Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Multi-Objective Scientific-Workflow Scheduling with Data Movement Awareness on Cloud

PEERASAK WANGSOM[1], KITTICHAI LAVANGNANANDA[1], and PASCAL BOUVRY[2]

[1]Data Science and Engineering Laboratory, School of Information Technology, King Mongkut's University of Technology Thonburi, Bangkok 10140, Thailand
[2]FSTC-CSC, SnT, University of Luxembourg, Esch-sur-Alzette, L-4364, Luxembourg

Corresponding author: Kittichai Lavangnananda (e-mail: kitt@sit.kmutt.ac.th).

This work was supported by School of Information Technology, King Mongkut's University of Technology Thonburi.

ABSTRACT Due to serving several purposes simultaneously, running scientific workflows on dynamic environments such as cloud computing, has become multi-objective scheduling. Among these purposes, Cost and Makespan are probably the most two primitive objectives. Another critical factor in a large-scale scientific workflow is tremendous amount of data during execution. Therefore, this work also includes Data Movement as an additional objective as it has a major impact on network utilization and energy consumption in network equipment in cloud data center. In considering these three objectives, this work proposes a framework for scheduling solutions which combines a new nodes clustering technique in *Directed Acyclic Graph* (DAG) model known as *Multilevel Dependent Node Clustering* (MDNC) and the multi-objective optimization, *Extreme Nondominated Sorting Genetic Algorithm-III* (E-NSGA-III). E-NSGA-III is the recent extension of Nondominated Sorting Genetic Algorithm (NSGA-III). Five well-known scientific workflows, CyberShake, Epigenomics, LIGO, Montage, and SIPHT are selected as testbeds, while the commonly known Hypervolume is chosen as the performance metric. In this work, MDNC is also experimented with both NSGA-III. Comparison among three approaches, E-NAGA-III alone, E-NAGA-III with Peer-to-Peer clustering and E-NAGA-III with MDNC are carried out. The superiority of the proposed framework among them and its limitation are discussed.

INDEX TERMS Cloud computing, cost, data movement, Directed Acyclic Graph (DAG), Extreme Nondominated Sorting Genetic Algorithm-III (E-NSGA-III), makespan, Multilevel Dependent Node Clustering (MDNC), multi-objective optimization, Nondominated Sorting Genetic Algorithm-III (NSGA-III), Peer-to-Peer clustering, scientific workflows.

I. INTRODUCTION

IN scientific research community, several scientific workflows such as astronomy, bioinformatics, and physics usually are large-scale batch processing, they are also computation and data-intensive applications [1], [2]. They generally comprise more than thousand tasks in the whole workflow. This is usually carried out over distributed systems such as cluster, grid and cloud computing. To enable execution chain and data flow of multiple tasks, *Directed Acyclic Graph* (DAG) is the common method where a node in the graph denotes a task, and a directed edge symbolizes execution direction [1], [3]. With the DAG model, task schedulers can freely allocate tasks to distributed computing machines, nevertheless maintaining correct execution must also be determined in order to maximize the execution efficiency.

Recently, the execution of scientific workflows is commonly carried out on *Infrastructure-as-a-Service* (IaaS) cloud because it provides unlimited, self-manageable, scalable computing resources with affordable price [4]. The most popular service of IaaS for running scientific workflows is probably by means of *Virtual Machines* (VMs) due to its various type of cost, processing capacity, disk storage, and network bandwidth. Users can select a proper VM type for a particular task and they are usually charged by considering VM usage time in a pay-per-use model. Basically, in running scientific workflows, cloud users consider cost and makespan, which is the total execution time to finish all tasks in a workflow, as the first two priorities [4]. They prefer a scheduling plan offering reasonable cost at acceptable time. Users can balance cost and makespan objectives by allocating tasks to suitable VM type, and determining the start and stop time of each VM. However, another issue of real concern is data movement due to the data-driven nature of scientific workflows. Apart from the fact that these workflows usually produce tremendous data, the data exchange among tasks also occurs during the execution. This could lead to major congestion of traffic in cloud data center, which in turn, has a significant impact on network usage and energy consumption by network equipment.

With respect to energy consumption in cloud data center, the commonly believed factors which have direct influence, are energy demands by power distribution, cooling system, and network equipment [5], [6]. Data movement is a major concern in network equipment as it uses tremendous amount of energy. It has been reported that when computing resources (e.g. servers) in cloud data center have 15% load and fully generate energy proportionality, network equipment may consume as much as 50% of the overall power consumption [6]. Hence, reducing data movement can benefit cloud service providers in terms of energy cost. It must be borne in mind the right balance must be struck between the quality of service and efficient scheduling plan. This is where good solutions which optimize the three objectives mentioned becomes a challenging issue.

Dealing with data traffic in a cloud data center can be done at the network and application layers. However, doing so at the network layer is mostly concerned with transmission control of data packet according to network architecture such as network topology. Major concerns of such work lie in reducing latency and failure rate, and maximizing network utilization [7]. Moreover, eliminating traffic at the network layer cannot minimize cost and makespan for cloud users directly. Hence, this work adopts the approach of removing data transfer at the application layer by putting consecutive dependent tasks at the same machine. Scheduler which adopts the conventional DAG model does not allow unnecessary data transfer among tasks in workflows to be detected. This work takes advantage of *Peer-to-Peer clustering* [8] by grouping dependent tasks and allocating them to the same VM in order to decrease data movement. The work in [9] introduced a useful measure called 'Data Locality Ratio' where a portion of VM locality can be measured against other localities. Peer-to-Peer clustering can also improve VM locality as higher portion is preferable when running applications over distributed virtual environments. While Peer-to-Peer clustering manages to improve the efficiency in reducing data movement, it is limited to only three scenarios of connection dependent nodes as described in [8]. Also, the approach does not allow dependent nodes which are not direct parent to be packed together.

To overcome the limitation of Peer-to-Peer clustering, this work proposes a new clustering technique, *Multilevel Dependent Node Clustering* (MDNC). This technique packs several dependent nodes in different levels of DAG workflows. Besides, grouping two nodes with parent and child relationship together, MDNC also packs two nodes which may have some node(s) in between where Peer-to-Peer does not allow. This extends the ability to create more clusters of tasks in DAG workflows as data movement cannot occur among tasks in the same cluster. The number of clusters also implies the ability to reduce data transfer. Furthermore, MDNC still preserves the advantage of Peer-to-Peer clustering by maintaining the parallelism capability of DAG workflows.

It is commonly known that scheduling over dynamic heterogeneous computing resources is an *NP*-hard problem. This is well explained in [10], [11]. There exist several previous works where *Evolutionary Algorithms* (EAs) have been applied in the execution of scientific workflows on cloud, particularly in multi-objective scheduling optimization [4]. This work selects the recently EA, *Extreme Nondominated Sorting Genetic Algorithm-III* (E-NSGA-III) [12], to generate scheduling solutions. E-NSGA-III was shown superior to *NSGA-II* [13] and *NSGA-III* [14], its original version, in task scheduling. Nevertheless, E-NSGA-III has not been applied with clustering techniques in a scheduling problem. For these reasons, E-NSGA-III is chosen together with MDNC and Peer-to-Peer clustering to solve the multi-objective workflow scheduling, where *Cost*, *Makespan*, and *Data Movement* are three objectives for optimization. NSGA-III is also selected for the performance comparison as it performed well when working with Peer-to-Peer clustering [8]. The performance of MDNC is evaluated by comparing against the original DAG and Peer-to-Peer clustering. Five well-known scientific workflows [15] from different fields are selected for evaluation in this study, these are of CyberShake, Epigenomics, LIGO, Montage, and SIPHT.

It is commonly known that scheduling over dynamic heterogeneous computing resources is an *NP*-hard problem. This is well explained in [10], [11]. There exist several previous works where *Evolutionary Algorithms* (EAs) have been applied in the execution of scientific workflows on cloud, particularly in multi-objective scheduling optimization [4]. This work selects the recently EA, *Extreme Nondominated Sorting Genetic Algorithm-III* (E-NSGA-III) [12], to generate a scheduling solution. E-NSGA-III was shown superior to *NSGA-II* [13] and *NSGA-III* [14], its original version, in task scheduling. Nevertheless, E-NSGA-III has not been applied with clustering techniques in scheduling problem. For these reasons, E-NSGA-III is chosen together with MDNC and Peer-to-Peer clustering to solve the multi-objective workflow scheduling, where *Cost*, *Makespan*, and *Data Movement* are three objectives for optimization. NSGA-III is also selected for the performance comparison as it performed well when working with Peer-to-Peer clustering [8]. The performance of MDNC is evaluated by comparing against the original DAG and Peer-to-Peer clustering. Five well-known scientific workflows [15] from different fields are selected for evaluation in this study, these are of CyberShake, Epigenomics, LIGO,

Montage, and SIPHT.

The organization of the paper is as follows. It begins with description of related works in Section 2. The formulation of workflow scheduling problem is discussed in Section 3. Section 4 presents the description of MDNC. E-NSGA-III is described in Section 5 and Section 6 elaborates the multi-objective scheduling by using E-NSGA-III with MDNC. Section 7 presents characteristics of the five scientific workflows. This is followed by Section 8 and Section 9 where the results and the discussions of this study are presented respectively. The paper is concluded in Section 10 where the contributions are summarized and future direction are suggested.

II. RELATED WORK

It is widely known that optimizing workflow scheduling in a heterogeneous distributed environment is an *NP*-hard problem [10], [11]. Hence, plentiful researches have carried out on various aspects over the last two decades. Running multiple tasks in a workflow over distributed systems requires suitable tools for defining the precedence of task relationship and data dependency, therefore the DAG model is widely used for this. Recently, large and complex scientific workflows are often executed on IaaS cloud providing heterogeneous VMs with different cost and performance. To allocate multiple tasks to dynamic machines (e.g. VM), several scheduling objectives have been studied from different perspectives such as cloud user and service provider perspectives. Related works in this section are divided into two groups. The first group discusses previous works where DAG model is involved and the second is those which involve scheduling of workflow on cloud.

A. DAG MODEL

DAG workflow has been used in various types of parallel and distributed systems over several decades. For a multiprocessor machine having more than one central processing units, computation and communication time are major parts which impact the performance. Reducing communication time while limiting process time was challenging in multiprocessor systems. Linear clustering scheme [16] and Dominant Sequence Clustering (DSC) [17] had been proposed for this. Both works used computation and communication time of tasks in a workflow to determine a cluster of tasks. They packed two or more tasks together. Grouping did not increase the computation time of workflow. However, the determination of clusters requires both computation and communication time. This led to some drawbacks as if both of these two times of tasks were changed, clusters had to be revised.

Another common approach in DAG workflow is horizontal clustering where type and level of tasks are considered whilst clustering [18]. In this approach, at each level of workflow, tasks having the same type were packed to the same cluster. The approach was able to increase the utilization of VMs. Similarly, Bag of Tasks [19], enhanced the VM utilization by grouping all types of tasks at the same level. Nevertheless, due to aiming to improve the utilization of VMs, both horizontal clustering and BoT did not consider with parallelism of workflow where grouping several dependent tasks at the same level might degrade the performance of workflow.

Peer-to-Peer clustering [8] is the recent technique for grouping tasks of DAG workflows. It has been proposed for reducing data movement and increasing VM utilization. Unlike works in [16] and [17], Peer-to-Peer used only the DAG structure without computation and communication time to determine clusters. It also preserved the workflow parallelization while eliminating data transfer. The limitation of Peer-to-Peer was that it was applicable to only in three scenarios of DAG, these were of a cluster of consecutive Peer-to-Peer nodes, a cluster of Peer-to-Peer nodes at the start node, and a cluster of Peer-to-Peer nodes at the end node. This limitation did not allow the Peer-to-Peer technique to apply DAG workflows where all three scenarios did not exist such as CyberShake and SIPHT.

Due to the importance of communication task in cloud applications such as cloud gaming, cloud storage, cloud backup and online office, the communication-aware model of cloud applications, called CA-DAG [20], presented communication task as another type of node in DAG instead of using edge. In CA-DAG, circular node stood for computation task while square node typified communication task. This allowed making separate resource allocation decisions, assigning VMs to handle computing jobs and network resources for information transmissions.

As data transferring has a significant impact to processing of workflows over distributed systems, this work aims to eliminate unnecessary data movement at the application level. To date, there has not been any previous works on the use of DAG in aiding scheduler. Also, Peer-to-Peer clustering had been limited to three scenarios as discussed above. This work attempts to improve Peer-to-Peer clustering further, which also uses DAG as a basis, in reduction of data movement and in enhancing VM utilization.

B. WORKFLOW SCHEDULING

Due to its NP-hard nature, the problem of workflow scheduling on cloud is still active in both areas of optimization and cloud computing. Several previous works have proposed algorithms to deal with this problem as listed in Table 1. These algorithms are presented with an aim to allocate multiple tasks to heterogeneous VMs according to their objectives. They can be divided into two categories, heuristic algorithm, and metaheuristic based on EAs. Most of the heuristic algorithms [19], [21]–[26] limit their problems to single-objective optimization under some constraints while the second category [27]–[34] may have up to three objectives. Referring to Table 1, cost and time-based measurement (e.g. makespan, and execution time) are common objectives. Similarly, the work in [4] has been reported that most priority objectives of cloud users are cost and makespan. Moreover, Table 1 reaffirms that CyberShake, Epigenomics, LIGO, Montage, and SIPHT are the most widely used scientific workflows in the problem of workflow scheduling on cloud.

TABLE 1. Algorithms of workflow scheduling on cloud

Proposed Algorithm	Algorithm Type	Objectives	Constraints	Workflows
IaaS Cloud Partial Critical Path (IC-PCP) and IaaS Cloud Partial Critical Path with Deadline Distribution (IC-PCPD2) [21], 2012	Heuristic	Cost	Deadline	CyberShake, Epigemomics, LIGO, Montage, SIPHT
Cloud-based Workflow Scheduling (CWSA) [22], 2016	Heuristic	Makespan	-	CyberShake, SIPHT
Multiclouds Partial Critical Paths with Pretreatment (MCPCPP) [23], 2016	Heuristic	Cost	Deadline	CyberShake, Epigemomics, LIGO, Montage, SIPHT
Budget-driven Algorithm [19], 2017	Heuristic	Execution time	Budget	CyberShake, Epigemomics, LIGO, Montage, SIPHT
Dynamic Scheduling of Bag of Tasks based workflows (DSB) [24], 2018	Heuristic	Cost	Deadline	CyberShake, Epigemomics, LIGO, Montage, SIPHT
Partition Problem based Dynamic Provisioning and Scheduling (PPDPS) [25], 2018	Heuristic	Cost	Deadline	CyberShake, Epigemomics, LIGO, Montage, SIPHT
Budget Deadline Aware Scheduling (BDAS) [26], 2019	Heuristic	Cost	Budget, Deadline	CyberShake, Epigemomics, LIGO, Montage, SIPHT
Multi-objective Strategies based on NSGA-II (MS-NSGA-II) [27], 2012	Evoluationary	Cost, Execution time	-	CyberShake, Epigemomics, LIGO, Montage, SIPHT
Deadline Based Resource Provisioning and Scheduling (DBRPS) [28], 2014	Evoluationary	Cost	Deadline	CyberShake, LIGO, Montage, SIPHT
Evolutionary Multi-objective Optimization based on NSGA-II (EMO-NSGA-II) [29], 2016	Evoluationary	Cost, Makespan	-	CyberShake, Epigemomics, LIGO, Montage, SIPHT
Cost Effective Genetic Algorithm (CEGA) [30], 2016	Evoluationary	Cost	Deadline	CyberShake, Epigemomics, LIGO, Montage
Security and Cost-Aware Algorithm based on Genetic Algorithm (SCA-GA) [31], 2017	Evoluationary	Cost	Deadline, Security level	CyberShake, Epigemomics, SIPHT
Greedy-Ant [32], 2017	Evoluationary	Speedup	-	CyberShake, SIPHT
Hybrid Bio-inspired Metaheuristic for Multi-objective Optimization (HBMMO) [33], 2018	Evoluationary	Cost, Makespan, Load balance of VMs	-	CyberShake, Epigemomics, LIGO, Montage, SIPHT
Fluctuation-Aware and Predictive Scheduling based on Genetic Algorithm (FAP-GA) [34], 2018	Evoluationary	Cost	Deadline	CyberShake, Epigemomics, Montage

This paper focuses on the tools based on EAs as they have intensively used in solving multi-objective optimization problems. It has been reported that Multi-Objective Genetic Algorithm (MOGA), Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) and are some of the popular algorithms for this task [4]. Referring to Table 1, there are five algorithms based on the genetic algorithm including Multi-objective Strategies based on NSGA-II (MS-NSGA-II) [27], Evolutionary Multi-objective Optimization based on NSGA-II (EMO-NSGA-II) [29], Cost Effective Genetic Algorithm (CEGA) [30], Security and Cost-Aware algorithm based on Genetic Algorithm (SCA-GA) [31], and Fluctuation-Aware and Predictive Scheduling based on Genetic Algorithm (FAP-GA) [34]. While MS-NSGA-II and EMO-NSGA-II considered cost and makespan (i.e. execution time) as scheduling objectives, the rest of them were concerned with minimizing execution cost while satisfying the deadline constraints.

One of the challenges of using the genetic algorithm is a chromosome representation which can be converted to a solution to the problem. MS-NSGA-II used two chromosomes presenting the solution where the first chromosome referred to the association of tasks and VM nodes and the second one denoted task ordering. Besides cost and execution time, MS-NSGA-II was also concerned with the communication overhead. Hence, this overhead was used in calculating execution time. Unlike MS-NSGA-II, EMO-NSGA-II implemented three-row chromosome s for solution representation where the first row denoted task order, the second row typified VM ID, and the last one presented VM type. This proprietary representation required specific implementation for genetic operators where population initialization, order crossover, task-to-instance crossover, and order mutation of EMO-NSGA-II also were proposed for generating and manipulating a chromosome. Similarly, CECA and FAP-GA used the same approach as EMO-NSGA-II for chromosome representation, however, while EMO-NSGA-II dealt with minimizing cost and makespan, CECA and FAP-GA considered makespan as the deadline constraint where obtained solutions ought to have makespan value lower than the predefined deadline.

Besides the deadline, SCA-GA also included the security level as an additional constraint. To represent a specific concern, SCA-GA used a group of slots referring to a task. For example, using four slots represented the task where the first, second, third, and fourth slots referred to task ID, VM ID, VM type, and security level respectively. Therefore, if there were five tasks in a workflow, the length of the chromosome of SCA-GA was equal to 20 slots. Also, SCA-CA used overhead models of security level as a part for computing makespan where each security level reflected the overhead time for data encryption and decryption when data was exchanged among VMs.

Instead of using the genetic algorithm, Deadline Based Resource Provisioning and Scheduling (DBRPS) [28], Greedy-Ant [32], and Hybrid Bio-inspired Metaheuristic for Multi-objective Optimization (HBMMO) [33] had been implemented based on PSO, ACO, and Symbiotic Organisms Search (SOS) [35] respectively. To represent a solution in DBRPS, the length of the particle of PSO was equal to the number of tasks in a workflow where the value of each slot was a real number used for mapping VM instances. Once the algorithm terminated, the values in the particle were converted to be a scheduling plan. Due to the consideration of resource provisioning, DBRPS considered the start and shutdown time for calculating makespan when a VM was launched and terminated. Greedy-Ant extended the ACO-based workflow scheduling techniques where two main contributions of Greedy-Ant were introducing two-phase operations and presenting new heuristic information. In each iteration, Greedy-Ant mainly operated two phases where, in the first phase, an ant colony was employed to search task sequences and then the allocation of VMs by a greedy policy was performed in the second phase. Moreover, Greedy-Ant applied task dependency in a workflow as new heuristic information for updating the pheromone of ants.

While DBPS and Greedy-Ant focused on single-objective optimization, HBMMO dealt with multi-objective workflow scheduling where cost, makespan and load balance of VMs were considered. HBMMO was implemented by using a hybrid approach. It combined SOS and the heuristic algorithm, called Predict Earliest Finish Time (PEFT) [36]. HBMMO used organisms of the ecosystem for presenting population. Each organism referred to a valid feasible schedule of a workflow and the length of the organism equaled the number of tasks where the position of each slot in the organism denoted the task and the value presented VM ID. HBMMO generated the initial organisms of the ecosystem by PEFT for determining task priority and allocating each task to suitable VMs according to its priority.

In the multi-objective optimization of the workflow scheduling problem, most of the previous works had focused on the most common objectives, Cost and Makespan as they are the two most common for user requirements. Nevertheless, in execution of large and complex scientific workflows, an end user may make a request for scheduling plan from cloud providers who are in charge of cloud data center and infrastructure in offering cloud services. They may also be concerned with additional objectives in order to gain higher revenue and reduce operation cost. Therefore, this research includes Data Movement as another important objective since it has a significant impact on both network utilization and energy consumption. In order to achieve this, this work proposes a novel clustering technique, MDNC, fulfilling the gap of Peer-to-Peer clustering. The description of MDNC is presented in Section IV.

III. WORKFLOW-SCHEDULING PROBLEM FORMULATION

In this section, models for workflow and cloud system are presented. Definition and problem formulation of three objective functions are also elaborated. For ease of description, important notations and their definitions used throughout this paper are listed in Table 2.

TABLE 2. Notations and their descriptions

Notations	Descriptions
W	Scientific workflow represented by DAG model
T	Tasks in the workflow W
E	Dependency edges in the workflow W
VM	Set of virtual machines provided in a IaaS cloud
$Cost$	Cost objective function
MS	Makespan objective function
DM	Data Movement objective function
n	Number of tasks in the workflow
T_j	The task j, j = $1, ..., n$
S_j	Start time of task T_j
C_j	Completion time of task T_j
RPT_j	Reference processing time of task T_j
ET_j	Estimated runtime of task T_j
CT_j	Communication time of task T_j
E_{ij}	The edge between task T_i and T_j (i.e. T_i is parent of T_j)
k	Number of VM instances used for executing the workflow
VM_m	The VM m, m = $1, ..., k$
VM_{mj}	VM_m that executes task T_j
BH_m	Billed hour of VM_m
CH_m	Unit cost per hour of VM_m
SVM_m	Start time of VM_m
CVM_m	Completion time of VM_m
SLD_{mj}	Slowdown rate of VM_m that executes task T_j
DS_{ij}	Data size sent by task T_i to task T_j
DTR_{ij}	Data transfer rate (e.g. 1 Gbps) between task T_i and T_j
$Move_{ij}$	Data movement between task T_i and T_j
$tasks(VM_m)$	List of tasks on VM_m
$previous(T_j)$	List of the previous tasks of task T_j at the same VM_m
$parents(T_j)$	List of the parent tasks of task T_j
$PreC_j$	Maximum completion time of tasks in $previous(T_j)$
$ParC_j$	Maximum completion time of tasks in $parents(T_j)$

A. WORKFLOW MODEL

DAG is a common representation of a workflow, its tasks and dependencies. While a node in the graph represents a task in the workflow, a directed edge typifies dependency between two connected tasks. Let a workflow W is a set of (T,E) where $T=\{T_1,T_2,T_3,\dots,T_n\}$ is the set of tasks having n different tasks and $E=\{E_{ij} \mid (1 \leq i \leq n, 1 \leq j \leq n, i \neq j)\}$ is the set of dependency edges between two tasks. An edge E_{ij} denotes that task T_i is the parent of task T_j so that T_j cannot be executed until T_i is finished. In scientific workflows, the output of the parent task is usually used as the input data of the child task. This leads to data transferring if the parent and the child task are not in the same computing machine.

In order to create a workflow scheduling, this work assumes that each task T_j has its reference processing time, denote by RPT_j. This processing time is used for computing the estimated runtime of T_j, denoted by ET_j, after T_j is allocated to a specific VM. In addition, each edge E_{ij} has a weight representing the data size sent from task T_i to task T_j, denoted by DS_{ij}. To calculate data transfer time, the data transfer rate (e.g. 1 Gbps) is represented as the speed of network, and DTR_{ij} is the data transfer rate between T_i and T_j. In case of T_j having multiple inputs from its parents, the assumption is that transferring of multiple input files is sequential and the communication time of T_j, denoted by CT_j,

TABLE 3. List of VM types based on Amazon EC2 [28]

VM Type	EC2 Units	Processing Capacity (MFLOPS)	Cost/Hour	Slowdown Ratio
m1.small	1	4,400	\$0.06	26.00
m1.medium	2	8,800	\$0.12	13.00
m1.large	4	17,600	\$0.24	6.50
m1.xLarge	8	35,200	\$0.48	3.25
m3.xLarge	13	57,200	\$0.50	2.00
m3.doubleXlarge	26	114,400	\$1.00	1.00

is the total time spent in receiving all input files.

B. CLOUD SYSTEM MODEL

The target system in this work is IaaS cloud which offers various types of VM at different costs, as well as processing, memory, and storage capacities. From user point of view, commercial IaaS providers, (such as Amazon Web Service (AWS), Google Cloud Engine, and Window Azure), provide unlimited VM instances where users can create and launch an infinite number of VMs. Formally, suppose that an infinite set $VM=\{VM_1,VM_2,VM_3,\dots\}$ is all VM instances in an IaaS system. After a scheduling plan for a workflow execution is proposed, the number of VMs is also determined. If k is the number of all VM instances used for running a workflow, VM_m denotes VM in the m index where $VM_m \in VM$ and $1 \leq m \leq k$.

This work adopts the same VM types as in [28] which is based on Amazon EC2, the VM model of AWS provider. Each VM type consists of virtual processor, processing capacity, cost, and slowdown ratio as shown in Table 3. A virtual processor is represented by the processor unit of Amazon EC2, while processing capacity is typified in MFLOPS (Million Floating Point Operations Per Second). Slowdown ratio is the ratio of performance degradation comparing to the fastest VM, which is m3.doubleXlarge, in terms of MFLOPS. In most cases, the VM cost is charged per hour interval and each partial VM hour consumed is billed as a full hour.

To execute a scientific workflow in IaaS, the following common practice is assumed.

1) The workflow is executed in a single cloud data center.
2) The scientific workflow is batch processing, and a scheduling plan is off-line mode which means that the scheduling plan is done before the workflow execution starts.
3) The task scheduling is considered as non-preemptive scheduling which implies that a running task cannot be interrupted in the middle of the execution until its execution is finished.
4) The minimum number of VMs executing a workflow is one machine, while the maximum case is equal to the number of tasks because it is possible that all tasks in a workflow are assigned to a single VM or each task is allocated to an individual VM.
5) Each task can only be allocated to a single VM.
6) Each VM can run several tasks, but it can process only one task at a time.
7) Once a scheduling plan is proposed, the list and the execution order of tasks in each VM are also determined.
8) There is no transfer time for the tasks at the same VM.
9) The data transfer rate among VMs is assumed to be the same.
10) The per-pay-use model is applied in per hour period, and a partial running hour of VM is charged as one full period.

C. MULTI-OBJECTIVE SCHEDULING

Workflow scheduling in this work can be considered as multi-objective optimization where more than one conflicting objectives must be optimized simultaneously. Therefore, a set of solutions has to satisfy all different objectives. The multi-objective optimization can be modeled formally as :

$$\begin{aligned} &minimize\ F = (f_1(x), f_2(x), f_3(x), \dots, f_o(x)) \\ &subject\ to \quad x \in X \end{aligned} \tag{1}$$

where X is the decision space and $f(x)$ comprises o objective functions. In a single optimization problem, the best single solution can be found. This is in contrast to a multi-objective problem where multiple solutions are accepted due to the difficulty in finding the optimum for all conflicting objectives. To compare the performance among possible solutions, a common measurement, *Pareto Dominance*, is used where in the minimization problem, solution a is said to dominate solution b if and only if,

$$\begin{aligned} &\forall i \in \{1,\dots,o\} : f_i(a) \leq f_i(b) \quad \wedge \\ &\qquad \exists j = \{1,\dots,o\} : f_j(a) < f_j(b) \end{aligned} \tag{2}$$

where $a,\ b \in X$. A solution that is not dominated by others is called *Nondominated* or *Pareto solution* and the set of all Pareto optimal solutions is commonly known as *Pareto Front*. For an optimal solution in this front, the improvement in the value of any objective function usually implies decreasing in some of the other objective values.

In this work, as the multi-objective scheduling for scientific workflow on cloud is to minimize *Cost*, *Makespan* and *Data Movement*, therefore, the problem can be formulated as follows :

$$minimize\ F = (Cost, MS, DM) \tag{3}$$

where *Cost* is the function of cost objective, *MS* denotes the makespan objective function, and *DM* is the data movement objective function. The description of three objective functions is provided in the following subsections.

1) Cost

The cost of all VMs running a workflow can be determined by the following equations :

$$Cost = \sum_{m=1}^{k} BH_m \times CH_m \tag{4}$$
$$BH_m = CVM_m - SVM_m \tag{5}$$
$$CVM_m = max_{v \in tasks(VM_m)}(C_v) \tag{6}$$
$$SVM_m = min_{v \in tasks(VM_m)}(S_v). \tag{7}$$

Referring to (4), BH_m denotes the billed hour of VM_m while CH_m is the unit cost per hour of VM_m and depends on the type of VM_m as listed in Table 3. Both BH_m and CH_m are used in determination of the *Cost*. Note that BH_m is the different time between the start time (SVM_m) and the completion time (CVM_m) of VM_m. SVM_m is the minimum start time of all tasks executed on VM_m and CVM_m is the maximum completion time of all tasks on VM_m. In order to compute these two values, all tasks executed on VM_m have to be determined where they can be retrieved by using *tasks(*VM_m*)* function.

2) Makespan

Makespan is often used interchangeably to mean maximum completion time [4], [37]–[42]. It represents the finish time of the last task in the workflow and can be determined by the following equations :

$$MS = max_{j \in T}(C_j) \tag{8}$$
$$C_j = S_j + CT_j + ET_j \tag{9}$$
$$S_j = max(PreC_j, ParC_j) \tag{10}$$
$$PreC_j = max_{p \in previous(T_j)}(C_p) \tag{11}$$
$$ParC_j = max_{q \in parents(T_j)}(C_q) \tag{12}$$
$$CT_j = \sum_{i \in parents(T_j)} DS_{ij} \times DTR_{ij} \tag{13}$$
$$ET_j = RPT_j \times SLD_{mj}. \tag{14}$$

Equation (9) indicates three components used for computing the completion time of T_j, these are of S_j, CT_j, and ET_j. In order to determine S_j, the maximum value between $PreC_j$ and $ParC_j$ is selected where $PreC_j$ is the maximum completion time of the previous tasks of T_j and $ParC_j$ is the maximum completion time of the parent tasks of T_j. $PreC_j$ ensures that a VM processes one task at a time, while $ParC_j$ guarantees that a child task starts once all its parents finished their execution. Referring to T_j, its previous tasks on the same VM and its parent tasks can be retrieved by *previous(*T_j*)* and *parents(*T_j*)* function respectively. Moreover, CT_j is the summation of the communication time of T_j receiving all data files from its parent and ET_j is RPT_j multiplied by the slowdown ratio of VM_m that runs T_j (SLD_{mj}). This ratio is given in Table 3.

3) Data Movement

In this work, data movement refers to the number of data transfer during the workflow execution. It can be formulated by (15) and (16) as below :

$$DM = \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} Move_{ij} \tag{15}$$

$$Move_{ij} = \begin{cases} 1, & \text{if } E_{ij} \text{ exists and } VM_{mi} \neq VM_{mj} \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

The determination of of data movement between task T_i and task T_j is defined as $Move_{ij}$ in (16). $Move_{ij}$ is equal to 1 if there is E_{ij} and VM_{mi} is not the same as VM_{mj}, otherwise it is 0.

IV. MULTILEVEL DEPENDENT NODE CLUSTERING

For the reduction of data movement of workflow, this paper proposes the task clustering technique, *Multilevel Dependent Node Clustering* (MDNC), in order to avoid unnecessary movement at the application level. As mentioned earlier, DAG represents the execution order of tasks in a workflow and data flow between tasks. Usually, parent tasks supply outputs to their children as input data, which consequently cause data movement. In a single machine, this circumstance can be omitted due to all tasks taken place together. In distributed systems, however, tasks are allocated and processed across different machines (e.g. VMs) in order to take advantage of parallel processing. Data movement can be apparent when two dependent tasks, having a parent-child relationship, are assigned to different VMs. Once data movement occurs, the penalty time of transferring data depends on a ration of data locality appearing among VM locality, Host locality, Rack locality, and Off-rack locality as discussed in [9] where among these models, VM locality is most preferred due to prevention of data movement. Off-rack locality ought to be avoided as much as possible because of having the largest penalty time. The challenge of the allocation of tasks over distributed systems lies in balancing between parallel processing and data movement.

Similar to Peer-to-Peer clustering, MDNC is concerned with the parallelism of a workflow. It does not pack independent tasks in the same cluster in order to allow them to be simultaneous execution. However, MDNC focuses not only on consecutive dependent tasks as Peer-to-Peer does but also on dependent tasks having an indirect connection in different levels. The level of each task can be determined in DAG, as shown in Fig. 1(a), where starting tasks are at the first level and their children are in the second level and so on. In case of a child task having several parents in multiple levels, its level is the one after the highest level of its parents. Fig. 1(b) and Fig. 1(c) reveal the difference between the result of Peer-to-Peer clustering and that of MDNC in the same original DAG. Peer-to-Peer clustering groups Peer-to-Peer tasks having a direct connection in consecutive levels only as shown in Fig. 1(b), while MDNC makes possible

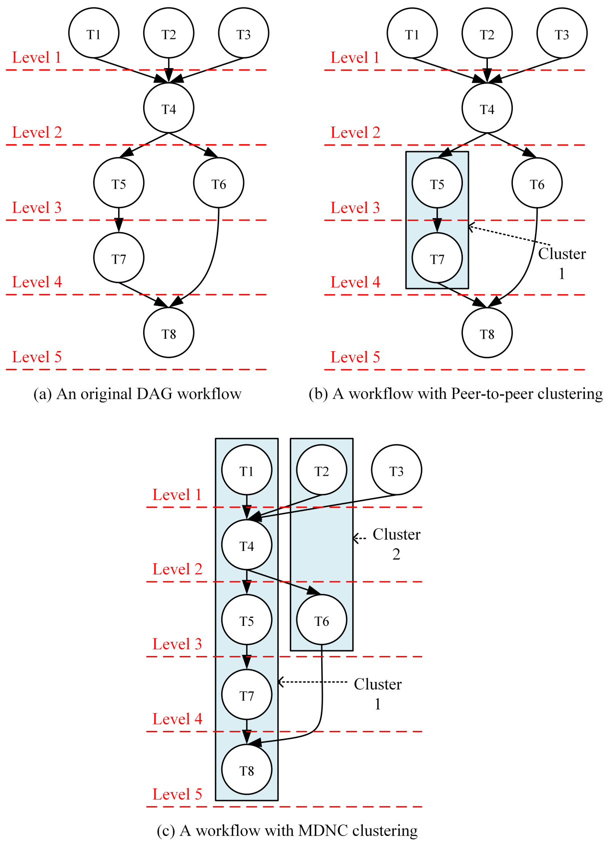
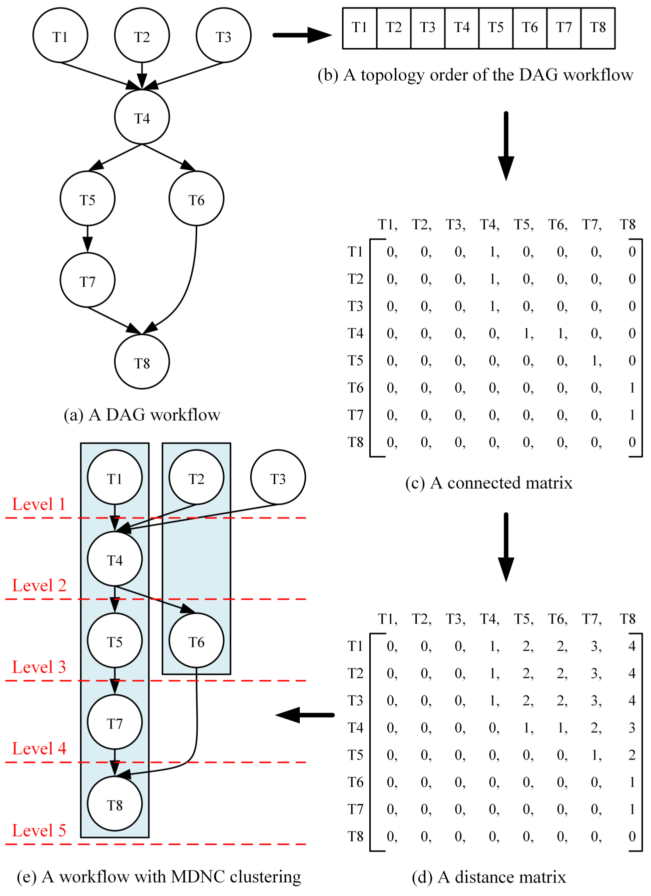

T1
T2
T3
T4
T5
T6
T7
T8
Level 1
Level 2
Level 3
Level 4
Level 5
(a) An original DAG workflow
(b) A workflow with Peer-to-peer clustering
Cluster 1
Cluster 2
(c) A workflow with MDNC clustering

FIGURE 1. The difference between Peer-to-Peer clustering and MDNC outcomes.



	T1,	T2,	T3,	T4,	T5,	T6,	T7,	T8
T1	0,	0,	0,	1,	0,	0,	0,	0
T2	0,	0,	0,	1,	0,	0,	0,	0
T3	0,	0,	0,	1,	0,	0,	0,	0
T4	0,	0,	0,	0,	1,	1,	0,	0
T5	0,	0,	0,	0,	0,	0,	1,	0
T6	0,	0,	0,	0,	0,	0,	0,	1
T7	0,	0,	0,	0,	0,	0,	0,	1
T8	0,	0,	0,	0,	0,	0,	0,	0

	T1,	T2,	T3,	T4,	T5,	T6,	T7,	T8
T1	0,	0,	0,	1,	2,	2,	3,	4
T2	0,	0,	0,	1,	2,	2,	3,	4
T3	0,	0,	0,	1,	2,	2,	3,	4
T4	0,	0,	0,	0,	1,	1,	2,	3
T5	0,	0,	0,	0,	0,	0,	1,	2
T6	0,	0,	0,	0,	0,	0,	0,	1
T7	0,	0,	0,	0,	0,	0,	0,	1
T8	0,	0,	0,	0,	0,	0,	0,	0

FIGURE 2. Steps in MDNC clustering.

for dependent nodes which do not connect directly to be in the same cluster (i.e. *Cluster 2*) as shown in Fig. 1(c). This feature does not violate the parallel processing capacity of DAG and it also aids formation of more clusters. Moreover, in order to manage the critical path of DAG (i.e. the longest path), MDNC searches and packs all tasks in the critical path as the first cluster of DAG as in *Cluster 1* in Fig. 1(c).

The process of constructing the MDNC cluster is presented as in Fig. 2(a) to Fig. 2(e). Fig. 2(a) shows an example of the DAG workflow having eight tasks. The initial process is to determine a topological order of the DAG workflow as depicted in Fig. 2(b). The common algorithms for topological sorting (e.g. Kahn 's algorithm [43]) have linear time complexity which is $O(T + E)$ where T denotes the number of tasks in DAG and E is that of edges [44]. For ease of constructing a cluster, all tasks are represented by a matrix in the conventional sense as shown in Fig. 2(c). The next step is computing a distance matrix as depicted in Fig. 2(d) which denotes the distance among tasks in the workflow. The distance value of the matrix element represents the longest path traveling between two tasks. Referring to Fig. 2(a), there are two paths from task *T1* to task *T8*; the shortest path via task *T6* having three hops and the longest path via task *T5* having four hops. In this case, the distance value from *T1* to *T8* is equal to four. Fig. 3 describes the pseudocode for computing the distance matrix. The algorithm processes over the connected matrix in the row-major approach where each row is iterated and then its columns are repeatedly accessed. The basic idea is finding the accumulated hops of each task to its dependent tasks. In the topological order, it guarantees that a parent task always has a lower index of its children. Hence, the two inner loops (line 4 and line 6) take this advantage by iterating only tasks having a higher index than the current parent index.

Algorithm Compute Distance Matrix

```
1:  $T \leftarrow number\ of\ tasks;$
2:  $matrix \leftarrow connected\ matrix\ with\ T \times T\ size;$
3:  for $i \leftarrow 1$ to $T$ do
4:      for $j \leftarrow i + 1$ to $T$ do
5:          if $matrix[i][j] > 0$ do
6:              for $k \leftarrow j + 1$ to $T$ do
7:                  if $matrix[j][k] > 0$ do
8:                      $matrix[i][k] = matrix[i][j] + matrix[j][k];$
9:                  end if
10:             end for
11:         end if
12:     end for
13: end for
```

FIGURE 3. Algorithm for converting a connected matrix to distance matrix.

Once the distance matrix is constructed, the MDNC cluster is determined by using the algorithm depicted in Fig. 4. As mentioned above, the first MDNC cluster is a group of tasks in the longest path of DAG. Referring to Fig. 4, the maximum hop in the distance matrix is used as the first target of the algorithm (line 6). The maximum hop is decremented each round and the algorithm terminates when the maximum hop becomes zero (line 5). In each iteration of the target hop, each row of the distance matrix is iterated for assessment whether it holds the target hop (line 9). To reduce time processing, the second outer while loop uses the variable *minRow* for skipping tasks in the row index already done and continuingly processing the rest of other tasks (line 8 –11). In case of the current row having the target hop, its association task is used for initial MDNC cluster (line 13 –14), otherwise, its task is skipped. Referring to Fig. 2(d), *T1* at the first row is used for creating the first MDNC cluster due to its row containing the maximum hop (which is four hops in this case). In some DAG workflows, it may be possible to have more than one end tasks at the end of the workflow, the algorithm marks the first end task appearing in the topological order as the destination of current path (line 16) and the last task being added to a cluster (line 34). Hence, the next step is finding the intermediary tasks from the initial task to the destination task. In this example, *T1* is the start task while *T8* is the target task. In order to acquire in-between tasks to the cluster, the algorithm starts from the nearest available task to the most further one (line 18 –33). Fortunately, each row of the distance matrix is managed by topological order, so that the distance is already sorted from the smallest distance to the largest. In the case of *T1*, task *T4* will be added as the task in the nearest hop. In some cases, it is possible that there are more than one tasks having the same hop for the initial task (e.g. *T5* and *T6* in case of *T1*). To make sure of finding the right path, the algorithm selects the task that has the total hop from the initial task to the destination task equal to the maximum hop (line 22). For determining the second intermediary task of *T1*, *T5* will be added to cluster because the distance from *T1* to *T5* is two hops, the distance from *T5* to *T8* is two hops and in total, the number of accumulated hops is equal to the maximum hop which is four. Once an intermediary task is put into the cluster, its value in all rows will be set to be zero (line 24) for marking that it is not available for other clusters (i.e. they have already been clustered). For the same reason, all rows of a destination task, which are represented by *matrix.column(dest).allRows*, will change to zero (line 35). Also, when the algorithm finishes processing of each row in the matrix, all column values of the current row, denoted by *matrix.row(i).allColumns*, will be zero marking that the row is already processed (line 36). Finally, an MDNC cluster is stored in a list (line 37).

Apart from the fact that MDNC reduces data movement during the execution of DAG workflow, it also decreases problem complexity and the search space due to scheduler dealing with clusters of tasks instead of an individual single task. The next two sections describe the representation of genetic algorithm and illustrate the use of MDNC in the multi-objective scheduling on cloud using E-NSGA-III.

```
Algorithm Construct MDNC
 1: $T \leftarrow number\ of\ tasks$;
 2: $matrix \leftarrow distance\ matrix\ with\ T \times T\ size$;
 3: $clusterList \leftarrow \emptyset$;                    /* list of MDNC clusters */
 4: $minRow \leftarrow T$;
 5: while $matrix.maxValue > 0$ do
 6:     $maxHop \leftarrow matrix.maxValue$;
 7:     $i \leftarrow 1$;
 8:     if $minRow < T$ do
 9:         $i \leftarrow minRow$;
10:         $minRow \leftarrow T$;
11:     end if
12:     while $i\ in\ T$ and $matrix.maxValue = maxHop$ do
13:         if $maxHop\ in\ matrix.row(i)$ do
14:             $mdncList \leftarrow \emptyset$;          /* list of tasks in a MDNC cluster */
15:             $mdncList.add(i)$;
16:             $dest \leftarrow matrix.row(i).maxValue.firstIndex$;
17:             $hop \leftarrow matrix.row(i).minValueNotZero$;
18:             while $hop < maxHop$ do
19:                 $isHopFound \leftarrow false$;
20:                 $j \leftarrow i + 1$;
21:                 while $j\ in\ T$ and $isHopFound\ not\ true$ do
22:                     if $matrix[i][j] = hop$ and
                          $matrix[i][j] + matrix[j][dest] = maxHop$ do
23:                         $mdncList.add(j)$;
24:                         $matrix.column(j).allRows \leftarrow 0$;
25:                         $isHopFound \leftarrow true$;
26:                         $hop \leftarrow hop + 1$;
27:                     end if
28:                     $j \leftarrow j + 1$;
29:                 end while
30:                 if $isHopFound\ not\ true$ do
31:                     $hop \leftarrow hop + 1$;
32:                 end if
33:             end while
34:             $mdncList.add(dest)$;
35:             $matrix.column(dest).allRows \leftarrow 0$;
36:             $matrix.row(i).allColumns \leftarrow 0$;
37:             $clusterList.add(mdncList)$;
38:         else if $i < minRow$ do
39:             $minRow \leftarrow i$;
40:         end if
41:         $i \leftarrow i + 1$;
42:     end while
43: end while
```

FIGURE 4. Algorithm for constructing MDNC clustering.

V. E-NSGA-III

In order to address multi-objective scheduling on cloud, this work adopts E-NSGA-III, the improvement of NSGA-III. E-NSGA-III proposed to include extreme solutions at the initial population for improving the diversity of solutions. Both NSGA-III and E-NSGA-III use nondominated solutions with elitism approach in acquiring the convergence of solutions. The nondominated solutions, also called *Pareto solutions*, refer to solutions having at least one objective better than all other solutions and elitism typifies the selection approach that allows nondominated parent population surviving for the next generation. NSGA-III employed a set of well-distributed reference points to maintain the uniform distribution among solutions.

Fig. 5 shows 15 reference points of NSGA-III in a three-

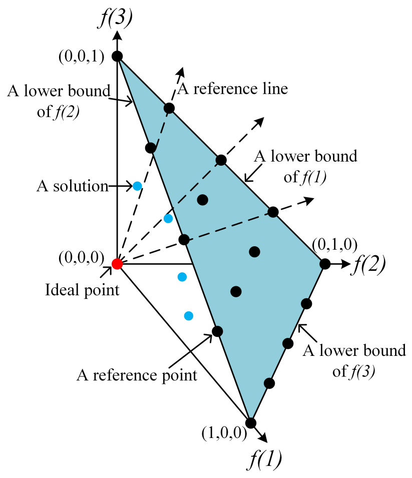


FIGURE 5. Reference points of NSGA-III in three-objective problems.

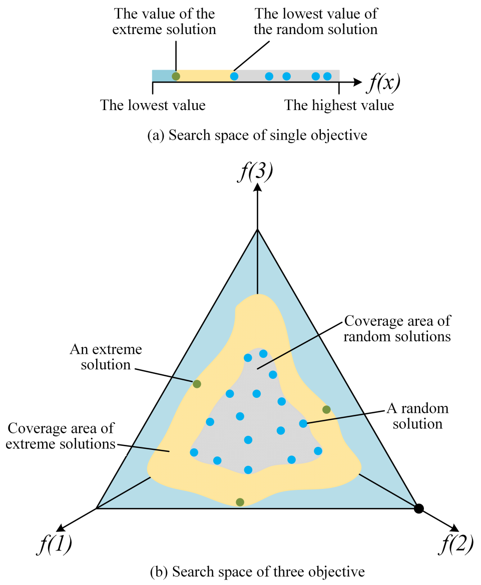

The value of the extreme solution
The lowest value of the random solution
f(x)
The lowest value
The highest value
(a) Search space of single objective
f(3)
Coverage area of random solutions
An extreme solution
A random solution
Coverage area of extreme solutions
f(1)
f(2)
(b) Search space of three objective

FIGURE 6. Illustration the coverage area of extreme solutions.

objective problem where each objective is divided into four parts. The number and locations of the reference points are determined by *Normal-Boundary Intersection* (NBI) [45]. The position of each reference point indicates normalized objective values ranging from 0 to 1 and each reference point has a reference line which is a logical line laid down from an ideal point to the reference point. The closest solution of each reference line will be associated with the reference point of that line as depicted in Fig. 5. Before solutions are associated with reference points, however, each objective value of a solution is normalized where a solution holding the lowest value is mapped to be 0 and located in the lower bound of each objective, while a solution carrying the highest value is mapped to 1. This can lead to the limitation of NSGA-III because if the lowest value of each objective was considerably high which, in fact, did not reflect the possible realistically solution. This may result in a situation where diversity of solutions may not be well distribution covering possible search spaces.

In E-NSGA-III, the best solution of the single objective should be one of the Pareto solutions when that objective is combined with other objectives as multi-objective problems. This can speed up the discovery of solutions in the search space, E-NSGA-III puts the best solution of a specific objective in the first generation as an extreme solution. For simplicity and efficiency, extreme solutions can be common cases usually suggested without the expensive and exhausted computation and the number of these solutions is often equal to the number of objectives [12].

Extreme solutions are put in order to support the algorithm in two aspects. Each extreme solution could be the lowest value of its associated objective and consequently, it reveals the possible lower space of the problem. Fig. 6(a) and Fig. 6(b) demonstrate coverage areas of random solutions and extreme solutions where the grey area is search space covered by the random solutions while the yellow one is coverage area of the extreme solutions. The first population is usually generated randomly so that they are often unpredictable and are rarely close to the possible lowest value. Adding extreme solutions is a short cut to reach the minimum value as shown in Fig. 6(a). Another aspect is the fact that extreme solutions can share their ability and characteristic to their offsprings with the intention that these children can fill the gap between the extreme solutions and the rest of the solutions where is the yellow areas in Fig. 6(a) and Fig. 6(b). Introduction of extreme solutions can also improve the diversity of solutions.

Fig. 7 presents the overview steps of E-NSGA-III where the green box is the extra step differing from the original NSGA-III. Extreme solutions which are the same size as the problem objectives (M) are produced in the initial population and then the rest of the solutions are randomly generated in order to fulfill the required population size (N). Crossover and mutation operations are performed for generating offsprings of the parent population. The number of offsprings is also the same number of N. Both parent and children generations are combined and each individual is assessed by objective functions. The objective values are used for population ranking by nondominated sorting where the first rank refers to an individual which is not dominated by others, and the second rank is individual dominated by only members in the first rank and so on. In the selection process, if the members in the first rank are equal to N, all of them will be selected. Otherwise, *Niche-Preserving Operation* [14] is performed. It is an iterative procedure for mapping the closest population to the reference points. However, it is possible that some reference points could not be associated with any individual. In each case, an individual is randomly selected in order to fill the missing reference point until the size of the selected population is equal to N. Once the termination criterion is completed (for example the number of generations is reached), the Pareto

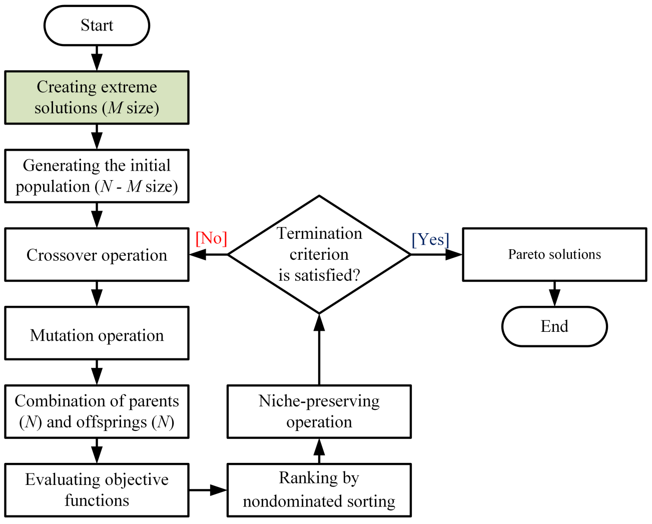

Start
Creating extreme solutions (M size)
Generating the initial population (N - M size)
Crossover operation
Mutation operation
Combination of parents (N) and offsprings (N)
Evaluating objective functions
Ranking by nondominated sorting
Niche-preserving operation
Termination criterion is satisfied?
[No]
[Yes]
Pareto solutions
End

FIGURE 7. Steps of E-NSGA-III.

solutions are finally determined.

The performance of E-NSGA-III in a multi-objective scheduling problem has been studied in [12]. The results revealed that E-NSGA-III outperforms NSGA-II and NSGA-III in terms of Hypervolume and Pareto front. This work is the first attempt to combine E-NSGA-III with the MDNC technique in order to improve the quality of solution in the multi-objective workflow scheduling on cloud. The detail of applying E-NSGA-III with the MDNC technique is elaborated in the next section.

VI. MULTI-OBJECTIVE WORKFLOW SCHEDULING USING E-NSGA-III WITH MDNC

The genetic algorithm is classified as a metaheuristic algorithm which refers to a method providing an upper-level template and operation. Users need to customize running steps and data representation of a problem matching a template of the algorithm when applying in real-world problems [46]. This section elaborates on the application of E-NSGA-III with MDNC for workflow scheduling problems. Fig. 8 presents the overview steps of transforming a DAG workflow to a chromosome in E-NSGA-III and converting a solution to a scheduling plan. In the initial step, tasks in the DAG workflow are clustered by the MDNC technique as depicted in Fig. 8(b). Consequently, the length of a chromosome is equal to the number of clusters plus the number of unable clustered tasks. Once the algorithm terminates, tasks in a chromosome are unfolded to full length to represent a scheduling plan.

The detail of chromosome representation and scheduling decoding are described in the following subsections. For ease of description, the scenario is assumed that the DAG workflow consists of eight tasks *(T1,T2,...,T8)* configured as in Fig. 8(a) and there are six VM types as stated in Table 3.

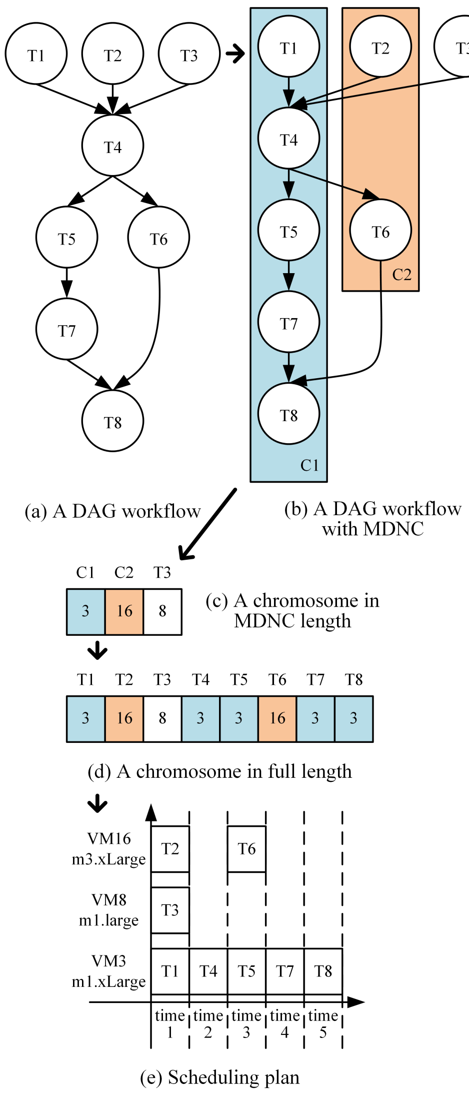


FIGURE 8. An example of a DAG workflow, chromosome representation and solution.

A. DATA REPRESENTATION

This work adopts the representation of task scheduling in NSGA-III that was proposed in [47]. However, the slight difference is that the length of a chromosome is determined after a DAG workflow is clustered by MDNC. Hence, the order in a chromosome indicates a topological order of clusters and unclustered tasks in the workflow. The integer value in the chromosome denotes VM specifications including ID and type. As this work allows each cluster or each task to be executed in any available VM type, the possible values are equal to the length of the chromosome multiply the number of VM types. In this scenario, there ought to be 18 possible values ranging from 0 to 17 used for representing VM id for a particular cluster and task. To determine VM type, VM id is interpreted by the modulo operation (MOD) (i.e. VM id MOD no. of VM types) where 0 refers to the m1.small type and 1 denotes the m1.medium type and so on. Referring to

Fig. 8(c), for instance, the first value in the chromosome is '3', indicating that the m1.xLarge type ought to be assigned to execute all tasks in Cluster *C1* (i.e. '3 MOD 6' is '3' which corresponds to the m1.xLarge type in Table 3). Similarly, as the second value in the chromosome is '16', indicating that the m3.xLarge type ought to be assigned to execute task *T2* and *T6* in Cluster *C2*.

B. SCHEDULING DECODING ALGORITHM

As a possible scheduling solution is encoded in a chromosome, a decoding process is required in order to obtain a scheduling plan. Prior to converting a solution to be a scheduling plan, a short chromosome in MDNC length ought to be unfolded to a chromosome in full length which is equal to the number of tasks in the DAG workflow as shown in Fig. 8(d).

To interpret a scheduling plan, this work adopts a decoding algorithm in [47] as shown in Fig. 9. Referring to Fig. 8(e), the example of a scheduling Gantt chart, task *T2* and *T6* are allocated at the same VM id, which is '37', having the m3.xLarge type. *T2* is executed at time slot 1. Even though time slot 2 of VM id '37' is free, *T6* cannot be assigned to that slot because its parent, task *T4*, is executed at time slot 2. Instead, *T6* is shifted to be processed at time slot 3. In the algorithm depicted in Fig. 9, a two-dimensional array is used for representing a scheduling plan where the number of rows reflects that of unique VM id in a solution and the number of columns denotes that of tasks plus one where the first column of each row indicates VM id and other rest columns refer to task id in each time slot.

Algorithm Scheduling Decoding

```
$T \leftarrow number\ of\ tasks$;
$solution \leftarrow solution\ array\ with\ VM\ id\ having\ T\ elements$;
$tasks \leftarrow task\ array\ with\ topological\ sorting\ having\ T\ elements$;
$V \leftarrow number\ of\ unique\ VM$;
$vms \leftarrow unique\ VM\ array\ with\ ascending\ sort\ having\ V\ elements$;

/* The output array where the first column of $plan$ represents VM id
and the rest columns are task id in each time slot */
$plan \leftarrow scheduling\ plan\ array\ with\ V \times (T+1)\ size$;

/* Initial $plan$ */
for $i \leftarrow 1$ to $V$ do
    $plan(i, 1) \leftarrow vms(i)$;
    $i \leftarrow i + 1$;
end for

for $j \leftarrow 1$ to $T$ do
    $task \leftarrow tasks(j)$;
    $vmId \leftarrow solution(j)$;
    $row \leftarrow plan.getRowOfVm(vmId)$;
    $col \leftarrow plan.getFirstEmptyColumnOfRow(row)$;
    if $task.hasParent\ is\ true$ do
        $parents \leftarrow task.getParentArray$;  /* Array of parents */
        for $k \leftarrow 1$ to $parents.size$ do
            $parrentCol \leftarrow plan.getColumnOfTask(parents(k))$;
            if $parentCol \geq col$ do
                $col \leftarrow parentCol + 1$;
            end if
        end for
    end if
    $plan[row][col] \leftarrow task$;
end for
```

FIGURE 9. Decoding algorithm [12].

C. EXTREME SOLUTIONS IN WORKFLOW SCHEDULING

Due to the requirement of E-NSGA-III, extreme solutions for the workflow scheduling problem have to be created in the first generation. As cost, makespan, and data movement are considered as the scheduling objectives, this work adopts the same strategy as proposed in [12] for creating the extreme solutions of these three objectives.

1) *Cost Solution*: The common solution of this case is all tasks are assigned to VM id '0' having the m1.small type which is the cheapest one.
2) *Makespan Solution*: The best solution of this case is scheduling all of the tasks to the m3.doubleXlarge type which is the highest processing capacity.
3) *Data Movement Solution*: The best case of this objective is executing all of the tasks in the same VM in order to avoid data movement.

Although in the application of E-NSGA-III with the MDNC technique, the data representation, the scheduling decoding, and the extreme solutions can be applied together with E-NSGA-III both with and without clustering technique. In case of using E-NSGA-III alone, the steps in Fig. 8(b) and Fig. 8(c) can be omitted. This work evaluates MDNC by comparing its performance against the two scenarios including E-NSGA-III without clustering technique and with Peer-to-Peer clustering. The comparison is based on five scientific workflows where their details are discussed in the following sections.

VII. SCIENTIFIC WORKFLOW CHARACTERISTICS

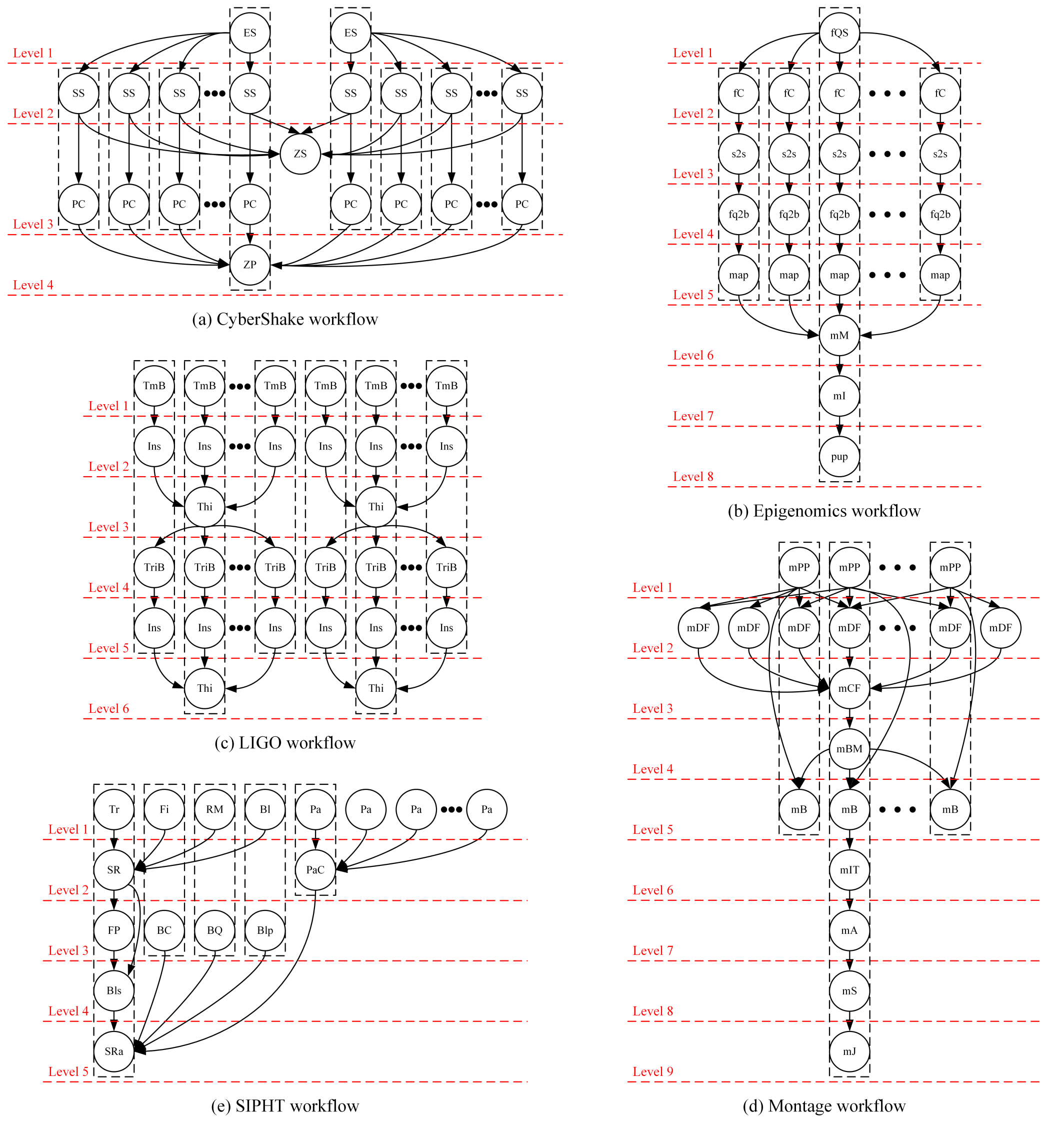
This study selects five well-known and often referred to scientific workflows for the performance evaluation. They are of CyberShake, Epigenomics, LIGO, Montage, and SIPHT. Their descriptions, examples, and portion of clustered nodes are discussed in the two following subsections.

A. DAG STRUCTURE OF FIVE SCIENTIFIC WORKFLOWS

This subsection presents a DAG structure of five scientific workflows. As a DAG model is used for typifying node dependencies, levels of all nodes in the workflow can be determined in conventional way where a starting node is in the first level and children of the starting node are in the second level and so on. In case of a child node having parents from multiple levels, its level is the next level to its highest parent level.

The examples of the multilevel DAG with MDNC of all workflows are illustrated in Fig. 10(a) to Fig. 10(e). In each Figure, a dashed line shows tasks that are clustered by the MDNC technique.

1) *CyberShake*: It is a seismology application that calculates probabilistic seismic hazard curves for geographic sites in the Southern California region [15],



(a) CyberShake workflow

(b) Epigenomics workflow

(c) LIGO workflow

(e) SIPHT workflow

(d) Montage workflow

FIGURE 10. DAG structure of MDNC of five scientific workflows.

[48]. CyberShake comprises five types of tasks in four different levels. Most of all tasks can be grouped by MDNC except for task ZS type as depicted in Fig. 10(a).

2) *Epigenomics*: It is developed by the USC Epigenome Center [49] and the Pegasus research team [50], and is used to automate various operations in genome sequence processing [15]. Eight types of tasks are managed in eight different levels and four of them are singleton-task type as depicted in Fig. 10(b). Note that all tasks in Epigenomics can be packed by MDNC.

3) *LIGO*: Laser Interferometer Gravitational Wave Observatory (LIGO) workflow is used to search for gravitational wave signatures in data collected by large-scale interferometers [15], [51]. Recently, its results are used for generating the first image of a black hole. LIGO comprises four types of tasks in six different levels as depicted in Fig. 10(c). All tasks in LIGO can also be packed by MDNC.

4) *Montage*: It is an astronomy application workflow created by NASA/IPAC. Montage is used to construct large image mosaics of the sky [15], [52]. It comprises nine types of tasks in nine different levels where six types are singleton-task type as depicted in Fig. 10(d). In this workflow, some tasks in the second level cannot be grouped by MDNC.

5) *SIPHT*: The sRNA Identification Protocol using High-throughput Technology (SIPHT) program is an automation workflow searching for sRNA encoding-genes in the National Center for Biotechnology Information (NCBI) database [15], [53]. It comprises 13 types of tasks in five levels where 12 types are

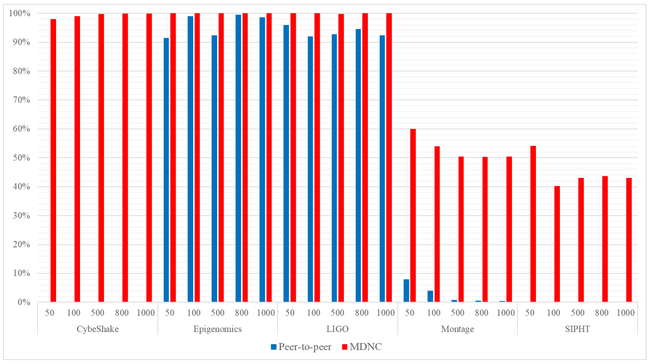


FIGURE 11. Percentage of clustered nodes in Peer-to-Peer and MDNC techniques.

singleton-task type as depicted in Fig. 10(e). In the first level of SIPHT, most of the tasks cannot be packed by MDNC.

B. PERCENTAGE OF CLUSTERED NODES

The complexity of workflow scheduling problem mainly depends on the number of tasks in a workflow. In order to improve the efficiency, grouping multiple tasks together benefits directly, particularly in this work to both reducing the search space of the problem and decreasing data movement. Hereafter, the proportion of nodes that can be grouped together is referred to as 'percentage of clustered nodes'.

Fig. 11 depicts the percentage of clustered nodes of both MDNC and Peer-to-Peer clustering techniques, red bars are the portion of MDNC, while blue ones are that of Peer-to-Peer clustering. Each workflow comprises five task sizes ranging from 50, 100, 500, 800 to 1,000. Note that CyberShake and SIPHT workflows cannot be clustered by Peer-to-Peer technique since their structures are not suitable for the predefined scenarios of Peer-to-Peer. In Epigenomics and LIGO, MNDC yields higher percentage of clustered nodes than Peer-to-Peer in all sizes. It also manages to gain a significantly higher percentile in Montage workflow. In case where Peer-to-Peer clustering cannot be applied such as CyberShake and SIPHT, MDNC manages to cluster 99.32% and 44.80% of all nodes (averages of five sizes) respectively.

VIII. PERFORMANCE EVALUATION

NSGA-III and E-NSGA-III are applied to original DAG, DAG with Peer-to-Peer clustering, and DAG with MDNC. For brevity, these three versions will be referred to as 'O-DAG', 'P2P-DAG', and 'MDNC-DAG' respectively. Five scientific workflows are used as testbeds where each one consists of five task sizes. The performance comparison is conducted in terms of *Hypervolume* [54], which is the most popularly used metric for multi-objective optimization problems [55].

TABLE 4. Parameter of NSGA-III and E-NSGA-III

Parameter	Value
DAG versions	O-DAG, P2P-DAG, and MDNC-DAG
Scheduling objectives	Cost, Makespan, and Data Movement
No. of tasks	50, 100, 500, 800, and 1,000
Reference points	91
Population size	92
Crossover operation	Simulated Binary Crossover (SBX) [58]
	- the distribution index = 30
	- the probability = 1.0
Mutation operation	Polynomial Mutation [59]
	- the distribution index = 30
	- the probability = 1/the length of chromosome
Stopping criteria	300 generations

A. EXPERIMENT SETUP

In this work, E-NSGA-III is implemented by extending NSGA-III of jMetal tool [56] in Java version and the Hypervolume indicator is also provided in jMetal.The parameters of NSGA-III and E-NSGA-III are summarized in Table 4. All parametric values used in this study followed those proposed in [14], [57]. An experiment consists of 30 runs, therefore, the average Hypervolume is presented as the result.

B. HYPERVOLUME

Hypervolume represents the volume of the space between a set of Pareto solutions and the inverted values of the reference solutions. The more distance of obtained solutions away from the inverted reference points, the larger Hypervolume is gained. Usually, optimal solutions of a problem are used as the reference solutions and their inverted values are used for computing the Hypervolume. However, in this work, optimal solutions are not existed, therefore nondominated solutions of all obtained solutions from all cases in the experiment are archived as the reference solutions. Their values in each objective are normalized ranging from '0' to '1' and then the inverted values of their normalized values are used for determining the Hypervolume.

Table 5 reveals the Hypervolume of O-DAG, P2P-DAG, and MDNC-DAG of NSGA-III and E-NSGA-III. In this work, the larger Hypervolume is preferred. Referring to Table 5, the values in the bold font indicate the highest Hypervolume in each size of each workflow, while those in the italic font are the lowest value. The value '0' typifies that there is no obtained solution that dominates the inverted reference solutions in all 30 runs.

C. COMPARISON AMONG CLUSTERING TECHNIQUES FOR EACH OBJECTIVE

With respect to the obtained solutions, there are many possible presentations for each scientific workflow. In this work, obtained solutions by E-NSGA-III are used for further investigation. For ease of comparison, solutions of workflows having 1,000 tasks are selected for presentation as shown in Fig. 12(a) to Fig. 12(e) due to their difficulty in generating Pareto solutions. The objective values of obtained solutions are normalized by comparing with the normalized values of

TABLE 5. Hypervolume of original DAG, DAG with Peer-to-Peer, and DAG with MDNC

Workflow	Size	NSGA-III			E-NSGA-III		
		O-DAG	P2P-DAG	MDNC-DAG	O-DAG	P2P-DAG	MDNC-DAG
CyberShake	50	*0*	N/A	*0*	0.070017	N/A	**0.103981**
	100	*0*	N/A	*0*	0.202680	N/A	**0.248492**
	500	*0*	N/A	*0*	**0.128773**	N/A	0.128485
	800	*0.006053*	N/A	0.034048	0.035347	N/A	**0.163653**
	1000	*0.008323*	N/A	0.228983	0.195973	N/A	**0.353505**
	Average	*0.002875*	N/A	0.052606	0.126558	N/A	**0.199623**
Epigenomics	50	*0.026185*	0.601715	0.718088	0.473422	0.563584	**0.737065**
	100	*0*	0.138697	0.148843	0.080501	0.209013	**0.219450**
	500	*0*	0.030454	0.117842	0.205742	0.274666	**0.309234**
	800	*0*	0.011160	0.017873	0.094941	0.222044	**0.296379**
	1000	*0*	0.008947	0.019562	0.029118	0.071306	**0.129356**
	Average	*0.005237*	0.158195	0.204441	0.176745	0.268123	**0.338297**
LIGO	50	*0*	0.053565	0.281022	0.161872	0.275378	**0.283535**
	100	*0*	0.026622	0.179280	0.152327	0.289468	**0.297893**
	500	*0*	0.018429	0.090839	0.040676	0.156273	**0.157147**
	800	*0*	0.007852	0.051300	0.187718	0.243477	**0.255688**
	1000	*0*	0.007344	0.051949	0.176596	0.268672	**0.297000**
	Average	*0*	0.022762	0.130878	0.143838	0.246653	**0.258252**
Montage	50	*0*	*0*	*0*	**0.409813**	0.339605	0.279629
	100	*0*	*0*	*0*	0.208704	**0.272838**	0.245101
	500	*0*	*0*	*0*	**0.100944**	0.084471	0.066825
	800	*0*	*0*	*0*	**0.115668**	0.016646	0.082284
	1000	*0*	*0*	*0*	0.161150	**0.201177**	0.172615
	Average	*0*	*0*	*0*	**0.199256**	0.182947	0.169291
SIPHT	50	*0*	N/A	0.017911	**0.229907**	N/A	0.143784
	100	*0.005567*	N/A	0.080004	**0.354595**	N/A	0.168261
	500	*0.015990*	N/A	0.040911	**0.195327**	N/A	0.161846
	800	*0.000263*	N/A	0.015683	0.106421	N/A	**0.182691**
	1000	*0.011098*	N/A	0.024461	**0.169301**	N/A	0.143246
	Average	*0.006584*	N/A	0.035794	**0.211110**	N/A	0.159966

*N/A = Not available

the reference solutions used for calculating Hypervolume. Therefore, it is possible to have a value higher than 1 in case a solution has value higher than the maximum value of the reference solutions, but less than 0 is not possible as this work is concerned with minimization. The reference solutions are a collection of solutions that has the lowest value. An outlier, a solution having value either lower than the lower quartile of box plot or higher than the upper quartile, is marked in the red plus sign.

IX. DISCUSSIONS

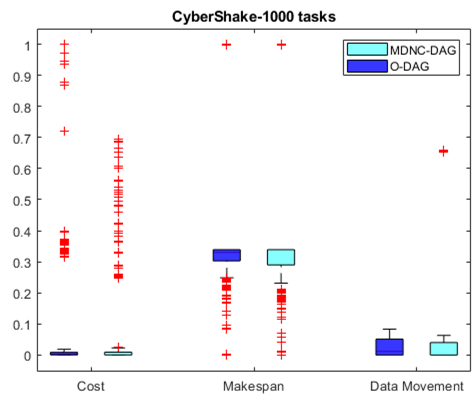
In overall, referring to Table 5, solutions of E-NSGA-III yield better Hypervolume than those of NSGA-III. A set of solutions generated by NSGA-III cannot reach the highest Hypervolume in all 25 cases. Also, most of the lowest Hypervolume solutions appear in NSGA-III. Especially, in Montage workflow, none of those solutions can dominate the inverted reference points. Therefore, it can be concluded that E-NSGA-III outperforms NSGA-III in five scientific workflows. For the ease explanation, the rest of the discussion focuses on the Hypervolume of E-NSGA-III in all three versions of DAG.

Concerning the E-NSGA-III Hypervolume in Table 5, the results can be concluded in two perspectives. MDNC-DAG can help E-NSGA-III generate solutions with higher Hypervolume than other DAG versions in most cases of CyberShake, Epigenomics, and LIGO workflows (14 of 15 cases). On the other hand, in Montage and SIPHT workflows, the solutions of DAG without clustering yield better Hypervolume than those of P2P-DAG and MDNC-DAG in most cases (7 of 10 cases).

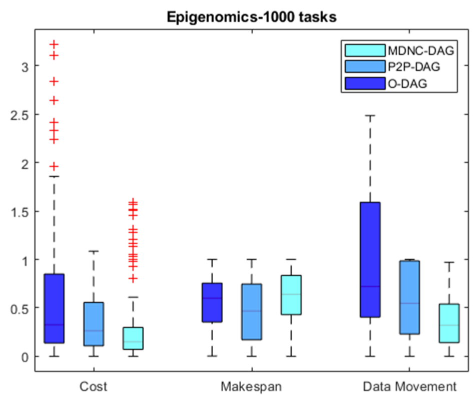
As shown in the results, MDNC-DAG reveals both advantage and limitation for clustering DAG workflows. In CyberShake, Epigenomics, and LIGO where MDNC outperformed in general, these workflows have higher clustered node portion having more than 90%. By contrast, in Montage and SIPHT having the percentage of clustered nodes lower than 60%, MDNC yields poor performance. In general, the higher the percentage of clustered nodes, the better it is for applying E-NSGA-III with the MDNC technique. Therefore, workflow with higher percentage of clustered nodes is preferable.

Furthermore, with respect to DAG structures of five scientific workflows, CyberShake, Epigenomics, and LIGO share the same characteristic that they have more than one levels having a high number of nodes. This characteristic shows a balanced number of nodes in different levels, where nodes can be easily grouped together. In general, MDNC may take advantage of this type of structure as more clusters can be generated.

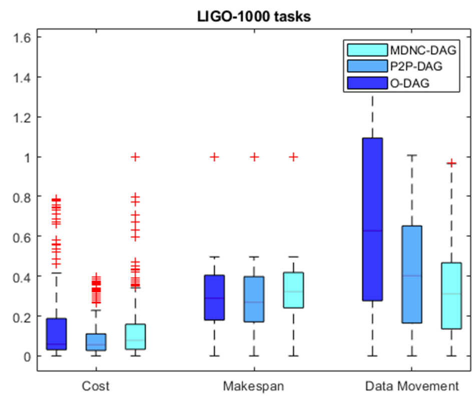
In Montage and SIPHT workflows, on the other hand, they have only one level having high numbers of nodes and that level dominates other levels due to its nature of having more nodes than the rest. The single dominated level obstructs MDNC in grouping nodes in different levels because many



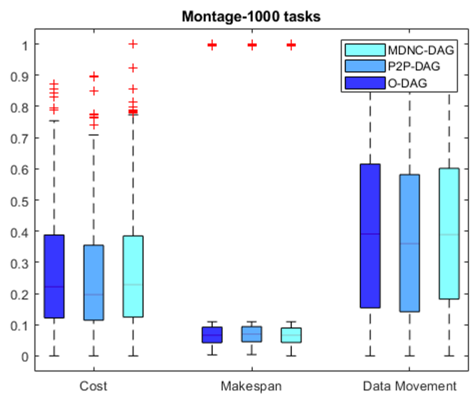
(a) CyberShake with 1,000 tasks



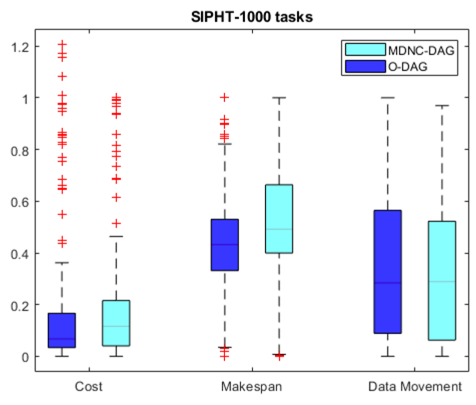
(b) Epigenomics with 1,000 tasks



(c) LIGO with 1,000 tasks



(d) Montage with 1,000 tasks



(e) SIPHT with 1,000 tasks

FIGURE 12. Obtained solutions of E-NSGA-III in five workflows.

nodes cannot be packed with other nodes in different levels. Consequently, the single dominated level leads to a lower percentage of the clusters in general.

Referring to Fig. 12(a) to Fig. 12(e), normalized values of cost and data movement reveal confliction with makespan especially solutions of CyberShake, Montage, and SIPHT workflows. Hence, striking a good balance in minimizing cost, makespan, and data movement in these three workflows are relatively difficult. Bearing in mind that reducing data movement leads to the reuse of a VM for processing tasks and consequently helps to decrease the cost of VMs.

Therefore, in general, it can be concluded that a workflow having a balanced number of nodes in different levels is suitable for the MDNC technique in creating high percentage of clustered nodes. Nevertheless, in case of using E-NSGA-III, a workflow having the single dominated level (e.g. Montage and SIPHT) may not be suitable for the application of the MNDC technique. E-NSGA-III in itself ought to be sufficient and an additional application of MNDC may not be necessary.

X. CONTRIBUTIONS AND CONCLUSIONS

In conclusion, running scientific workflows on cloud usually generates tremendous data and exchange that data among tasks during the execution. Moving the numerous data within cloud data center can lead to an increase in the usage of network bandwidth and energy consumption consumed by the network equipment. From cloud provider perspective, the reduction of data movement is becoming more critical. Meanwhile, cloud service providers have to maintain the quality of service where cost and makespan are common objectives for the cloud user perspective. This work probably is the earlier work in multi-objective scheduling problem that considers optimization from both cloud user and cloud provider perspectives.

The contributions of this work can be divided into two main perspectives, theoretical and application. From the first perspective, this work proposes MDNC, the clustering technique for reducing data movement, for a DAG workflow. MDNC can help to alleviate the unnecessary movement of data at the application level, without interrupting the parallelism of a workflow. This contribution, together with E-NSGA-III, the recent evolutionary algorithm, is able to provide solutions for multi-objective workflow scheduling based on the DAG model. Furthermore, this work considers that cloud system model can provide unlimited resources (e.g. VM) where a scheduler can freely determine the number and types of VM for workflow execution. This model can corporate with the cloud user perspective where cloud users can scale computing resources up or down anytime without the consideration of hardware shortage.

From the application perspective, this work demonstrates MDNC with NSGA-III and E-NSGA-III for dealing with the multi-objective workflow scheduling where Cost, Makespan, and Data Movement are the objectives. It also presents the working steps of applying MDNC for the DAG workflow and those of using E-NSGA-III in task scheduling. In the genetic algorithm, the data representation is designed to reflect the number of MDNC clusters and then the scheduling plan can be decoded from the obtained solutions. The work then has illustrated and evaluated the applications of NSGA-III and E-NSGA-III with MDNC on five well-known scientific workflows in the DAG model.

To alleviate data movement, the conventional DAG model is improved by implementing the MDNC technique for grouping dependent tasks together. MDNC also enables E-NSGA-III to enhance the quality of solutions by offering a scheduling plan with minimizing all three objectives. The reduction of data movement also helps to save the cost where both cloud service provider and user can benefit. With respect to the results in this work, E-NSGA-III with MDNC demonstrates its advantage in DAG workflows with the higher clustered node portion. The work also identifies the limitation in a situation of imbalance level.

Further studies can be extended in several facets. Similar attempts can be done in green computing objective which is the recent concern in the field of cloud computing. This suggests inclusion of energy consumption, energy renewable sources, and CO_2 emissions in the multi-objective problems. From E-NSGA-III perspective, it can also be applied to other areas of multi-optimization such as multi-objective portfolio in finance field and multi-objective optimal design in engineering and manufacturing fields. As the evolutionary algorithm in this work based on E-NSGA-III where an improvement by noting the behavior of NSGA-II and NSGA-III, E-NSGA-III merits further investigation, especially on the use of 'crowding distance' in NSGA-II and 'reference points' in NSGA-III. This may lead to an even more efficient multi-objective based on evolutionary algorithm than E-NSGA-III. Finally, the clustering technique can be investigated to deal with a DAG workflow having an imbalance structure where MDNC is still showing a limitation.

REFERENCES

[1] M. A. Rodriguez and R. Buyya, "A taxonomy and survey on scheduling algorithms for scientific workflows in iaas cloud computing environments," Concurrency and Computation: Practice and Experience, vol. 29, no. 8, p. e4041, 2017, e4041 cpe.4041. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4041

[2] M. Atkinson, S. Gesing, J. Montagnat, and I. Taylor, "Scientific workflows: Past, present and future," Future Generation Computer Systems, vol. 75, pp. 216 – 227, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17311202

[3] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. F. da Silva, M. Livny, and K. Wenger, "Pegasus, a workflow management system for science automation," Future Generation Computer Systems, vol. 46, pp. 17 – 35, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X14002015

[4] M. Guzek, P. Bouvry, and E. Talbi, "A survey of evolutionary computation for resource management of processing in cloud computing," IEEE Computational Intelligence Magazine, vol. 10, no. 2, pp. 53–67, May 2015.

[5] Q. Liao and Z. Wang, "Energy consumption optimization scheme of cloud data center based on sdn," Procedia Computer Science, vol. 131, pp. 1318 – 1327, 2018, recent Advancement in Information and Communication Technology:. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050918307075

[6] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," SIGARCH Comput. Archit. News, vol. 38, no. 3, pp. 338–347, Jun. 2010. [Online]. Available: http://doi.acm.org/10.1145/1816038.1816004

[7] T. A. Nguyen, D. Min, E. Choi, and T. D. Tran, "Reliability and availability evaluation for cloud data center networks using hierarchical models," IEEE Access, vol. 7, pp. 9273–9313, 2019.

[8] P. Wangsom, K. Lavangnananda, and P. Bouvry, "Multi-objective scheduling for scientific workflows on cloud with peer-to-peer clustering," in 2019 11th International Conference on Knowledge and Smart Technology (KST), Jan 2019, pp. 175–180. [Online]. Available: https://ieeexplore.ieee.org/document/8687412/

[9] P. Wangsom, K. Lavangnananda, and P. Bouvry, "Measuring data locality ratio in virtual MapReduce cluster using WorkflowSim," in Proceedings of the 2017 14th International Joint Conference on Computer Science and Software Engineering, JCSSE 2017, 2017.

[10] M. R. Garey and D. S. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman Co., 1990.

[11] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz, Handbook on Scheduling: From Theory to Applications. Springer Publishing Company, Incorporated, 2014.

[12] K. Lavangnananda, P. Wangsom, and P. Bouvry, "Extreme solutions nsga-iii (e-nsga-iii) for scientific workflow scheduling on cloud," in 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Dec 2018, pp. 1139–1146. [Online]. Available: https://ieeexplore.ieee.org/document/8614209/

[13] 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, April 2002.

[14] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," IEEE Transactions on Evolutionary Computation, vol. 18, no. 4, pp. 577–601, Aug 2014.

[15] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," Future Generation Computer Systems, vol. 29, no. 3, pp. 682 – 692, 2013, special Section: Recent Developments in High Performance Computing and Security. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X12001732

[16] A. Gerasoulis and T. Yang, "On the granularity and clustering of directed acyclic task graphs," IEEE Transactions on Parallel and Distributed Systems, vol. 4, no. 6, pp. 686–701, June 1993.

[17] Tao Yang and A. Gerasoulis, "Dsc: Scheduling parallel tasks on an unbounded number of processors," IEEE Transactions on Parallel and Distributed Systems, vol. 5, no. 9, pp. 951–967, Sep. 1994.

[18] W. Chen and E. Deelman, "Workflowsim: A toolkit for simulating scientific workflows in distributed environments," in 2012 IEEE 8th International Conference on E-Science, Oct 2012, pp. 1–8.

[19] M. A. Rodriguez and R. Buyya, "Budget-driven scheduling of scientific workflows in iaas clouds with fine-grained billing periods," ACM Trans. Auton. Adapt. Syst., vol. 12, no. 2, pp. 5:1–5:22, May 2017. [Online]. Available: http://doi.acm.org/10.1145/3041036

[20] D. Kliazovich, J. E. Pecero, A. Tchernykh, P. Bouvry, S. U. Khan, and A. Y. Zomaya, "Ca-dag: Modeling communication-aware applications for scheduling in cloud computing," Journal of Grid Computing, vol. 14, no. 1, pp. 23–39, Mar 2016. [Online]. Available: https://doi.org/10.1007/s10723-015-9337-8

[21] S. Abrishami, M. Naghibzadeh, and D. H. Epema, "Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds," Future Generation Computer Systems, vol. 29, no. 1, pp. 158 – 169, 2013, including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X12001008

[22] B. P. Rimal and M. Maier, "Workflow scheduling in multi-tenant cloud computing environments," IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 1, pp. 290–304, Jan 2017.

[23] B. Lin, W. Guo, N. Xiong, G. Chen, A. V. Vasilakos, and H. Zhang, "A pretreatment workflow scheduling approach for big data applications in multicloud environments," IEEE Transactions on Network and Service Management, vol. 13, no. 3, pp. 581–594, Sep. 2016.

[24] N. Anwar and H. Deng, "Elastic scheduling of scientific workflows under deadline constraints in cloud computing environments," Future Internet, vol. 10, no. 1, 2018. [Online]. Available: https://www.mdpi.com/1999-5903/10/1/5

[25] V. Singh, I. Gupta, and P. K. Jana, "A novel cost-efficient approach for deadline-constrained workflow scheduling by dynamic provisioning of resources," Future Generation Computer Systems, vol. 79, pp. 95 – 110, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17307264

[26] V. Arabnejad, K. Bubendorfer, and B. Ng, "Budget and deadline aware e-science workflow scheduling in clouds," IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 1, pp. 29–44, Jan 2019.

[27] C. Szabo and T. Kroeger, "Evolving multi-objective strategies for task allocation of scientific workflows on public clouds," in 2012 IEEE Congress on Evolutionary Computation, June 2012, pp. 1–8.

[28] M. A. Rodriguez and R. Buyya, "Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds," IEEE Transactions on Cloud Computing, vol. 2, no. 2, pp. 222–235, April 2014.

[29] 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, May 2016.

[30] J. Meena, M. Kumar, and M. Vardhan, "Cost effective genetic algorithm for workflow scheduling in cloud under deadline constraint," IEEE Access, vol. 4, pp. 5065–5082, Aug 2016.

[31] H. Y. Shishido, J. C. Estrella, C. F. M. Toledo, and M. S. Arantes, "Genetic-based algorithms applied to a workflow scheduling algorithm with security and deadline constraints in clouds," Computers & Electrical Engineering, vol. 69, pp. 378 – 394, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045790617312259

[32] B. Xiang, B. Zhang, and L. Zhang, "Greedy-ant: Ant colony system-inspired workflow scheduling for heterogeneous computing," IEEE Access, vol. 5, pp. 11 404–11 412, 2017.

[33] N. Anwar and H. Deng, "A hybrid metaheuristic for multi-objective scientific workflow scheduling in a cloud environment," Applied Sciences, vol. 8, no. 4, 2018. [Online]. Available: https://www.mdpi.com/2076-3417/8/4/538

[34] W. Li, Y. Xia, M. Zhou, X. Sun, and Q. Zhu, "Fluctuation-aware and predictive workflow scheduling in cost-effective infrastructure-as-a-service clouds," IEEE Access, vol. 6, pp. 61 488–61 502, 2018.

[35] M.-Y. Cheng and D. Prayogo, "Symbiotic organisms search: A new metaheuristic optimization algorithm," Computers & Structures, vol. 139, pp. 98 – 112, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045794914000881

[36] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 3, pp. 682–694, March 2014.

[37] W. Raaymakers and A. Weijters, "Makespan estimation in batch process industries: A comparison between regression analysis and neural networks," European Journal of Operational Research, vol. 145, no. 1, pp. 14 – 30, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S037722170200173X

[38] I. Chaouch, O. B. Driss, and K. Ghedira, "Elitist ant system for the distributed job shop scheduling problem," in Advances in Artificial Intelligence: From Theory to Practice, S. Benferhat, K. Tabia, and M. Ali, Eds. Cham: Springer International Publishing, 2017, pp. 112–117.

[39] V. Nagarajan and M. Sviridenko, "Tight bounds for permutation flow shop scheduling," in Integer Programming and Combinatorial Optimization, A. Lodi, A. Panconesi, and G. Rinaldi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 154–168.

[40] B. Dorronsoro, G. Danoy, P. Bouvry, and A. J. Nebro, Multi-objective Cooperative Coevolutionary Evolutionary Algorithms for Continuous and Combinatorial Optimization. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 49–74.

[41] T. T. Tran and J. C. Beck, "Logic-based benders decomposition for alternative resource scheduling with sequence dependent setups," in Proceedings of the 20th European Conference on Artificial Intelligence, ser. ECAI'12. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2012, pp. 774–779.

[42] P. Florin, A Fault Tolerant Decentralized Scheduling in Large Scale Distributed Systems. Hershey, PA, USA: IGI Global, 2010, pp. 566–588.

[43] A. B. Kahn, "Topological sorting of large networks," Commun. ACM, vol. 5, no. 11, pp. 558–562, Nov. 1962. [Online]. Available: http://doi.acm.org/10.1145/368996.369025

[44] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and Stein Clifford, Introduction to algorithms. MIT Press, 2009.

[45] I. Das and J. Dennis, "Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems," SIAM Journal on Optimization, vol. 8, no. 3, pp. 631–657, 1998. [Online]. Available: https://doi.org/10.1137/S1052623496307510

[46] E.-G. Talbi, Metaheuristics: From Design to Implementation. Wiley Publishing, 2009.

[47] P. Wangsom, K. Lavangnananda, and P. Bouvry, "The application of nondominated sorting genetic algorithm (nsga-iii) for scientific-workflow scheduling on cloud," in The 8th Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2017), Dec 2017, Conference Proceedings, pp. 269–287.

[48] P. Maechling, E. Deelman, L. Zhao, R. Graves, G. Mehta, N. Gupta, J. Mehringer, C. Kesselman, S. Callaghan, D. Okaya, H. Francoeur, V. Gupta, Y. Cui, K. Vahi, T. H. Jordan, and E. H. Field, "Scec cybershake workflows - automating probabilistic seismic hazard analysis calculations," in Workflows for e-Science, Scientific Workflows for Grids, 2007.

[49] P. W. Laird, "The USC epigenome center," Epigenomics, vol. 1, no. 1, pp. 29–31, oct 2009. [Online]. Available: https://www.futuremedicine.com/doi/10.2217/epi.09.12

[50] "Application showcase: Epigenomics," https://pegasus.isi.edu/application-showcase/epigenomics/, accessed: 2019-06-19.
[51] B. P. Abbott et al., "LIGO: the laser interferometer gravitational-wave observatory," Reports on Progress in Physics, vol. 72, no. 7, p. 076901, jun 2009.
[52] G. B. Berriman, E. Deelman, J. C. Good, J. C. Jacob, D. S. Katz, C. Kesselman, A. C. Laity, T. A. Prince, G. Singh, and M.-H. Su, "Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand," in SPIE Astronomical Telescopes + Instrumentation, vol. 5493. SPIE, 2004.
[53] J. Livny, H. Teonadi, M. Livny, and M. K. Waldor, "High-throughput, kingdom-wide prediction and annotation of bacterial non-coding rnas," PloS one, vol. 3, no. 9, pp. e3197–e3197, 2008. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/18787707 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2527527/
[54] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," IEEE Transactions on Evolutionary Computation, vol. 3, no. 4, pp. 257–271, Nov 1999.
[55] N. Riquelme, C. Von Lücken, and B. Baran, "Performance metrics in multi-objective optimization," in 2015 Latin American Computing Conference (CLEI), Oct 2015, pp. 1–11.
[56] A. J. Nebro, J. J. Durillo, and M. Vergne, "Redesigning the jmetal multi-objective optimization framework," in Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, ser. GECCO Companion '15. New York, NY, USA: ACM, 2015, pp. 1093–1100. [Online]. Available: http://doi.acm.org/10.1145/2739482.2768462
[57] H. Ishibuchi, R. Imada, Y. Setoguchi, and Y. Nojima, "Performance comparison of nsga-ii and nsga-iii on various many-objective test problems," in 2016 IEEE Congress on Evolutionary Computation (CEC), July 2016, pp. 3045–3052.
[58] R. B. Agrawal, K. Deb, and R. B. Agrawal, "Simulated binary crossover for continuous search space," Complex systems, vol. 9, no. 2, pp. 115–148, 1995.
[59] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms. New York, NY, USA: John Wiley & Sons, Inc., 2001.

PEERASAK WANGSOM received the B.Sc. degree from Chiang Mai University, Chiang Mai, Thailand, in 1999, and the M.Sc. degree in Computer Science from Thammasat University, Bangkok, Thailand, in 2009. He is now pursuing a Ph.D. in Computer Science at School of Information Technology (SIT), King Mongkut 's University of Technology Thonburi (KMUTT). His research involves multi-objective optimization, evolutionary algorithms, and scheduling of distributed systems.

KITTICHAI LAVANGNANANDA received his B.Sc. in Computational Science from Hull University, in 1985 and M.Sc. in Computing from Cardiff University in 1987, U.K. He completed his Ph.D. studies in Artificial Intelligence at Mechanical Engineering Centre (MEC) at Cardiff University, U.K. in 1995. At present, he is an Associate Professor and Head of Software Division Technology at the School of Information Technology (SIT), King Mongkut 's University of Technology Thonburi (KMUTT), Thailand.

He is an active research member of the Data and Knowledge Engineering Laboratory (D-Lab) at SIT. His research interest is in the Computational Intelligence related areas (Data Mining, Evolutionary Computation, Machine Learning, Neural Networks) and their applications. He is also involved in academic activities under various capacities. He is a Senior Member of IEEE Association and a member of the Editorial Board for Cogent Engineering Journal.

PASCAL BOUVRY earned his undergraduate degree in Economical & Social Sciences and his Master degree in Computer Science with distinction ('91) from the University of Namur, Belgium. He went on to obtain his Ph.D. degree ('94) in Computer Science with great distinction at the University of Grenoble (INPG), France.

He is a full Professor at the University of Luxembourg, "Chargé de Mission auprés du Recteur" in charge of the University High Performance Computing service, and heading the PCOG (Parallel Computing and Optimization Group) at the SnT. He is active in various scientific committees and technical workgroups (IEEE CIS Cloud Computing vice-chair, IEEE TCSC GreenIT steering committee, ERCIM WG, ANR, COST TIST, etc.).

He is the Luxembourg national delegate for PRACE (Partnership for Advanced Computing in Europe). The mission of PRACE is to enable high impact scientific discovery and engineering research and development across all disciplines to enhance European competitiveness for the benefit of society.

He is part of the editorial boards of IEEE Transactions on Sustainable Computing, Springer journal on Communications and Sustainable Computing, IET Cyber- Physical Systems: Theory & Applications, and Elsevier journal in Swarm and Evolutionary Computation.