Hindawi
Scientific Programming
Volume 2017, Article ID 5342727, 17 pages
https://doi.org/10.1155/2017/5342727

Research Article

Multiobjective Level-Wise Scientific Workflow Optimization in IaaS Public Cloud Environment

Phyo Thandar Thant,[1] Courtney Powell,[2] Martin Schlueter,[2] and Masaharu Munetomo[2]

[1] *Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan*
[2] *Information Initiative Center, Hokkaido University, Sapporo, Japan*

Correspondence should be addressed to Phyo Thandar Thant; phyothandarthant@ist.hokudai.ac.jp

Received 18 July 2017; Accepted 25 October 2017; Published 21 December 2017

Academic Editor: Emiliano Tramontana

Cloud computing in the field of scientific applications such as scientific big data processing and big data analytics has become popular because of its service oriented model that provides a pool of abstracted, virtualized, dynamically scalable computing resources and services on demand over the Internet. However, resource selection to make the right choice of instances for a certain application of interest is a challenging problem for researchers. In addition, providing services with optimal performance at the lowest financial resource deployment cost based on users' resource selection is quite challenging for cloud service providers. Consequently, it is necessary to develop an optimization system that can provide benefits to both users and service providers. In this paper, we conduct scientific workflow optimization on three perspectives: makespan minimization, virtual machine deployment cost minimization, and virtual machine failure minimization in the cloud infrastructure in a level-wise manner. Further, balanced task assignment to the virtual machine instances at each level of the workflow is also considered. Finally, system efficiency verification is conducted through evaluation of the results with different multiobjective optimization algorithms such as SPEA2 and NSGA-II.

1. Introduction

Scientific experiments in fields such as bioinformatics, astronomy, elementary particle physics, and life science require the storing and processing of big data. The adoption of cloud computing to process these scientific data has increased recently as cloud enables data correlations, pattern mining, data predictions, and data analytics in a cost-efficient manner. Cloud provides resources such as networks, storage, applications, and servers can be allocated from a shared resource pool with minimal management or interaction [1]. In general, cloud computing provides Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) in a pay-per-use model. Of these services, IaaS is the most frequently utilized as it provides customers with an elastic facility to provision or release virtual machines (VMs) in the cloud infrastructure [2]. The elastic nature of cloud facilitates changing of resource quantities and characteristics to vary at runtime, thus dynamically scaling up when there is a greater need for additional resources and scaling down when the demand is low [3]. As a result of its elastic resource provisioning mechanism, the cloud is widely used in several areas including large-scale scientific applications. With improved service support and cloud bursting technologies, users do not need to provision their resources for the worst-case scenarios. In the context of cloud computing, an application's execution cost means the monetary cost of renting resources from the cloud service provider. Addressing the issues of minimizing the resource usage with the best performance is an important research area in an IaaS cloud.

Scientific applications consist of thousands of tasks requiring heavy computation and data transfer. Workflow tasks also have certain dependencies during execution. Modeling of these scientific applications is necessary for effective processing. The most widely used representation model is in the form of directed acyclic graph (DAG) in which the structure of the workflow indicates the order of execution of tasks. Our optimization framework deals with scientific workflow in the form of DAG during optimization experimentations. In addition, it is assumed that the monetary cost for workflow execution in the cloud is based on the amount of resources used, that is, task execution cost correlated to the total number of CPU cycles for all tasks. By using cloud infrastructure, customers can also reduce the processing cost of workflow applications with the aid of the cloud's pay-per-use model. Moreover, a task that shares the same time interval with a previous task hosted in the same instance might not produce extra cost, reducing the overall workflow execution cost.

Khajemohammadi et al. [4] proposed a leveled fast workflow scheduling strategy that minimizes cost and time in a grid environment. However, in that approach, resource utilization is only possible within predefined set of service resources, which is irrelevant because of the dynamicity and elasticity of the cloud. Moreover, a service only processes one task at a time within a level. Our approach differs from the previous work in several aspects: (1) instead of a predefined set of service resources, randomized search on the number of VMs and various machine instance types is conducted during execution to benefit from cloud elasticity and monetary facility, (2) more than one task can share a single virtual machine instance for better resource utilization, (3) balanced task assignment is conducted to balance the load in each machine instance as well as reducing the wait time to proceed to the next level of tasks, and (4) minimizing failure of machine instances to improve the reliability during workflow execution is considered. In that way, the system searches for the Pareto-optimal solutions that exhibit the best performance with minimized failure at the lowest financial cost.

Our optimization idea is based on the widely used scientific workflows from the Pegasus project [5], which have control and dependencies. The experimental results obtained comprise a series of optimized VMs with optimized machine instance type for each level of the workflow with their respective makespan time, cost, and minimized machine instance failure. The current version of this paper is an extension of the previous studies [6] in the following aspects. First, we include another objective function which is machine instance failure minimization function, during optimization. Second, we extend our cloud system model from only four machine instance types of academic private cloud to 25 machine instance types of public cloud. The system implementation now conducts optimization on three minimization objectives (makespan, cost, and failures) with 25 virtual machine instance types from AWS. Third, we perform experiments and analysis on three real-world workflows using SPEA2 and NSGA-II algorithms to show the system effectiveness on three objective functions.

The remainder of this paper is organized as follows. Section 2 presents the theoretical background associated with scientific workflows, SPEA2 and NSGA-II. Section 3 outlines the proposed system, the objective functions used, and the proposed workflow execution optimization algorithm. Section 4 discusses the performance evaluation conducted. Section 5 presents related work. Finally, Section 6 concludes and outlines intended future work.

2. Background

This section presents the relevant theoretical background associated with the paper. First, scientific workflows and their features are discussed. Then, the multiobjective evolutionary algorithms SPEA2 [7] and NSGA-II [8] are presented.

2.1. Scientific Workflows.

Scientific workflows are essential to facilitate and automate the processing of high-volume scientific data in large distributed computing structure such as grid [9]. These workflows describe the set of tasks needed to carry out computational experiments and provide scientists with the ability to expose, share, and reuse their work. The goals of these workflows are (i) to save human cycles by enabling scientists to focus on domain-specific aspects of their work, rather than dealing with complex data management and software issues, and (ii) to save machine cycles by optimizing workflow execution on available resources [10]. Scientific workflows are created with a corresponding workflow editor that usually provides a repository with predefined workflow activities.

In general, a scientific workflow is modeled as a directed acyclic graph (DAG). A DAG workflow, W, is defined as $W = (T, D)$, where $T = \{T_0, T_1, \ldots, T_n\}$ is a set of tasks and $D = \{(T_i, T_j) \mid T_i, T_j \in T\}$ is a set of data or control dependencies [7]. Each task has its respective execution time, task name, input data size, and resulting output data size. For a DAG with multiple entries and exits, it is necessary to add a pseudo T_{entry} and or a pseudo T_{exit} with no control and dependencies. In this study, we assumed that all DAG workflows have only one T_{entry} and one T_{exit}, and optimization is considered for the actual processing steps inside the workflow. A DAG is represented in either XML or JSON format and parsing is required to get the necessary information for further processing. A simple DAG is illustrated in Figure 1 with respective task labels.

2.2. Multiobjective Evolutionary Algorithms (MOEAs).

Multiobjective evolutionary algorithms (MOEAs) have been proposed for finding multiple Pareto-optimal solutions in one single simulation run. MOEAs are an effective way to find the Pareto-optimal solutions among conflicting objectives. They produce nondominated solutions with respect to minimization or maximization of objectives for user satisfaction.

2.2.1. Improved Strength Pareto Evolutionary Algorithm (SPEA2).

SPEA2 [7] is an improved version of the original Strength Pareto Evolutionary Algorithm (SPEA) by means of an improved fitness assignment scheme, nearest neighbor density estimation technique, and new truncation method.

The algorithm starts with a population and an archive (external set). Then it performs iterations for the evolution process. During the update operation in the evolution process, filling up with the dominated individuals or truncating individuals if necessary to fit in the fixed-size archive occurs. Unlike SPEA, SPEA2 only considers members of the archive to participate in the mating selection process.

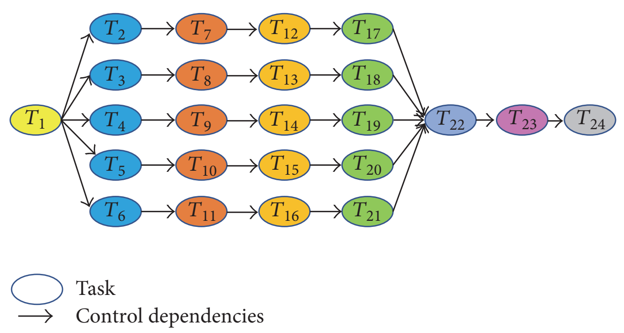


Figure 1: An example DAG workflow.

The objective of the algorithm is to locate and maintain a front of nondominated solutions, ideally a set of Pareto-optimal solutions. This is achieved by using an evolutionary process (with surrogate procedures for genetic recombination and mutation) to explore the search space and a selection process that uses a combination of the degree to which a candidate solution is dominated (strength) and an estimation of the density of the Pareto front as an assigned fitness. An archive of the nondominated set is maintained separate from the population of candidate solutions used in the evolutionary process, providing a form of elitism.

2.2.2. Nondominated Sorting Genetic Algorithm (NSGA-II). In 1995, nondominated sorting genetic algorithm (NSGA) [11], which improved computational complexity via a nonelitism approach and a sharing parameter, was developed. In 2002, Deb et al. developed NSGA-II [8], an improved version of NSGA which rectifies the problems found in NSGA. NSGA-II is a popular multiobjective optimization algorithm that is widely used in several application domains. The main features of NSGA-II are low computational complexity, parameter-less diversity preservation, and an elitism approach. The major components of NSGA-II include fast nondominated sorting and crowding distance assignment.

Fast Nondominated Sorting. Elitism is introduced in NSGA-II by storing all nondominated solutions discovered, starting with the initial population [12]. Elitism enhances the convergence properties towards the Pareto-optimal set. Sorting the individuals in the population according to the level of nondomination, each solution must be compared with every other solution to determine if it is dominated. Firstly, two entities are calculated for each solution: (1) n_p, the number of solutions that dominate solution p, and (2) S_p, a set of solutions that solution p dominates. All solutions that have $n_p = 0$, that is, nondominated solutions, are identified and placed in a current front list as first nondominated front list. Then, the algorithm repeatedly finds the second front, third front, and so on until all fronts are identified or the specified population size is reached.

Crowding Distance Algorithm. After completing nondominated sorting, crowding distance is assigned. In this process, all the individuals in the population are assigned a crowding distance value and individual selection is conducted based on rank and crowding distance. Crowding distance comparison compares the individuals within the same front and distance calculation is conducted. In other words, crowding distance involves finding the Euclidean distance between each individual in a front based on their m objectives in an m-dimensional hyperspace. The individuals in the boundary are always selected because they have infinite distance assignment [13].

The solutions found in NSGA-II are said to be Pareto-optimal if they are not dominated by any other solutions in the solution space. Further, the set of all feasible nondominated solutions in the solution space is referred to as the Pareto-optimal set, and for a given Pareto-optimal set the corresponding objective function values in the objective space are called the Pareto front. The goal of multiobjective optimization is to identify the solutions in the Pareto-optimal set. In this proposed optimization system, we use NSGA-II because it is one of the most popular optimization approaches in several domains. Figure 2 depicts the process flow of NSGA-II optimization.

3. Proposed Workflow Optimization Framework

Search-based workflow optimization on makespan, machine instance failure, and virtual machine deployment cost in cloud computing environment is applied in our system with SPEA2 and NSGA-II. The details of our system are given in the ensuing sections.

3.1. System Overview. Figure 3 shows the architecture of our proposed multiobjective workflow optimization system. The operation of the system is as follows. First, workflow input in the form of a DAX (a DAG in XML representation) is given to the system for optimization and the system parses the necessary information at each level of tasks in the workflow DAG. Then, the system performs makespan minimization, virtual machine deployment cost minimization, and machine instance failure minimization based on the specific machine instance type throughout the search space.

3.2. Level-Wise Optimization. Level-wise workflow optimization is conducted in order not to violate the dependencies of workflow tasks in scientific workflow executions in the cloud. The system objective functions and balanced task assignment in the proposed cloud system model are described in the following sections in detail.

3.2.1. Objective Functions. The optimization problem involves resource optimization for workflow execution on the cloud infrastructure that gives the optimized makespan, cost, and VM instance failures. The specified parameters include the number of VM instances and the type of machine instances at each workflow level. The number of machine instances is encoded as integer values where each value specifies the number of VM instances for the specific workflow level. Further, the types of VM instances are also encoded as integer, where

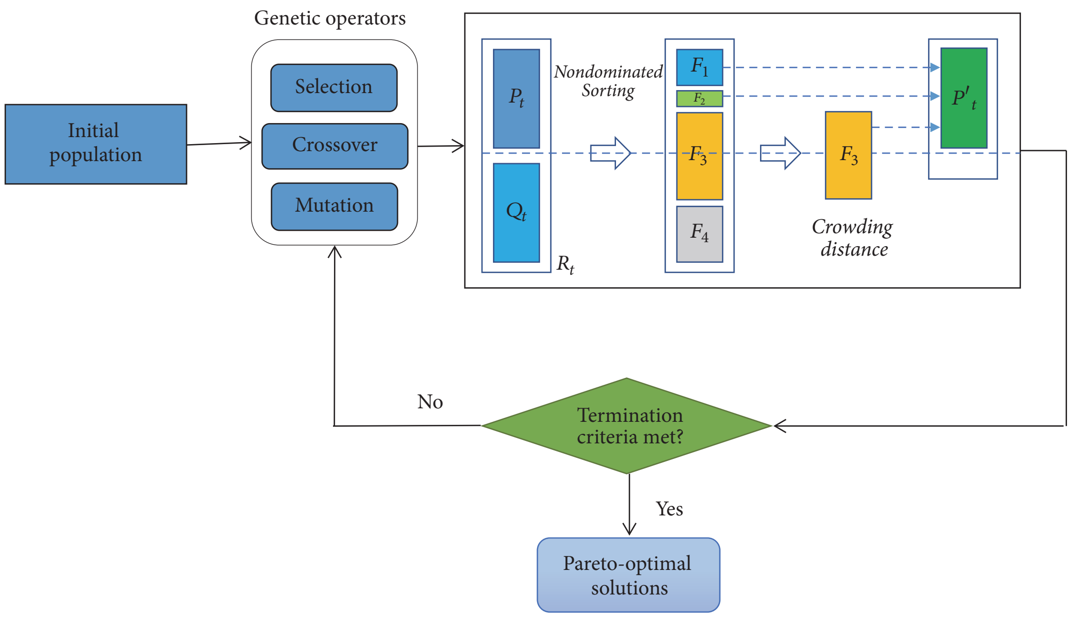


Figure 2: Process flow of NSGA-II optimization.

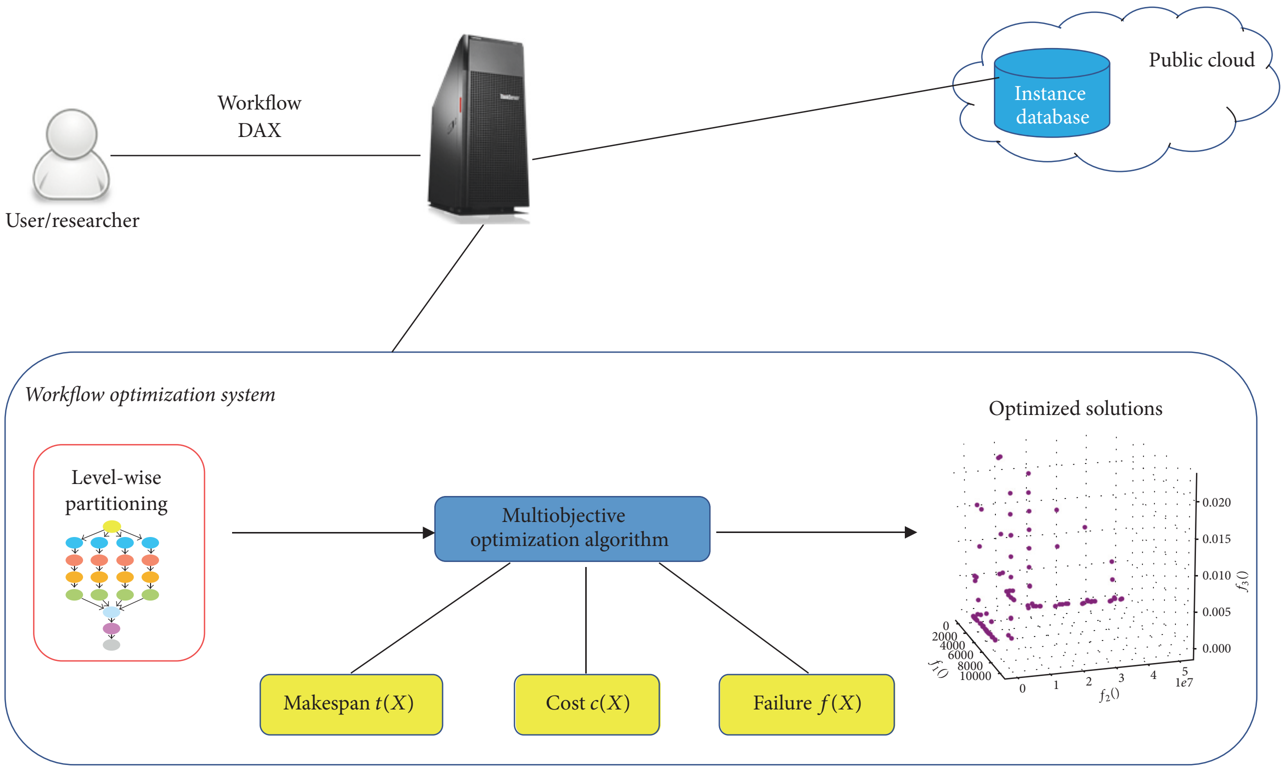


Figure 3: Proposed multiobjective scientific workflow optimization system.

each integer value specifies one of the machine instance types described in Table 2. Then, the functions that relate workflow makespan, cost, and VM instance failures for fitness evaluations are defined. The objective functions utilized in the proposed system are geared towards minimizing the overall makespan for workflow execution, machine instance deployment cost, and instance failure during workflow execution. They are expressed as follows:

$$\min t(X), \min c(X), \min f(X), \tag{1}$$

Table 1: Machine instance CPU performance.

Number	Instance type	CPU type	PassMark CPU Mark	Speedup over m3 CPU performance (%)
(1)	m3	Intel Xeon E5-2670 v2	14975	—
(2)	m4	Intel Xeon E5-2686 v4	17795	18
(3)	c3	Intel Xeon E5-2680 v2	16341	9
(4)	c4	Intel Xeon E5-2666 v3	24877	66
(5)	r3	Intel Xeon E5-2670 v2	14975	—

where $X = [x_1, x_2, \ldots, x_l, \ldots, x_m]$ is the number of VMs and types of VMs in each level of the DAG; $t(X)$ is total workflow execution time; $c(X)$ is total workflow machine instance deployment cost; $f(X)$ is total workflow machine instance failure probability.

In scientific workflow applications, execution dependencies among tasks are important. Thus, level-wise execution resource selection is conducted in our proposed system. First, we optimize the makespan and the number of VMs at each execution level. Then, following optimization at all levels in the workflow, an optimized execution plan is identified for the overall workflow.

Moreover, the deployment cost differs per machine instance type used and the number of machine instances used during workflow optimization. According to the machine instance type, the runtime of each task in the workflow is also different. For the failure minimization case, the system uses the availability factor of the virtual machine instances provided by Amazon Web Services (AWS). No user-defined constraints are considered at this time and automatic makespan, failure, and resource usage cost optimization are performed by the system. The three objective functions for level-wise makespan cost, instance deployment cost, and virtual machine instance failure are calculated as follows:

(1) Level-wise makespan cost:

$$t(X) = \sum_{k=1}^{L} \sum_{j=1}^{\lceil |T_k|/x_k \rceil} \max \bigcup_{l=1+(j-1)x_k}^{x_k \times j} T_k(l), \quad (2)$$

where $|T_k|$ is total number of tasks in level k; x_k is number of machine instances in level k; $T_k(l)$ is execution time of the task in position l in the cluster task set with respect to the instance type; L is total number of levels in DAG.

(2) Level-wise virtual machine deployment cost:

$$c(X) = \sum_{k=1}^{L} \left(\sum_{j=1}^{\lceil |T_k|/x_k \rceil} \left(\sum_{l=1+(j-1)x_k}^{x_k \times j} T_k(l) \times P\left(x_j^k\right) \right) \right), \quad (3)$$

where $P(x_j^k)$ is price of the jth virtual machine at kth workflow level.

(3) Level-wise virtual machine failure probability:

$$f(X) = \frac{\sum_{k=1}^{L} \left(\prod_{j=1}^{x_k} P_j^{\text{fail}} \right)}{L}, \quad (4)$$

where P_j^{fail} is failure probability of the jth virtual machine.

3.2.2. Cloud System Model. In our study, the cloud is modeled as a set of virtual machine instances, $\text{Ins} = \{\text{ins}_1, \text{ins}_2, \ldots, \text{ins}_j\}$. Each instance, ins_j, can be one of 25 machine instance types and instance costs are calculated based on instance prices for Tokyo region in AWS pricing [14]. Table 2 shows the instance pricing used in our resource usage calculation. The performance difference of different machine instance types is specified based on the CPU used as well as the number of CPU cores in each machine instance. We refer to the CPU performance of various instances from the PassMark benchmark [15]. Instance type m3 is assumed to be the lowest performance machine instance with the lowest price. The CPU performance differences of m4, c3, c4, and r3 instance types with respect to m3 machine instance type are assumed to be as listed in Table 1.

The performance speed up of each machine instance group is specified based on the number of CPU cores used in each machine instance. In addition, we also assume that all virtual machine instances share data from the same data center. Thus, data communication and data transfer costs are constant.

3.2.3. Balanced Task Assignment. Level-wise balanced task assignment to the VMs is conducted in scientific workflow execution because the fine-grained tasks in each workflow must be processed in their dependency order. The objective of balanced task assignment is to reduce the waiting time to continue to the next level during workflow execution and to balance the loads in each machine instance at each workflow level allowing resource sharing facility. For those purposes, balanced clustering of tasks at each level is considered before execution in their respective virtual machine. Figure 4 shows the task clustering for balanced task assignment process. There are two stages for balanced clustering: (1) task sorting and (2) balanced clustering. In task sorting, all the tasks in each level are sorted according to their runtime. Then, the task cluster size decision for each task group in the sorted task list is conducted according to the following equation:

If (number of virtual machines for ith level $= n$), $\text{NC} = n$ $(1 \le n \le p)$

$$C_{\text{size}_j} = \frac{T_i}{\text{NC}} + s_k, \quad j = 1, \ldots, \text{NC}, \; k = 1, \ldots, n, \quad (5)$$

where

Table 2: Machine instance pricing.

Number	Instance type	CPU (number of cores)	Memory (GB)	Price per second ($)
(1)	m3.medium	1	3.75	0.00002667
(2)	m3.large	2	7.5	0.00005361
(3)	m3.xlarge	4	15	0.00010694
(4)	m3.2xlarge	8	30	0.00021389
(5)	m4.large	2	8	0.00004833
(6)	m4.xlarge	4	16	0.00009667
(7)	m4.2xlarge	8	32	0.00019306
(8)	m4.4xlarge	16	64	0.00038639
(9)	m4.10xlarge	40	160	0.00096583
(10)	m4.16xlarge	64	256	0.00154528
(11)	c3.large	2	3.75	0.00003556
(12)	c3.xlarge	4	7.5	0.00007083
(13)	c3.2xlarge	8	15	0.00014194
(14)	c3.4xlarge	16	30	0.00028361
(15)	c3.8xlarge	32	60	0.00056750
(16)	c4.large	1	3.75	0.00003694
(17)	c4.xlarge	4	7.5	0.00007361
(18)	c4.2xlarge	8	15	0.00014750
(19)	c4.4xlarge	16	30	0.00029472
(20)	c4.8xlarge	36	60	0.00058944
(21)	r3.large	2	15	0.00005556
(22)	r3.xlarge	4	30.5	0.00011083
(23)	r3.2xlarge	8	61	0.00022167
(24)	r3.4xlarge	16	122	0.00044333
(25)	r3.8xlarge	32	244	0.00088667

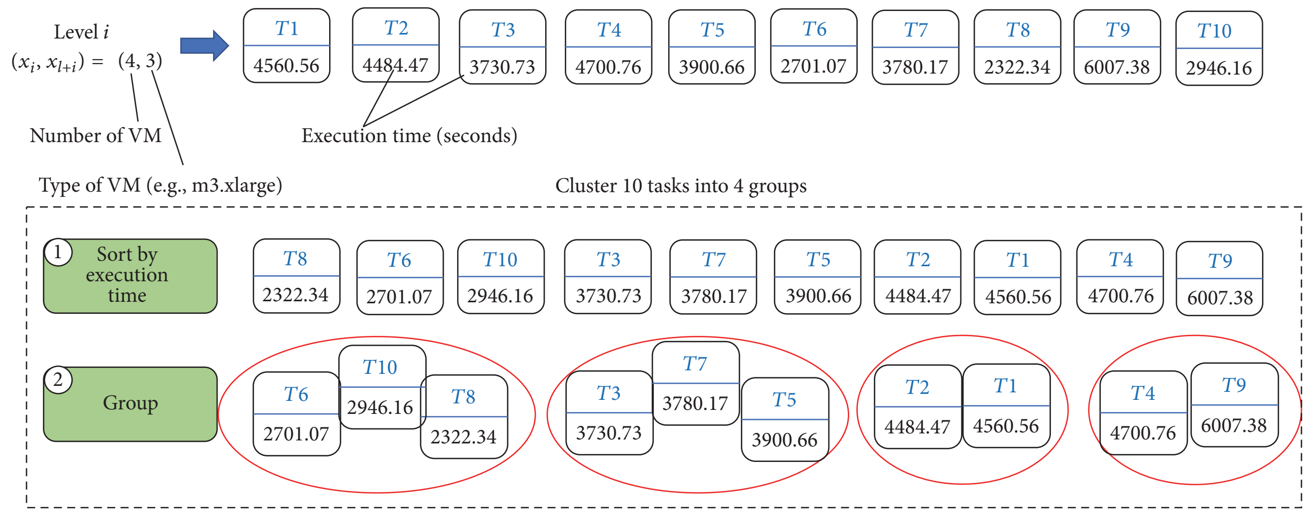


Figure 4: Level-wise balanced task assignment.

```
Input: Workflow DAG.
Output: Task Objects, Search Space (SP), total number of levels ($L$).
Step 1. Parse the workflow DAG
Step 2. Label tasks
Step 3. Assign workflow tasks to objects
Step 4. Retrieve maximum workflow level for chromosome length determination in optimization
Step 5. Retrieve maximum parallelization degree for search space determination
```

Algorithm 1: Preprocessing.

```
Input: Task Objects, Search Space (SP), total number of levels ($L$), $N$: Population size, $G$: number of generations,
$O$: number of objectives, MP: mutation probability, CP: crossover probability.
Output: Pareto optimal solutions, $W_{\text{opt}}$.
Processing
Step 1. Generate random initial population, $P_0$ and empty archive $\overline{P_0} = \Phi,\ t = 0$.
    (1.1) Randomly generate integer chromosome $C_i$ $(i = 1, 2, \ldots, N)$ for $P_0$, $\overline{P_0}$.
        (truncate if necessary)
    (1.2) Calculate the fitness of each random chromosome
    (1.3) Copy all non-dominated individuals in $P_t$ and $\overline{P_t}$ to $\overline{P_t + 1}$
Step 2. Evolve population $(t, P_t, \overline{P_t}, \overline{P_t + 1})$.
    (2.1) Select parents using tournament selection, $\overline{P_t + 1}$
    (2.2) Perform two-point dynamic crossover operation on parents with probability CP to produce offspring
        population.
    (2.3) Perform mutation operation with probability MP on random points applied to offspring
    (2.4) Calculate fitness of new offspring population and update population
    (2.5) Use non-dominated sorting to divide into several non-domination levels $F_1, F_2, \ldots, F_l$.
    (2.6) Update $\overline{P_t + 1}$
Step 3. Repeat Step 2 until termination condition is met.
Step 4. Output the list of optimal solutions at Pareto front, $W_{\text{opt}}$.
```

Algorithm 2: Workflow optimization using SPEA2.

$$s_k = \begin{cases} 0, & \text{if } T_i\%\ \text{NC} = 0 \\ 1, & \text{if } 0 < T_i\%\ \text{NC} < \text{NC} \end{cases} \quad T_i = \text{total number of tasks in } i\text{th level},\ \text{NC} = \text{number of task clusters}. \tag{6}$$

3.2.4. Virtual Machine Instance Failure. In general, a cloud uses a large number of commodity servers; thus, the possibility of failure exists. Failures of cloud based resources may lead to failures of application processing. Consequently, these failures can have a significant impact on the performance of the workflow execution. Thus, it is necessary to minimize the possibility of failures in the system. Workflow task clustering sometimes may result in a higher failure rate if more than one task is assigned to a single machine instance [3]. In this proposed system, we calculated the failure of each virtual machine instance based on AWS availability with certain additional weighted value. First, the original failure probability of a virtual machine instance is assumed to be $p_f = 0.02337$. We assume that the more tasks in the virtual machine, the higher the weight of failure of that virtual machine. Weight values are calculated based on the number of tasks in the level and the original failure probability value. Each machine instance failure probability P_j^{fail} is calculated as follows:

$$P_j^{\text{fail}} = p_f + w_j,$$

$$w_j = \left(N_{tj} - 1\right)\left(\frac{p_f}{T_k}\right), \tag{7}$$

where N_{tj} is total number of tasks in the jth virtual machine.

3.3. Multiobjective Workflow Optimization Algorithm. This section presents the SPEA2 and NSGA-II based optimization algorithms, which are the major component of the proposed system. The system uses NSGA-II because it outperforms other contemporary MOEAs, such as PAES [16] and SPEA [17], in terms of finding a diverse set of solutions and in converging near the true Pareto-optimal set [8]. SPEA2 is used to compare the Pareto-optimal solutions obtained from NSGA-II optimization. Algorithm 1 describes the preprocessing steps necessary for parsing the workflow tasks in DAG format to task objects. The workflow optimization algorithms using SPEA2 and NSGA-II are described in Algorithms 2 and 3, respectively.

```
Input: Task Objects, Search Space (SP), total number of levels ($L$), $N$: Population size, $G$: number of generations,
$O$: number of objectives, MP: mutation probability, CP: crossover probability.
Output: Pareto optimal solutions, $W_{\text{opt}}$.
Processing
Step 1. Generate random initial population, IP.
        (1.1) Randomly generate integer chromosome $C_i$ ($i = 1, 2, \ldots, N$).
        (1.2) Calculate the fitness of each random chromosome
        (1.3) Rank population $P$ according to non-domination
Step 2. Evolve population ($N, P, G$).
        (2.1) Select parents using tournament selection, ($P_1, P_2$).
        (2.2) Perform two-point dynamic crossover operation on parents ($P_1, P_2$) with probability CP to produce
              offspring population.
        (2.3) Perform mutation operation with probability MP on random points applied to offspring
        (2.4) Calculate fitness of new offspring population and update population
        (2.5) Use non-dominated sorting to divide $P \leftarrow P \cup \{x^C\}$ into several non-domination levels $F_1, F_2, \ldots, F_l$.
        (2.6) Calculate Crowding Distance of all solutions
        (2.7) Identify the worst solution $x' \in F_l$ and set $\leftarrow P' \setminus \{x'\}$.
Step 3. Repeat Step 2 until termination condition is met.
Step 4. Output the list of optimal solutions at Pareto front, $W_{\text{opt}}$.
```

Algorithm 3: Workflow optimization using NSGA-II.

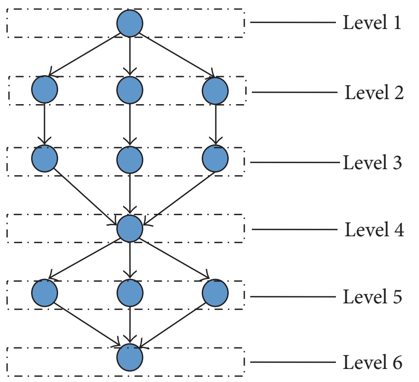


Figure 5: Level-wise workflow partitioning.

3.4. System Implementation. Scientific workflows support and automate the execution of error-prone, repetitive tasks in scientific applications. It combines activities and computations to solve scientific problems. Task partitioning is important during workflow execution because it can affect the performance of the application. Three approaches to workflow partitioning before executing are reported in the literature: (1) horizontal workflow partitioning, (2) vertical workflow partitioning, and (3) arbitrary workflow partitioning.

In our proposed system, we use level-wise workflow partitioning which is a horizontal partitioning approach. We chose level-wise partitioning of structured scientific workflows because these workflows consist of different numbers of tasks with different resource requirements at each level of the workflow. The level-wise approach enables elastic provisioning of resources and load balancing at each workflow level. Level-wise partitioning of a simple DAG workflow is illustrated in Figure 5.

3.4.1. DAX Parsing. In the parsing stage, task information specified in the directed acyclic graph is retrieved. In this system, necessary information such as total number of levels in the workflow, task name and their relevant execution time, and the maximum parallelization degree for the input workflow to perform automatic optimization are retrieved. Subsequent chromosome length determination depends on the total number of levels in the workflow. Moreover, task labelling is also conducted at each level of the workflow to facilitate level-wise optimization of workflow tasks within the optimization process during workflow parsing.

3.4.2. Multiobjective Optimization. Following parsing and task labelling of the workflow, the system begins the multiobjective optimization process. First, random initial chromosome parameter lists comprising the number of VMs and the type of VMs suitable for each level of the workflow are generated to initiate the process. Next, fitness calculations are performed for the three objective functions. Genetic operators such as selection, crossover, and mutation are then chosen to be used during evolution. Then, the evolution process proceeds until the specified termination criterion is met. At this point, the Pareto front solutions from the evolution process are retrieved and the optimized number of VMs with optimized instance type at the Pareto front for optimal makespan and optimal deployment cost with the least failure for each workflow level are recommended for processing the scientific workflow applications in the cloud.

3.4.3. Chromosome Representation. The chromosome of the proposed system includes genes which represent the number of VMs and the type of VMs used for each level of the workflow. The system uses a fixed-length integer-encoding scheme, one of the most common representations of chromosomes in genetic algorithms (GA). The genes in the

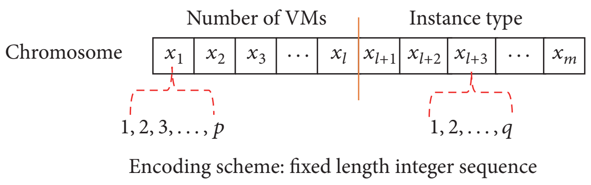

Number of VMs
Instance type
Chromosome
x_1 x_2 x_3 $\cdots$ x_l x_{l+1} x_{l+2} x_{l+3} $\cdots$ x_m
$1, 2, 3, \ldots, p$
$1, 2, \ldots, q$
Encoding scheme: fixed length integer sequence

FIGURE 6: Chromosome structure for optimization during workflow execution.

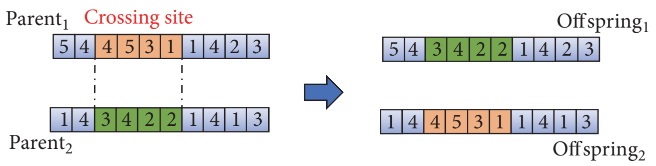


FIGURE 7: Crossover operation.

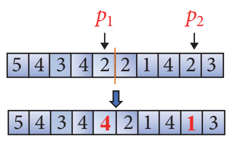


FIGURE 8: Mutation operation.

chromosomes are in the form of $\{1, p\}$ and $\{1, q\}$ with each gene representing the number of VMs for each level of workflow execution for parallelization, while p signifies the maximum number of concurrent tasks allowable in that workflow level, and q signifies the total possible number of instance types provided by the academic cloud system. A single gene represents one level and the chromosome length depends on the maximum number of levels in the workflow. The chromosome length, $2L$, where L is the highest level of the workflow DAG, varies according to the input workflow structure. Specifically, the chromosome length for a simple workflow in Figure 5 is ($2L = 12$), where $L = 6$ and the gene value ranges are $(1, p) = (1, 3)$ and $(1, q) = (1, 25)$, respectively. An example chromosome for the optimization process is illustrated in Figure 6.

3.4.4. Genetic Operators. The genes in the chromosomes represent number of VMs and type of VMs at each workflow level and the proposed workflow optimization processes the execution of tasks in a level-wise manner to preserve the dependencies between tasks. During optimization, the genetic operators are used to guide the multiobjective optimization algorithm towards optimal solutions to a given problem. Three main operators, selection, crossover, and mutation, collaborate for successful processing.

(1) Selection. The select operation gives us a straightforward way of choosing offspring for the next generation. Tournament selection is used in this system as it reduces the computational cost associated with selection. In tournament selection, random T (user-defined tournament selection size) individuals from the population are chosen, their fitness values are compared, and the fittest is used in the recombination process. The higher the T value, the higher the selection pressure in tournament selection.

(2) Crossover. The crossover operation takes two parent strings called parent$_1$ and parent$_2$ and generates two offspring, offspring$_1$ and offspring$_2$. The decision of whether to perform a crossover operation on the current pair of parent chromosomes is made according to the specified crossover probability (CP). If the random probability is less than CP, a crossing site is selected and partial exchange occurs between the two parents to produce the offspring. If the probability is greater than CP, the crossing site will be the entire chromosome length, which has no effect because of the crossover operator. Two-point crossover is applied in the proposed system, as illustrated in Figure 7.

(3) Mutation. Mutation is important because it allows the evolutionary process to explore new potential solutions to the problem. To avoid inbreeding in a small population problem, mutation is applied with small probability (MP) after the crossover operation. The selected MP should be reasonable because if it is too high or too low, a good solution will not be found during evolution. Random two-point mutation is used during evolution of the proposed system, as illustrated in Figure 8.

4. Performance Evaluation

This section outlines the experimental environment and presents the experimental results obtained for various scientific workflow applications. The simulation experiments were conducted on an Intel core i7 Dell PC with 16 GB RAM with Pegasus workflow dataset. The system implementation was

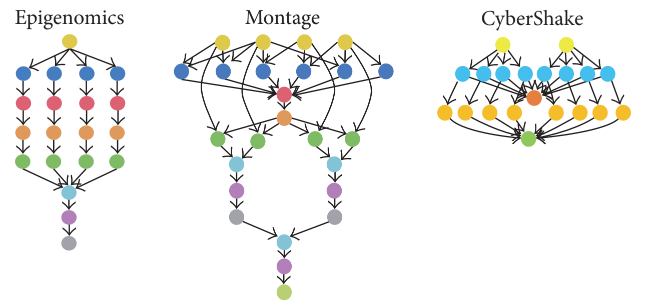


Figure 9: Structure of the experimental scientific workflows.

Table 3: Experimental workflows.

Number	Workflow application	Number of task nodes (input size)			
(1)	Epigenomics	24	46	100	997
(2)	Montage	25	50	100	1000
(3)	CyberShake	30	50	100	1000

conducted using the Python programming language and the well-known DEAP framework [18].

4.1. Experimental Scientific Workflow Applications. To demonstrate the proposed offline workflow optimization system, the experiments were conducted using three scientific workflow DAG applications (Epigenomics, Montage, and CyberShake) from Pegasus [5, 19], with various input data sizes. Epigenomics is a CPU-bound bioinformatics workflow with eight levels of workflow tasks. Montage is a collection of programs that combines multiple shots of astronomical images to generate a custom mosaic image [20]. CyberShake is a seismology workflow application that is used by the Southern California Earthquake Center to characterize earthquake hazards in a region [3]. The structures of the applications are illustrated in Figure 9. Table 3 shows the scientific workflows used for system evaluation along with their respective input data sizes.

4.2. Experimental Analysis. Figures 10–13 show cost-makespan tradeoff solutions identified by the proposed system using NSGA-II and SPEA2. The experiments were conducted with a population size of 100 evolving over 500 generations with crossover probability 0.7 and mutation probability of 0.0625, 0.055, and 0.125—in accordance with the number of levels in the given workflow. The experimental results indicate that the system reduced the makespan effectively with acceptable cost tradeoff. The effectiveness of the system is reflected more in the larger data size workflows, in which makespan is significantly improved with only a small increase in deployment cost. For failure minimization objective, the system gives only some significant effectiveness in smaller input data size workflows compared with other two objective functions.

Figure 10 shows cost-makespan tradeoff solutions for three workflows with four input data size after 500 generations. Although the optimization was conducted on three objective functions, the distribution of the solutions is presented in two dimensions (2D) as 2D provides better clarity for comparison than 3D. In the figures, it is clear that makespan and deployment cost functions are strongly correlated with a negative slope. That is, the higher the makespan, the lower the deployment cost, and vice versa. As our objective functions are minimizing functions, we would like to choose the optimal solution which is as close to the X-Y origin as possible. The values in the figures also illustrate a gentle slope in small input data size that becomes steeper with increased data size. For the Montage case, there are gaps in the Pareto front; the reason will be determined in future studies. The scatter plots in Figure 11 illustrate the cost-failure tradeoff solutions. The values in the figures highlight that the two functions are not strongly correlated. Nevertheless, there is a weak correlation between these two functions except Montage 1000 task nodes and CyberShake 1000 task nodes. In those extra-size input workflows, all Pareto front solutions recommend to use similar instance type but with different number of machine instances at each workflow level. Moreover, the failure probability values of all solutions are very similar with only small differences in the 7th position of the fractional result values. This can affect the cost-failure tradeoff solutions of Montage 1000 and CyberShake 1000 to be a linear function with no correlation. For these reasons, the best optimal solution for the three objective functions should be selected based on the makespan, cost, and failure, respectively. In our evaluation, we identified two extreme points and one mid-point (the one closest to the origin) to be makespan best, time best, and mid-point solution to find out the system effectiveness. We found out that the system improves by 5.6, 10 times for makespan, cost for Epigenomics case with failure of 0.00559256, 1.5, 4.5 times improvement with failure of 0.0149133 for Montage, and 9, 9 times improvement with failure of 0.008948 for CyberShake, respectively. Figures 12 and 13 show the experimental results using SPEA2 algorithm and the results are quite similar for all three workflows.

For the evaluation of the evolutionary algorithms, several quality indicators such as convergence, diversity, inverse generational distance (IGD), and hypervolume exist. Among these indicators, convergence and diversity values of the optimal solutions obtained from the Pareto fronts over five independent test runs for 500 generations were used to calculate the average values and standard deviations listed in Table 4. These quality indicator calculations were conducted according to the DEAP framework, the smaller the convergence and diversity values the better. Convergence is a measure of the distance between the obtained nondominated front and the true Pareto-optimal front. Convergence metric, γ, is calculated by computing the minimum Euclidean distance of each obtained solution from the chosen solutions on the Pareto-optimal front and then averaging those distances [13]. The metric is calculated as below:

$$\text{Convergence}, \gamma = \frac{\sum_{i=1}^{N} |d_i|}{N}, \qquad (8)$$

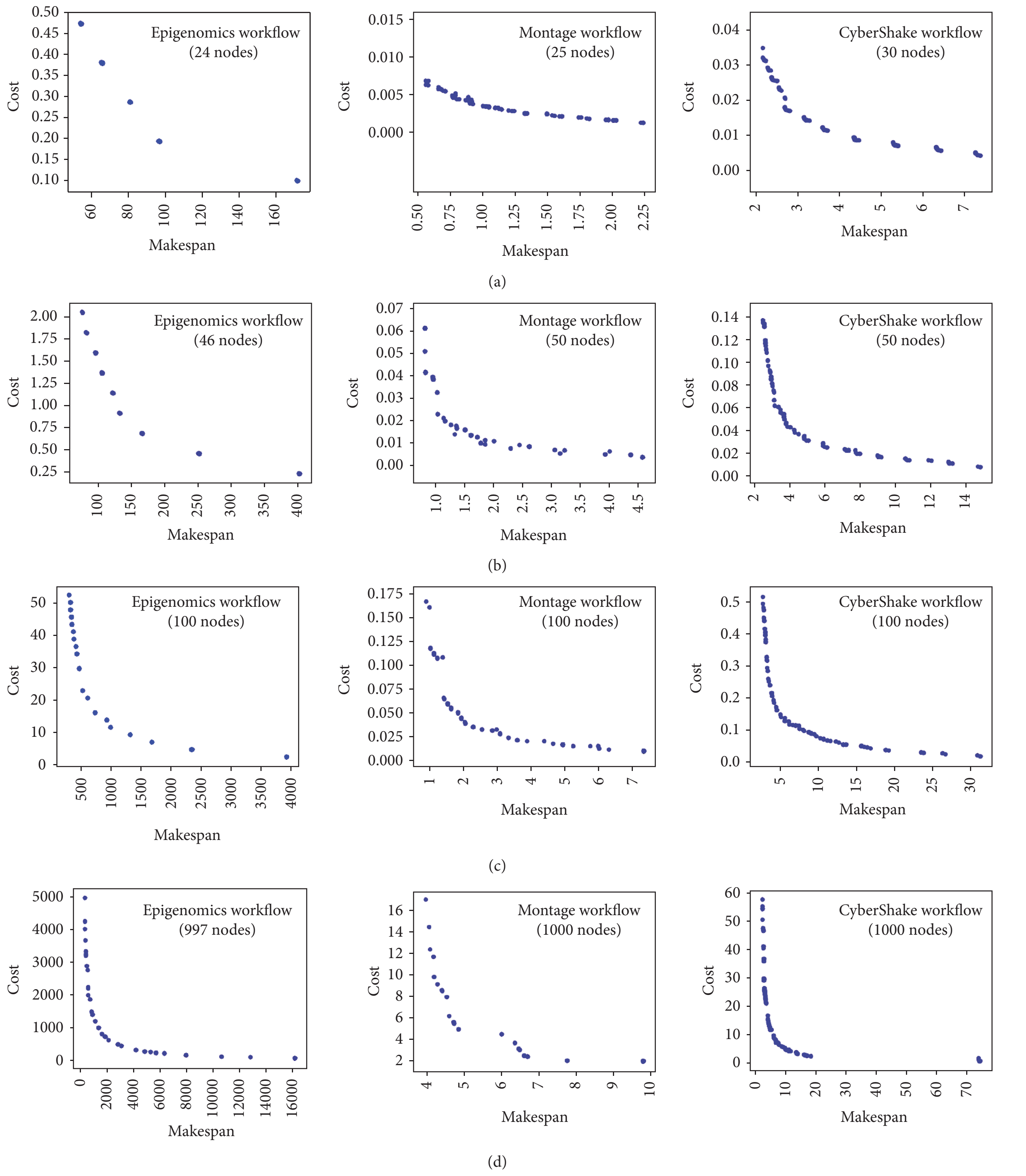
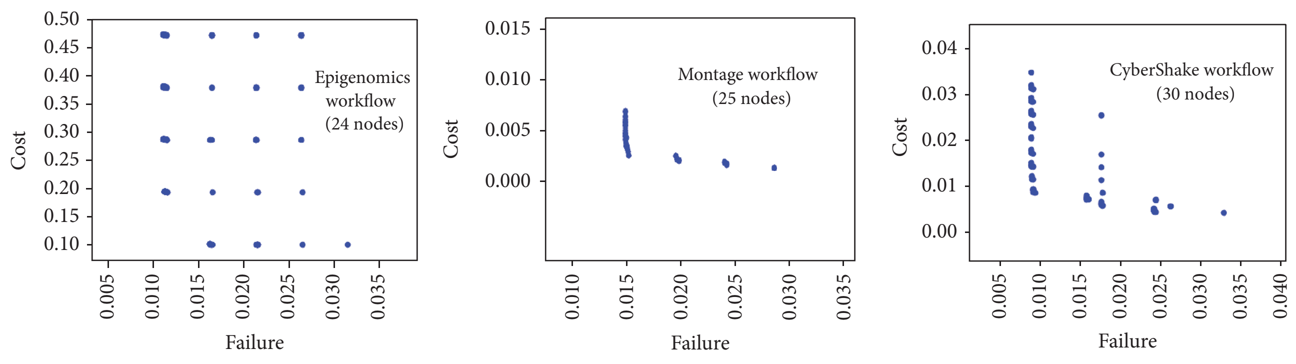
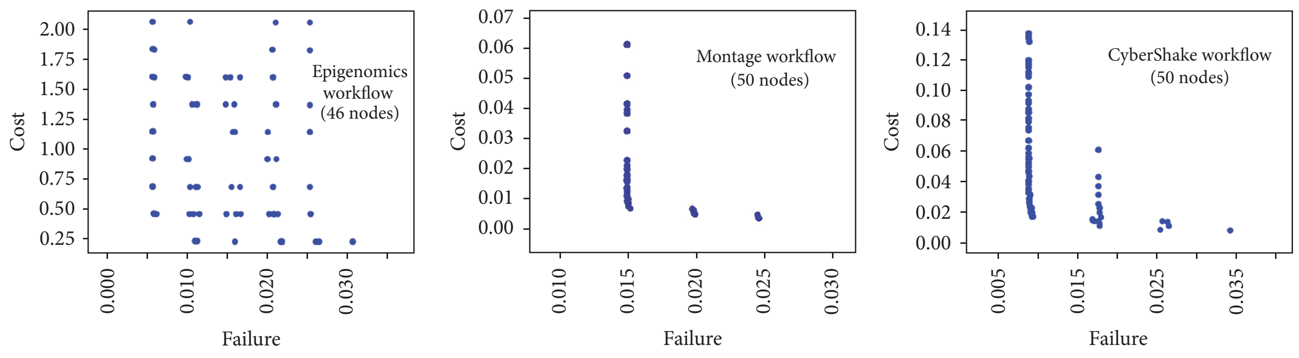


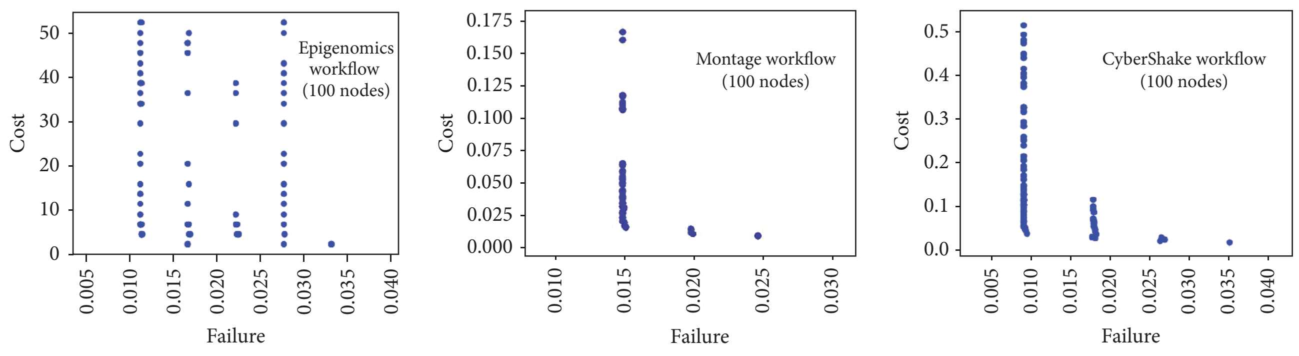
FIGURE 10: Cost_Makespan Pareto solutions (a), (b), (c), and (d) for three workflows with small, medium, large, and ex-large input data size using NSGA-II.



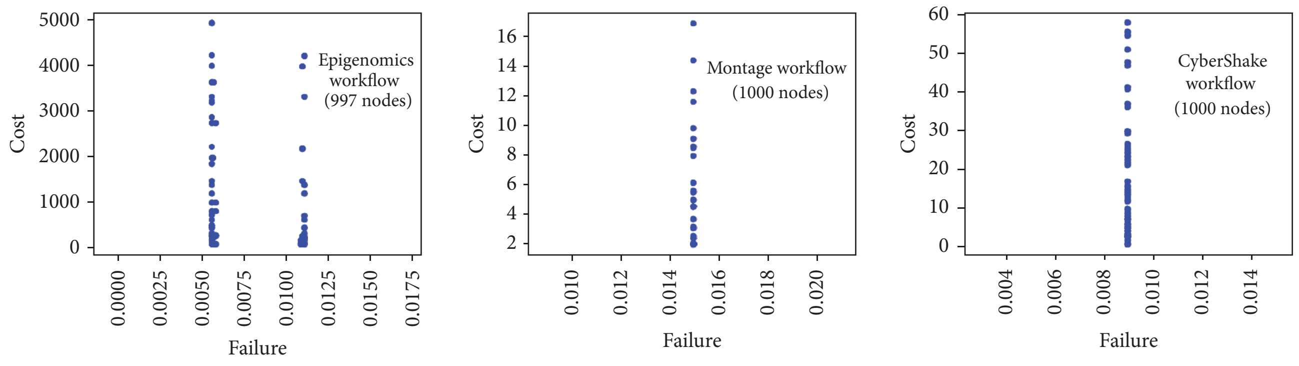
(a)



(b)



(c)



(d)

Figure 11: Cost_Failure Pareto solutions (a), (b), (c), and (d) for three workflows with small, medium, large, and ex-large input data size using NSGA-II.

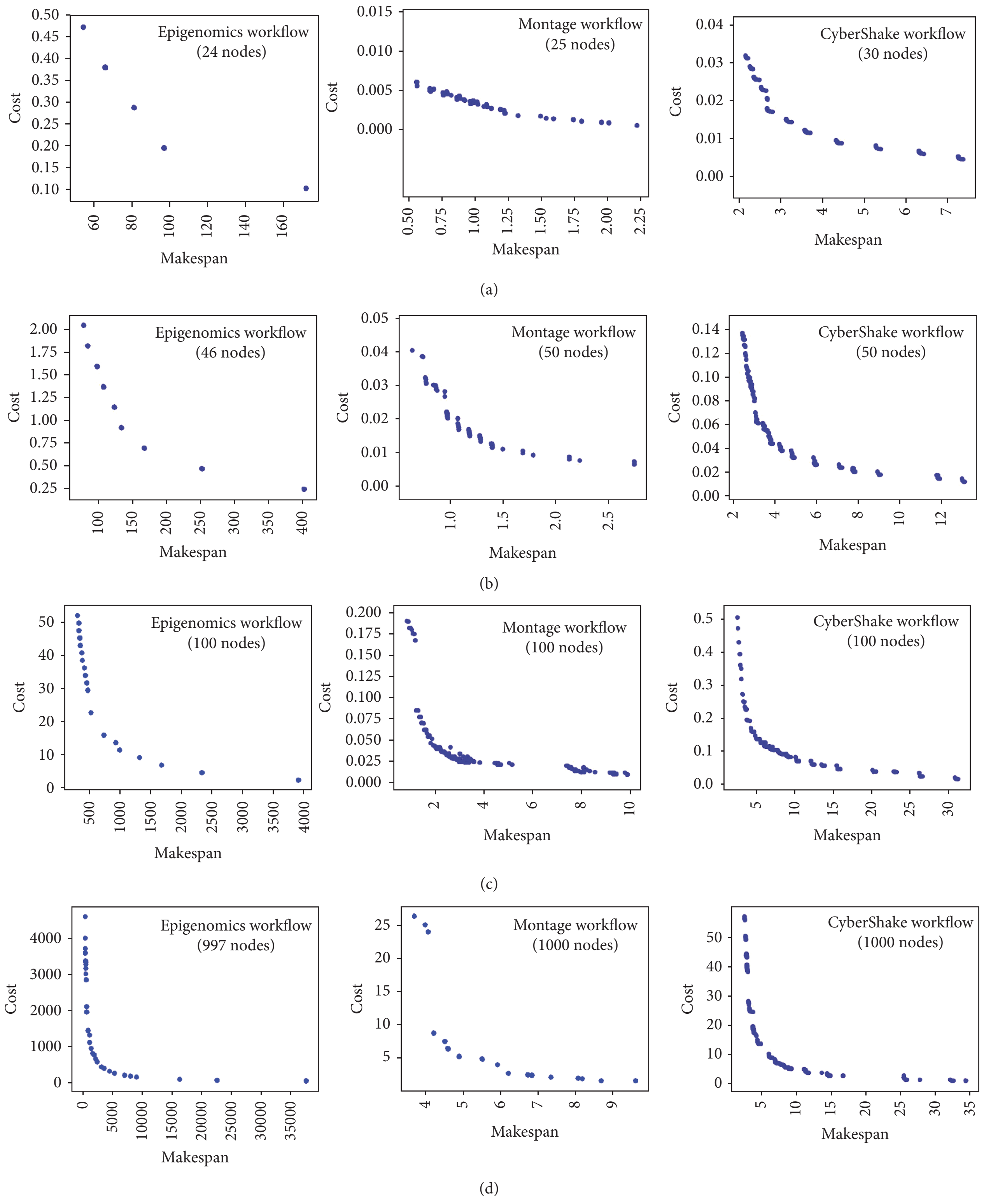
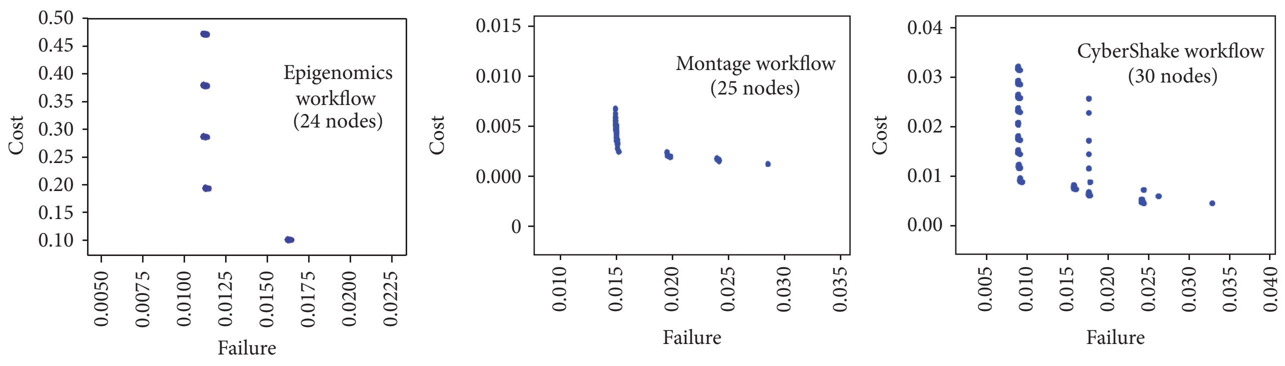
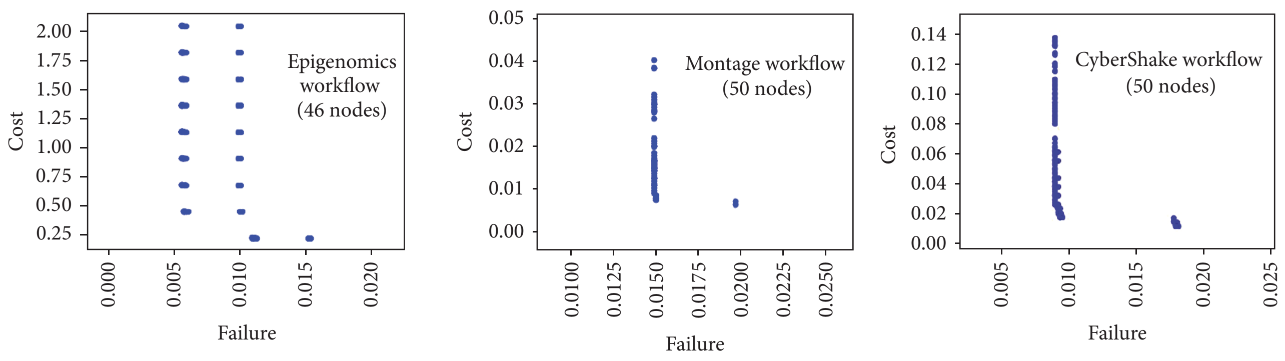


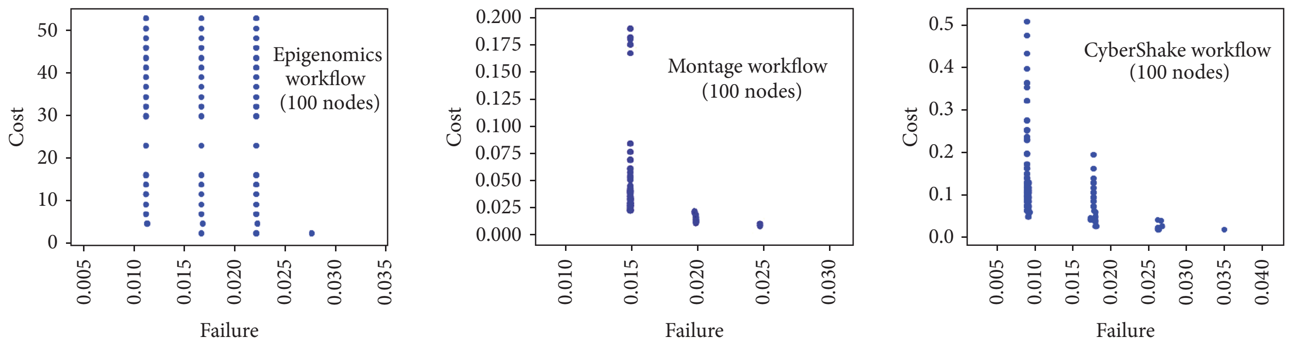
Figure 12: Cost_Makespan Pareto solutions (a), (b), (c), and (d) for three workflows with small, medium, large, and ex-large input data size using SPEA2.



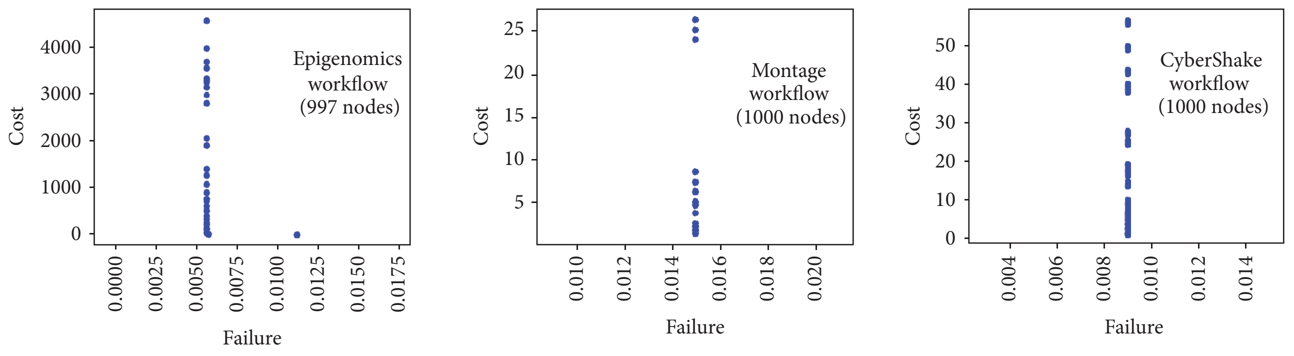
(a)



(b)



(c)



(d)

FIGURE 13: Cost_Failure Pareto solutions (a), (b), (c), and (d) for three workflows with small, medium, large, and ex-large input data size using SPEA2.

Table 4: Convergence, diversity, and execution time for SPEA2 and NSGA–II.

Workflow	Algorithm	Avg. Conv. ($\pm\sigma$)	Avg. Div. ($\pm\sigma$)	Avg. Exec. (sec) ($\pm\sigma$)
Epigenomics_24	SPEA2	0.0240 (±0.0011)	1.8554 (±0.0367)	406.5189 (±14.9)
	NSGA-II	0.0294 (±0.0177)	1.7936 (±0.0992)	74.5451 (±6.5)
Montage_25	SPEA2	0.0060 (±0.0039)	1.0104 (±0.4130)	372.7346 (±26.9)
	NSGA-II	0.0059 (±0.0005)	1.2770 (±0.1713)	68.5270 (±16.2)
CyberShake_30	SPEA2	0.0094 (±0.0012)	1.3485 (±0.0019)	414.6732 (±11.0)
	NSGA-II	0.0079 (±0.0012)	1.4026 (±0.0642)	53.9493 (±1.8)
Epigenomics_46	SPEA2	0.1661 (±0.2064)	1.7670 (±0.0390)	431.8035 (±30.5)
	NSGA-II	0.1118 (±0.0857)	1.7152 (±0.0918)	71.5525 (±5.5)
Montage_50	SPEA2	0.0033 (±0.0040)	1.4672 (±0.1394)	366.6482 (±21.9)
	NSGA-II	0.0100 (±0.0031)	1.2758 (±0.1414)	63.8932 (±4.6)
CyberShake_50	SPEA2	0.0209 (±0.0077)	1.3629 (±0.1168)	415.7871 (±17.6)
	NSGA-II	0.0224 (±0.0060)	1.3372 (±0.9048)	56.6219 (±3.5)
Epigenomics_100	SPEA2	0.1005 (±0.0518)	1.7229 (±0.0079)	388.3032 (±13.0)
	NSGA-II	0.1423 (±0.1108)	1.7188 (±0.0004)	74.4787 (±6.3)
Montage_100	SPEA2	0.0178 (±0.0140)	1.1626 (±0.2073)	375.2320 (±13.3)
	NSGA-II	0.0178 (±0.0055)	1.2122 (±0.1533)	71.6682 (±10.1)
CyberShake_100	SPEA2	0.0731 (±0.0116)	1.0607 (±0.0361)	388.4410 (±10.4)
	NSGA-II	0.0721 (±0.0240)	1.2178 (±0.0665)	60.2484 (±5.0)
Epigenomics_997	SPEA2	2.3735 (±1.7707)	1.5694 (±0.0521)	545.0587 (±16.3)
	NSGA-II	12.652 (±6.5924)	1.5404 (±0.0777)	225.4969 (±15.7)
Montage_1000	SPEA2	0.0027 (±0.0044)	1.5974 (±0.1057)	546.3927 (±27.2)
	NSGA-II	0.0573 (±0.0188)	1.6398 (±0.1235)	216.1495 (±4.0)
CyberShake_1000	SPEA2	0.2783 (±0.0532)	1.1775 (±0.1606)	522.4185 (±18.7)
	NSGA-II	0.3458 (±0.1087)	1.3284 (±0.0589)	170.0315 (±3.7)

where d_i is Euclidean distance between the obtained solutions and the nearest member of the best nondominated front.

Diversity is a measure of the sufficiency of the obtained nondominated solutions to represent the range of the Pareto-optimal front [19]. The equation for diversity calculation is described in [8] as follows:

$$\text{Diversity}, \Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - \overline{d} \right|}{d_f + d_l + (N-1)\,\overline{d}}, \tag{9}$$

where d_f, d_l are Euclidean distances between the extreme solutions and the boundary solutions; d_i is a solution on the best nondominated front d_i, $i = 1, 2, 3, \ldots, (N-1)$; $\overline{d}$ is the average of all distances d_i.

The current research work focuses on optimization of scientific workflow executions in the cloud through multiobjective evolutionary algorithms such as SPEA2 and NSGA-II. According to the experimental results, the ability to identify the Pareto-optimal solutions of SPEA2 and NSGA-II is virtually the same. In some cases, SPEA2 gives better convergence and diversity values than NSGA-II. However, a longer execution time is required by SPEA2 which is very important for the search process during workflow optimization. In multiobjective evolutionary algorithms such as SPEA2 and NSGA-II, the most important parts of the algorithms are in fitness assignment based on Pareto-domination: domination counts, nondominated sorting, and identification of the nondominated solutions. These processes can be done within $O(MN^2)$, where M is the number of objectives and N is the population size in both SPEA2 and NSGA-II. The time complexities of SPEA2 and NSGA-II are beyond the scope of this paper, but interested readers can refer to Jessen [21] for further details on the time complexities of multiobjective evolutionary algorithms.

5. Related Work and Discussion

The use of cloud in scientific experiments is a popular alternative because transferring large volumes of data to processing nodes is impractical in this era of continuously increasing big data. However, the right virtual machine instance selection for a specific application is a challenging problem for the cloud users. Thus, optimization system approaches that easily facilitate optimized cloud resource selection are actively being researched. Thus, various single-objective and multiobjective optimization strategies have been proposed.

The proposed single-objective optimization strategies primarily focused on developing effective scheduling strategies to reduce makespan. For example, Tanaka and Tatebe [20] proposed a data-aware scheduling strategy that reduces makespan by minimizing data movement between cluster nodes was presented. However, their strategy can only operate with homogeneous resources during workflow execution and is not suitable for use in cloud environments. Chen and Deelman [1] proposed a workflow scheduling strategy that uses horizontal or vertical task clustering techniques to reduce the schedule overhead during workflow execution. Yan et al. [22] proposed a novel probability evaluation-based scheduling algorithm to address the problem of workflow deadline guarantee. They showed that their approach is effective in terms of adaptability and predictability under deadline constraints, but improvements in deadline distribution are still necessary. Kaur and Mehta [23] proposed workflow scheduling in clouds using augmented shuffled frog leaping algorithm (ASFLA). They improved the original shuffled frog leaping algorithm (SFLA) to speed up the overall execution time of the workflow. Experiments conducted showed significant reduction in overall workflow execution time, but resource usage cost was not accounted for in the system.

Poola et al. [24] presented a resource scheduling algorithm for workflow execution on heterogeneous cloud resources along with makespan and cost minimization. However, the robustness and effectiveness of their proposed system depend on an increased budget. Zhu et al. [2] proposed a multiobjective workflow scheduling approach for cloud infrastructure. In their proposed approach, they enhance the evolution process of their optimization by introducing two novel crossover and mutation operators. They conducted experiments based on actual pricing and resource parameters on Amazon EC2 but did not consider task dependencies during workflow executions and simply assumed that all tasks can be executed in parallel. Zhang et al. [25] proposed biobjective workflow optimization on energy consumption and reliability in heterogeneous computing systems.

Single-objective and multiobjective workflow execution optimization has been conducted on cluster, grid, and cloud infrastructure in recent years. Single-objective optimization approaches are focused on minimizing the workflow makespan through scheduling [20, 22, 23] and task clustering approach [1]. However, single-objective optimization is not sufficient for data intensive workflows. Our proposed method is focused on multiobjective workflow execution optimization in the cloud infrastructure. Although similar work [23, 24] has been carried out on cloud infrastructure, the approaches employed have limitations in terms of resource elasticity and task dependencies. Moreover, previous studies do not cover important issues associated with VM instances such as VM failure probability. Our proposed approach addresses these issues as follows:

(1) Randomized search-based resource selection to better fit resource usage

(2) Level-wise task processing approach for task dependencies and resource elasticity

(3) Balanced task clustering for better load balancing and resource sharing

(4) Optimized workflow execution on three objectives (including cloud SLA related objectives)

In this way, the proposed system helps with the selection of the right resources through multiobjective optimization during workflow executions that benefits both stakeholders (users and cloud service providers).

6. Conclusion and Future Work

Machine instance resource selection for scientific workflow applications with the best performance, the cheapest deployment cost, and minimum failure probability is still challenging in the cloud. Without proper resource selection, it may lead to user dissatisfaction (owing to high costs and unreliable result data) which can affect the service providers' business. In this study, we addressed this problem by modeling the workflow optimization problem as a three-objective optimization problem using SPEA2 and NSGA-II in the cloud. During workflow optimization, a fixed-length integer chromosome was used to represent a random number of VMs that can be deployed in each level of the workflow. The optimization process was conducted based on the random initial chromosome population and the respective fitness values of the chromosomes calculated. Our experimental results obtained on three real-world workflows available from the ongoing Pegasus research project [5] show that the proposed system reduces makespan with an acceptable cost tradeoff and instance failure. The results also indicated that the two objective functions (makespan and instance cost) are strongly related to each other, whereas failure minimization objective values are weakly related to or have no relation with the other two functions in some cases. Moreover, we found out that c4.xlarge machine instance type was a dominant instance type during optimization according to our model. In future work, we plan to include result comparisons with other multiobjective optimization algorithms such as NSGA-III and MOEA/D. In addition, many-objective optimization with makespan and budget constraints during workflow execution will be considered.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by CREST, Japan Science and Technology Agency (Grant no. JPMJCR1501), and Nitobe School, Hokkaido University.

References

[1] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," in *Proceedings of the 2012 IEEE 8th International Conference on E-Science, e-Science 2012*, USA, October 2012.

[2] Z. Zhu, G. Zhang, M. Li, and X. Liu, "Evolutionary multi-objective workflow scheduling in cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1344–1357, 2016.

[3] D. A. Prathibha, B. Latha, and G. Sumathi, "Efficient scheduling of workflow in cloud enviornment using billing model aware task clustering," *Journal of Theoretical and Applied Information Technology*, vol. 65, no. 3, pp. 595–605, 2014.

[4] H. Khajemohammadi, A. Fanian, and T. A. Gulliver, "Efficient Workflow Scheduling for Grid Computing Using a Leveled Multi-objective Genetic Algorithm," *Journal of Grid Computing*, vol. 12, no. 4, pp. 637–663, 2014.

[5] Pegasus Workflow Generator. http://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator, 2016.

[6] P. T. Thant, C. Powell, M. Schlueter, and M. Munetomo, "A level-wise load balanced scientific workflow execution optimization using NSGA-II," in *Proceedings of the 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pp. 882–889, Madrid, Spain, 2017.

[7] E. Zitzler, M. Laumanns, and L. Thiele, SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, 2001.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[9] G. Scherp, *A Framework for Model Driven Scientific Workflow Engineering [Dissertation, thesis]*, Kiel Computer Science Series, University of Oldenburg, Germany, 2013.

[10] B. Ludashcer, M. Weske, T. McPhillips, and S. Bowers, "Scientific Workflows: Business as Usual?" in *Proceedings of the 7th International Conference on Business Process Management, BPM 2009*, pp. 31–47, Springer, 2009.

[11] N. Srinivas and K. Deb, "Multiobjective function optimization using nondominated sorting genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1995.

[12] C. Chitra, *Performance Comparison of Multi-objective Evolutionary Algorithms for QoS Routing Problems in Computer Networks [Dissertation, thesis]*, 2011.

[13] S. Kushwaha, *Multiobjective optimization of cluster measures in Microarray Cancer data using Genetic Algorithm Based Fuzzy Clustering [Dissertation, thesis]*, National Institute of Technology Rourkela, India, 2013.

[14] Amazon Service, Cloud. https://aws.amazon.com/ec2/instance-types/, 2017.

[15] PassMark Benchmark, CPU. https://www.umed.pl/zp/pliki/11412.pdf, 2017.

[16] J. Knowles and D. Corne, "The pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, vol. 1, pp. 98–105, July 1999.

[17] E. Zitzler, "Evolutionary algorithms for multiobjective Optimization: Methods and applications Doctoral dissertation ETH 13398," in *Swiss Federal Institute of Technology (ETH)*, Zurich, Switzerland, 1999.

[18] F.-M. De Rainville, F.-A. Fortin, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: A Python framework for Evolutionary Algorithms," in *Proceedings of the 14th International Conference on Genetic and Evolutionary Computation, GECCO'12*, pp. 85–92, USA, July 2012.

[19] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M. Su, and K. Vahi, "Characterization of scientific workflows," in *Proceedings of the 3rd Workshop on Workflows in Support of Large-Scale Science (WORKS '08)*, 10, 1 pages, IEEE, November 2008.

[20] M. Tanaka and O. Tatebe, "Workflow scheduling to minimize data movement using multi-constraint graph partitioning," in *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012*, pp. 65–72, Canada, May 2012.

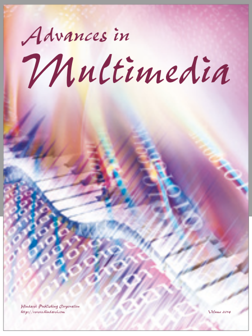
[21] M. T. Jensen, "Reducing the Run-Time Complexity of Multiobjective EAs: The NSGA-II and Other Algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 5, pp. 503–515, 2003.

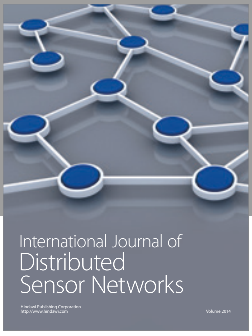
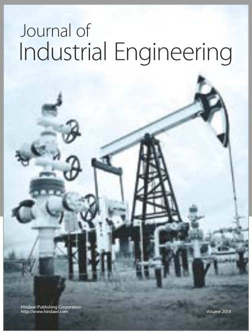
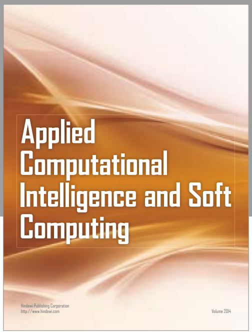
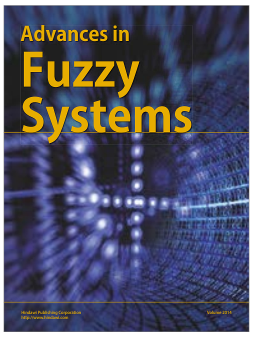
[22] C. Yan, H. Luo, Z. Hu, X. Li, and Y. Zhang, "Deadline guarantee enhanced scheduling of scientific workflow applications in grid," *Journal of Computers (Finland)*, vol. 8, no. 4, pp. 842–850, 2013.

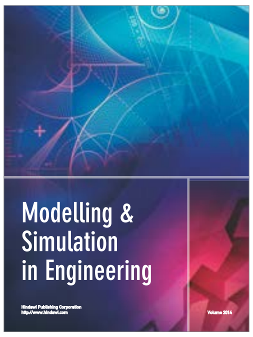
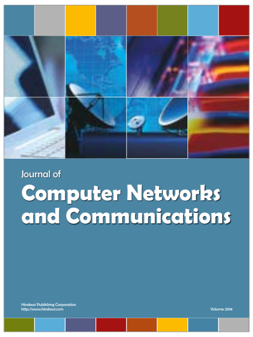
[23] P. Kaur and S. Mehta, "Resource provisioning and work flow scheduling in clouds using augmented Shuffled Frog Leaping Algorithm," *Journal of Parallel and Distributed Computing*, vol. 101, pp. 41–50, 2017.

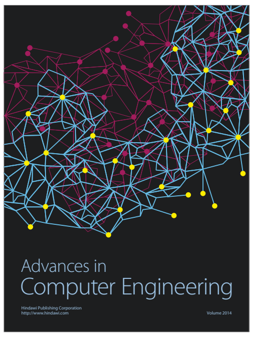
[24] D. Poola, S. K. Garg, R. Buyya, Y. Yang, and K. Ramamohanarao, "Robust scheduling of scientific workflows with deadline and budget constraints in clouds," in *Proceedings of the 28th IEEE International Conference on Advanced Information Networking and Applications, IEEE AINA 2014*, pp. 858–865, Canada, May 2014.

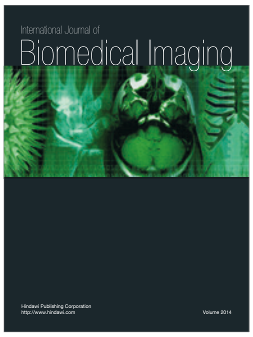
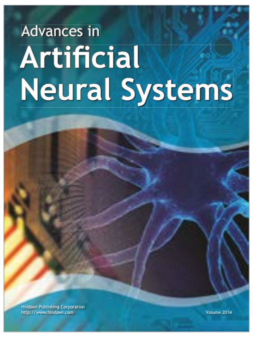
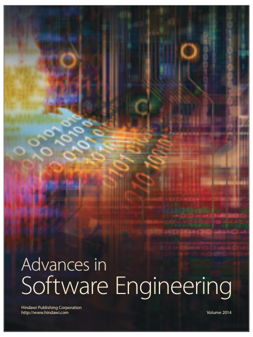
[25] L. Zhang, K. Li, C. Li, and K. Li, "Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems," *Information Sciences*, 2015.

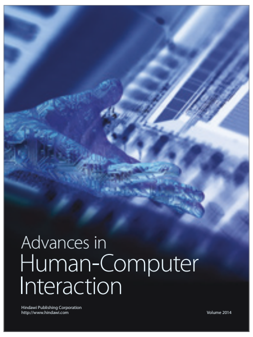
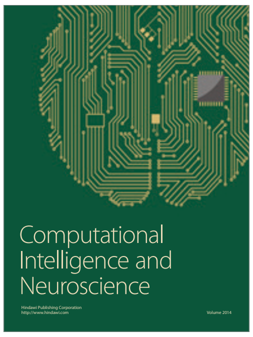

Advances in
Multimedia

The Scientific World Journal

International Journal of
Distributed Sensor Networks

Journal of
Industrial Engineering

Applied Computational Intelligence and Soft Computing

Advances in
Fuzzy Systems


Modelling & Simulation in Engineering

Journal of
Computer Networks and Communications

Advances in
Artificial Intelligence

Advances in
Computer Engineering

International Journal of
Computer Games Technology

International Journal of
Biomedical Imaging

Advances in
Artificial Neural Systems

Advances in
Software Engineering

Journal of
Robotics

Advances in
Human-Computer Interaction

Computational Intelligence and Neuroscience

International Journal of
Reconfigurable Computing

Journal of
Electrical and Computer Engineering