



Cloud customers service selection scheme based on improved conventional cat swarm optimization

Danlami Gabi^{1,2} · Abdul Samad Ismail³ · Anazida Zainal³ · Zalmiyah Zakaria³ · Ajith Abraham⁴ · Nasiru Muhammed Dankolo³

Received: 15 February 2019 / Accepted: 5 March 2020 / Published online: 17 March 2020
© The Author(s) 2020

Abstract

With growing demand on resources situated at the cloud datacenters, the need for customers' resource selection techniques becomes paramount in dealing with the concerns of resource inefficiency. Techniques such as metaheuristics are promising than the heuristics, most especially when handling large scheduling request. However, addressing certain limitations attributed to the metaheuristic such as slow convergence speed and imbalance between its local and global search could enable it become even more promising for customers service selection. In this work, we propose a cloud customers service selection scheme called Dynamic Multi-Objective Orthogonal Taguchi-Cat (DMOOTC). In the proposed scheme, avoidance of local entrapment is achieved by not only increasing its convergence speed, but balancing between its local and global search through the incorporation of Taguchi orthogonal approach. To enable the scheme to meet customers' expectations, Pareto dominant strategy is incorporated providing better options for customers in selecting their service preferences. The implementation of our proposed scheme with that of the benchmarked schemes is carried out on CloudSim simulator tool. With two scheduling scenarios under consideration, simulation results show for the first scenario, our proposed DMOOTC scheme provides better service choices with minimum total execution time and cost (with up to 42.87%, 35.47%, 25.49% and 38.62%, 35.32%, 25.56% reduction) and achieves 21.64%, 18.97% and 13.19% improvement for the second scenario in terms of execution time compared to that of the benchmarked schemes. Similarly, statistical results based on 95% confidence interval for the whole scheduling scheme also show that our proposed scheme can be much more reliable than the benchmarked scheme. This is an indication that the proposed DMOOTC can meet customers' expectations while providing guaranteed performance of the whole cloud computing environment.

Keywords Cloud computing · Scheduling · Cat swarm optimization · Pareto dominance

✉ Danlami Gabi
danlami@cs.umu.se

Abdul Samad Ismail
abdsamad@utm.my

Anazida Zainal
anazida@utm.my

Zalmiyah Zakaria
zalmiyah@utm.my

Ajith Abraham
ajith.abraham@ieee.org

Nasiru Muhammed Dankolo
ndmuhammad@live.utm.my

¹ Department of Computer Science, Faculty of Science, Kebbi State University of Science and Technology, Aliero, Kebbi State, Nigeria

² Department of Computing Science, Umea University, Umeå, Sweden

³ Department of Computer Science, School of Computing, Universiti Teknologi Malaysia, Johor, Malaysia

⁴ Machine Intelligence Research Labs, Scientific Network for Innovation and Research Excellence, Auburn, WA 98071, USA

1 Introduction

Cloud computing is a platform of choice providing distributed resources (e.g., virtual machine, storage and bandwidth) to meet increasing demand of customers. The environment provides cost-effective solution for running business applications through virtualized technologies [1–4]. Services that are made available by cloud environment are affordable using the concept of pay-per-use (PPU) pricing models. Three types of service model are associated with the cloud computing environment: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [5]. The cloud customers usually interact remotely with the SaaS layer to run their applications (e.g., datacenter) [5]. The SaaS layer thus functions with the support of PaaS layer. The PaaS layer provides interactive mechanisms between cloud customers and service providers. The PaaS thus allows cost-efficient development and deployment of applications [7, 8]. With the provision of numerous advantages (e.g., deployment of PPU pricing model, maintaining application having the same integrated environment) by the PaaS layer, the cloud customers can now have the means of using same integrated software application development environment [9, 10]. On the other hand, the IaaS layer is responsible for providing services to the cloud customers in terms of the infrastructures. The cloud customers have no control over managing the infrastructures; rather, they only utilize the available resources situated at the IaaS layer through service request [6]. In another development, the IaaS provides pool of resources of varied types which are leased according to customers' request. This paper provides customers with techniques for optimal mapping of their choice of resources at the IaaS layer to meet their computation need [7, 11, 12].

At the IaaS layer, virtual machines are heterogeneous. Some requests are said to have a very high demand for virtual machines, while others require more storage [13, 14]. In most instances, the customers demand resources that have minimum costs of execution. Virtual machines with high processing speed usually incur high processing cost [15–18]. In Amazon EC2 for instance, a cloud customer has the privilege of accessing and controlling set of virtual machines that run inside the data-center of the service provider, while being charged for a specific time the virtual machine has been allocated. At this instance, the customer satisfaction is measured based on the Quality of Service (QoS) (e.g., execution time, cost and storage) he/or she experiences [19, 20]. To provide customers of cloud computing resources with better options of selecting their resource preferences, there is need for an optimal resource scheme [15, 16, 21, 22]. In recent

development, high-level research conducted in cloud computing task and resource scheduling focuses on time and cost models. These models capture customers' QoS experience, where techniques of metaheuristic are exploited in their evaluations. Although the metaheuristic techniques are attributed with certain limitations (e.g., local and global imbalance, local optimality) [14, 23], they have been proven to be more efficient in reducing the complexity of a task scheduling problem when finding an optimum or near-optimal solution [14, 24, 25]. Several researchers [26–31] also focus on investigating relationship between local optimality and slow convergence and preferred solutions to addressing these concerns.

The conventional cat swarm optimization (CSO) is a metaheuristic optimization technique put forward by Chu and Tsai in 2007. This technique mimics the behavior of natural cat and has relatively proven better in terms of convergence speed than the particle swarm optimization (PSO) [32]. It has both the global and local search (also known as the seeking and tracing mode) and a control variable called the mixed ratio (MR) that determine whether the current position at which the cat is standing is in either seeking or tracing mode. As attributed to most metaheuristic techniques, the CSO suffers slow convergence speed that can lead to its entrapment at the local optima, and global and local imbalance causing instability of the solution. Orthogonal Taguchi approach is a greedy-based technique that when applied can increase the convergence speed of the CSO to avoid being trapped at the local optima [13, 20]. In this paper, we proposed Dynamic Multi-Objective Orthogonal Taguchi-Cat (DMOOTC) scheme. In the proposed scheme, we exploit advantages of the orthogonal Taguchi approach to avoid the DMOOTC from being entrapped at the local optima. To provide customers with choices of service preferences, we incorporated the Pareto dominance strategy in the proposed DMOOTC scheme. The simulation results show that our proposed DMOOTC scheme can provide customers with better service choices compared to the benchmarked schemes.

The contribution of this paper is as follows:

- Development of a multi-objective task scheduling model for cost and computation time.
- An improvement is proposed to the CSO algorithm using the orthogonal Taguchi approach and Pareto dominance.
- Development of a DMOOTC scheme to solve the multi-objective task scheduling model.

The rest of this article is organized as follows: Review on related work is discussed in Sect. 2. Section 3 discusses the CSO technique. Orthogonal Taguchi approach is discussed in Sect. 4. Section 5 provides discussion on Pareto

dominance strategy. The proposed models are discussed in Sect. 6. Section 7 discusses the proposed DMOOTC scheme. Discussion on simulation results is provided in Sect. 8. Section 9 discusses the performance metrics used in the evaluation of the schemes. Discussion on the simulation results is provided in Sect. 10. Section 11 provides discussion on the statistical results, and finally, Sect. 12 concludes the paper.

2 Related works

Task scheduling is one of the most research areas in cloud computing. Researchers have shown that scheduling strategies such as heuristic and metaheuristic can be promising when applied to deal with scheduling problems in cloud datacenters. Metaheuristics such as particle swarm optimization (PSO), genetic algorithm (GA), ant colony optimization (ACO) and cat swarm optimization (CSO) are few examples of metaheuristic techniques that can handle large scheduling problems. Their improvement with trajectory-based (e.g., simulated annealing (SA)) and greedy-based techniques such as the orthogonal Taguchi approaches can further improve their performances toward providing more efficient solutions [33–35]. Few among researchers that exploit these advantages are discussed in the following:

In Wei et al. [36], a Compounded Local Mobile Cloud Architecture (LMCpri) with dynamic priority queue is proposed to solve a multi-objective scheduling problem. In their proposed approach, a priority-based positioning technique based on auction processing is incorporated to store jobs upon arrival from the cloud customers. Then, a Non-Static Genetic Algorithm II (NSGA-II) is introduced for scheduling tasks on resources to achieve minimum processing time and decrease the request cost. According to the researchers, simulation results show their proposed algorithm can provide better performance than PSO and sequential scheduling algorithms in terms of minimum total execution time and cost. However, improvement in their proposed method is still possible since mutation process exhibited by the GA can lead to local trapping due to slow convergence speed. In another development, Liu et al. [37] proposed a Single Site Virtual Machine Provisioning (SSVP) approach and ActGreedy to minimize task execution time and monetary cost. In their proposed approach, a single-site initialization module is used to ensure virtual machine provisioning and multisite data transfer. At the instance of task execution, a virtual machine is not allowed to restart for the execution of any continuous activities on the site due to the fact that they are grouped and scheduled as a fragment. According to the researchers, simulation results show their developed SSVP

can generate better VM-provisioning plan for customers to achieve minimum task execution time and monetary cost compared to the benchmarked techniques.

In Zuo et al. [14], a Multi-Objective Ant Colony Optimization (MOACO) algorithm is proposed. Their objective is to minimize the makespan time and budgetary cost. A cost model that reflects the relationship between customers' resource and budgetary cost is introduced in evaluating the efficiency of their proposed algorithms. According to the researchers, simulation results show their proposed algorithm has achieved minimum execution cost compared to that of the benchmarked algorithms. However, the updating process of pheromone exhibited by the ants along the path can lead to local trapping. Duan et al. [38] in their part proposed a communication and storage-aware multi-objective task scheduling algorithm, which is based on sequential cooperative game. Their goal is to optimize the execution time and economic cost. In their proposed approach, the individual players are considered behaving selfishly. The global knowledge of all players is computed by ordering each customer's task in decreasing order. The simulation results show their proposed algorithm can achieve better solution in terms of makespan, cost, system-level efficiency and fairness in less execution time compared to Grid-Min-Min, Grid-Max-Min and Grid-Suffrage.

In Verma and Kaushal [39], a Hybrid Particle Swarm Optimization (HPSO) algorithm is proposed. Their objective is to provide resources that can guarantee customers with minimum execution time and processing cost under deadline and budget constraint. In their studies, trade-off values are adopted to provide customers with the choices of selecting their service preferences. According to the researchers, simulation results shows their proposed HPSO scheduling approach can reduce execution time and execution cost as compared to the Non-Static Genetic Algorithm II (NSGA-II), Multi-Objective Particle Swarm Optimization (MOPSO) and ϵ -Fuzzy Dominance sort-based Discrete Particle Swarm Optimization (ϵ -FDPSO) scheduling algorithm. However, the global optimization process exhibited by the PSO may not always guarantee the required optimum solution. Panda and Jana [34] in their part proposed a Multi-Objective Task Scheduling (MOTS) optimization algorithm for heterogeneous multi-cloud environment. Their goal is to minimize both makespan time and total execution cost. In their method, two phases of task scheduling processes were adopted. The first phase goes through normalization process to scale values between 0 and 1. In the second phase, the normalization process is performed by dividing the Expected Time to Compute (ETC) matrix and cost matrix element into their corresponding maximum values. As put forward by the researchers, simulation results show their proposed method

has outperformed the two benchmarked algorithms in achieving minimum execution time, minimum total cost of execution and improves the average cloud utilization. Ramezani et al. [15] introduced a Multi-Objective Particle Swarm Optimization (MOPSO) algorithm to provide customers with better service choices to deal with the challenges of high computation time and cloud performance. As stated by the researchers, their simulation results show they can achieve an optimal solution in a reasonable amount of time. However, incorporating a novel technique can further improve their solution in terms of generating trade-offs for consumers' service preferences during task scheduling.

Gabi et al. [33] put forward an Orthogonal Taguchi-Based Cat Swarm Optimization (OTB-CSO) algorithm to improve the performance of cloud environment. The researchers exploited the advantages of Taguchi method to achieve task mapping on appropriate virtual machines. The simulation results according to the researchers show their proposed algorithm can reduce makespan compared to Min–Max, Hybrid Particle Swarm Optimization with Simulated Annealing (HPSO-SA) and Particle Swarm Optimization with Linear Descending Inertia Weight (PSO-LDIW). However, the improvement in their OTB-CSO to handle a multi-objective optimization problem is required to meet customers' expectation. Liu et al. [40] dwelt on Improved Min–Min algorithm for cloud computing environment. Their objective is to achieve QoS, dynamic priority model and minimum cost of service delivery to customers. In their scheduling process, static priority rule and dynamic changing factors are used for providing the scheduling of higher priority tasks first. The results of their simulation as indicated by the researchers show their proposed algorithm can increase resource utilization, ensure longer task is executed at reasonable time and meet customers' QoS requirement compared to the benchmarked algorithm. In another development, Beegom and Rajasree [41] in their part put forward a new variant of continuous Particle Swarm Optimization (PSO) algorithm called the Integer-PSO. The Integer-PSO adopts Pareto optimality using a weighted sum approach. Their goal is to minimize the makespan and execution cost. In their scheduling process, a model as a constraint biobjective optimization for the makespan and cost is developed. The efficiency of their proposed algorithm is tested using the developed model. Simulation results as shown by the researchers indicate that their proposed Integer-PSO has outperformed the Smallest Position Value (SPV) rule-based PSO technique in terms of achieving minimum makespan and execution cost.

To ensure effective scheduling on heterogeneous virtual machines and reduce task execution time, Akbari and Rashidi [42] proposed a Multi-Objective Scheduling

Cuckoo Optimization Algorithm (MOSCOA). In their proposed approach, each cuckoo is used as a scheduling solution. As tasks are placed in order of their arrivals, the cuckoo technique in turn does the mapping to the most appropriate virtual machines as it enables the movement of tasks toward the global optima region using a target immigration operator. The technique was later evaluated using large number of random graphs and real-world application. The simulation results show their proposed MOSCOA algorithm is much more superior in terms of performance compared to their previously proposed task scheduling algorithm. Voicu et al. [43] introduced Multi-Objective and Multi-Constrained (MOMC) task scheduling algorithm for scheduling tasks in Hadoop system. Their objective is to minimize deadline and budget. The simulation results according to the researchers show their proposed MOMC method can provide better performance in Hadoop system.

In Bilgaiyan et al. [44], a Multi-Objective Cat Swarm Optimization (MOCSO) algorithm is proposed. Their goal is to improve the performance of cloud environment in terms of minimum execution cost, makespan and CPU idle time. In their task scheduling process using the proposed MOCSO, a control variable known as the mixed ratio is used to decide the best virtual machines to assign tasks. The experimental results according to the researchers show their proposed MOCSO algorithm can achieve minimum execution cost, makespan time and CPU idle time than Multi-Objective Particle Swarm Optimization. On the other hand, Xu et al. [45] in their part proposed a Multi-Objective Genetic Optimization Algorithm (MOGA). Their goal is to minimize the average completion time, total completion time and ensure load balancing on virtual machines. In their scheduling process, large tasks are divided into multiple sub-tasks using the chromosomes encoding. Each chromosome length signifies length of the sub-tasks with smaller tasks mapped on virtual machines. To determine the performance of their proposed MOGA, three different fitness functions models were designed to evaluate the fitness of each chromosome according to their objectives. The simulation results presented by the researchers show their proposed MOGA can achieve the minimized average completion time, total completion time and ensure load balancing on virtual machines with faster convergence than the benchmarked single-objective genetic algorithms. In their part, Milani and Navin [46] proposed a multi-objective scheduling algorithm based on PSO technique. Their objective is to minimize the total execution time, average waiting time and number of missed tasks. The researchers exploit the PSO technique to propose a scheduling approach that can allocate tasks on the best virtual machines. To investigate how efficient is their solution, a fitness function model is developed. The experimental

results as put forward by the researchers show their proposed algorithm can achieve minimum execution time, waiting time and missed tasks compared to First Come First Served (FCFS), Shortest Process Next (SPN) and Highest Response Ratio Next (HRRN).

Jena [47] proposed a multi-objective Two-State PSO (TSPSO) algorithm. Their aim is to reduce the energy consumption and makespan at the cloud datacenters. In their scheduling method, selection of best virtual machine is introduced using the nondominance strategies. As the tasks are scheduled across virtual machines, the two-stage PSO generates two fitness functions and are compared to determine the solution that is nondominant. The nondominant solution is then chosen to represent the optimum solution of the task scheduling process. According to the researchers, the simulation results show their proposed TSPSO algorithm can minimize the energy and makespan compared to Best Resource Selection (BRS) and Random Scheduling Algorithm (RSA). In Khajehvand et al. [48], the researchers introduced a hybrid First-Fit Cost-Time Trade-Off (FCTT) and Workflow Planning Cost-Based (WPC) model to minimize the runtime and execution cost of scheduled tasks on virtual machines. In their scheduling process, large task is divided into sub-tasks which are sorted in nonincreasing manner. In their proposed algorithms, a bottom-up traversal technique is later incorporated to assign each sub-task a rank. The child sub-tasks are first allocated virtual machines for their execution. The parent tasks are later executed only when the child tasks completed their execution on the virtual machines. According to the researchers, their simulation results show the proposed FCTT can reduce task runtime and execution cost compared to MOGA and Best Effort (BE) algorithms. However, task updates method exhibited by their proposed WPC technique can lead to longer execution time since the

techniques. Although the metaheuristics are promising than the heuristic techniques, their improvements using a trajectory-based technique like the simulated annealing as well as the greedy-based techniques like the orthogonal Taguchi approach can enable the metaheuristics to become a potential solution in solving a multi-objective task scheduling problem in cloud computing environment. Therefore, this paper addresses the concern of customers service selection strategy using an improved metaheuristic algorithm to meet customers' QoS expectations with a focus on multi-objective task scheduling.

3 Cat swarm optimization

Chu and Tsai [32] proposed the CSO technique. The technique mimics the common behavior of natural cat. The CSO technique has two modes of operation: resting (seeking) and chasing (tracing) mode. The two modes are also referred to as the global and local search. A control factor within CSO known as the mixed ratio (MR) is used to determine the current position of the cat. The cat position also signifies solution (fitness) set. The velocity of the cat is associated with a dimension and a fitness value. As the cat progresses closer to the solution (fitness), it updates itself each time with better results at the memory until all the cats achieve the best solution (fitness) [20, 32, 49]. The following sections explain the seeking and tracing modes [20, 49, 50].

3.1 Seeking mode

The seeking mode is known as the global search process of the CSO technique [50]. Algorithm 1 shows the pseudocode for the seeking mode [49, 50].

Algorithm 1: Seeking Mode Process

1. Generate Y copies of cats k -th, i.e Z_{dq} where, $(1 \leq q \leq Y)$ and $(1 \leq d \leq D)$ where, D is the overall dimension of the problem and Y is the seeking memory pool.
 2. Change at random the dimensions of the cats as per CDC by applying mutation operator to Z_{dq} . Where CDC is the count dimension to change
 3. Determine the fitness values of all changed cats.
 4. Discover the best cats (non-dominant) based on their fitness values.
 5. Replace the position of the k -th cat after picking a cat at random from Y .
-

performance of the algorithm depends on its update process.

Form the literature reviewed so far, high complexity, slow convergence and imbalance between global and local search are some of the drawbacks of the metaheuristic

3.2 Tracing mode

The tracing mode corresponds to the local search process of the CSO. The pseudocode for the tracing mode is shown in Algorithm 2 [20, 49, 50].

Algorithm 2: Tracing Mode

1. Compute and update cat k -th velocity using the velocity in Equation (1)

$$\vec{V}_{k,d}(t) = \vec{V}_{k,d}(t-1) + \left(c_1 r_1 \cdot (\vec{X}_{gbest} - \vec{X}_{k,d}(t-1)) \right) \tag{1}$$

$d = 1, 2, \dots, M$

Where c is the constant value of acceleration, r is a random number in the range of $[0,1]$. $\vec{V}_{k,d}$ is the velocity of the cat k -th; $(t + 1)$ is the iteration number; \vec{X}_{gbest} is the best position of the cat.

2. Add new velocity by computing the current (new) position of the k -th cat using Equation (2):

$$\vec{X}_{k,d}(t) = \vec{X}_{k,d}(t-1) + \vec{V}_{k,d}(t-1) \tag{2}$$

Where, $\vec{X}_{k,d}$ is the position where the cat is currently standing; $\vec{V}_{k,d}$ is the velocity of the cat and $(t + 1)$ is the iteration number.

3. Determine fitness values of all cats.
4. Update achieve contents with the best cats.

3.3 The need for cat swarm optimization improvement

To provide efficient scheduling in cloud datacenters with the goal of meeting customers’ expectations, the CSO then needs to be improved. However, global search of the CSO does not always assure superior solutions when the search space increases. The number of cats that always move into the global search mode of the CSO always exceeds that of the local search. Thus, its convergence toward a stable solution becomes difficult, leading to its entrapment at the local optima [32]. On the other hand, for each iteration, the global and local search modes exhibited by the CSO are independently carried out. These also cause its velocity and position update to perform similar process. This can lead to high computation time during task scheduling on cloud computing environment. In another concern, the CSO can only handle a single-objective optimization problem. Imbalance between global and local search of the CSO becomes another challenge. Hence, there is a need for improving CSO to make it efficient for service provisioning in cloud computing [33].

4 Taguchi orthogonal array

Taguchi method is a greedy approach put forward by Genichi Taguchi [51]. The Taguchi method uses an orthogonal array (OA) matrix representation for its experiment. It involves the study of a large number of design variables to achieve efficient results using few numbers of simulation runs [52]. According to Taguchi, for any two-level orthogonal array (2OA) with Z factors where Z represents the number of designed factors, each factor will be based on two levels. Taguchi formulated a general symbol shown in Eq. (3) for the establishment of an OA with n levels of Z factors [53, 54]:

$$L_n(2^{n-1}), \tag{3}$$

where $n - 1$ represents the number of columns in two-level orthogonal array; $n = 2k$ is the number of experiments corresponding to the n rows and columns; 2 represents the number of levels required for each factor Z ; and k is a positive integer ($k > 1$). The matrix in Table 1 shows the values in the column are mutually orthogonal. According to Taguchi, for any pair of columns, the combinations of all factors at each level occur at an equal number of times. As described in [45], to allocate six factors each with two levels “ $L_8(2^6)$,” only six columns are needed for the run of the experiment. Hence, $L_8(2^7)$ orthogonal seems sufficient, since there are seven columns. The L_8 is an indication that eight experimental runs will be conducted by studying seven variables at two levels. The value “7” represents the dimension of the problem. The main objective of adopting the Taguchi approach is to find an optimal solution in a reasonable amount of time [51, 55]. Detail about the Taguchi method can be found in [33].

Here, the value 1 at each column represents the first set of factors to be considered for the experiment, while the value 2 represents the second set of experiments.

Table 1 $L_8(2^7)$ OA [27]

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

4.1 The Taguchi optimization algorithm

The Taguchi method can serve as a better approach in reducing the execution time of a task when used for solving task scheduling problem in cloud computing. In cloud task scheduling, the total cost of execution is mostly influenced by amount of time tasks are executed on a virtual machine. Hence, incorporation of Taguchi optimization algorithm into a conventional CSO can be a potential solution to achieve the desired results. The pseudocode for the Taguchi optimization algorithm is shown in Algorithm 3 [54, 55].

Algorithm 3: Taguchi Optimization Algorithm

1. Select two-level Orthogonal array $L_n(2^{n-1})$ for the matrix experiment. Where at this instance, n represent the number of tasks to be schedule on a virtual machine.
 2. at this instances, n represent the number of tasks to be schedule on virtual machine.
 3. Generate two sets of velocities $\vec{V}_{set_{1,k,d}}(t-1)$ and $\vec{V}_{set_{2,k,d}}(t-1)$ according to **Equation (15)**
 4. Update the original velocity according to **Equation (17)**
 5. Add new velocity by computing the current (new) position of k-th cat using **Equation (18)**
 6. Calculate the fitness value of all cats using **Equation (14)** in accordance with the Orthogonal array: $L_n(2^{n-1})$
-

5 Pareto dominance strategy

For any task scheduling, the chances of locating an optimal solution that can meet customers’ expectations in terms of minimum execution time and cost are becoming harder in a large search space like the cloud computing environment [56]. Due to multi-criteria requirements associated with customers’ request, the concept of optimality needs to be achieved. Multi-objective optimization approach is a potential solution to solving this kind of problem. It is characterized with trade-off factors, where each trade-off solution corresponds to a specific order of importance of the objectives [18]. Currently, the Pareto optimization strategy is the most widely adopted strategy for solving several multi-objective problems. Individual objectives can be combined using Pareto dominance strategy to achieve their Pareto front [50]. The Pareto optimal solution that represented best possible among these objectives without worsening another objective is chosen as the best candidate solution [24, 39, 53].

Task scheduling to meet the expectation of customers involves dealing with multi-objective problems. The Pareto optimization approach can be adopted to provide customers as many nondominant solutions as possible by allowing a set of trade-offs in terms of execution time and execution cost [37, 57]. Although study shows that the actual customers’ service preferences are quite difficult to predict,

the cloud customers’ attention can be drawn to the trade-off points P^* known as the Pareto front, where customers are allowed to select their service preferences in terms of virtual machines that can provide them with the minimum execution time and cost [48, 58]. At this instance, the customers are left to optimize their service preferences by selecting the best trade-offs [57, 58]. The main goal of this research is to make sure that the Pareto optimal set is discovered in a minimum amount of time for all the tasks scheduled on virtual machines using our proposed technique. This study is based on the following definitions in solving the multi-objective task scheduling problem.

Definition 1.1 Multi-objective Optimization problem.

A typical multi-objective optimization problem can be expressed as a minimization of a K components of a vector function f_i in the form [15, 49]:

$$\text{Min } \vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), f_3(\vec{x}), \dots, f_k(\vec{x})], \quad (4)$$

where $\vec{x} \in \mathcal{X} = \{x_1, \dots\}$ represents the vector of decision variable such that $f_i : R^n \rightarrow R, i = \{1, 2, \dots, k\}$ are the objective functions in a universe U . $\vec{f}(\vec{x})$ is the multi-objective function.

Definition 1.2 Pareto Dominance

In Pareto dominance, two different participants can be judged on how good they are in terms of their performances. Given two candidate solutions \vec{x}_1 and \vec{x}_2 from U , vector \vec{x}_1 is said to dominate vector \vec{x}_2 (denoted as $(\vec{x}_1) \prec (\vec{x}_2)$) if and only if:

$$f_i(\vec{x}_1) \leq f_i(\vec{x}_2), \quad \forall i \in \{1, \dots, k\} \quad (5)$$

$$f_i(\vec{x}_1) < f_i(\vec{x}_2), \quad \exists i \in \{1, \dots, k\} \quad (6)$$

Equation (5) shows that \vec{x}_1 dominates \vec{x}_2 in all objectives, while Eq. (6) shows \vec{x}_1 dominates for at least one of the objectives [7].

Definition 1.3 Pareto Optimal

If for all means \vec{x}_1 is not dominated by any other solution, \vec{x}_1 is then considered to be the nondominant (Pareto optimal) solution.

Definition 1.4 Pareto Optimal set

Set of all solutions $\vec{x}_1 \in U$ correspond to the Pareto optimal.

Definition 1.5 Pareto Front (P*)

The fitness value of the solution is called trade-off or P* if and only if Eq. (7) holds. This is the optimal solution of a multi-objective optimization problem that is comprised of a set of solutions:

$$f(\vec{x}_1) \prec f(\vec{x}_2), \neg \exists \vec{x}_1 \in U \tag{7}$$

where \neg is true if $\vec{x}_1 \in U$; \exists means there exist $\vec{x}_1 \in U$; \in represents an element of U ; and U is the universal set.

The framework that describes the scheduling process using the developed scheme is discussed in the following.

6 The system scheme

Cloud computing consists of several datacenters that are usually managed by the cloud service providers. For any cloud, virtual machines are dynamically created and deployed in datacenters based on task availabilities. These virtual machines are heterogeneous in nature, having different characteristics in terms of memory and sizes. Our assumption is that one datacenter is not sufficiently enough to handle our task scheduling problem. Therefore, two datacenters each with 20 virtual machines are sufficiently enough for our task scheduling problem. Our proposed system scheme is illustrated in Fig. 1. The scheme integrates Pareto optimization strategy for generating a set of trade-offs in finding the best schedule that can minimize the execution time and cost. Three modules (customers, Pareto generator and scheduler) are defined in the proposed scheme, where each of the modules consists of sub-modules that carry out the scheduling process. The proposed scheduling scheme adopts the DMOOTC algorithm to make the scheduling decisions. The global and local resource managers within the scheme work together with the scheduler to achieve near-optimal solution. In the scheduling process, customers submit their request based on certain resource requirements. The task manager is responsible for estimating the amount of resources to execute the customer’s requests. On the other hand, the local resource manager is responsible for monitoring and managing local virtual nodes and to obtain information about processing elements and memory information and bandwidth which are later submitted to the global manager for subsequent

forwarding to the scheduler. The Pareto generator then generates a set of trade-offs according to the customer’s requirement which are presented to the customer to select his/or her choice. The customer opts for his/or her choice of service preference (i.e., the best virtual machine), and the process continues with the scheduler. The DMOOTC task scheduling algorithm then judges whether available resource has met requirement of the task in terms of time and cost. Then, the proposed DMOOTC algorithm dispatches the customer’s task on the chosen virtual resources. The main contribution of this work lies in that the proposed DMOOTC task scheduling algorithm can address uncertainties by allowing customers to realize better performance to cost and time ratio in cloud computing environment.

6.1 Problem description

The problem is first represented by using a set of independent tasks waiting to be scheduled on sequence of heterogeneous virtual machines. $V = \{v_k | m \geq k \geq 1\}$ is a set of virtual machines, where m is the number of virtual machines. $T = \{t_i | n \geq i \geq 1\}$ represents the tasks’ groups, and n is the overall number of tasks [37]. Our goal is to dynamically assign each task $t_i \forall i = \{1, 2, \dots, n\}$ as customers’ requests on appropriate virtual machines $v_k \forall k = \{1, 2, \dots, m\}$ in order to determine the timing and execution cost of the tasks. We assume the following in our scheduling problem: (1) Two datacenters are used for the tasks scheduling; (2) the datacenters are said to belong to the same service provider; (3) tasks are assigned to virtual machines dynamically where the total number of all possible schedules is $(n!)^m$ for the problem with n number of tasks and m number of virtual machines; (4) preemptive scheduling allocation is not allowed; and (5) the cost of using a virtual machine for a time quantum varies from one to the other. By adopting the Expected Time to Compute (ETC) matrix as shown in Eq. (8), our goal is to dynamically assign each virtual machine v_k with the right computing capacity to appropriate customers’ request in order to find the optimum value of the total execution time and the total execution cost [34, 53]:

$$ETC = \begin{bmatrix} t_1 v_1 & t_1 v_2 & \dots & \dots & t_1 v_k \\ t_2 v_1 & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ t_n v_1 & \dots & \dots & \dots & t_n v_k \end{bmatrix} \tag{8}$$

6.2 The proposed multi-objective task scheduling model

Our proposed multi-objective time and cost model is formulated from the problem description. The model reflects

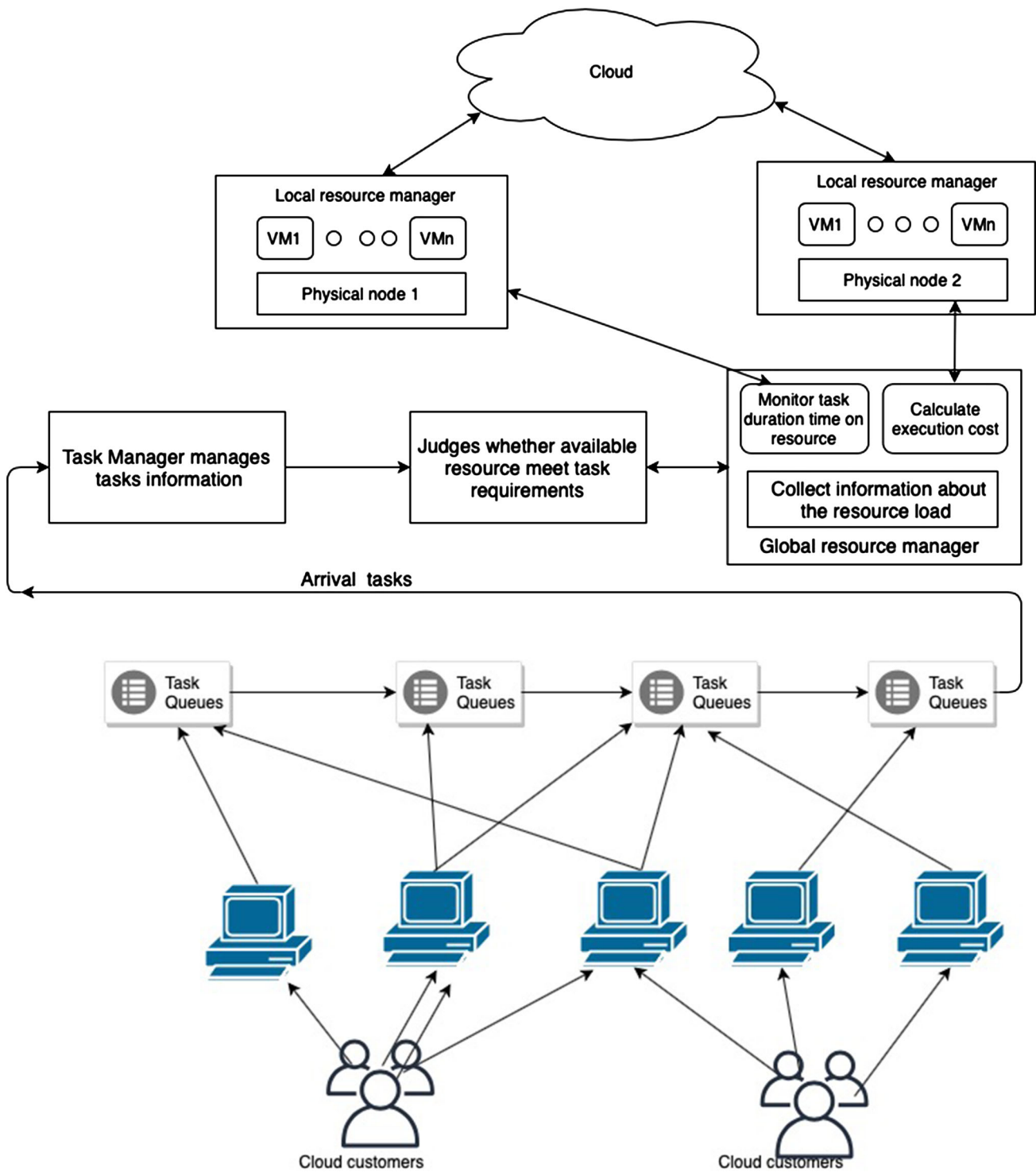


Fig. 1 System model

the relationship between time and cost [14]. A combined method put forward in [14, 15, 25] was used in the formulation of the multi-objective time–cost model. In our assumption for the formulation of the time and cost model, all virtual machines are said to belong to the same service provider, ignoring the cost of data transfer [39].

6.2.1 Execution time model

Let $T = \{t_i | n \geq i \geq 1\}$ denote the set of tasks and $V = \{v_k | m \geq k \geq 1\}$ the set of heterogeneous virtual machines. Assuming $t_i \forall i = \{1, 2, \dots, n\}$ is to be scheduled on $v_k \forall k = \{1, 2, \dots, m\}$, the execution time $exec_k$ of all tasks

processed on a virtual machine is computed using Eq. (9) [15, 25]. The total execution time $Texec_k$ of all tasks t_i processed on all virtual machines v_k is computed using Eq. 10 [25]:

$$exec_k = \sum_{i=1}^n x_{ik} \cdot \frac{t_i v_k}{npe_k \cdot v_{kmips}}, \tag{9}$$

$$Texec_k = \sum_{k=1}^m \sum_{i=1}^n x_{ik} \cdot \frac{t_i v_k}{npe_k \cdot v_{kmips}} \tag{10}$$

where $exec_k$ is the execution time of running tasks on one virtual machine; x_{ik} is equal to 1 if task t_i is assigned to a virtual machine, otherwise $x_{ik} = 0$; t_i is the task whose length is given in million instructions (MIs); v_{kmips} is the virtual machine speed whose unit is given in million instructions per second (MIPS); and npe_k is the number of processing elements of a virtual machine.

6.2.2 Execution cost model

The proposed cost model is a multi-objective task scheduling model that captures the customers’ QoS requirement. The model permits charging a customer based on the amount of time the virtual machine spent executing his/or her request [35]. The time quantum [37] of a virtual machine is the smallest discrete unit use by the service providers to define the cost of a virtual machine in either per second or on hourly basis. In this study, we assume the cost of memory and central processing unit (CPU) are all included in the monetary cost of a virtual machine [25]. For instance, assume for every one-minute N of using a virtual machine, the price specified by the service provider is given as 0.5 dollars per hour. For time quantum in minutes of using a virtual machine, the execution cost can be computed as $\frac{N \cdot 0.5}{60}$ dollars [37]. Assuming the cost v_{kcost} of executing tasks on a virtual machine per hour (/h) is known, Eq. (11) holds for the execution cost exe_{cost_k} of tasks t_i on a virtual machine per time quantum in second [25, 37]:

$$exe_{cost_k} = \frac{1}{3600} \cdot v_{kcost} \cdot \sum_{i=1}^n x_{ik} \cdot \frac{t_i v_k}{npe_k \cdot v_{mipsk}} \tag{11}$$

where v_{kcost} is the monetary cost of a virtual machine per time quantum in US dollar per hour:

$$x_{ik} = \begin{cases} 1, & \text{if } t_i \text{ is assign on a virtual machine.} \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

When more than one $v_k \forall k = \{1, 2, \dots, m\}$ is used by a service provider to execute many tasks, the total tasks execution cost $TTexe_{cost_k}$ by all virtual machine in a data-center in a datacenter can be computed using Eq. (13):

$$TTexe_{cost_k} = \frac{1}{3600} \cdot v_{kcost} \cdot \sum_{k=1}^m \sum_{i=1}^n x_{ik} \cdot \frac{t_i v_k}{npe_k \cdot v_{mipsk}} \tag{13}$$

The multi-objective task scheduling mathematical model can be expressed as follows:

$$Min \quad F(X) = [Texec_k(X), TTexe_{cost_k}(X)] \tag{14}$$

subject to

$$\sum_{k=1}^m x_{ik} = 1, \quad \forall i = 1, 2, \dots, n; x_{ik} \in \{0, 1\}, \forall i, k$$

Equation (14) is the proposed multi-objective optimization time–cost model that captured customers’ QoS requirement.

7 The multi-objective scheduling method based on the proposed DMOOTC scheme

The proposed DMOOTC scheme consists of two phases (global and local search) that are combined to solve the multi-objective task scheduling optimization problem. The following attributes were considered to arrive at an optimal solution: the tasks number, the number of virtual machines and other relevant parameters such as count dimension to change (CDC). Each cat symbolizes the choice of virtual machine used for the task schedule. This is encoded in $[1 \times n]$ vector, with n belonging to a number of tasks. We also assume that each virtual machine in a datacenter has different cost per time quantum.

Based on expected time to compute (ETC), when tasks are scheduled on a virtual machine by our proposed DMOOTC algorithm, it uses two-level orthogonal array $L_n(2^{n-1}) \forall n \geq N + 1$, where N represents task number. Each task is assigned to a cat (also known as the virtual machine). Each of the cats has a dimension D , and the models associated with each cats are based on two objective functions: the total execution time ($Texec_k(X)$) and total execution cost ($TTexe_{cost_k}(X)$). When a cat traverses all tasks, it formed a feasible solution. Each cat has both position and the velocity vector. The position of the cat symbolizes the solution attained by the cat. A mixed ratio (MR) is a control factor that is used to specify two groups of cats. The cats are moved into either seeking or tracing mode at random using the value of the MR. When the cat reaches its desired targets, its fitness value is computed based on the defined objective function ($Texec_k$ and $TTexe_{cost_k}$). This process of assigning tasks to virtual machine mimics the process of the orthogonal approach. As the velocity of cat points the cat toward achieving near-optimal solution, two sets of candidate velocity vectors $\vec{V}_{set1_{k,d}}(t)$ and $\vec{V}_{set2_{k,d}}(t)$ are generated as follows:

$$\begin{cases} \vec{V}_{set1_{k,d}}(t) = w_1 \cdot \vec{V}_{k,d}(t-1) + (c_1 r_1 \cdot (\vec{X}_{gbset_d}(t-1) - \vec{X}_{k,d}(t-1))) \\ \vec{V}_{set2_{k,d}}(t) = w_2 \cdot \vec{V}_{k,d}(t-1) + (c_1 r_1 \cdot (\vec{X}_{ibset_d}(t-1) - \vec{X}_{k,d}(t-1))) \end{cases} \tag{15}$$

such that:

$$\vec{VO}_{k,d}(t) = \begin{cases} \vec{V}_{set1_{k,d}}(t-1), & \text{if orthogonal array element is "1"} \\ \vec{V}_{set2_{k,d}}(t-1), & \text{otherwise} \end{cases} \quad (16)$$

where $\vec{VO}_{k,d}(t)$ represents two candidate velocity sets; d is dimension of the solution space; \vec{X}_{gbset_d} represents the global best position attained by the cat; \vec{X}_{lbset_d} represents the local best position of the cat; w_1, w_2 are the controlled factors; r_1 represents uniform random number in the range of $[0, 1]$; c_1 represents a constant value of the acceleration;

$\vec{X}_{k,d}$ represents the position of the cat; and t , is the number of iterations. To update the velocity, the velocity among the two velocity sets with the best optimum solution is selected using the condition in Eq. (17):

$$\vec{V}_{k,d}(t) \begin{cases} \max v, & \text{if } [\vec{V}_{k,d}(t-1) + \vec{VO}_{k,d}(t-1)] > \text{maximum velocity,} \\ \vec{V}_{k,d}(t-1) + \vec{VO}_{k,d}(t-1) & \text{otherwise} \end{cases} \quad (17)$$

where $\max v$ is the maximum velocity attained by the cat; $\vec{V}_{k,d}$ represents the velocity attained by the cat; and $\vec{VO}_{k,d}(t)$ represents the two candidate solutions. A dominant strategy is used to compare the optimum solution and

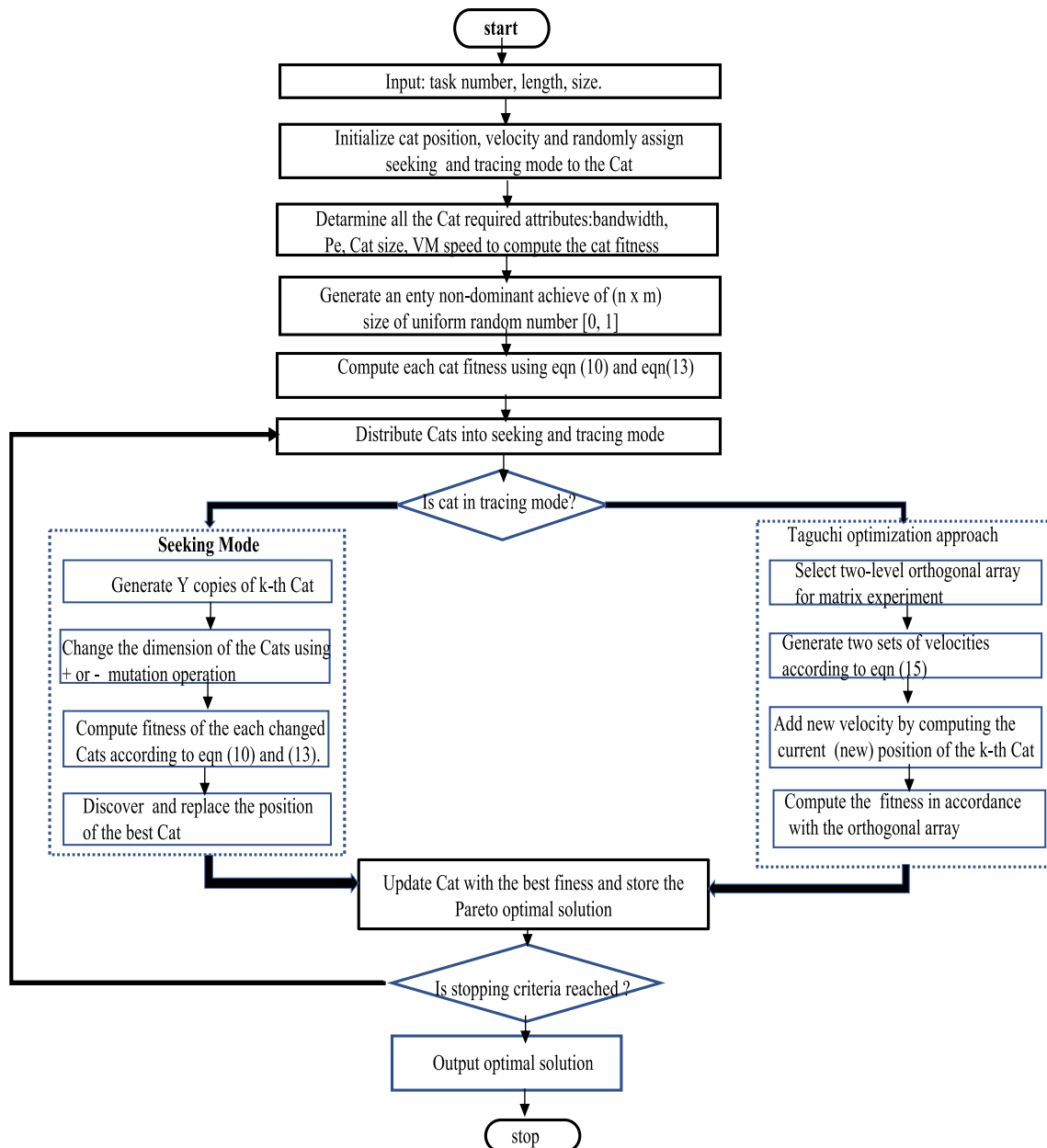


Fig. 2 Flowchart of the DMOOTC scheduling algorithm

is stored at the archive where the final velocity that should formulate the latest velocity is selected. This velocity returns optimal solution which is used to compute the new position of the cat as indicated in Eq. (18):

$$\vec{X}_{k,d}(t) = \vec{X}_{k,d}(t-1) + \vec{V}_{k,d}(t-1). \quad (18)$$

The quality of solution is evaluated using a fitness function. Every cat is assessed based on the value of the fitness function $QoS(\vec{X})$ in Eq. (19):

$$QoS(\vec{X}) = \sum_{j=1}^m W_j f_j(\vec{X}_i), \{ \forall \vec{X}_i \in Archive \} \quad (19)$$

where m is the number of objective functions and W_j is the preference weight for every objective function ($f_j(\vec{X}_i)$).

Algorithm 4 provides the pseudocode for the developed DMOOTC task scheduling algorithm, while Fig. 2 illustrates the flowchart of the scheduling algorithm.

Algorithm 4: DMOOTC-Based Algorithm

INPUT: Task number, lengths, initialize mixed ratio MR ; virtual machine (VM) number and their required attributes (number of processing elements, unit cost of using one VM);

Generate an empty non-dominant archive of $(n \times m)$ size of uniform random number $[0, 1]$

Compute all cats according to defined objective (Fitness) functions

OUTPUT: Optimal Task Schedule

Identify best optimal solution for the trade-off values.

Compare fitness functions of all cats, keep position with best fitness value into the archive

1. **While** condition is not reached
2. **DO**
3. increment_iteration_number \leftarrow t + 1
4. **If** ($MR \leftarrow 0.2$)
5. Generate y (where $y = SMP$) copies of $k - th$ cat
6. Change at random the dimension of cats as per CDC using + or - mutation operation
7. Determine the fitness of changed cats.
8. Discover suitable cats based on their fitness values.
9. **Else**
10. Select two-level orthogonal Taguchi array matrix such that, $L_n(2^{n-1})$, where n represent task number.
11. Generate two sets of velocities according to Equation (15) and determine the dimension of the scheduling problem that corresponds to task number N .
12. Calculate the fitness values of the n experiments using Equation (19) and store Pareto optimal value in the archive
13. **Endif**
14. Generate two candidate solutions X_{ibest} and X_{gbest} from the archive, pick current best solution based on non-dominant sort as X_{ibest} and corresponding best position as X_{gbest}
15. **If** ($X_{ibest} > X_{gbest}$)
16. $X_{ibest} = X_{gbest}$
17. X_{ibest} is now the global best
18. **If (termination condition reached)**
19. **Output** position of best minimum total execution time
20. **Output** position of the best minimum total execution cost
21. **Else**
22. Go to step 2
23. **Endif**
24. **Endif**
25. **EndWhile**

Table 2 Computer and utility software specifications

Windows specifications	Software
Intel® Core™ i5-5200U CPU @ 3.60 GHz 3.60 GHz	Eclipse-java-luna-SR2-win32-x86-64
Window 10 (64-bit), x64-based processor	CloudSim 3.0.3 [41, 54, 55]
4 GB DDR3L RAM	
1000 GB(1 TB) SATA-3G HDD	
Intel® Core™ i5-5200U CPU @ 3.60 GHz 3.60 GHz	

Table 3 Configured properties for the cloud datacenter

	Parameter	Values
Datacenter	No. of datacenters	2
	No. of hosts in a datacenter	1
	Host RAM	2 GB
	Storage	ITB
	Bandwidths	10 GB/s
Tasks	Accumulated host processing power	1,000,000 MIPS
	No. of tasks	[20–100]
	Lengths	[100, 1000] Mis
	File size	[200, 400] MB
	Output size	[300]
	Number of CPUs	[1–4]
	Virtual machine	Virtual machine ID
Virtual machines monitor		Xen
Accumulated RAM		0.5 GB
Accumulated storage		10 GB
Bandwidth		1 GB/s
Virtual machine processing power		1000–10000 MIPS
Number of processing elements		1 to 4
Virtual machine policy		Time-shared
Cost per unit of virtual machine		0.17\$ – 1.25\$/hour
Cost of using memory	0.05\$/hour	

8 Simulation environment

Table 2 shows specifications of the computer system and utility software that we used in the simulation of our proposed DMOOTC scheme. The proposed DMOOTC scheme is benchmarked against Multi-Objective Particle Swarm Optimization (MOPSO) [15], Multi-Objective Ant Colony Optimization (MOACO) [24] and Min–Min [40] task scheduling schemes. The selection of properties for the datacenter host, task and virtual machines is used as in [2, 15, 40]. The estimated cost of a unit virtual machine over a time quantum is adopted as used in [24]. This estimated cost comprises both the computing cost and memory cost and varies from one to the other depending on the capacity. The values for the inertia weight and coefficient factors (c_1c_2) for MOACO, MOPSO and DMOOTC were specified as used in [59]. Table 3 shows properties of

the datacenters, tasks and virtual machines, while Table 4 indicates the parameter settings for the scheduling algorithms.

9 Performance metrics

This study considers four performance metrics to evaluate the efficiency of the developed scheme which are discussed in the following:

9.1 Execution time

The time a task spent executing on a computing resource (i.e., virtual machine) is significant to cloud customers. To measure the performance of our proposed DMOOTC scheme, the model in Eq. (10) is used.

Table 4 Parameter settings for the scheduling schemes

Algorithm	Parameter	Value
MOPSO	Particle size	100
	Self-recognition coefficients (c_1c_2)	2.0
	Uniform random number (R_1)	[0,1]
	Maximum iteration	1000
	Inertia weight (W)	90–40%
	Mixed ratio	2%
	Cat size	100
DMOOTC	Count dimension to change	5%
	Cat size	100
	Self-recognition coefficients (c_1)	2.0
	Uniform random number (R_1)	[0,1]
	Maximum iteration	1000
	Inertia weight (W)	90–40%
	Mixed ratio	2%
MOACO	Pheromone persistence α	0.3
	Importance of pheromone (γ)	1
	Importance of resource innate attribute (β)	1
	Pheromone evaporation value (ρ)	0.3
	Iteration number	1000
	Number of ant m	100

9.2 Execution cost

The cost of a service is that a customer paid for the cloud services he/she has consumed [14, 40]. This is derived from the execution cost. In our study, all virtual machines adopted are heterogeneous with varying cost specification per time quantum. Therefore, virtual machines with high speed are said to return high execution cost. Our aim is to provide customers with better choice of virtual machines that will minimize the execution cost using the model developed in Eq. (13).

9.3 Performance improvement rate

The performance improvement rate (PIR) is computed in percentage using Eq. (20). It helps to investigate the efficiency of our proposed DMOOTC scheme toward achieving better performance compared to the benchmarked schemes [2, 22]:

$$\text{PIR}(\%) = \left(\frac{\text{Execution time}(\text{other scheme}) - \text{Execution tim}(\text{DMOOTC})}{\text{Execution time}(\text{other scheme})} \right) \cdot 100 \quad (20)$$

9.4 Quality of Service

The Quality of Service (QoS) represents the fitness of the proposed DMOOTC scheduling scheme based on any combined objective factors. It is used to reveal the quality of standard provided to the customers. The developed algorithm is designed to achieve better QoS [40]. When the execution time and execution cost values are small, QoS value must be higher. In this study, Eq. (19) is adopted in the evaluation of the QoS of the four scheduling schemes [15, 60].

10 Simulation results and discussion

Two task scheduling scenarios are considered in our experiments. The simulation results are elaborated in these sections. For the first scenario, task instances from 20 to 100 were used with 20 heterogeneous virtual machines, while for the second scenario, High Performance Computing (HPC2N) Net log [61] containing 527, 371 task instances was used. The properties for the datacenters, host and virtual machine settings are similar in configuration to those of the first scenario but vary in terms of the task sizes.

10.1 First scenario

For the first scenario, ten independent simulation runs were conducted in revealing the efficiency of our proposed DMOOTC scheme. Tables 5, 6 and 7 show the average value of the simulation runs obtained in terms of execution time and execution cost. Table 5 shows that the Min–Min scheduling scheme was able to achieve better results in the task scheduling interval from 20 to 50. As the scheduling intervals increase over time (see 70 to 100 tasks), performance of the Min–Min decreases to an unprecedented level. On the other hand, the MOACO scheme thus performs better in the task scheduling interval from 20 to 40. Its performances also degrades with an increase in task sizes above 40. Weakness in performance recorded by the MOACO scheme is probably caused by the traversing process of the ant colony approach, which usually leads to its entrapment at the local optimal region during the search process. An improvement over MOACO scheme is seen in MOPSO task scheduling scheme. The MOPSO achieves better performance in terms of both the execution time and execution cost under the scheduling interval of 50 to 100 compared to MOACO and Min–Min scheduling schemes.

The performance shown by our proposed DMOOTC scheduling scheme is quite remarkable compared to Min–Min, MOACO and MOPSO schemes. Quality-of-Service expectations of customers are paramount to a service

Table 5 Results of the execution time and cost for the scheduling schemes

Task	Improved Min–Min		MOACO		MOPSO		DMOOTC	
	Execution time (s)	Execution cost (\$/h)	Execution time (s)	Execution cost (\$/h)	Execution time (s)	Execution cost (\$/h)	Execution time (s)	Execution cost (\$/h)
20	21.46	4.72	27.54	6.05	26.86	5.90	25.40	5.47
30	39.16	8.61	56.32	12.39	70.29	15.45	53.58	11.78
40	60.44	13.29	87.56	19.26	92.29	20.59	53.92	11.86
50	66.62	14.65	163.49	35.96	147.28	32.40	129.19	28.42
60	209.82	46.15	243.89	53.65	206.50	45.43	162.77	35.80
70	318.81	70.13	312.78	68.10	287.21	63.18	240.12	52.82
80	505.02	111.10	407.54	89.65	334.99	73.69	312.28	68.70
90	768.78	169.13	674.89	148.47	538.09	118.37	331.62	72.94
100	1096.40	241.21	754.90	166.07	663.34	145.93	454.52	99.98

Improved Min–Min Improved Minimum Job First Algorithm, *MOACO* Multi-Objective Ant Colony Optimization, *MOPSO* Multi-Objective Particle Swarm Optimization, *DMOOTC* Dynamic Multi-Objective Orthogonal Taguchi-Cat Algorithm, *hr* hour

Table 6 Comparison of estimated QoS with cost–time weighted factor = −0.5

Task	Improved Min–Min	MOACO	MOPSO	DMOOTC
20	− 10.72	− 13.79	− 13.43	− 12.52
30	− 19.57	− 28.25	− 35.14	− 26.78
40	− 30.22	− 44.01	− 46.80	− 26.96
50	− 33.31	− 82.56	− 73.64	− 64.59
60	− 104.91	− 123.76	− 103.25	− 81.39
70	− 159.53	− 159.34	− 143.61	− 120.06
80	− 252.52	− 208.84	− 269.09	− 156.13
90	− 468.95	− 351.34	− 347.71	− 162.82
100	− 548.25	− 394.86	− 331.68	− 212.10

Improved Min–Min Improved Minimum Job First Algorithm, *MOACO* Multi-Objective Ant Colony Optimization, *MOPSO* Multi-Objective Particle Swarm Optimization, *DMOOTC* Dynamic Multi-Objective Orthogonal Taguchi-Cat Algorithm

Table 7 Comparison of estimated QoS with time–cost weighted factors = −0.5 and − 0.9

Task	Improved Min–Min	MOACO	MOPSO	DMOOTC
20	− 19.31	− 24.80	− 24.17	− 22.54
30	− 35.24	− 50.78	− 63.26	− 48.22
40	− 54.40	− 79.03	− 84.25	− 48.53
50	− 59.96	− 147.95	− 132.55	− 116.27
60	− 188.85	− 220.71	− 185.86	− 146.27
70	− 286.94	− 284.40	− 258.49	− 216.11
80	− 450.93	− 371.86	− 246.30	− 281.06
90	− 709.96	− 621.31	− 484.29	− 298.47
100	− 986.82	− 696.82	− 597.02	− 409.08

Improved Min–Min Improved Minimum Job First Algorithm, *MOACO* Multi-Objective Ant Colony Optimization, *MOPSO* Multi-Objective Particle Swarm Optimization, *DMOOTC* Dynamic Multi-Objective Orthogonal Taguchi-Cat Algorithm

provider for continuous demand of its services. A service provider is expected to meet each customer’s expectations within the service-level agreement. In Tables 6 and 7, results of the QoS based on the four scheduling schemes are shown. The weighted factors − 0.5 and − 0.9 are introduced to serve as a control parameter for customers’ service selection in terms of virtual machine types to guarantee minimum execution time and execution cost. Upon successful simulation runs, the total QoS by Min–Min, MOACO, MOPSO, and that of the proposed DMOOTC schemes are − 2792.41, − 2497.66, − 2076.19 and − 1586.55. Comparisons show that the proposed DMOOTC scheme has returned the highest QoS value of −1586.55 in contrast to those obtained by the Min–Min, MOACO and MOPSO scheduling schemes. In Table 8

precisely, the total values for the execution time, execution cost and QoS obtained are reported to provide insight on the performance of the proposed schemes. We further show how significant our proposed DMOOTC scheme in terms of percentage improvement rate as indicated in Table 9. In the overall performance, the proposed DMOOTC scheduling scheme is able to achieve minimum execution time with 42.87%, 35.47% and 25.49% improvement compared to that of the Min–Min, MOACO and MOPSO task scheduling schemes. In a similar development, DMOOTC scheme is also able achieve 38.62%, 35.32% and 25.56% improvement compared to the benchmarked schemes. Figures 3 and 4 show the trend of the performances of the scheduling schemes. From the trend

Table 8 Total QoS obtained by the scheduling schemes

	Improved Min–Min	MOACO	MOPSO	DMOOTC
Total execution time	3086.51	2732.94	2366.85	1763.40
Total execution cost	631.84	599.60	520.94	387.77
Total estimated QoS based on cost and time weights = −0.5	− 1627.98	− 1406.75	− 1364.35	− 863.35
Total estimated QoS based on cost and time weights = − 0.5 and − 0.9	− 2792.41	− 2497.66	− 2076.19	− 1586.55

Improved Min–Min Improved Minimum Job First Algorithm, *MOACO* Multi-Objective Ant Colony Optimization, *MOPSO* Multi-Objective Particle Swarm Optimization, *DMOOTC* Dynamic Multi-Objective Orthogonal Taguchi-Cat Algorithm

Table 9 Improvement in percentage

	Improved Min–Min	MOACO	MOPSO	DMOOTC
<i>Total execution time</i>	3086.51	2732.94	2366.85	1763.40
PIR (%) over Improved Min–Min		11.455	23.31	42.87
PIR (%) over MOACO			13.39	35.47
PIR (%) over MOPSO				25.49
<i>Total execution cost</i>	631.84	599.60	520.94	387.77
PIR (%) over Improved Min–Min		5.10	17.55	38.62
PIR (%) over MOACO			13.11	35.32
PIR (%) over MOPSO				25.56

Improved Min–Min Improved Minimum Job First Algorithm, *MOACO* Multi-Objective Ant Colony Optimization, *MOPSO* Multi-Objective Particle Swarm Optimization, *DMOOTC* Dynamic Multi-Objective Orthogonal Taguchi-Cat Algorithm, *PIR*(%) Performance Improvement Rate (in percentage)

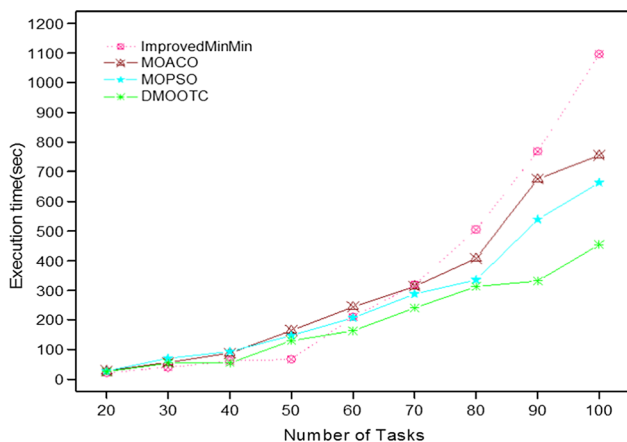


Fig. 3 Total execution time

illustrated in these figures, it has shown that our proposed DMOOTC scheme has the ability to provide customers with the best services in terms of virtual machines that can guarantee them with the minimum execution time and execution cost compared to the benchmarked schemes.

The achievements displayed by our customers service selection scheme are as a result of the incorporation of orthogonal Taguchi at its local search procedure which helps the scheme in traversing all cats (virtual machines), and the use of Pareto dominance strategy provides the best

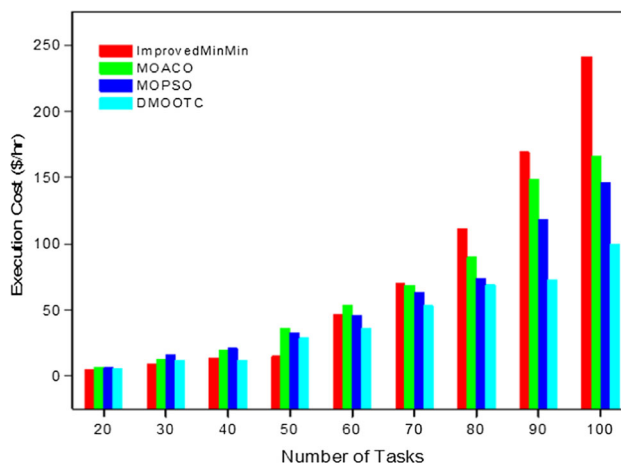


Fig. 4 Total execution cost

optimum solutions of the multi-objective in terms of execution time and execution cost.

10.2 Second scenario

In this scenario, we used large-workload-containing HPC2N dataset to determine the performances of the four scheduling schemes. The Parallel Workload Archive—HPC2N is made available by the High-Performance Computing Center North (HPC2N). It is a setlog with

Table 10 Results of the execution time and cost for the scheduling schemes

Task	Improved Min–Min		MOACO		MOPSO		DMOOTC	
	Execution time (s)	Execution cost (\$/h)	Execution time (s)	Execution cost (\$/h)	Execution time (s)	Execution cost (\$/h)	Execution time (s)	Execution cost (\$/h)
100	486.68	107.07	967.56	212.86	1193.18	262.50	1134.25	249.53
200	1452.91	319.64	1786.67	393.06	2893.53	632.57	2564.98	564.29
300	3857.67	848.68	7865.24	1730.35	6975.66	1534.64	4587.67	1009.28
400	6231.47	1370.92	9078.56	1997.28	10411.32	2290.49	5089.76	1119.74
500	9918.75	2182.12	11436.78	2516.09	11458.76	2520.92	8986.08	1976.93
600	13810.89	3038.39	15451.23	3399.27	12764.87	2808.27	10679.87	2349.57
700	18380.38	4043.68	16895.87	3717.09	14023.48	3085.16	13867.32	3050.81
800	22984.71	5056.63	19417.56	4271.86	18513.75	4073.02	16894.90	3716.87
900	29688.63	6531.48	24980.23	5495.65	23278.90	5121.35	21856.23	4808.37
1000	35424.75	7793.44	29672.78	6528.01	26867.23	5910.79	25785.45	5672.79

Improved Min–Min Improved Minimum Job First Algorithm, *MOACO* Multi-Objective Ant Colony Optimization, *MOPSO* Multi-Objective Particle Swarm Optimization, *DMOOTC* Dynamic Multi-Objective Orthogonal Taguchi-Cat Algorithm, *hr* hour

Table 11 Estimated QoS with time–cost weight factor = – 0.5

Task	Improved Min–Min	MOACO	MOPSO	DMOOTC
100	– 243.35	– 483.80	– 596.62	– 567.15
200	– 726.50	– 893.38	– 1446.85	– 1282.56
300	– 1928.95	– 3932.86	– 3488.04	– 2293.97
400	– 3115.92	– 4539.55	– 5205.97	– 2545.03
500	– 4959.67	– 5718.73	– 5729.73	– 4493.31
600	– 6905.86	– 7726.08	– 6382.82	– 5340.26
700	– 9190.75	– 8448.45	– 7012.16	– 6934.08
800	– 11493.05	– 9709.37	– 10183.12	– 8447.96
900	– 14845.22	– 12490.87	– 11640.16	– 10928.78
1000	– 17713.45	– 14837.29	– 13434.43	– 12893.51

Improved Min–Min Improved Minimum Job First Algorithm, *MOACO* Multi-Objective Ant Colony Optimization, *MOPSO* Multi-Objective Particle Swarm Optimization, *DMOOTC* Dynamic Multi-Objective Orthogonal Taguchi-Cat Algorithm

Table 12 Estimated QoS with time–cost weight factors = –0.5 & – 0.9

Task	Improved Min–Min	MOACO	MOPSO	DMOOTC
100	– 438.03	– 870.83	– 1073.90	– 1020.85
200	– 1307.66	– 1608.05	– 2604.27	– 2308.56
300	– 3472.02	– 7078.95	– 6278.31	– 4129.04
400	– 5608.51	– 8170.98	– 9370.50	– 4580.93
500	– 8927.17	– 10293.45	– 10313.23	– 8087.74
600	– 12930.22	– 13906.57	– 11488.77	– 9612.20
700	– 16542.90	– 15206.79	– 12621.56	– 12481.01
800	– 20686.94	– 17476.39	– 16662.94	– 15205.92
900	– 26720.67	– 22482.97	– 20951.72	– 19671.27
1000	– 31883.35	– 26706.40	– 24181.32	– 23207.69

Improved Min–Min Improved Minimum Job First Algorithm, *MOACO* Multi-Objective Ant Colony Optimization, *MOPSO* Multi-Objective Particle Swarm Optimization, *DMOOTC* Dynamic Multi-Objective Orthogonal Taguchi-Cat Algorithm

Table 13 Total QoS obtained by the scheduling schemes

	Improved Min–Min	MOACO	MOPSO	DMOOTC
Total execution time	142,236.84	137,552.48	128,380.68	111,446.51
Total execution cost	31,292.05	30,261.52	28,239.71	24,518.18
Total estimated QoS based on cost and time weights = -0.5	$-71,122.72$	$-68,780.38$	$-65,119.90$	$-48,792.35$
Total estimated QoS based on cost and time weights = -0.5 and -0.9	$-128,517.47$	$-123,801.38$	$-115,546.52$	$-100,305.31$

Improved Min–Min Improved Minimum Job First Algorithm, *MOACO* Multi-Objective Ant Colony Optimization, *MOPSO* Multi-Objective Particle Swarm Optimization, *DMOOTC* Dynamic Multi-Objective Orthogonal Taguchi-Cat Algorithm

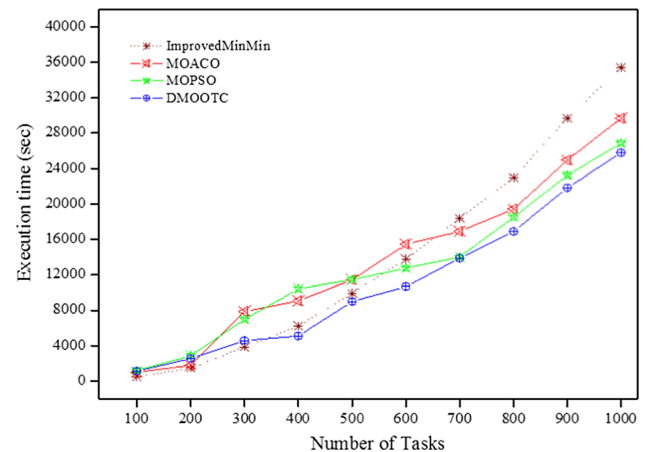
Table 14 Improvement in percentage

	Improved Min–Min	MOACO	MOPSO	DMOOTC
<i>Total execution time</i>	142,236.84	137,552.48	128,380.68	111,446.51
PIR (%) over Improved Min–Min		3.29	9.74	21.64
PIR (%) over MOACO			6.66	18.97
PIR (%) over MOPSO				13.19
<i>Total execution cost</i>	31,292.05	30,261.52	28,239.71	24,518.18
PIR (%) over Improved Min–Min		3.29	9.75	21.64
PIR (%) over MOACO			6.66	18.97
PIR (%) over MOPSO				13.19

Improved Min–Min Improved Minimum Job First Algorithm, *MOACO* Multi-Objective Ant Colony Optimization, *MOPSO* Multi-Objective Particle Swarm Optimization, *DMOOTC* Dynamic Multi-Objective Orthogonal Taguchi-Cat Algorithm, *PIR*(%) Performance Improvement Rate (in percentage)

information on about 527, 371 tasks. The HPC2N workload log was freely provided by Ake Sandgren, who also helped with background information and interpretation, while Michael Jack assisted in making sure the log is hosted into the archive for general usage [2]. We stored the workload in a sac folder at the CloudSim. The datacenter broker within the CloudSim is configured to make use of the scheduling schemes as the main scheduler instead of using the default scheduler. Tasks are pooled from the sac folder containing the HPC2N dataset for each run of the simulation based on assigned scheduling intervals (100–1000) using each of the scheme sets for the simulation.

The results after simulations are indicated in Tables 10, 11, 12, 13 and 14. These results attest to the fact that our proposed DMOOTC scheme can provide minimum execution time and execution cost compared to improved Min–Min, MOACO and MOPSO schemes. More precisely, Table 13 summarizes the whole results in terms of total execution time and cost. The proposed DMOOTC scheduling scheme is able to achieve -48792.35 QoS value when a customer selects -0.5 weight factor for his/ or her cloud service and also achieves -1000305.31 for time and cost weight factors of -0.5 and -0.9 . Table 14 reports a significant improvement gained by our proposed DMOOTC scheme over improved Min–Min, MOACO and MOPSO scheduling schemes in terms of execution time and execution cost. The proposed DMOOTC scheme has

**Fig. 5** Total execution time

achieved 21.64%, 18.97% and 13.17% improvement compared to improved Min–Min, MOACO and MOPSO algorithms. The continuous display of performance by our proposed DMOOTC scheduling scheme is attributed to the incorporation of orthogonal Taguchi approach and the use of Pareto dominance strategy to provide customers with the choices of virtual machines that help in retuning an optimum solution of the multi-objective problem. Likewise, figures are used to further show the trend in performance of the scheduling scheme under different scheduling intervals as shown in Figs. 5 and 6.

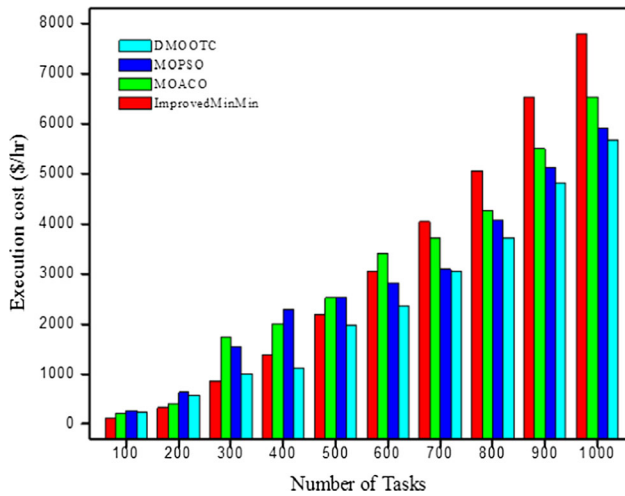


Fig. 6 Total execution cost

11 Statistical analysis on 95% confidential interval

A statistical analysis based on 95% confidence interval is provided to show how significant our obtained results for both scenarios compared to those of the benchmarked

schemes. Tables 15 and 16 present the computed 95% confidence intervals for both scenarios according to Eq. (21) [62]:

$$\text{Confidential Interval (CI)} = \bar{x} \pm t \frac{s}{\sqrt{n}} \tag{21}$$

where \bar{x} represents the mean; t represents t-distribution that is derived from the t-distribution table; s represents the standard deviation of the sample data and n represents the number of samples. The smaller the value of the confidence interval, the more precise our estimate. For results shown in Tables 15 and 16, the 95% confidence intervals obtained by our proposed DMOOTC scheduling scheme are less compared to those obtained by the benchmarked schemes. This means that there is a significant difference in the results obtained by our proposed DMOOTC scheduling schemes compared to the benchmarked schemes. It can be concluded that our proposed DMOOTC scheduling scheme can provide cloud customers with better services that will meet their expectations and adapt the elasticity of cloud computing environment than the benchmarked scheduling schemes.

Table 15 Computed 95% confidence interval for the first scenario

	Improved Min–Min	MOACO	MOPSO	DMOOTC
Degree of freedom	9	9	9	9
Confidence level	0.025	0.025	0.025	0.025
<i>t</i> -distribution	2.262	2.267	2.267	2.262
Mean	342.94	303.21	262.98	195.93
Standard deviation	378.38	264.49	218.04	148.25
Lower bound	72.28	114.02	107.01	90.04
Upper bound	613.60	492.40	418.95	254.14
95% confidential interval	(72.28, 613.60)	(114.02, 492.40)	(107.01, 418.95)	(90.04, 254.14)

Table 16 Computed 95% confidence interval for the second scenario

	Improved Min–Min	MOACO	MOPSO	DMOOTC
Degree of freedom	9	9	9	9
Confidence level	0.025	0.025	0.025	0.025
<i>t</i> -distribution	2.262	2.267	2.267	2.262
Mean	11,868.01	11986.63	11279.27	9517.90
Standard deviation	10,155.57	7995.00	7048.25	6979.87
Lower bound	3378.74	3412.51	3211.13	2709.68
Upper bound	20,357.28	20,560.75	19,347.41	16,326.12
95% confidential interval	(3378.74, 20,357.28)	(3412.51, 20,560.75)	(3211.13, 19,347.41)	(2709.68, 16,326.12)

12 Conclusion

Scheduling of cloud service for the purpose of meeting the expectations of each customer in cloud computing is a non-deterministic polynomial times (NP-hard) hard problem. Solutions required to facilitate the provisioning of better cloud service are rather too complex to develop. In this paper, we proposed a cloud customers service selection scheme known as Dynamic Multi-Objective Orthogonal Taguchi-Cat (DMOOTC) that served as an ideal solution. The proposed DMOOTC scheduling scheme not only considers meeting customers' QoS expectations, but also facilitates the provisioning of several service choices for customers to select their service preference. Two computing scenarios were adopted in evaluation of the efficiency of our proposed DMOOTC scheduling scheme via simulation. The simulation results obtained in both scenarios show our proposed DMOOTC scheduling scheme had returned minimum execution time and cost for all scheduled tasks and also provided better QoS compared to the benchmarked schemes. We further revealed the significance of our proposed DMOOTC scheduling scheme using statistical analysis based on 95% confidence interval. The statistical results obtained by our proposed DMOOTC scheduling scheme are quite significant than those obtained by the benchmarked schemes. The overall performances displayed by our proposed DMOOTC scheme is as a result of the incorporation of an orthogonal Taguchi strategy at its local search, which facilitates better task mapping on virtual machines and the use of Pareto dominance strategy that provides customers with several service choices to select their preference. Further studies are therefore necessary to investigate the scalability of the proposed DMOOTC scheduling scheme using large workloads.

Acknowledgements Open access funding provided by Umea University. The use of Universiti Teknologi Malaysia facilities and its services is hereby acknowledged with thanks.

Compliance with ethical standards

Conflict of interest There is no conflict of interest for this manuscript. All authors agreed to its submission.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Gui Z, Yang C, Xia J, Huang Q, Liu K, Li Z, Yu M et al (2016) A service brokering and recommendation mechanism for better selecting cloud services. *PLoS ONE* 9(8):e105297. <https://doi.org/10.1371/journal.pone.0105297>
2. Abdullahi M, Ngadi MA, Abdulhamid SM (2016) Symbiotic Organism Search optimization-based task scheduling in cloud computing environment. *Future Gener Comput Syst* 56(2016):640–650
3. Thanasias V, Lee C, Hanif M, Kim E, Helal S (2016) VM capacity-aware scheduling within budget constraints in IaaS clouds. *PLoS ONE* 11(8):e0160456. <https://doi.org/10.1371/journal.pone.0160456>
4. Gabi D (2014) Surveillance on security issues in cloud computing: a view on forensic perspective. *Int J Sci Eng Res* 5(5):1246–1252
5. Gabi D, Ismail AS, Zainal A (2015) Systematic review on existing load balancing techniques in Cloud Computing. *Int J Comput Appl* 125(9):16–24
6. Meena M., Bharadi VA (2016). Performance analysis of cloud based software as a service (SaaS) model on public and hybrid cloud. In: Proceedings of the 2016 symposium on colossal data analysis and networking (CDAN). 18–19 March. Indore, Madhya Pradesh, India, pp 1–6
7. Tassabehji R, Hackney R (2016) Hey You? Get Off My Cloud: evaluation of cloud Service Models for Business Value within Pharma X. *J Adv Manag Sci Inf Syst* 2:48–52
8. Zhang Y, Qian C, Lv J, Liu Y (2017) Agent and cyber-physical system based self-organizing and self-adaptive intelligent shop-floor. *IEEE Trans Ind Inf* 13(2):737–747
9. Furht B (2010) Cloud computing fundamentals. In: Furht B, Escalante A (eds) *Handbook of cloud computing*. Springer, New York, pp 3–19
10. Raza HM, Adenola FA, Nafarieh A, Robertson W (2015) The slow adoption of cloud computing and its workforce. *Proc Comput Sci* 52(2015):1114–1119
11. Zhan Z-H, Liu X-F, Gong Y-J, Zhang J, Chung HS-H, Li Y (2015) Cloud computing resource scheduling and a survey of its evolutionary approaches. *J ACM Comput Survey* 47(4):1–33
12. Cui H, Liu X, Yu T, Zhang H, Fang Y, Xia Z (2017) Cloud service scheduling algorithm research and optimization. *Hindawi Secur Commun Netw* 2017:1–7
13. Gabi D, Ismail AS, Zainal A, Zakaria Z (2018) Quality of Service (QoS) task scheduling algorithm with taguchi orthogonal approach for cloud computing environment. In: Saeed F, Gazem N, Patnaik S, Saed Balaid A, Mohammed F (eds) *Recent Trends in Information and Communication Technology*. IRICT 2017. Lecture notes on data engineering and communications technologies, vol 5. Springer, Cham, pp 641–649
14. Zuo L, Shu L, Dong S, Zhu C, Hara T (2015) A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. *IEEE Access J Rapid Open Access Publ* 3:2687–2699
15. Ramezani F, Lu J, Taheri J, Hussain FK (2013). Task scheduling optimization in cloud computing applying multi-objective particle swarm. *LNCS 8274*, 2013: Springer, Berlin, pp 237–251
16. Zhang F, Cao J, Li K, Khan US, Hwang K (2014) Multi-objective scheduling of many tasks in cloud platforms. *Futur Gener Comput Syst* 37(2014):309–320
17. Xie Z, Shao X, Xin Y (2016) A scheduling algorithm for cloud computing system based on the driver of dynamic essential path. *PLoS ONE* 11(8):e0159932. <https://doi.org/10.1371/journal.pone.0159932>

18. Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, New York
19. Subashini G, Bhuvaneshwari MC (2011) Non dominant particle swarm optimization for scheduling independent tasks on heterogeneous distributed environments. *Int J Adv Soft Comput Appl* 3(1):1–17
20. Gabi D, Ismail AS, Zainal Zakaria Z (2017) Solving task scheduling problem in cloud computing environment using orthogonal Taguchi-cat algorithm. *Int J Electr Comput Eng (IJECE)* 7(3):1489–1497
21. Souza Pardo MH, Centurion AM, Franco Eustáquio PS, Carlucci Santana RH, Bruschi SM, Santana MJ (2016) Evaluating the influence of the client behavior in cloud computing. *PLoS ONE* 11(7):e0158291. <https://doi.org/10.1371/journal.pone.0158291>
22. Abdulhamid SM, Abd Latiff MS, Abdul-Salaam G, Hussain Madni SH (2016) Secure scientific applications scheduling technique for cloud computing environment using global league championship algorithm. *PLoS ONE* 11(7):e0158102. <https://doi.org/10.1371/journal.pone.0158102>
23. Tran D-H, Cheng M-Y, Cao M-T (2015) Hybrid multiple-objective artificial bee colony with differential evolution for the time–cost–quality tradeoff problem. *Knowl Based Syst* 74(2015):176–186
24. Lakra AV, Yadav DK (2015) Multi-objective task scheduling algorithm for cloud computing throughput optimization. *Proc Comput Sci J* 48(2015):107–113
25. Ramezani F, Lu J, Taheri J, Hussain FK (2015) Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments. *World Wide Web* 18:1737–1757. <https://doi.org/10.1007/s11280-015-0335-3>
26. Rubio JJ, Lughofer E, Meda-Campaña JA, Páramo LA, Novoa JF, Pacheco J (2018) Neural network updating via argument Kalman filter for modeling of Takagi-Sugeno fuzzy models. *J Intell Fuzzy Syst* 35(2):2585–2596
27. Soares AM, Fernandes BJT, Bastos-Filho CJA (2018) Pyramidal neural networks with evolved variable receptive fields. *Neural Comput Appl* 29(12):1443–1453
28. Rubio JJ (2017) USNFIS: uniform stable neuro fuzzy inference system. *Neurocomputing* 262:57–66
29. Liu Y, Wang Z, Yuan Y, Alsaadi FE (2018) Partial-nodes-based state estimation for complex networks with unbounded distributed delays. *IEEE Trans Neural Netw Learn Syst* 29(8):3906–3912
30. Rubio JJ (2009) SOFMLS: online self-organizing fuzzy modified least-squares network. *IEEE Trans Fuzzy Syst* 17(6):1296–1309
31. Li X, Li H, Sun B, Wang F (2018) Assessing information security risk for an evolving smart city based on fuzzy and grey FMEA. *J Intell Fuzzy Syst* 34(4):2491–2501
32. Chu S-C, Tsai P-W (2007) Computational intelligence based on the behavior of cats. *Int J Innov Comput Inf Control* 3(2007):163–173
33. Gabi D, Ismail AS, Zainal A, Zakaria Z, Abraham A (2016) Orthogonal Taguchi-based cat algorithm for solving task scheduling problem in cloud computing. *Neural Comput Appl* (2016) 30(6):1845–1863
34. Panda SK, Jana PK (2015) A Multi-objective task scheduling algorithm for heterogeneous multi-cloud environment. In: *Proceedings of the international conference on electronic design, computer networks and automated verification (EDCAV)*, 2015. IEEE
35. Pacini E, Mateos C, Garino CG (2015) Balancing throughput and response time in online scientific Clouds via Ant Colony Optimization. *Adv Eng Softw* 84:31–47
36. Wei X, Sun B, Cui J, Xu G (2016) A multi-objective compounded local mobile cloud architecture using priority queues to process multiple jobs. *PLoS ONE* 11(7):e0158491. <https://doi.org/10.1371/journal.pone.0158491>
37. Liu J, Pacitti E, Valduriez P, de Oliveira D, Mattoso M (2016) Multi-objective scheduling of Scientific Workflows in multisite clouds. *Futur Gener Comput Syst* 63(2016):76–95
38. Duan R, Prodan R, Li X (2014) Multi-objective game theoretic scheduling of bag-of-tasks workflows on hybrid clouds. *IEEE Trans Cloud Comput I* 2:29–42
39. Verma A, Kaushal S (2015) Cost-time efficient scheduling plan for executing workflows in the cloud. *J Grid Comput* 2015:495–506. <https://doi.org/10.1007/s10723-015-9344-9>
40. Liu G, Li J, Xu J (2013) An improved min-min algorithm in cloud computing. *Adv Intell Syst Comput*. https://doi.org/10.1007/978-3-642-33030-8_8
41. Beegom ASA, Rajasree MS (2014) A particle swarm optimization based pareto optimal task scheduling in cloud computing. *Lect Notes Comput* 8795:79–86
42. Akbari M, Rashidi H (2016) A multi-objectives scheduling algorithm based on cuckoo optimization for task allocation problem at compile time in heterogeneous systems. *Expert Syst Appl* 60(2016):234–248
43. Voicu C, Pop F, Dobre C, Xhafa F (2014) MOMC: multi-objective and multi-constrained scheduling algorithm of many tasks in Hadoop. In: *Proceedings of the ninth international conference on P2P, parallel, grid, cloud and internet computing (3PGCIC)*; 2014: IEEE
44. Bilgaiyan S, Sagnika S, Das M (2015) A multi-objective cat swarm optimization algorithm for workflow scheduling in cloud computing environment. *Int J Soft Comput* 10(1):37–45
45. Xu Z, Xu X, Zhao X (2015) Task scheduling based on multi-objective genetic algorithm in cloud computing. *J Inf Comput Sci* 12(4):1429–1438
46. Milani FS, Navin AH (2015) Multi-objective task scheduling in the cloud computing based on the patrice swarm optimization. *Int J Inf Technol Comput Sci* 7(5):61–66
47. Jena RK (2015) Multi-objective task scheduling in cloud environment using nested PSO framework. *Proc Comput Sci J* 57(2015):1219–1227
48. Khajehvand V, Pedram H, Zandieh M (2014) Multi-objective and scalable heuristic algorithm for workflow task scheduling in utility grids. *J Optim Ind Eng* 14(2014):27–36
49. Pradhan PM, Panda G (2012) Solving Multi-objective problems using cat swarm optimization. *Int J Expert Syst Appl* 39(2012):2956–2964
50. Shojaei R, Faragardi R. H, Alaei S, Yazdani N (2012) A New cat swarm optimization based algorithm for reliability-oriented task allocation in distributed systems. In: *symposium on sixth international telecommunications (IST)*. IEEE
51. Taguchi G, Chowdhury S, Taguchi S (2000) *Robust engineering: learn how to boost quality while reducing costs and time to market*. McGraw-Hill, New York
52. Tsai P-W, Pan J-S, Chen S-M, Lio B-Y (2012) Enhanced parallel cat swarm optimization based on taguchi method. *Expert Syst Appl* 39(2012):6309–6319
53. Tsai J-T, Fang J-C, Chou J-H (2013) Optimized tasks scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. *Comput Oper Res* 40(2013):3045–3055
54. Chang H-C, Chen Y-P, Liu T-K, Chou J-H (2015) Solving the flexible job shop scheduling problem with makespan optimization by using a hybrid Taguchi-genetic algorithm. *IEEE J Mag* 3:1740–1754
55. Tsai J-T, Liu T-K, Ho W-H, Chou J-H (2008) An improved genetic algorithm for job-shop scheduling problems using Taguchi-based crossover. *Int J Adv Manuf Technol* 2008(38):987–994

56. Saule C, Giegerich R (2015) Pareto optimization in algebraic dynamic programming. *Algorithms Mol Biol* 10(22):1–20
57. Kalra M, Singh S (2015) A review of metaheuristic scheduling techniques in cloud computing. *Egypt Inf J* 16(3):275–295
58. Farahabady HRM, Lee CY, Zomaya YA (2014) Pareto-optimal cloud bursting. *IEEE Trans Parallel Distrib Syst* 25(10):2670–2682
59. Eberhart RC, Shi Y (2000) Comparing inertia weights and constriction factors in particle swarm optimization. In: *Proceedings of the IEEE conference on evolutionary computation (ICEC)*. IEEE
60. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R (2010) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw Pract Exp* 41(1):23–50
61. HPC2N.TheHPC2NSethlog (2016) http://www.cs.huji.ac.il/labs/parallel/workload/l_hpc2n/index.html. Accessed 12 Apr 2018
62. Hosmer DW, Lemeshow S (1992) Confidence interval estimation of interaction. *Epidemiology* 3(5):452–456

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.