Task Scheduling Optimization in Cloud Computing Applying Multi-Objective Particle Swarm Optimization

Fahimeh Ramezani, Jie Lu, and Farookh Hussain

Decision Systems & e-Service Intelligence Lab
Centre for Quantum Computation & Intelligent Systems
School of Software, Faculty of Engineering and Information Technology
University of Technology, Sydney, P.O. Box 123 Broadway NSW 2007 Australia
`Fahimeh.Ramezani@student.uts.edu.au`,
`{Jie.Lu,Farookh.Hussain}@uts.edu.au`

Abstract. Optimizing the scheduling of tasks in a distributed heterogeneous computing environment is a nonlinear multi-objective NP-hard problem which is playing an important role in optimizing cloud utilization and Quality of Service (QoS). In this paper, we develop a comprehensive multi-objective model for optimizing task scheduling to minimize task execution time, task transferring time, and task execution cost. However, the objective functions in this model are in conflict with one another. Considering this fact and the supremacy of Particle Swarm Optimization (PSO) algorithm in speed and accuracy, we design a multi-objective algorithm based on multi-objective PSO (MOPSO) method to provide an optimal solution for the proposed model. To implement and evaluate the proposed model, we extend Jswarm package to multi-objective Jswarm (MO-Jswarm) package. We also extend Cloudsim toolkit applying MO-Jswarm as its task scheduling algorithm. MO-Jswarm in Cloudsim determines the optimal task arrangement among VMs according to MOPSO algorithm. The simulation results show that the proposed method has the ability to find optimal trade-off solutions for multi-objective task scheduling problems that represent the best possible compromises among the conflicting objectives, and significantly increases the QoS.

Keywords: Cloud computing, Task Scheduling, Multi-Objective Particle Swarm Optimization, Jswarm, Cloudsim.

1 Introduction

Cloud computing provides new business opportunities for both service providers and requestors (e.g. organizations, enterprises, and end users) by means of a platform for delivering Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). A Cloud encloses IaaS, PaaS, and/or SaaS inside its own virtualization infrastructure to carry out an abstraction from its underlying physical assets [1].

Task scheduling problems which relate to the efficiency of the whole cloud computing facilities, are one of the most famous combinatorial optimization problems, and

© Springer-Verlag Berlin Heidelberg 2013

play a key role in improving flexible and reliable systems. The main purpose is to schedule tasks to adaptable resources in accordance with adaptable time, which involves establishing a proper sequence whereby tasks can be executed under transaction logic constraints [2]. The scheduling algorithms in distributed systems usually have the goal of spreading the load on processors and maximizing their utilization while minimizing the total task execution time [3]. In the cloud environment, the number of tasks in a workflow as well as the number of available resources can grow quickly, especially when virtual resources are allocated. Calculating all possible task-resource mappings in cloud environment and selecting the optimal mapping is not feasible, since the complexity would grow exponentially with the number of tasks and resources [4]. The use of a heuristic algorithm ensures an acceptable runtime of the scheduling algorithm itself since it significantly reduces the complexity of the search space. This provides a compromise between scheduling runtime and optimality of the assignment. Of the heuristic optimization algorithms, genetic algorithm, fuzzy-genetic algorithm, multi-objective genetic algorithm, swarm optimization and normal best-oriented ant colony have been applied in previous works for optimizing task scheduling, mostly with two main objectives: (1) to minimize the task execution time and (2) to minimize cost in the cloud environment.

Although a significant amount of research has been done in this area, the majority assumed that the objective functions in their multi-objective optimization model are not in conflict with each other and have the same trend. Therefore, they applied single-objective evolutionary algorithms to solve their optimization problem. In this study we develop a comprehensive multi-objective task scheduling optimization model to optimize cloud utilization and QoS, with in the objective functions are in conflict with each other. Considering the supremacy of PSO for solving task scheduling optimization in cloud and grid environments [5-9], we develop an algorithm based on Multi-Objective Particle Swarm Optimization (MOPSO) to solve our model. The feasibility and the advantages of applying MOPSO for task scheduling in any distributed environments, has not been investigated previously in the literature. In the proposed model, the criteria of QoS, including response time and service cost, are considered to determine the optimization objectives for task distribution in cloud computing. To implement and evaluate the proposed optimization model, we extend Jswarm [10] to Multi-objective Jswarm and apply it in the Cloudsim toolkit [11] as the task scheduling algorithm. We analyze the implementation results and compare our method with three other methods to prove its efficiency. Simulation result shows that the proposed model significantly increases the QoS in comparison with previous works. In fact, the proposed model is able to determine good trade-off solutions that offer the best possible compromises among the optimization objectives, and help clouds providers to maintain the expected level of QoS, or to improve it after creating new optimal task distribution schema. The paper contributions can be summarized as:

1) Develop a new multi-objective task scheduling model to minimize both task execution and transferring time, and task execution cost
2) Develop a MOPSO-based algorithm to solve the proposed task scheduling model
3) Extend the Jswarm and Cloudsim packages to evaluate our model

The rest of this paper is organized as follows. In Section 2, related works for task scheduling optimization methods are described. In Section 3, we propose our multi-objective model for optimal task scheduling, followed by a developed MOPSO-based algorithm for solving the proposed multi-objective task scheduling model in Section 4. We evaluate our developed model and analyze the simulation results, in Section 5. Finally we present our conclusion and future works in Section 6.

2 Related Works for Task Scheduling Optimization

The task scheduling problem in distributed computing systems is an NP-hard optimization problem which plays an important role in optimizing cloud utilization. It also effects on QoS in the cloud environment by optimizing service cost and service response time. Song et al. [12] proposed a general task selection and allocation framework to be directly applicable in a dynamic collaboration environment and improve resource utilization for primary cloud provider. Their framework utilizes an adaptive filter to select tasks and a modified heuristic algorithm (Min-Min) to allocate tasks. A trade-off metric is developed as the optimization goal of the heuristic algorithm, so that it is able to manage and optimize the trade-off between the QoS of tasks and the utilization of resources. The authors considered four criteria for resource utilization in their approach, namely: resource requirement of CPU, memory, hard-disk and network bandwidth. In addition they considered two objectives: they tried to allocate tasks to a physical machine by maximizing the remaining CPU capacity, and maximizing the utilization of whole resources. Li et al. [13] applied another heuristic optimization approach to propose an algorithm called Normal Best-Oriented Ant Colony Optimization (NBOACO). They applied their experimental results in a simulation environment to prove that a better scheduling result with shorter total-task-finish time and mean-task finish time, and batter load balance can be achieved by their proposed algorithm in compared to the Ant Colony Optimization algorithm (ACO). To achieve better results in task scheduling, Li et al. [14] took resource allocation pattern into account and proposed a task and resource optimization mechanism. Their approach contains two online dynamic task scheduling algorithms: dynamic cloud list scheduling and dynamic cloud min-min scheduling. These algorithms were designed to schedule tasks for the IaaS cloud system with preemptive tasks and task priorities. They considered tasks map and task types (Advance Reservation or Benefit- effort), to determine tasks priorities. Their algorithms dynamically adjust the resource allocation based on updated actual task execution which can be calculated by applying the information about the resource status. This information is pulled from other clouds and aggregated by clouds' servers managers.

Zomaya et al. [3] and Xiao et al. [2] applied Genetic Algorithm (GA) to develop a load-balancing algorithm whereby optimal or near optimal task allocation can evolve during the operation of the parallel computing system. To enhance the accuracy of GA results for the task scheduling process, Tayal [15] purposed an optimized algorithm based on the fuzzy-genetic algorithm optimization which makes a scheduling decision by evaluating the entire group of tasks in the job queue. To adapt the GA operator's value (selection; crossover; mutation) during the run of the GA, they designed an algorithm for the fuzzy setting of GA parameters. They considered three parameters for the triangular function which are: (1) execution time, (2) work load and (3) objective function[1]. Juhnke et al. [4] proposed a multi-objective scheduling algorithm for cloud-based workflow applications by applying Pareto Archived Evolution Strategy which is a type of GA which are capable of dealing with multi-objective optimization problems. When the constituent workflow tasks in a cloud environment are geographically distributed – hosted by different cloud providers or data centers of the same provider – data transmission can be the main bottleneck. The multi-objective genetic algorithm therefore takes data dependencies between BPEL (Business Process Execution Language for Web Services) workflow steps into account and assigns them to cloud resources based on the two conflicting objectives of execution cost and execution time according to the preferences of the user, and provides additional resources when needed.

Task assignment has been found to be an NP-Complete problem, thus GA has been used for solving this problem. However, GA may not be the best method. Lei et al. [6] and Salman et al. [5] have illustrated that the particle swarm optimization (PSO) algorithm is able to obtain a better schedule than GA in grid computing and distributed systems. Not only is the solution quality of PSO algorithm better than GA in most of the test cases, it also runs faster than GA [7].

Considering that user applications may incur large data retrieval and execution costs. Chen and Tsai [16] suggested that the cost arising from data transfers between resources as well as execution costs, should be taken into account in the optimization of task scheduling. They therefore presented a Discrete Particle Swarm Optimization (DPSO) approach for tasks allocation. They proposed a meta-heuristic optimization approach based on PSO for finding the near optimal tasks allocation with reasonable time. The approach seeks to dynamically generate an optimal task allocation so that tasks can be completed in a minimal period of time while still utilizing resources in an efficient way. Similarly, Guo et al. [7] proposed a PSO algorithm which is based on a small position value rule to formulate a model for task scheduling that would minimize the overall time of execution and transmission. They compared and analyzed PSO with crossover, mutation and local search algorithms based on particle swarm. The experiment results demonstrate that the PSO algorithm converges and performs more quickly than the other two algorithms in a large scale. Hence the authors concluded that the PSO is more suitable for task scheduling in cloud computing. As an expansion of [16] and [7], Liu et al. [8] introduced several meta-heuristic adaptations to the particle swarm optimization algorithm to deal with the formulation of efficient schedules and presented the Variable Neighborhood Search Particle Swarm Optimization (VNPSO) algorithm as a method for solving the resulting scheduling problem. They formulated the scheduling problem for workflow applications with security constraints in distributed data-intensive computing environments and presented a novel security constraint model. They introduced VNPSO as an algorithm which can be applied in distributed data-intensive applications to meet specific requirements, including workflow constraints, security constraints, data retrieval/transfer, job

[1] Objective function represents the time that processor *i* will have finished the previously assigned jobs and E[t][i] is the predicted execution time that task *t* is processed on processor *i*.

interaction, minimum completion cost, flexibility and availability. The authors benchmarked the proposed algorithm with a multi-start particle swarm optimization and multi-start genetic algorithm. The empirical results illustrate that VNPSO is more feasible and effective than two other baselines. According to the outputs, VNPSO balances the global exploration and local exploitation for scheduling tasks very well.

Despite the efficiency of PSO-based single objective algorithms, they are not practical for solving a multi-objective task scheduling problems for minimizing both cost and time in a cloud environment, because these two objectives are in conflict with one another and there does not exist a single solution that simultaneously optimizes each of them. To confront with this drawback, MOPSO is applied in this paper to solve such problems and determine the best possible trade-off among the objectives while also providing higher QoS.

3 A Multi-Objective Model for the Optimal Task Scheduling Problem

We develop a multi-objective model for optimizing task scheduling considering three aspects of task scheduling optimization problem including: task execution time, task transferring time, and task execution cost. To determine this problem we combine and improve the methods which are proposed in [7, 8] for formulating task execution time. In this model, to minimize time consumption, not only total tasks execution time is minimized, but also minimize the maximum tasks execution time. Applying this method, the highest level of time consumption for task executing is also restricted. To formulate the multi-objective model, the following variables are defined:

$\mathrm{n} = $ The number of arrival tasks

$\mathrm{T} = \{\mathrm{t}_1, \mathrm{t}_2, \ldots, \mathrm{t}_\mathrm{n}\} = $ Set of arrival tasks

$\mathrm{N}_{\mathrm{PM}} = $ The number of Physical Machines (PMs) in cloud

$\mathrm{m} = $ The number of VMs

$\mathrm{VM}_\mathrm{j} = $ Virtual Machine j, $\mathrm{j} = \{1,2,\ldots,\mathrm{m}\}$

$$\mathrm{PM}_\mathrm{z} = \{\mathrm{k} | \mathrm{VM}_\mathrm{k} \epsilon \text{ z th PM}, \mathrm{z} \in \{1,2,\ldots,\mathrm{N}_{\mathrm{PM}}\}\}$$
$$= \text{The set of VMs which are located in zth PM}$$

$$\mathrm{SP}_\mathrm{p} = \{\mathrm{k} | \mathrm{VM}_\mathrm{k} \epsilon \text{ Pth Cloud provider}, \mathrm{P} \in \{1,2,\ldots,\mathrm{cp}\}\}$$
$$= \text{The set of VMs which are asigned to } P\text{th provider}$$

$\mathrm{B}_{\mathrm{ck}} = $ The bandwidth between center and VM_k

cp = Number of cloud providers

$\tilde{\mathrm{C}}_\mathrm{p} = $ Maximum capacity for provider p

$\mathrm{x}_{\mathrm{ik}} = 1$ if task i is assigned to VM_k and $\mathrm{x}_{\mathrm{ik}} = 0$, otherwise

$DE_{ik} = $ The amount of data that task i assigns to the VM_k

$VMm_k =$ The amount of memory of $\mathrm{VM_k}$

$VMc_k =$ The amount of capacity of $\mathrm{VM_k}$

$Pcost_j =$ The cost of one unit VM for jth provider (USD per hour)

$r_p =$ The total number of VMs supplied by provider k that have executed tasks in the period time pt

Applying these variables, we can calculate the following functions:

$$Texe_k = \sum_{i=1}^{n} x_{ik} * \frac{DE_{ik}}{VMm_k * VMc_k} \tag{1}$$

where $Texe_k$ denotes the task execution time on $\mathrm{VM_k}$. Using this, the total task execution time is calculated as:

$$Texe = \sum_{k=1}^{m} Texe_k \tag{2}$$

The total tasks transferring time is determined as:

$$Ttrans = \sum_{k=1}^{m}\sum_{i=1}^{n} x_{ik} * \frac{DE_{ik}}{B_{ck}} \tag{3}$$

The total task execution cost for providers (USD per hour) is:

$$Cexe = \sum_{p=1}^{cp} (Pcost_p * r_p * (\sum_{k \in SP_p} Texe_k)) \tag{4}$$

and r_p is determined using following equation:

$$r_p = \sum_{k \in SP_p} min(\sum_{i=1}^{n} x_{ik}, 1) \tag{5}$$

Problem:

$$min\, f(Time) = (Texe + Ttrans) + \left(max_{i=1}^{k} Texe_i\right) \tag{6}$$

$$min\, f(Cost) = Cexe \tag{7}$$

Subject to

$$\sum_{k=1}^{\mathrm{m}} x_{ik} = 1, \forall\, i = 1, \dots, n$$

$$x_{ik} \in \{0,1\}, \forall i = 1, \dots, n \,\&\, k = 1, \dots, m$$

$$0 \le r_p \le \tilde{C}_p \;, \forall p = 1,2, \dots, cp$$

4 MOPSO-Based Algorithm for Solving the Multi-Objective Task Scheduling Problem

In this section we first provide preliminary definition and explanation of MOPSO method. Then, we explain our proposed MOPSO-based algorithm that will be used to solve the proposed model in Section 3.

4.1 Multi-Objective Particle Swarm Optimization Method

Optimization problems that have more than one objective function are rather common in every field or area of knowledge. In such problems, the objective functions are normally in conflict with respect to each other, which means that there is no single solution for these problems. Instead, the aim is to find good trade-off solutions that represent the best possible compromises among the objectives [17]. A multi-objective optimization problem is of the form:

$$\mathrm{Min}\ \ \vec{\mathrm{F}}(\vec{\mathrm{x}}) \coloneqq [\mathrm{f}_1(\vec{\mathrm{x}}), \mathrm{f}_2(\vec{\mathrm{x}}), \dots, \mathrm{f}_\mathrm{k}(\vec{\mathrm{x}})] \tag{8}$$

where $\vec{X} = (x_1, x_2, \dots, x_k)$ is the vector of decision variables; $f_i: R^n \to R, i = 1, \dots, k$ are the objective functions. Let particle $\vec{X}_1 = (x_1, x_2, \dots, x_k)$ represent a solution to (1). A solution $\vec{X}_2$ dominates $\vec{X}_1$ if $f_j(\vec{X}_1) \geq f_j(\vec{X}_2)$ for all *j=1,..,k* and $f_j(\vec{X}_1) > f_j(\vec{X}_2)$ for at least one *j=1,…,k*. A feasible solution $\vec{X}_1$ is called Pareto optimal (non-dominated) if there is no other feasible solution $\vec{X}_2$ that dominates it. The set of all objective vectors $F(\vec{X}_1)$corresponding to the Pareto optimal solutions is called the Pareto front (P*). Thus, the goal is to determine the Pareto optimal set from the set F of all the decision variable vectors (particles) [18-21].

In PSO, particles are flown through hyper dimensional search space. Changes to the position of the particles within the search space are based on the social–psychological tendency of individuals to emulate the success of other individuals. The position of each particle is changed according to its own experience and that of its neighbors. Let $\vec{X}_i(t)$ denote the position of particle i, at iteration t. The position of $\vec{X}_i(t)$ is changed by adding a velocity $\vec{V}_i(t+1)$ to it as follows:

$$\vec{\mathrm{X}}_\mathrm{i}(\mathrm{t}+1) = \vec{\mathrm{X}}_\mathrm{i}(\mathrm{t}) + \vec{\mathrm{V}}_\mathrm{i}(\mathrm{t}+1) \tag{9}$$

The velocity vector reflects the socially exchanged information and, in general, is defined in the following way:

$$\vec{\mathrm{V}}_\mathrm{i}(\mathrm{t}+1) = \mathrm{W}\vec{\mathrm{V}}_\mathrm{i}(\mathrm{t}) + \mathrm{C}_1\mathrm{r}_1\left(\vec{\mathrm{x}}_{\mathrm{pbest}_\mathrm{i}} - \vec{\mathrm{X}}_\mathrm{i}(\mathrm{t})\right) + \mathrm{C}_2\mathrm{r}_2\left(\vec{\mathrm{x}}_{\mathrm{gbest}_\mathrm{i}} - \vec{\mathrm{X}}_\mathrm{i}(\mathrm{t})\right) \tag{10}$$

where C_1 is the cognitive learning factor and represents the attraction that a particle has towards its own success; C_2 is the social learning factor and represents the attraction that a particle has towards the success of the entire swarm; W is the inertia weight, which is employed to control the impact of the previous history of velocities on the current velocity of a given particle; $\vec{x}_{pbest_i}$ is the personal best position of the particle i; $\vec{x}_{pbest}$ is the position of the best particle of the entire swarm; and $r_1, r_2 \in [0,1]$ are random values [17]. In MOPSO all Pareto optimal solutions are stored in an archive and $\vec{x}_{gbest_i}$ is chosen from this archive.

4.2 MOPSO-Based Algorithm

We develop a MOPSO-based algorithm to solve the proposed multi-objective task scheduling problem presented in Section 3. MOPSO finds the optimal task scheduling pattern, minimizing task execution time, task transfer time, and task execution cost. In the task scheduling model, we have n tasks $\{t_1, t_2, \ldots, t_n\}$ that should be assigned to m VMs $\{vm_1, vm_2, \ldots, vm_m\}$ to be executed (Table1). All particle positions $\vec{X}_i = (x_1, x_2, \ldots, x_n)$ determined by MOPSO by applying Equations 9 and 10, are vectors with continuous values, but we need their corresponding discrete values to determine the number of chosen VM for executing tasks. Therefore, we convert the particles' continuous position values vector to discrete vectors $d(\vec{X}_i) = (d_1, d_2, \ldots, d_n)$ applying the Small Position Value (SPV) rule [7].

Table 1. Task scheduling pattern (task mapping)

Tasks	t_1	t_2	t_3	t_4	t_5	…	t_n
VM number = Particle position	vm_7	vm_4	vm_5	vm_7	vm_3	…	vm_m

Particle position in Table 1 is a possible solution $d(\vec{X}_i) = (d_1, d_2, \ldots, d_n) = (7, 4, 5, 7, 3, .., m)$ after converting the continuous position values to discrete. According to this possible solution VMs: vm_7, vm_4, vm_5, vm_7, …, and vm_m are chosen to execute t_1, t_2, t_3, t_4, …, and t_n respectively. Considering this fact, every particle in our MOPSO model has n dimensions to assign n tasks to m VMs, and this model has two fitness functions: (1) minimizing task execution and transferring time ($f(Time)$), (2) minimizing tasks execution cost: ($f(Cost)$). Every particle will be assessed considering these fitness functions and all Pareto optimal solutions stored in an archive. In this paper we assume:

$$QoS\,(\vec{X}_i) = \sum_{j=1}^{m} W_j f_j(\vec{X}_i), \{\forall\, \vec{X}_i \in \text{Archive}\} \tag{11}$$

where m is the number of objective functions and W_j is the preference weight for every objective function $(f_j(\vec{X}_i))$. We then rank Pareto optimal solutions (archive members) on the basis of the number of functions that they minimized, and the maximum value of QoS. Then $\vec{X}_{gbest_i}$ is randomly chosen from the top ten. The MOPSO-based algorithm is summarized as follow:

Step 1. Initialize population: determine random position and velocity for every particle in the swarm population

```
Step 2. Initialize archive: archive members are non-dominated solutions ($n$ dimen-
        sions particles whose position is a Pareto optimal solution)
Step 3. Convert continuous position values vector of $\vec{X_i}$ to discrete vector $d(\vec{X_i})$
        using SPV rule to determine allocated VM for every arrival task.
Step 4. Determine the value of $DE_{ik}, VMm_k, VMc_k, Pcost_j, r_p$ and $\mathrm{B}_{\mathrm{ck}}$ based on
        $d(\vec{X_i})$ to calculate the value of every fitness function.
Step 5. Evaluate population according to defined fitness functions:
        Step 5.1. Minimize tasks execution/transferring time (Equation (6))
        Step 5.2. Minimize tasks execution cost (Equation (7))
Step 6. Update the archive contents: delete dominated members from archive and
        store the Pareto optimal (non-dominated) solutions in the archive.
Step 7. Sort archive members based on the number of function that they minimized
        and the maximum value of $QoS\left(\vec{X}\right) = W_1 * f(Time) + W_2 * f(Cost)$
Step 8. Choose $\vec{X}_{gbest}$ from top 10 sorted members in the archive randomly
Step 9. Choose $\vec{X}_{pbest_i}$for every particle: If the current position of the particle domi-
        nates best position of the particle, use current position as new best position
        for the particle
Step 10. Compute inertia weight and learning factors
Step 11. Compute new velocity and new position of the particles based on MOPSO
         formulations (Equations (9) and (10))
Step 12. If maximum iteration is satisfied then
             Step 12.1. Output $d(\vec{X}_{gbest})$ position as the best task scheduling pattern
                        (task mapping))
         Else
             Step 12.2 Go to Step 3
```

5 Simulation Results

In this section, we aim to prove the efficiency of our multi-objective task scheduling method. We first describe the simulation environment. Then, we explain how the Jswarm and CloudSim packages are extended to implement the method, and finally the performance and evaluation section is presented.

5.1 Environment Description

We design the simulation by assuming that we have three PMs (data centers), five VMs, five cloud providers and ten arrival tasks (cloudlets). We assume every VM belongs to one provider. Data and information about VMs and tasks (cloudlets) are summarized in Tables 2 and 3:

Table 2. Properties of VMs

VM Id	MIPS	VM image size	VM memory (Ram)	Bandwidth	The number of CPUs	VMM name
0	256	10000	512	10000	4	Xen
1	300	1000	256	1000	1	Xen
2	256	1000	512	10000	2	Xen
3	256	1000	512	1000	1	Xen
4	256	100	256	10	1	Xen

Table 3. Properties of tasks

Task Id	Length	File Size	Output Size	The number of required CPUs
0	250000	300	300	1
1	25000	300	300	1
2	250000	300	300	1
3	25000	300	300	1
4	250000	300	300	1
5	250000	300	300	1
6	25000	300	300	1
7	250000	300	300	1
8	250000	300	300	1
9	25000	300	300	1

5.2 Implementation

To implement the proposed method, we extend Jswarm package [10] to MO-Jswarm by converting the PSO algorithm to MOPSO algorithm. To achieve this goal, we first change the evaluation method in Swarm class by adding four new functions to: determine non-dominated (Pareto optimal) solutions, insert non-dominated solutions in the archive, determine the dominated solutions in the archive, and update achieve. In the first function, for every iteration, the positions of the particles (possible solutions) are assessed considering all the fitness functions (objectives) and the non-dominated solutions are determined, then they are inserted in the archive by applying the second function. Non-dominated solutions in the archive are assessed by the third function to find dominated solutions in the archive, and in the fourth function, dominated solutions in the archive are deleted. The archive members are then sorted. We then change the velocity and position calculation methods in ParticleUpdate class. We also make Particle, Neighborhood, SwarmRepulsive, VariableUpdate classes compatible with the new multi-objective calculations. We then extend the Cloudsim toolkit [11] by applying MO-Jswarm as its task scheduling algorithm. The bindCloudletToVm() method in the DatacenterBrocker class of Cloudsim is responsible for assigning tasks to VMs, and the MO-Jswarm has the ability to determine the optimal tasks arrangement among VMs according to the MOPSO algorithm.

The objective functions in the proposed multi-objective task scheduling model are applied as the fitness functions in MO-Jswarm. In our model, we have 20 particles and the optimal results are obtained after 2000th iteration of the MOPSO algorithm in

MO-Jswarm. Cloudsim allocates tasks to VMs in an optimal way based on the results of the developed MOPSO-based algorithm in MO-Jswarm.

5.3 Evaluation

To evaluate the proposed method, we firs perform the simulation under the environment that we define in Section 5.1.The output results are illustrated in Table 5. As can be seen from the results, cloudlets (tasks) 1, 2, 4, 7, 8 and 9 are assigned to VM_0, and cloudlets 0, 3, 5 and 6 are allocated to VM_2.

Table 4. Cloudsim outputs for the proposed model using MO-Jswarm

Tasks	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
VM numbers = Particle position	vm_2	vm_0	vm_0	vm_2	vm_0	vm_2	vm_2	vm_0	vm_0	vm_5

The graphs for task execution/transferring time ($f(Time)$), and task execution cost ($f(Cost)$) using our MOPSO-based algorithm are illustrated in Fig. 1. As can be seen, $f(Time)$ fluctuates from 420 to 60 seconds, and $f(Cost)$ decreases from 37 to around 14 USD per hour, within 2000 iterations. $f(Time)$ decreases while $f(Cost)$ rises dramatically in interval [140, 260] iteration. This shows that some solutions (task scheduling pattern) that minimize one objective can maximize another objective. In the obtained optimal solution at iteration 2000, $f(Time)$ and $f(Cost)$ are equal to 60 seconds and 14 USD respectively.

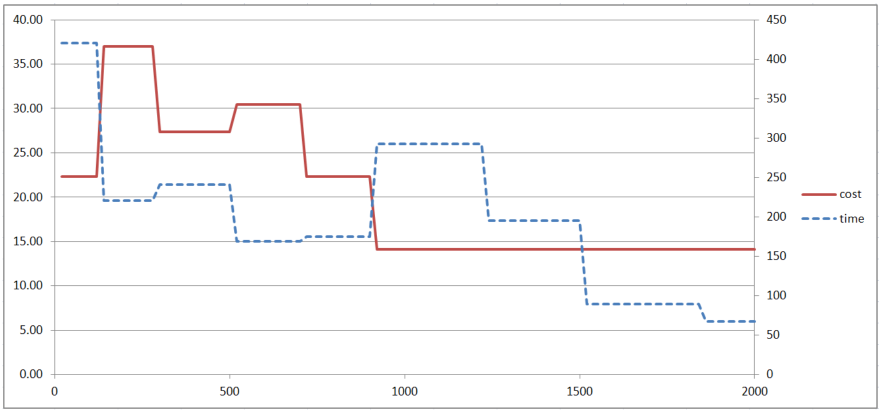

Fig. 1. The value of objective functions: Task execution/transferring time and task execution cost

We compare our proposed multi-objective method of solving task scheduling problem with conflicting objective with three different methods. The defined methods are explained as follows:

- Method 1: in which a single-objective optimization model is applied to minimize tasks execution/transferring time with objective function $f(Time)$. The corresponding value of $f(Cost)$ is simply calculated using the optimal solution of the model in Equation 7.
- Method 2: in which a single-objective optimization model is applied to minimize tasks execution cost with objective function $f(Cost)$. The corresponding value of $f(Time)$ is simply calculated using the optimal solution of the model in Equation 6.
- Method 3: in which a single-objective optimization model is applied to minimize $f(Time)$ and $f(Cost)$, and weighted aggregation of these objectives is considered as the single-objective of the model:

$$W_1 * f(Cost) + W_2 * f(Time) \tag{12}$$

The PSO algorithm in Jswarm package is used to solve the optimization models in these three methods. In the third comparison, we compare our method with the optimization method proposed by [7] which has two objectives: (1) execution time and (2) execution cost and is solved by PSO-based single-objective algorithms. In this model, the weighted aggregation of the objectives is considered as a single optimization objective, which means the optimal solutions for the single-objective optimization problem are Pareto optimal solutions to the multi-objective optimization problem, and the conflict between the objectives is neglected. Based on the Cloudsim output results, the optimal solutions (best particle position) resulted from every method are illustrated in Table 5.

Table 5. Cloudsim outputs for three methods using Jswarm.

Tasks	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
Optimal solution in Method 1	vm_1	vm_3	vm_0	vm_1	vm_0	vm_2	vm_1	vm_3	vm_0	vm_1
Optimal solution in Method 2	vm_0	vm_4	vm_0	vm_4	vm_0	vm_0	vm_0	vm_0	vm_0	vm_0
Optimal solution in Method 3	vm_0	vm_0	vm_0	vm_0	vm_0	vm_0	vm_0	vm_0	vm_0	vm_0

According to the first method, the optimal tasks execution/transferring time is 32.6. We calculate the corresponding tasks execution cost which is equal to 54.36 applying the optimal solution in $f(Cost)$. In the second method, the optimal task execution cost is 12.21, and the corresponding tasks execution/transferring time applying the optimal solution in $f(Time)$ is equal to 6335. In third method, the optimal task execution/transferring time and tasks execution cost is equal to -3235.66 applying weighted aggregation of the objectives. To estimate the QoS which results from every method, the following equation is utilized by assuming the same preference weight (-0.5) for both cost and time:

$$QoS\left(\vec{X}\right) = (-0.5) * f(Time) + (-0.5) * f(Cost) \tag{4}$$

The optimal results of all methods are summarized in Table 6.

Table 6. Comparison results

User preference weight Cost weight = Time weight = -0.5	Task execution and transferring time (Seconds)	Task execution cost (USD)	Estimated QoS
PSO in Method 1	32.6	54.36	-43.48
PSO in Method 2	6335	12.21	-3173.6
PSO in Method 3	-3235.66		-3235.66
MOPSO in the proposed Method	60	15.00	-37.5

As can be seen from the comparison results in Table 6, the estimated QoS in Methods 1, 2 and 3 are -43.48, -3173.6 and -3235.66 respectively. The estimated QoS in our proposed method is equal to -37.5, that is significantly increased in contrast to the QoS obtained in Methods 2 and 3. It is also higher than the QoS resulted from Method 1. Although, $f(Time)$ in Method 1 (i.e. 32.6) and $f(Cost)$ in Method 2 (i.e. 12.21) have their minimum values, they did not result the highest QoS. In addition, the QoS has its lowest value in Method 3. In this case, the approaches in Methods 1 and 2 are even better than the multi-objective optimizing approach in Method 3.

As the result, the proposed method determines an optimal trade-off solution for the multi-objective task scheduling problem with objective functions that are in conflict with one another, and determines the best possible compromises among the objectives, thereby significantly increasing the QoS.

6 Conclusion and Future Works

In this paper we propose a multi-objective model for optimizing task scheduling that considers three aspects of the task scheduling optimization problem: task execution time, task transferring time, and task execution cost. We also design a MOPSO algorithm to solve the proposed task scheduling model. To evaluate our proposed method we first extend the Jswarm package, change it to a multi-objective PSO and convert it to MO-Jswarm. Then we extend the Cloudsim toolkit by applying MO-Jswarm as its task scheduling algorithm. The bindCloudletToVm() method in the DatacenterBrocker class of Cloudsim is responsible for assigning tasks to VMs, and the MO-Jswarm has the ability to determine the optimal task arrangement among VMs according to the MOPSO algorithm. The experimental results in the simulation environment show that the proposed optimization model has the ability to determine the best trade-off solutions compared to recent task scheduling approaches; it provides the best possible coincidences among the objectives and achieves the highest QoS. The decision support system that we have designed, implemented and validated in our work, could be made part of the virtualization layer. This would enable data center operators to make use of this system for load balancing.

In our future work, we will implement the proposed model in a real cloud environment. We also will consider task priorities and types in our optimization model. In addition, we will extend our model to minimize energy consumption by not choosing VMs on idle PMs as new hosts for executing tasks. Furthermore, we will compare the MOPSO with other multi-objective evolutionary algorithms such as MOGA, to find the most efficient and reliable algorithm that not only determines the optimal task scheduling pattern but also obtains the solution in the shortest possible time.

Acknowledgment. The first author would like to thank Dr. Vahid Behbood for his help and comments.

References

1. Celesti, A., Fazio, M., Villari, M., Puliafito, A.: Virtual machine provisioning through satellite communications in federated cloud environments. Future Generation Computer Systems 28(1), 85–93 (2012)
2. Xiao, Z.J., Chang, H.Y., Yi, Y.: An optimization m ethod of w orkflow dynamic scheduling based on heuristic GA. Computer Science 34(2) (2007)
3. Zomaya, A.Y., Yee-Hwei, T.: Observations on using genetic algorithms for dynamic load-balancing. IEEE Transactions on Parallel and Distributed Systems 12(9), 899–911 (2001)
4. Juhnke, E., Dörnemann, T., Böck, D., Freisleben, B.: Multi-objective scheduling of bpel workflows in geographically distributed clouds. In: 4th IEEE International Conference on Cloud Computing, pp. 412–419 (2011)
5. Salman, A., Ahmad, I., Al-Madani, S.: Particle swarm optimization for task assignment problem. Microprocessors and Microsystems 26(8), 363–371 (2002)
6. Lei, Z., Yuehui, C., Runyuan, S., Shan, J., Bo, Y.: A task scheduling algorithm based on pso for grid computing. International Journal of Computational Intelligence Research 4(1), 37–43 (2008)
7. Guo, L., Zhao, S., Shen, S., Jiang, C.: Task scheduling optimization in cloud computing based on heuristic algorithm. Journal of Networks 7(3), 547–553 (2012)
8. Liu, H., Abraham, A., Snášel, V., McLoone, S.: Swarm scheduling approaches for workflow applications with security constraints in distributed data-intensive computing environments. Information Sciences 192(0), 228–243 (2012)
9. Behbood, V., Lu, J., Zhang, G.: Fuzzy bridged refinement domain adaptation: Long-term bank failure prediction. International Journal of Computational Intelligence and Applications 12(01) (2013), doi:10.1142/S146902681350003X
10. Cingolani, P.: `http://jswarm-pso.sourceforge.net/`
11. Calheiros, R.N., Ranjan, R., De Rose, C.A.F., Buyya, R.: Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services. Arxiv preprint arXiv:0903.2525 (2009)
12. Song, B., Hassan, M.M., Huh, E.: A novel heuristic-based task selection and allocation framework in dynamic collaborative cloud service platform. In: 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp. 360–367 (2010)
13. Li, J., Peng, J., Cao, X., Li, H.-y.: A task scheduling algorithm based on improved ant colony optimization in cloud computing environment. Energy Procedia 13, 6833–6840 (2011)

14. Li, J., Qiu, M., Ming, Z., Quan, G., Qin, X., Gu, Z.: Online optimization for scheduling preemptable tasks on iaas cloud systems. Journal of Parallel and Distributed Computing 72(5), 666–677 (2012)
15. Tayal, S.: Tasks scheduling optimization for the cloud computing systems. International Journal of Advanced Engineering Sciences and Technologies 5(2), 111–115 (2011)
16. Chen, Y.M., Tsai, S.Y.: Optimal provisioning of resource in a cloud service. IJCSI International Journal of Computer Science Issues 7(6), 1694–1814 (2010)
17. Mahmoodabadi, M.J., Bagheri, A., Nariman-zadeh, N., Jamali, A.: A new optimization algorithm based on a combination of particle swarm optimization, convergence and divergence operators for single-objective and multi-objective problems. Engineering Optimization, 1–20 (2012)
18. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
19. Alves, M.J.: Using MOPSO to solve multiobjective bilevel linear problems. In: Dorigo, M., Birattari, M., Blum, C., Christensen, A.L., Engelbrecht, A.P., Groß, R., Stützle, T. (eds.) ANTS 2012. LNCS, vol. 7461, pp. 332–339. Springer, Heidelberg (2012)
20. Gao, Y., Zhang, G., Lu, J., Wee, H.-M.: Particle swarm optimization for bi-level pricing problems in supply chains. Journal of Global Optimization 51(2), 245–254 (2011)
21. Lu, J., Zhang, G., Ruan, D.: Multi-objective group decision making: Methods, software and applications with fuzzy set techniques. Imperial College Press, London (2007)