-to-understand concept, compact structure, having only a few control parameters, ease of use, and high convergence characteristics [29,24].

3.1.1. Initialization

Randomly select the initial parameter values uniformly on the intervals(x_j^-, x_j^+), which are lower and upper bound for each parameter. Each of the individual vectors undergoes mutation, recombination, and selection.

3.1.2. Mutation

In generation G, each individual vector X_i^G ($i=1, 2, \ldots, p_s$) in the population becomes a target vector, where p_s denotes the population size. There are some mutation strategies reported in the literature, three common ones are listed in Eqs. (3.1)–(3.3). For each target vector X_i^G, DEA applies a differential mutation operation to generate a mutant individual V_i^G, according to the following equations:

$$V_i^G = X_j^G + F(X_k^G - X_q^G) \tag{3.1}$$

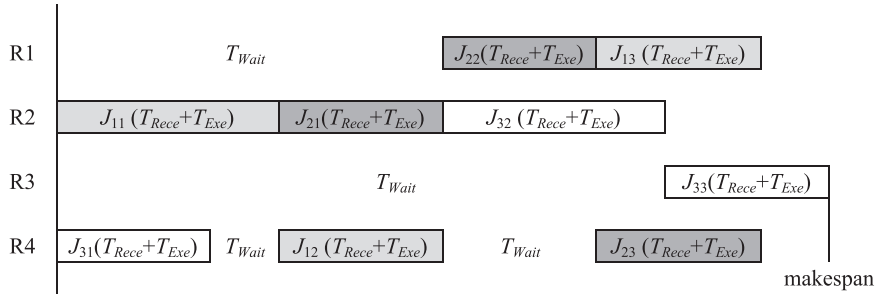

Fig. 1. Gantt chart.

$$V_i^G = X_{best}^G + F(X_k^G - X_q^G) \tag{3.2}$$

or

$$V_i^G = X_{best}^G + F(X_j^G + X_k^G - X_q^G - X_r^G)\,, \tag{3.3}$$

where X_j^G, X_k^G, X_q^G, and X_r^G are randomly taken from the population such that j, k, q, and r belong to $\{1, 2, \ldots, p_s\}$ and $j \neq k \neq q \neq r$; X_{best}^G is the best individual at generation G; the mutation factor F is a constant parameter which lies in the range [0, 2]. In the event that mutation causes an element v_{ij}^G in the vector $V_i^G = \left[v_{i1}^G, v_{i2}^G, \ldots, v_{i\rho}^G\right]$ to shift outside the allowable interval $(x_j^-,\ x_j^+)$ then v_{ij}^G is set to x_j^- if $v_{ij}^G < x_j^-$, and x_j^+ if $v_{ij}^G > x_j^+$.

3.1.3. Recombination

Each mutant vector V_i^G recombines with its respective parent X_i^G through crossover operation to produce its final offspring vector U_i^G. The genes of U_i^G are inherited from V_i^G and X_i^G, determined by the crossover rate p_c, in which $p_c \in [0, 1]$, in the following:

$$U_{ji}^G = \begin{cases} V_{ji}^G & \text{if } \text{rand}_j(0,1) \le p_c \\ X_{ji}^G & \text{otherwise} \end{cases} \tag{3.4}$$

where $j = 1, 2, \ldots, \rho$, which is the total number of design variables. The $\text{rand}_j(0, 1)$ is the jth evaluation of a uniform random number generator from the interval [0, 1].

3.1.4. Selection

Each generated child is evaluated with the fitness function, and based on this evaluation either the child vector U_i^G or its parent vector X_i^G is selected for the next generation as follows:

$$X_i^{G+1} = \begin{cases} U_i^G & \text{if } f(U_i^G) \le f(X_i^G) \\ X_i^G & \text{otherwise} \end{cases} \tag{3.5}$$

3.2. Taguchi method

The Taguchi method is a robust-design method [23,22,32] that uses statistical experimental design concepts to evaluate and implement improvements for products, processes or equipment systems. The objective of the method is to improve quality by minimizing the effects of causes but without eliminating the causes of variations.

The orthogonal array (OA) of the Taguchi method enables study of a large number of design variables with a small number of experiments. The better combinations of design variables are indicated by the orthogonal arrays and by the signal-to-noise ratios. The underlying concept of the Taguchi method is to maximize a performance measure (here, signal-to-noise ratio) by using orthogonal arrays to run a partial set of experiments. This work uses a two-level OA (2OA) with Q factors, where Q is the number of design factors (variables) and each factor has two levels. To establish an OA with two levels of Q factors, $L_n(2^{n-1})$ represents $n-1$ columns, n separate experiments corresponding to the n rows, and $n = 2^k$ where k is a positive integer ($k > 1$) and $Q \le n-1$. If $Q < n-1$ only the front Q columns are used, and the other $n-1-Q$ columns are ignored. For example, to allocate six factors, each of which has two levels, only six columns are needed; therefore, $L_8(2^7)$ is sufficient for this purpose because it has seven columns.

The signal-to-noise ratio (η_i), which is measured in decibels, refers to the mean-square-deviation of the objective function. According to the Taguchi definition, $\eta_i = -10\ \log((1/\tau)\ \sum_{j=1}^{\tau}(1/y_{ij}^2))$ for the larger-the-better characteristic, and $\eta_i = -10\ \log((1/\tau)\sum_{j=1}^{\tau}(y_{ij}^2))$ for the smaller-the-better characteristic where $i = 1, 2, \ldots, n$ and where n is the number of experiments for an OA $L_n(2^{n-1})$. Let y_{ij} denote the experimental output value of the ith row of OA $L_n(2^{n-1})$ where $j = 1, 2, \ldots, \tau$ and where τ is the number of experiments for the i^{th} row of OA. When building the response table, the effects of different factors can be defined as follows:

$$E_{\beta l} = \text{sum of } \eta_i \text{ for factor } \beta \text{ at level } l, \tag{3.6}$$

where i is the experiment number, β is the factor name, and l is the level number.

3.3. Multi-objective approach

Many real-world design or decision making problems involve simultaneous optimization of multiple objectives. These objectives, for example, minimizing time, money, distance, storage, are often conflicting, which means that improvement on one objective value results in the degradation of another. Instead of seeking for a single optimal solution with respect to one objective, multi-objective optimization aims at seeking for the set of so called Pareto-optimal solutions. In single-objective optimization, one attempts to obtain the best solution. However, in a multi-objective case, one cannot identify a single solution that simultaneously optimizes each objective. While searching for solutions, one reaches points such that, when attempting to improve an objective further, other objectives suffer as a result. A solution is called non-dominated solution or Pareto-optimal solution, if it cannot be eliminated from consideration by replacing it with another solution which improves an objective without worsening another one. For example, assuming a minimization problem, a solution x is said to dominate solution x' if and only if $f_i(x) \le f_i(x')$ for all the objectives f_i ($i = 1, 2, \ldots, m$, where m is the number of objectives) and $f_j(x) < f_j(x')$ for at least one objective. A point x^* is a Pareto-optimal solution if there is no point x such that $f(x)$ dominates $f(x^*)$. The set of all the Pareto-optimal solutions is called the Pareto set and its image in the objective space is the Pareto front. The rest of the solutions are knows as dominated solutions. Since none of the solutions in the non-dominated set is absolutely better than any other, any one of them is an acceptable solution. Therefore, the choice of one solution over the other requires problem knowledge and a number of problem-related factors [28].

Since genetic algorithms (GAs) work with a population of points, a number of Pareto-optimal solutions may be captured using GAs. Non-dominated sorting genetic algorithm II (NSGA-II) has been developed to solve multi-objective optimization problems. NSGA-II implemented both aspects of Goldberg's suggestion [15] in a better way. The ranking classification was performed according to the non-dominance of the individuals in the population and a distribution of the non-dominated points was maintained using a niche formation technique. Both these aspects caused the distinct non-dominated points to be found in the population [28,8,9].

4. The proposed approach for task scheduling and resource allocation

The proposed IDEA includes two calculated models for cloud computing and five new ideas for scheduling. They are shown in the following. The cost and time models are proposed for cloud scheduling. The cost model includes rent cost for processing and receiving subtask. The time model incorporates receiving, processing, and waiting time. (1) A subtask-resource-based technique is proposed to encode a schedule. The technique is useful to avoid generating infeasible individual on initialization, mutation, and crossover. (2) A mask mutation operator Θ to generate a mutated individual instead of a traditional mathematical operator F in DEA, it can effectively generate feasible and diverse individuals for task scheduling. (3) For considering diverse individuals, the subtasks are changed in mutation and the resources are recombined in crossover. (4) A systematic reasoning capability of the Taguchi method with orthogonal mask Θ is exploited for use in selecting subtasks during the recombination

operation. (5) The IDEA procedures combine the DEA with improved operators and the Taguchi method with orthogonal mask Θ for cloud scheduling. The DEA has a powerful global exploration capability on macro-space and uses fewer control parameters. The systematic reasoning ability of the Taguchi method is used to exploit the better individuals on micro-space to be potential offspring.

The proposed IDEA for task scheduling and resource allocation is described below.

4.1. Detailed procedures of IDEA procedure

The following demonstrates the use of IDEA approach to solve task scheduling and resource allocation.

4.1.1. Proposed cost and time models

In cloud computing environment, resources are located in different regions and have disparate processing abilities, characteristics (number of CPU cores, amount of main memory, etc.), and cost. Commercial cloud infrastructures used a simple cost computational model. For example, Amazon EC2 [3] charged users per day of resource usage. The proposed cost and time models made a finer grain estimate of the real infrastructure usage which helps the user estimating cost and time in specified resources that would be consumed for each run of an application. The proposed cost model includes rent cost for processing and receiving subtask. Let T_{Exe} and T_{Rece} be processing and receiving time, respectively, of a subtask. C_{Rent_P} and C_{Rent_R} are rent costs of per-unit time on cloud resources for processing and receiving subtask, respectively. Suppose that the cost of transferring data between one resource and the other one is ignore. Therefore, the total cost C_{Total} to implement all subtasks is shown as follows:

$$C_{Total} = \sum^{all\ subtasks} (C_{Rent\_Exe} + C_{Rent\_Rece}) \tag{4.1}$$

where $C_{Rent\_Exe} = T_{Exe} \times C_{Rent\_P}$ and $C_{Rent\_Rece} = T_{Rece} \times C_{Rent\_R}$. C_{Rent_Exe} and C_{Rent_Rece} are rent costs on cloud resources for processing and receiving subtask, respectively.

The maximum completion time of all the tasks is makespan, which is the time difference between the start and finish of a sequence of tasks on a resource. The proposed time model for computing total completion time on a resource includes receiving, processing, and waiting time. The total completion time T_{Total_i} on each resource to implement subtasks is $T_{Total\_i} = \sum T_{Rece} + \sum T_{Exe} + \sum T_{Wait}$, where T_{Exe}, T_{Wait}, and T_{Rece} are receiving, processing, and waiting time, respectively. The makespan is shown below:

$$Makespan = Max(T_{Total\_1}, T_{Total\_2}, \ldots, T_{Total\_m}), \tag{4.2}$$

where m is the number of given available resources.

Algorithm 1

Begin
  for $i = 1$ to p_s
    Create an initial individual $\left(x_{i1}^1, x_{i2}^1, \ldots, x_{i\rho}^1\right)$
    for $j = 1$ to ρ
      Generate two random numbers α and β, in which $\alpha, \beta \in [0, \rho]$
      Conduct the swap-change on two subtasks of $\left(x_{i1}^1, x_{i2}^1, \ldots, x_{i\rho}^1\right)$ at the positions α and β
    end
    Assign an available resource for each subtask by random.
    Obtain an individual vector $\left(x_{i1}^1, x_{i2}^1, \ldots, x_{i\rho}^1\right)$
  end
  Obtain p_s initial feasible individuals.
End

The trade-off between total cost and makespan was minimized by using the non-dominated sorting technique [28]. The Pareto set was generated by the proposed IDEA. The choice of one solution over the other depends on the user needs.

4.1.2. Encoding and initial population

The subtask-based representation is proposed to encode a sequence of subtasks. Each element denotes each subtask of a task, and a vector can be decoded into a schedule. The method is useful to avoid generating infeasible individual on reproduction, crossover, and mutation. For example, consider a case of three tasks each of which has three subtasks. A nine-dimensional vector is constructed as $OP = (1, 3, 2, 1, 2, 3, 2, 3, 1)$. The first element "1" of OP is the first subtask of task 1. The second element "3" of OP is the first subtask of task 3. The third element "2" of OP is the first subtask of task 2. The fourth element "1" of OP is the second subtask of task 1. The fifth element "2" of OP is the second subtask of task 2, and so on. Further, the subtask-based representation combines resources into a subtask-resource-based representation.

This study applied a subtask-resource-based technique to encode a schedule in which each individual includes a sequence of subtasks and resources. The encoding of tasks is as follows: The first task is expressed as "1", and the second task is expressed as "2", and other tasks are denoted in this manner. The number 1 of resources is denoted as "R_1", and so on for other resources. For example, a certain assignment is composed of three tasks, each task consists of three subtasks, and there are four available resources. The resource allocation for tasks is shown in Table 1. Suppose the individual was distributed as $X = \{(1, R_2), (3, R_4), (2, R_2), (1, R_4), (2, R_1), (3, R_2), (2, R_4), (3, R_3), (1, R_1)\}$, where the first element "1" represents the first subtask of task 1 and its resource allocation is R_2. The second element "3" represents the first subtask of task 3 and its resource allocation is R_4. The third element "2" represents the first subtask of task 2 and its resource allocation is R_2. The fourth element "1" represents the second subtask of task 1 and its resource allocation is R_4, and so on for the remaining elements.

For convenience and simplicity, an individual X_i^G is denoted as

$$X_i^G = (x_{i1}^G, x_{i2}^G, \ldots, x_{ij}^G, \ldots, x_{i\rho}^G) \tag{4.3}$$

where G denotes the current generation, $i = 1, 2, \ldots, p_s$, and p_s denotes the population size. The $x_{ij}^G (j = 1, 2, \ldots, \rho)$ includes the number of task and resource, where ρ is the number of subtasks. The initial subtasks of x_{ij}^1 are produced by the swap-change elements randomly and resource allocation for each subtask is randomly chosen from given available resources. Once the initial population is generated, the fitness value of each individual is evaluated and stored for future reference.

The initialization procedure uses the following algorithm to produce p_s individuals:

4.1.3. Mutation operation

In every generation, each individual vector $X_i^G (i = 1, 2, \ldots, p_s)$ in the population becomes a target vector. For target vectors $X_i^G (i \in 1, 2, \ldots, p_s)$ and the best vector X_{best}^G in the current generation G, the IDEA

Table 1
Resources allocation for the tasks.

Task no	Subtask 1	Subtask 2	Subtask 3
Task 1	R_2	R_4	R_1
Task 2	R_2	R_1	R_4
Task 3	R_4	R_2	R_3

uses the mask mutation operator Θ to generate a mutated individual V_i^G because a mutation scaling factor F in DEA is not suitable for task scheduling problems, by applying the following equation:

$$V_i^G = X_{best}^G \Theta (X_j^G \Theta X_k^G), \tag{4.4}$$

where X_j^G and X_k^G are randomly selected from the population such that j and k belong to $\{1, 2, \ldots, p_s\}$ and $j \neq k$. The X_{best}^G represents the best individual in the current generation G. The Θ is a mask mutation factor, which is from an integer set S corresponding to the number of tasks. The S is randomly divided into the two sets S_1 and S_2, where $S_1 \cap S_2 = \varnothing$ and $S = S_1 \cup S_2$. The resource allocation remains unchanged in mask mutation.

For example, the steps of the mask mutation for a three-task four-resource problem are described below. X_{best} and X_1 were selected and shown below for mask mutation:

$$X_{best} = \{(1,R_2),\ (3,R_4),\ (2,R_2),\ (1,R_4),\ (2,R_1),\ (3,R_2),\ (2,R_4),\ (3,R_3),\ (1,R_1)\}$$

$$X_1 = \{(2,R_1),\ (1,R_3),\ (3,R_2),\ (3,R_2),\ (2,R_4),\ (1,R_3),\ (1,R_2),\ (3,R_4),\ (2,R_1)\}$$

The set S included three-task elements is $S = \{1, 2, 3\}$. The S is randomly divided into the two sets S_1 and S_2, $S_1 = \{1, 3\}$ and $S_2 = \{2\}$, respectively. The algorithm for mask mutation is shown below.

```
Begin
    for j=1 to 9
      if (element jth of X_best)∈S_1 then
        V_1 and V_2←element jth of X_best
      end
      if (element jth of X_1)∈S_2 then
        V_1 and V_2←element jth of X_1
      end
    end
End
```

The results in V_1 and V_2 are shown below, respectively:

$$V_1 = \{(1,R_2),\ (2,R_4),\ (3,R_2),\ (1,R_4),\ (2,R_1),\ (3,R_2),\ (3,R_4),\ (1,R_3),\ (2,R_1)\}$$

$$V_2 = \{(1,R_1),\ (2,R_3),\ (3,R_2),\ (1,R_2),\ (2,R_4),\ (3,R_3),\ (3,R_2),\ (1,R_4),\ (2,R_1)\}$$

4.1.4. Crossover operation

The IDEA for crossover only recombines the resource allocation and the tasks remain unchanged. Each mutant vector V_i^G recombines with its respective parent X_i^G through crossover operation to produce its offspring vector U_i^G. The genes of U_i^G are inherited from V_i^G and X_i^G, determined by the crossover rate p_c, in which $p_c \in [0, 1]$, by using Eq. (3.4).

For example, the crossover operation for a three-task four-resource problem is described below. X_1 and V_1 were selected and shown below for resource crossover. p_c was 0.8.

$$X_1 = \{(2,R_1),\ (1,R_3),\ (3,R_2),\ (3,R_2),\ (2,R_4),\ (1,R_3),\ (1,R_2),\ (3,R_4),\ (2,R_1)\}$$

$$V_1 = \{(1,R_2),\ (2,R_4),\ (3,R_2),\ (1,R_4),\ (2,R_1),\ (3,R_2),\ (3,R_4),\ (1,R_3),\ (2,R_1)\}$$

Let a random sequence of nine numbers from the range [0, 1] is 0.98, 0.24, 0.08, 0.82, 0.17, 0.65, 0.35, 0.95, and 0.46. The result in U_1 is shown below:

$$U_1 = \{(1,R_1),\ (2,\ R_4),\ (3,R_2),\ (1,R_2),\ (2,R_1),\ (3,R_2),\ (3,R_4),\ (1,R_4),\ (2,R_1)\}$$

4.1.5. Generating improved offspring by Taguchi method

Since the mask mutation operator Θ can affect the performance of the IDEA, performance improvements are made by using orthogonal mask Θ instead of random mask Θ for systematic reasoning. An orthogonal mask Θ was motivated by an OA of the Taguchi method, which uses an OA to study a large number of decision variables with a small number of experiments. The best combinations of decision factors are determined by the OA and by the signal-to-noise ratios. The key concept of the Taguchi method is to maximize the signal-to-noise ratios, which are used as a performance measure, by using the OA to run a partial set of experiments.

The Taguchi method generates a better trial individual U_t^G by using a target vector X_i^G and a mutated vector V_i^G to conduct matrix experiments of a 2OA and to build the response table. The orthogonal mask Θ is from an integer set $\tilde{U}$ corresponding to the number of tasks. The $\tilde{U}$ is divided into two sets $\tilde{U}_1$ and $\tilde{U}_2$ by using a 2OA, and the resource allocation remains unchanged. After analysis of the $E_{\beta l}$ yields the important factors, these factors are used to find the best combination after a Taguchi trial. For the two-level problem, when $E_{\beta 1} > E_{\beta 2}$ indicates that the first level factor of this factor has the greatest influence, where $\beta = 1, 2, \ldots, Q$, the factor is placed in set $\tilde{U}_1$. Otherwise, the second-level factor has greatest influence, and this factor is place in set $\tilde{U}_2$. This sequential comparison process yields the best combination [36]. The detailed steps for the matrix experiment are described as follows.

```
Begin
    Generate an integer set Ũ corresponding to the number of tasks
    Ũ = Ũ_1 ∪ Ũ_2 and Ũ_1 ∩ Ũ_2 = ∅
    for no. of Ũ
        if element of Ũ ↔ element "1" of 2OA
            then Ũ_1 ← element of Ũ
            else Ũ_2 ← element of Ũ
        end
    end
    for j = 1 to no. of individual length
        if (element j^th of X_i^G) ∈ Ũ_1 then
            U_i ← element j^th of X_i^G
        end
        if (element j^th of V_i^G) ∈ Ũ_2 then
            U_i ← element j^th of V_i^G
        end
    end
End
```

Then, conduct each experiment of a 2OA by using Algorithm 2, and compute the fitness value and the signal-to-noise ratio for each individual. Compare the effects of the various factors ($E_{\beta 1}$ and $E_{\beta 2}$), where $\beta = 1, 2, \ldots, Q$. When $E_{\beta 1} > E_{\beta 2}$ indicates that the first level factor of this factor has the greatest influence, where $\beta = 1, 2, \ldots, Q$, the factor is placed in set $\tilde{U}_1$. Otherwise, the second-level factor has greatest influence, and this factor is place in set $\tilde{U}_2$.

For example, for a five-task four-resource problem, each task has two subtasks. The $L_8(2^7)$ OA shown in Table 2 was selected. Here the tasks $\tilde{U} = \{1, 2, 3, 4, 5\}$ correspond to the OA factors A, B,

Table 2
$L_8(2^7)$ OA.

Experiment number	Factors						
	A	B	C	D	E	F	G
	Column numbers						
	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
2	1	1	1	2	2	2	2
3	1	2	2	1	1	2	2
4	1	2	2	2	2	1	1
5	2	1	2	1	2	1	2
6	2	1	2	2	1	2	1
7	2	2	1	1	2	2	1
8	2	2	1	2	1	1	2

C, D, E, and F. X_1 and V_1 were selected and shown below for Taguchi recombination. The method of generating sets $\tilde{U}_1$ and $\tilde{U}_2$ and the new individual are described as follows:

$$\tilde{U} = \{\,1 \quad 2 \quad 3 \quad 4 \quad 5\,\}$$

$$\Downarrow\tilde{U}_1\ \Downarrow\tilde{U}_2\ \Downarrow\tilde{U}_2\ \Downarrow\tilde{U}_1\ \Downarrow\tilde{U}_1 \quad \Longrightarrow \quad \tilde{U}_1=\{1, 4, 5\} \text{ and } \tilde{U}_2=\{2, 3\}$$

e.g., Experiment 3: 1 2 2 1 1

$$X_1 = \{(2,R_1),(1,R_3),(3,R_2),(3,R_2),(2,R_4),(1,R_3),(4,R_2),(5,R_4),(5,R_1),(4,R_3)\}$$
$$V_1 = \{(1,R_2),(2,R_4),(5,R_3),(3,R_2),(4,R_3),(1,R_4),(2,R_1),(3,R_2),(5,R_4),(4,R_1)\}$$
$$\Downarrow$$
$$U_1 = \{(1,R_2),(2,R_4),(3,R_3),(1,R_2),(4,R_3),(2,R_4),(5,R_1),(3,R_2),(5,R_4),(4,R_1)\}$$

4.2. Steps of the IDEA

Below are the detailed steps of the proposed IDEA for task scheduling and resource allocation.

1. Set parameters: population size p_s, crossover rate p_c, and function calls N_g.
2. Generate p_s initial feasible vectors X_i^1 ($i=1, 2, \ldots, p_s$) by using Algorithm 1, and evaluate their fitness values by using Eqs. (4.1) and (4.2).
3. Set the iteration index $G=1$.
4. Set the target index $i=1$.
5. Generate the mutated vector V_i^G by using the mask mutation.
6. Based on crossover rate p_c, obtain trial vector U_i^G inherited from X_i^G and V_i^G, and use Eqs. (4.1) and (4.2) to evaluate its fitness value.
7. Set the target index $i=i+1$. If $i > p_s$, then go to Step 8. Otherwise, return to Step 5.
8. Set the target index $t=1+p_s$.
9. Obtain the Taguchi trial vector U_t^G by implementing Algorithm 2.
10. Let the target index $t=t+1$. If $t > ((p_s/10)+p_s)$, then go to Step 11. Otherwise, return to Step 9.
11. For X_i^G, U_i^G, and U_t^G populations, sort the fitness values in increasing order.
12. Select the best p_s individuals as parents X_i^{G+1}($i=1, 2, \ldots, p_s$) of the next generation.
13. Let the iteration index $G=G+1$. If $G > N_g$, and go to Step 14. Otherwise, return to Step 4 and continue through Step 13.
14. Stop, and display the best individual and fitness value.

5. Design examples and comparisons

The performance index for measuring the quality of Pareto-optimal sets and two examples were used for a performance comparison of the IDEA, DEA (i.e., IDEA without the Taguchi method), and NSGA-II.

5.1. Performance index (PI)

Various performance indices for measuring the quality of Pareto-optimal sets have been proposed to compare the performance of different multi-objective optimization algorithms [41,44,20,7,6,42]. The four performance indices are often used in terms of the coverage ratio of two sets, the distance-based distribution, the maximum spread, and hyperarea (or hypervolume used with dimension above two) ratio [4,26,25,37,16,1]. Therefore, they are used in the study for measuring the quality of Pareto-optimal sets and described below.

Performance index of coverage ratio for measuring the quality of non-dominated sets obtained from different optimization algorithms was adopted in terms of the coverage ratio of two sets. In a minimization problem, Vector $\vec{a}$ dominates vector $\vec{b}$, notated as $\vec{a} \succ \vec{b}$, if and only if $\forall i$; $f_i(\vec{a}) \leq f_i(\vec{b})$ and $\exists i$; $f_i(\vec{a}) < f_i(\vec{b})$. When two non-dominated solution sets obtained by different optimizers are compared, the coverage ratio of two sets is shown below:

$$C(S_1,\ S_2) = |\{\, s_2 \in S_2 \ ;\ \exists\, s_1 \in S_1 \ :\ s_1 \succ s_2 \,\}|/|S_2|$$

where $C(S_1, S_2)$ is the value of ratio. The numerator, on the right-hand side of the equal sign, describes the number of the solutions S_2 dominated by S_1 and the denominator expresses the number of all solution S_2. It is worth mentioning that $C(S_1, S_2)$ does not have to be equal to $1-C(S_2, S_1)$i. Thus, both $C(S_1, S_2)$ and $C(S_2, S_1)$ should be given. The larger the coverage ratio is, the better the performance is.

Performance index for distribution can be calculated based on distance. The distance-based distribution PI is spacing (SP). The smaller the SP is, the better the performance is:

$$SP(S) = \sqrt{\frac{1}{|S|-1}\sum_{i=1}^{|S|}(d_i-\overline{d})^2}$$

$$d_i = \min_{s_k \in S \wedge s_k \neq s_i} \sum_{m=1}^{M} |f_m(s_i)-f_m(s_k)|$$

where $\overline{d}$ is the average of d_i. It is worth noticing that the distance is computed by the sum of absolute differences along each axis. Only the shortest distance from each point is used without sorting. The distance is summed up from $i=1$ to $|S|$. M is the number of objectives.

Performance index for measuring maximum spread of S is the distance (D) between the boundary solutions in S. The larger the D

is, the better the performance is:

$$D=\sqrt{\sum_{m=1}^{M}\left(\max_{i=1}^{|S|}f_m^i-\min_{i=1}^{|S|}f_m^i\right)^2}$$

Hyperarea (*HA*) relates to the size of the area that is dominated by *S*. The larger the area the solutions can dominate in the fitness space, the better. A normalized *HA* can be defined by

$$HA=\{\cup_i area_i|S_i\in S\}$$

where S_i is a non-dominated vector and $area_i$ is the area between vector S_i and the origin *O*'. The drawbacks of the PI are that *O*' needs to be given and that convex parts of the Pareto front are more preferable than concave parts.

5.2. Example 1: a five-task five-resource problem

There is a five-task five-resource problem for task scheduling and resource allocation on cloud computing. The five tasks contain 20 subtasks and five resources are available to perform these subtasks. The task set is $J=(J_1, J_2, J_3, J_4, J_5)$, where $J_1=(J_{11}, J_{12}, J_{13}, J_{14}, J_{15})$, $J_2=(J_{21}, J_{22}, J_{23})$, $J_3=(J_{31}, J_{32}, J_{33}, J_{34})$, $J_4=(J_{41}, J_{42})$, and $J_5=(J_{51}, J_{52}, J_{53}, J_{54}, J_{55}, J_{56})$. The five-resource set is $R=(R_1, R_2, R_3, R_4, R_5)$. Data size (*DS*) of a subtask is described in Table 3. Table 4 demonstrates rent cost (C_{Rent_P} and C_{Rent_R}) and processing capacity on each available resource. Table 5 shows processing time (T_{Exe}) of J_1 on each available resource. Other processing time for subtasks on each available resource could be computed by *DS* divided by processing capacity. Let transfer rate be 10 Mbps for uploading and downloading. The receiving time (T_{Rece}) for a subtask is computed by *DS* divided by 10 Mbps. The 2OA $L_8(2^7)$ was chosen for the IDEA to deal with the matrix experiments since the five tasks occur less than the seven factors in $L_8(2^7)$. The objective was simultaneously to minimize Eqs. (4.1) and (4.2) in finding the task scheduling to available resources.

The parameter settings were set by parameter sensitivity analysis for IDEA, DEA, and NSGA-II. Overall, the IDEA and DEA obtained good quality results in different 30 runs, especially when the settings for population size, crossover rate, and generation numbers were set to 100, 0.9, and 1000 (400 for the IDEA), respectively. These settings for population size, crossover rate, mutation rate, and generation numbers in the NSGA-II were set to 100, 0.9, 0.025, and 1000. Table 6 compares the proposed IDEA with the DEA and NSGA-II in terms of mean coverage ratio. The mean coverage ratio is computed by the average of $C(x_i, y_j)$, $1\le i\le 30$, $1\le j\le 30$, where *x* and *y* represent IDEA, DEA, and NSGA-II, and *x* and *y* is different. Each data set $C(x, y)$ has 900 values. The mean coverage ratios *C*(IDEA, DEA) (standard deviation) of 0.368 (0.162) and *C*(IDEA, NSGA-II) of 0.300 (0.132) are superior to the ratios *C*(DEA, IDEA) of 0.249 (0.157) and *C*(NSGA-II, IDEA) of 0.288 (0.150), respectively. Table 7 shows the results of performance comparisons of IDEA, DEA, and NSGA-II in terms of *SP*, *D*, and *HA* in different 30 runs of the five-task five-resource problem. The smaller the *SP* is and the larger the *D* and the *HA* are, the better. Therefore, the *SP*, *D*, and *HA* obtained by the IDEA are superior to those obtained by the DEA and NSGA-II. In summary, the above experimental results confirm that the IDEA outperforms both the DEA and NSGA-II in finding the better Pareto-optimal solutions.

García et al. [13] confirmed the use of the powerful non-parametric statistical tests to carry out comparisons for optimization methods. This study used Wilcoxon matched-pairs signed-rank test to compare two algorithms in single-problem analysis [39,13]. Let d_i be the difference between the performance scores of the two algorithms on *i*th out of *n* different runs. The differences are ranked according to their absolute values and average ranks are assigned in case of ties. Let T^+ be the sum of ranks for the different runs on which the second algorithm outperformed the first, and

Table 3
DS (unit: Giga Bytes, GB) of each subtask.

J_i	J_{i1}	J_{i2}	J_{i3}	J_{i4}	J_{i5}	J_{i6}
J_1	0.5	0.9	0.7	1.2	0.5	–
J_2	0.8	1.4	1.0	–	–	–
J_3	1.2	1.5	0.8	0.6	–	–
J_4	1.8	1.4	–	-	–	–
J_5	1.3	1.0	0.4	0.8	0.6	0.7

Table 4
Rent cost and processing capacity on an available resource.

Resource no.	Rent cost (USD/per hour)	Processing capacity (GB/per hour)
R_1	0.12	0.2
R_2	0.13	0.25
R_3	0.48	0.8
R_4	0.52	0.85
R_5	0.96	1.6

Table 6
Results of comparisons in terms of mean coverage ratio and Wilcoxon test in the five-task five-resource problem.

$C(S_1, S_2)$	Mean coverage ratio (standard deviation)
C(IDEA, DEA)	0.368 (0.162)
C(IDEA, NSGA-II)	0.300 (0.132)
C(DEA, IDEA)	0.249 (0.157)
C(DEA, NSGA-II)	0.357 (0.156)
C(NSGA-II, IDEA)	0.288 (0.150)
C(NSGA-II, DEA)	0.232 (0.117)
C(NSGA-II, SPEA2)	0.205 (0.148)
C(NSGA-II, IBEA)	0.197 (0.165)
C(SPEA2, NSGA-II)	0.203 (0.151)
C(IBEA, NSGA-II)	0.195 (0.159)
Wilcoxon test	*z* value (*p* value)
C(IDEA, DEA) vs. C(DEA, IDEA)	14.31 ($p<0.00001$)
C(IDEA, NSGA-II) vs. C(NSGA-II, IDEA)	2.145 ($p=0.0319$)
C(DEA, NSGA-II) vs. C(NSGA-II, DEA)	17.35 ($p<0.00001$)
C(NSGA-II, SPEA2) vs. C(SPEA2, NSGA-II)	1.163 ($p=0.2457$)
C(NSGA-II, IBEA) vs. C(IBEA, NSGA-II)	1.216 ($p=0.2238$)

Table 5
Processing time (T_{Exe}) of J_1 subtask on each available resource.

Resource no.	J_{11}	J_{12}	J_{13}	J_{14}	J_{15}
R_1	2.5 (=0.5/0.2)	4.5 (=0.9/0.2)	3.5 (=0.7/0.2)	6.0 (=1.2/0.2)	2.5 (=0.5/0.2)
R_2	2.0 (=0.5/0.25)	3.6	2.8	4.8	2.0
R_3	0.625 (=0.5/0.8)	1.125	0.875	1.5	0.625
R_4	0.588 (=0.5/0.85)	1.059	0.824	1.412	0.588
R_5	0.313 (=0.5/1.6)	0.563	0.438	0.75	0.313

T^- the sum of ranks for the opposite. Ranks of $di=0$ are split evenly among the sums and if there is an odd number of them, one is ignored:

$$T^+ = \sum_{d_i>0} \text{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \text{rank}(d_i) \text{ and } T^- = \sum_{d_i<0} \text{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \text{rank}(d_i)$$

Let T be the smaller of the two values, $T=\min(T^+, T^-)$. If T is less than or equal to the value of the distribution of Wilcoxon for n degrees of freedom, the null hypothesis of equality of means is rejected [13]. Also, to calculate the significance of the test statistic (T), the mean ($\overline{T}$) and standard error (SE_T) was defined as follows [11]:

$$\overline{T} = \frac{n(n+1)}{4} \text{ and } SE_T = \sqrt{\frac{n(n+1)(2n+1)}{24}}$$

where n is the sample size. Therefore, $z=(T-\overline{T})/SE_T$. If z is bigger than 1.96 (ignoring the minus sign) then it is significant at $p<0.05$.

Table 7
Results of performance comparisons of IDEA, DEA, NSGA-II, SPEA2, and IBEA in terms of *SP*, *D*, and *HA* in the five-task five-resource problem.

Algorithm	*SP* Average (standard deviation)	*D* Average (standard deviation)	*HA* Average (standard deviation)
IDEA	0.0125 (0.0011)	0.5945 (0.0365)	0.9538 (0.0031)
DEA	0.0197 (0.0019)	0.5327 (0.0434)	0.9536 (0.0038)
NSGA-II	0.0213 (0.0028)	0.5113 (0.0455)	0.9529 (0.0041)
SPEA2	0.0219 (0.0031)	0.5135 (0.0457)	0.9526 (0.0045)
IBEA	0.0226 (0.0037)	0.5109 (0.0501)	0.9531 (0.0039)

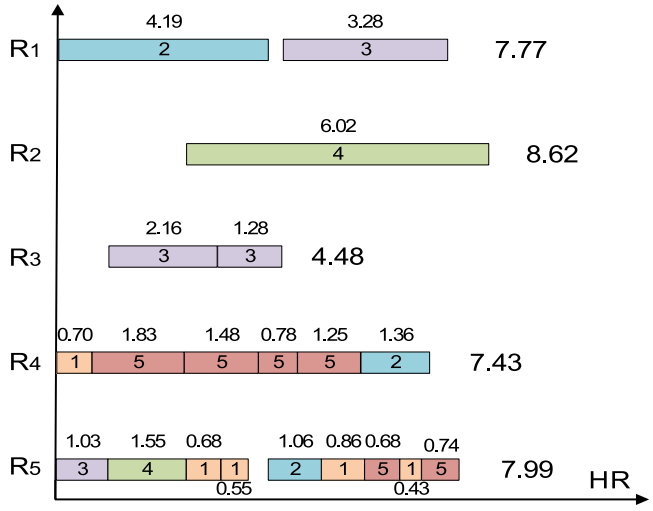

Fig. 2. The Gantt chart of the task scheduling in terms of having smaller makespan in the five-task five-resource problem.

The Wilcoxon test focused on the algorithm, IDEA, which had the higher mean coverage ratios. We studied the behavior of this IDEA with respect to the remaining algorithms (DEA and NSGA-II). The sample data came from using the IDEA, DEA, and NSGA-II at their coverage ratios in 30 independent runs. Each data set $C(x, y)$ has 900 values. Friedman's test gave the Chi square value of 606.06 and p-value of less than 0.00001 in the six data sets, which are C (IDEA, DEA), C(DEA, IDEA), C(IDEA, NSGA-II), C(NSGA-II, IDEA), C (DEA, NSGA-II), and C(NSGA-II, DEA). Given the p-value is lower than the level of significance considered $\alpha=0.05$, there is significant difference among the results in these test data. Attending to these results, a post-hoc statistical analysis could detect concrete differences among methods. The Wilcoxon z values in C(IDEA, DEA) vs. C(DEA, IDEA), C(IDEA, NSGA-II) vs. C(NSGA-II, IDEA), and C (DEA, NSGA-II) vs. C(NSGA-II, DEA) are 14.31 ($p<0.00001$), 2.145 ($p=0.0319$), and 17.35 ($p<0.00001$), respectively, as shown in Table 6. There is a significant difference between two algorithms, because Wilcoxon z value is bigger than 1.96. That is, the performance of IDEA really outperforms the performances of DEA and NSGA-II in finding the better Pareto-optimal solutions. Also, the performance of DEA is superior to that of NSGA-II.

The results of the three cases were selected by the IDEA as examples in describing their Gantt charts of task scheduling in terms of having smaller makespan, cost, and both. Figs. 2–4 show the Gantt charts of the task scheduling in terms of having smaller makespan, cost, and both. The makesapn and cost are 8.62 (hours) and 14.52 (USD), respectively, as shown in Fig. 2, when the objective is to have a smaller makespan. However, they are 66.14 (hours) and 10.85 (USD), respectively, as shown in Fig. 3, when the objective is to have a smaller cost. They are 23.74 (hours) and 12.27 (USD), respectively, as shown in Fig. 4, by considering trade-off between makespan and cost.

5.3. Example 2: a ten-task ten-resource problem

There is a ten-task ten-resource problem for task scheduling and resource allocation on cloud computing. The ten tasks contain 100 subtasks and 10 resources are available to perform these subtasks. The task set is $J=(J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8, J_9, J_{10})$, where $J_i=(J_{i1}, J_{i2}, J_{i3}, J_{i4}, J_{i5}, J_{i6}, J_{i7}, J_{i8}, J_{i9}, J_{i10})$ and $i=1, 2, \ldots, 10$. The ten-resource set is $R=(R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9, R_{10})$. *DS* of a subtask is described in Table 8. Table 9 demonstrates rent cost and processing capacity on each available resource. The processing time for subtasks on each available resource could be computed by *DS* divided by processing capacity. The 2OA $L_{16}(2^{15})$ was chosen for the IDEA to deal with the matrix experiments since the ten tasks occur less than the 15 factors in $L_{16}(2^{15})$. The objective was simultaneously to minimize Eqs. (4.1) and (4.2) in finding the task scheduling to available resources.

The parameter settings for IDEA, DEA, and NSGA-II in different 30 runs were the same as those of the five-task five-resource problem except the generation numbers were set to 5000 for the DEA and NSGA-II and 2000 for the IDEA. The makesapn and cost are 50.53 (hours) and 111.64 (USD), respectively, when the objective is to have a smaller makespan. However, they are 235.5

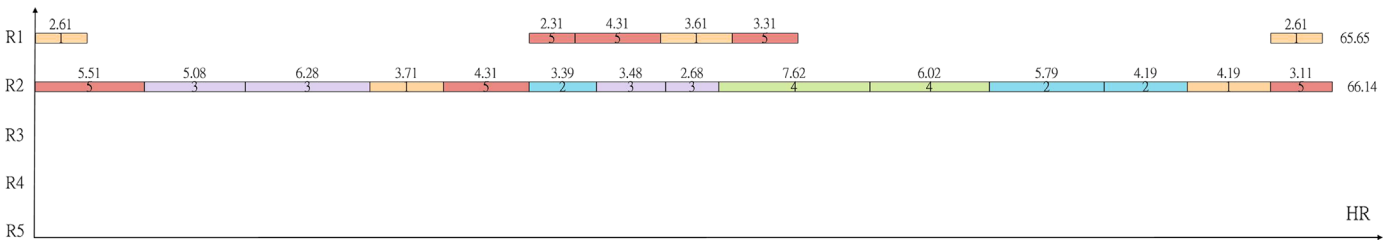

Fig. 3. The Gantt chart of the task scheduling in terms of having smaller cost in the five-task five-resource problem.

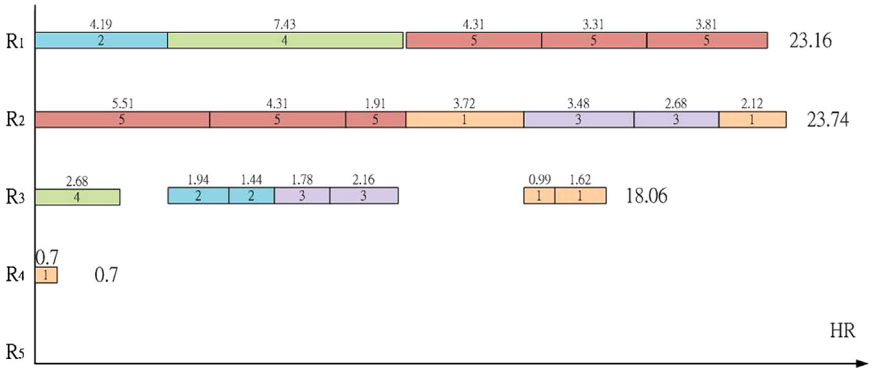

Fig. 4. The Gantt chart of the task scheduling in terms of considering trade-off between makespan and cost in the five-task five-resource problem.

Table 8
DS (unit: Giga Bytes, GB) of each subtask.

J_i	J_{i1}	J_{i2}	J_{i3}	J_{i4}	J_{i5}	J_{i6}	J_{i7}	J_{i8}	J_{i9}	J_{i10}
J_1	2.15	1.99	1.33	1.83	1.22	0.35	2.9	1.55	2.09	2.53
J_2	0.32	1.12	1.98	1.81	2.47	2.06	1.46	0.71	1.07	2.77
J_3	1.32	0.51	2.91	2.47	1.04	0.51	0.33	2.02	1.27	1.08
J_4	0.31	2.37	0.68	0.73	1.08	2.24	2.44	2.08	0.61	1.44
J_5	1.21	2.66	1.64	2.22	2.31	0.45	2.14	1.28	0.78	1.32
J_6	2.77	2.29	1.28	2.82	0.78	0.6	0.36	0.73	1.91	2.55
J_7	1.52	2.45	2.74	1.88	2.19	0.39	2.27	2.7	0.99	2
J_8	1.6	2.19	1.32	2.3	2.86	0.34	0.83	2.65	0.33	2.45
J_9	1.45	2.12	2.18	2.77	1.57	2.29	0.32	1.57	1.39	2.64
J_{10}	2.07	0.39	0.3	2.8	1.61	2.14	1.79	0.42	0.81	2.11

Table 9
Rent cost and processing capacity on an available resource.

Resource no.	Rent cost (USD/per hour)	Processing capacity (GB/per hour)
R_1	0.12	0.2
R_2	0.12	0.2
R_3	0.13	0.25
R_4	0.48	0.8
R_5	0.48	0.8
R_6	0.52	0.85
R_7	0.52	0.85
R_8	0.96	1.6
R_9	0.96	1.6
R_{10}	1.04	1.9

(hours) and 97.57 (USD), respectively, when the objective is to have a smaller cost. They are 88.47 (hours) and 105.99 (USD), respectively, by considering trade-off between makespan and cost. Table 10 compares the proposed IDEA with the DEA and NSGA-II in terms of mean coverage ratio. The mean coverage ratio is computed by the average of $C(x_i, y_j)$, $1 \le i \le 30$, $1 \le j \le 30$, where x and y represent IDEA, DEA, and NSGA-II, and x and y is different. Each data set $C(x, y)$ has 900 values. The mean coverage ratios C(IDEA, DEA) (standard deviation) of 0.506 (0.262) and C(IDEA, NSGA-II) of 0.701 (0.205) are superior to the ratios C(DEA, IDEA) of 0.286 (0.174) and C(NSGA-II, IDEA) of 0.052 (0.066), respectively. The Wilcoxon test focused on the algorithm, IDEA, which had the higher mean coverage ratios. We studied the behavior of this IDEA with respect to the remaining algorithms (DEA and NSGA-II). The sample data came from using the IDEA, DEA, and NSGA-II at their coverage ratios in 30 independent runs. Each data set $C(x, y)$ has 900 values. Friedman's test gave the Chi square value of 3435.2 and p-value of less than 0.00001 in the six data sets, which are C (IDEA, DEA), C(DEA, IDEA), C(IDEA, NSGA-II), C(NSGA-II, IDEA), C (DEA, NSGA-II), and C(NSGA-II, DEA). Given the p-value is lower than the level of significance considered $\alpha = 0.05$, there is significant difference among the results in these test data. Attending to these results, a post-hoc statistical analysis could detect concrete differences among methods. The Wilcoxon z values in C(IDEA, DEA) vs. C(DEA, IDEA), C(IDEA, NSGA-II) vs. C(NSGA-II, IDEA), and C (DEA, NSGA-II) vs. C(NSGA-II, DEA) are 18.42 ($p < 0.00001$), 25.98 ($p < 0.00001$), and 25.99 ($p < 0.00001$), respectively, as shown in Table 10. There is a significant difference between two algorithms, because Wilcoxon z value is bigger than 1.96. That is, the performance of IDEA really outperforms the performances of DEA and NSGA-II in finding the better Pareto-optimal solutions. Also, the performance of DEA is superior to that of NSGA-II. Table 11 shows the results of performance comparisons of IDEA, DEA, and NSGA-II in terms of *SP*, *D*, and *HA* in different 30 runs of the ten-task ten-resource problem. The smaller the *SP* is and the larger the *D* and the *HA* are, the better. Therefore, the *SP*, *D*, and *HA* obtained by the IDEA are superior to those obtained by the DEA and NSGA-II.

5.4. Discussions among multi-objective evolutionary algorithms for the two cloud scheduling problems

For providing more comparisons with other state-of-the-art multi-objective evolutionary algorithms to satisfy the curiosity of the researchers, SPEA2 [43] and IBEA [40] were used to solve the cloud scheduling problems. The mean coverage ratios C(NSGA-II, SPEA2), C(NSGA-II, IBEA), C(SPEA2, NSGA-II), and C(IBEA, NSGA-II) in the five-task five-resource and ten-task ten-resource problems are shown in Tables 6 and 10, respectively. Those values have shown no significant differences. The Wilcoxon z values in C (NSGA-II, SPEA2) vs. C(SPEA2, NSGA-II) and C(NSGA-II, IBEA) vs. C(IBEA, NSGA-II) are all less than 1.96 ($p > 0.05$), as shown in Tables 6 and 10. There are no statistically significant differences in C(NSGA-II, SPEA2) vs. C(SPEA2, NSGA-II) and C(NSGA-II, IBEA) vs. C (IBEA, NSGA-II). The performance results of SPEA2 and IBEA in terms of *SP*, *D*, and *HA* in different 30 runs in the five-task five-resource and ten-task ten-resource problems are shown in Tables 7 and 11, respectively. For the three performance measures, NSGA-II, SPEA2 and IBEA approaches have shown no significant performance differences in the five-task five-resource and ten-task ten-resource problems.

The mutation operation with DEA is performed by combinations of individuals rather than perturbing the genes in individuals with small probability as compared with GAs. DEA uses a non-uniform crossover instead of one randomly cut-point crossover as GAs. The most important characteristics of DEA, mutation and crossover, pass the objective functions topographical information toward the optimization process, and provide more efficient global optimization capability [30,31]. Therefore, in order to develop an effective method for cloud scheduling problems, the most

Table 10
Results of comparisons in terms of mean coverage ratio and Wilcoxon test in the ten-task ten-resource problem.

$C(S_1, S_2)$	Mean coverage ratio (standard deviation)
C(IDEA, DEA)	0.506 (0.262)
C(IDEA, NSGA-II)	0.701 (0.205)
C(DEA, IDEA)	0.286 (0.174)
C(DEA, NSGA-II)	0.696 (0.201)
C(NSGA-II, IDEA)	0.052 (0.066)
C(NSGA-II, DEA)	0.154 (0.102)
C(NSGA-II, SPEA2)	0.216 (0.152)
C(NSGA-II, IBEA)	0.209 (0.148)
C(SPEA2, NSGA-II)	0.218 (0.149)
C(IBEA, NSGA-II)	0.211 (0.152)
Wilcoxon test	z value (p value)
C(IDEA, DEA) vs. C(DEA, IDEA)	18.42 ($p < 0.00001$)
C(IDEA, NSGA-II) vs. C(NSGA-II, IDEA)	25.98 ($p < 0.00001$)
C(DEA, NSGA-II) vs. C(NSGA-II, DEA)	25.99 ($p < 0.00001$)
C(NSGA-II, SPEA2) vs. C(SPEA2, NSGA-II)	0.947 ($p = 0.3424$)
C(NSGA-II, IBEA) vs. C(IBEA, NSGA-II)	0.808 ($p = 0.4258$)

Table 11
Results of performance comparisons of IDEA, DEA, NSGA-II, SPEA2 and IBEA in terms of *SP*, *D*, and *HA* in the ten-task ten-resource problem.

Algorithm	*SP* Average (standard deviation)	*D* Average (standard deviation)	*HA* Average (standard deviation)
IDEA	0.0173 (0.0013)	0.6213 (0.0437)	0.9575 (0.0038)
DEA	0.0325 (0.0027)	0.5164 (0.0502)	0.9499 (0.0041)
NSGA-II	0.0389 (0.0039)	0.4727 (0.0593)	0.9326 (0.0052)
SPEA2	0.0392 (0.0041)	0.4821 (0.0597)	0.9312 (0.0058)
IBEA	0.0401 (0.0047)	0.4716 (0.0601)	0.9418 (0.0049)

important characteristics of DEA are preserved and used for the IDEA compared to NSGA-II with its operators.

NSGA varies from simple GA only in the way the selection operation works. The crossover and mutation operators remain as usual [28]. NSGA-II improves NSGA by using elitism, and keeps diversity without specifying any parameters by using a crowded tournament selection operator [8,9]. In the study, the operators of the NSGA-II use one randomly cut-point crossover on two chromosomes for resources and randomly swap mutation on one chromosome for subtasks, while those of the DEA use probability-based crossover on two individuals for resources and three vectors (including the best vector at present) combination with mask operator Θ for subtasks. The ability of the DEA on producing diverse offspring is better than that of the NSGA-II, because more vectors join in mutation. In macro-space, the DEA and NSGA-II have a powerful global exploration capability, but they lack of local search ability. The mechanism of the IDEA combines the DEA with improved operators and the Taguchi method with orthogonal mask Θ. In the computational experiment by using the systematic reasoning ability of the Taguchi method, it was confirmed that a new individual generated by each matrix experiment is superior to its two parents [33–35]. That is, potential individuals in micro-space can be exploited. In micro-space, the systematic reasoning mechanism of the orthogonal array with signal-to-noise ratio enhanced the performance of the IDEA by accelerating convergence to the global solution. The IDEA is well enhanced and balanced on exploration and exploitation. Therefore, the IDEA shows its effectiveness to optimize task scheduling and resource allocation compared with both the DEA and the NSGA-II.

6. Conclusions

By introducing the Taguchi method within the DEA framework, the proposed IDEA approach preserves the global solution finding merit of the DEA as a design tool to optimize task scheduling and resource allocation on cloud computing environment. The major contribution of the IDEA is the use of an OA of the Taguchi method as mask mutation operator to account for task-subtask encoding in order to generate improved offspring. Doing so obtains Pareto-optimal solutions within two objectives based on the proposed cost and time models so that further steps can be taken based on the non-dominated sorting technique. Performance comparisons of the proposed IDEA in the two cases in this study, which included the five-task five-resource and the ten-task ten-resource problems, confirm its high effectiveness and easy optimization. Furthermore, the results of the three cases at each task example were described in their Gantt charts of task scheduling in terms of having smaller makespan, cost, and both. They can provide decision makers an aid to make their decision when conflicting objectives are present.

Acknowledgment

This work was supported in part by the National Science Council, Taiwan, ROC, under Grant Numbers NSC99-2221-E-151-071-MY3, NSC100-2221-E153-001 and NSC101-2221-E153-003.

References

[1] Araujo DRB, Bastos-Filho CJA, Barboza EA, Chaves DAR, Martins-Filho JF. A performance comparison of multi-objective optimization evolutionary algorithms for all-optical networks design. In: IEEE symposium on computational intelligence in multicriteria decision-making; 2011. p. 89–96.
[2] Arora S, Hochbaum L. Approximation algorithms for NP-hard problems, PWS, Boston; 1997.
[3] Amazon Elastic Compute Cloud (Amazon EC2). Amazon Web Services LLC; 2009. ⟨http://aws.amazon.com/ec2/⟩.
[4] Bandyopadhyay S, Pal SK, Aruna B. Multiobjective GAs, quantitative indices, and pattern classification. IEEE Transactions on Systems, Man, and Cybernetics —Part B: Cybernetics 2004;34:2088–99.
[5] Berrichi A, Yalaoui F, Amodeo L, Mezghiche M. Bi-objective ant colony optimization approach to optimize production and maintenance scheduling. Computers & Operations Research 2010;37:1584–96.
[6] Coello CA, Lamont GB, Van Veldhuizen DA. Evolutionary algorithms for solving multi-objective problems. 2nd ed. Heidelberg: Springer; 2007.
[7] Deb K. Multi-objective optimization using evolutionary algorithms. New York: JohnWiley & Sons; 2004.
[8] Deb, K., Agrawal, S., Pratab, A., and Meyarivan, T., 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Proceedings of the parallel problem solving from nature VI conference. Lecture notes in computer science. p. 849–58.
[9] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 2002;6:182–97.
[10] Fanjul-Peyro L, Ruiz R. Size-reduction heuristics for the unrelated parallel machines scheduling problem. Computers & Operations Research 2011;38:301–9.
[11] Field A. Discovering statistics using SPSS. London: SAGE Publications; 2006.
[12] Gacias B, Artigues C, Lopez P. Parallel machine scheduling with precedence constraints and setup times. Computers & Operations Research 2010;37:2141–51.
[13] García S, Molina D, Lozano M, Herrera F. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. Journal of Heuristics 2009;15:617–44.
[14] Gromicho JAS, van Hoorn JJ, Saldanha-da-Gama F, Timmer GT. Solving the job-shop scheduling problem optimally by dynamic programming. Computers & Operations Research 2012;39:2968–77.
[15] Goldberg DE. Genetic algorithms in search, optimization and machine learning. Massachusetts: Addison-Wesley; 1989.
[16] Hsu CH, Tsou CS, Yu FJ. Multicriteria tradeoffs in inventory control using memetic particle swarm optimization. International Journal of Innovative Computing, Information and Control 2009;5:3755–68.
[17] Ho WH, Chou JH, Guo CY. Parameter identification of chaotic systems using improved differential evolution algorithm. Nonlinear Dynamics 2010;61:29–41.

[18] Kim MY, Lee YH. MIP models and hybrid algorithm for minimizing the makespan of parallel machines scheduling problem with a single server. Computers & Operations Research 2012;39:2457–68.
[19] Lin YK, Pfund ME, Fowler JW. Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems. Computers & Operations Research 2011;38:901–16.
[20] Okabe T, Jin Y, Sendhoff B. A critical survey of performance indices for multi-objective optimization. In: Proceedings of the 2003 congress on evolutionary computation; 2003. p. 878–85.
[21] Pandey, S., Wu, L., Guru SM, Buyya R. A particle swarm optimization- based heuristic for scheduling workflow applications in cloud computing environments. In: The 24th IEEE international conference on advanced information networking and applications (AINA); 2010. p. 400–7.
[22] Park SH. Robust design and analysis for quality engineering. London: Chapman & Hall; 1996.
[23] Phadke MS. Quality engineering using robust design. New Jersey: Prentice-Hall; 1989.
[24] Price KV, Storn RM, Lampinen JA. Differential evolution: a practical approach to global optimization. Berlin: Springer-Verlag; 2005.
[25] Reddy M, Kumar D. Multiobjective differential evolution with application to reservoir system optimization. Journal of Computing in Civil Engineering 2007;21:136–46.
[26] Santana-Quintero LV, Coello CAC. An algorithm based on differential evolution for multi-objective problems. International Journal of Computational Intelligence Research 2005;1:151–69.
[27] Senkul P, Toroslu I. An architecture for workflow scheduling under resource allocation constraints. Information Systems Journal 2005;30:399–422.
[28] Srinivas N, Deb K. Multiobjective optimization using nondominated sorting in genetic algorithms. Journal of Evolutionary Computation 1994;2:221–48.
[29] Storn R. System design by constraint adaptation and differential evolution. IEEE Transactions on Evolutionary Computation 1999;3:22–34.
[30] Storn R, Price KV. Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Berkeley, CA, Technical Report TR-95-012; 1995.
[31] Storn R, Price K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization II 1997:341–59.
[32] Taguchi G, Chowdhury S, Taguchi S. Robust engineering. New York: McGraw-Hill; 2000.
[33] Tsai JT, Chou JH, Liu TK. Optimal design of digital IIR filters by using hybrid Taguchi genetic algorithm. IEEE Transactions on Industrial Electronics 2006;53:867–79.
[34] Tsai JT, Chou JH, Liu TK. Tuning the structure and parameters of a neural network by using hybrid Taguchi-genetic algorithm. IEEE Transactions on Neural Networks 2006;17:69–80.
[35] Tsai JT, Liu TK, Chou JH. Hybrid Taguchi-genetic algorithm for global numerical optimization. IEEE Transactions on Evolutionary Computation 2004;8:365–77.
[36] Tsai JT, Liu TK, Ho WH, Chou JH. An improved genetic algorithm for job-shop scheduling problems using Taguchi-based crossover. International Journal of Advanced Manufacturing Technology 2008;38:987–94.
[37] Tsou CS, Chang SC, Lai PO. Using crowding distance to improve multi-objective PSO with local search. In: Felix TS Chan and Manoj Kumar Tiwari, editors. Swarm intelligence, focus on ant and particle swarm optimization; December 1, 2007. CC BY-NC-SA 3.0 license, ISBN: 978-3-902613-09-7.
[38] Warneke D, Kao O. Exploiting dynamic resource allocation for efficient parallel data processing in the cloud. IEEE Transactions on Parallel and Distributed Systems 2011;6:985–97.
[39] Wilcoxon F. Individual comparisons by ranking method. Biometrics 1945;1:80–3.
[40] Zitzler E, Ḱunzli S. Indicator-based selection in multiobjective search. Parallel problem solving from nature (PPSN VIII). Lecture notes in computer science. Birmingham, UK: Springer; 832–42.
[41] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation 1999;3:257–71.
[42] Zitzler E, Knowles J, Thiele L. Quality assessment of pareto set approximations. In: Branke J, Deb K, Miettinen K, Slowinski R, editors. Multiobjective optimization. Lecture notes in computer science, vol. 5252; 2008. p. 373–404.
[43] Zitzler E., Laumanns M., Thiele L. SPEA2: improving the strength pareto evolutionary algorithm. TIK Report Nr. 103. Computer Engineering and Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich; 2001.
[44] Zitzler E, Thiele L, Laumanns M, Fonseca CM, Gurnert da Fonseca V. Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation 2003;7:117–32.