Vol. 02, No. 1,(2018),15-26, **ISSN:2581-3242**

IJMLNCE JOURNAL

International Journal of Machine Learning and Networked Collaborative Engineering

Journal Homepage: http://www.mlnce.net/home/index.html

DOI : https://doi.org/10.30991/IJMLNCE.2018v02i01.003

A Scheduling Algorithm based on PSO Heuristic in Cloud Computing

[a]Nguyen Hoang Ha*, [b]Nguyen Hoang Nguyen

[a,b]*Hue University of Sciences, Vietnam*

[a]*nhha76@gmail.com, https://orcid.org/0000-0001-6986-4104*

[b]*nhnguyen.khdthue@gmail.com, https://orcid.org/0000-0002-2997-9393*

Abstract

The goal of the SaaS provider is the most protable; the user's goal is to meet requirements as quickly as possible but still within budget and deadline. The algorithm's aim gives the schedule to satisfy the objectives of the agents, this is a very difficult problem. This article studies heuristic PSO (Particle Swarm Optimization) and model of components in the cloud computing to propose a model of PaaS providers; admission control algorithm and scheduling for the user's requirements towards multi-objective optimization of time.The schedule given by the algorithm in order to: (1) optimizing the time for the user, (2) providing the greatest benefits for SaaS providers, (3) satisfying for the constraints of QoS (Quality of Service) of the user. The result of the algorithm is installed and compared with other algorithms on CloudSim.

Keywords

Quick Response code,

Shopping Mall,

Smart Trolley

1. Introduction

Cloud computing is a distributed computing model for large scale, it provides services to users by employing resources (hardware, software, storage resources, ...) via the internet[18].It provides services to users by employing resources via internet. Users may employ the various resources through their requirements and pay as they use. When users send requests together with the constraints as todeadline, budget, workload, arrival time, ... to SaaS providers, SaaS providers use PaaS to admission control, then conduct scheduling requirements as Figure 1. PaaS provider searches for suitable resources on IaaS to logical mapping to user requirements.

The problem of scheduling on cloud computing differs from the multi-processor scheduling problem in some features:

a) Can perform parallel tasks, tasks can always be completed on time.

b) Resources on cloud computing are provided by many vendors. So, at every moment, users can find the right resources.

* Corresponding author

Nguyen Hoang Ha

Email: nhha76@gmail.com

c) Each request can reuse the expiration period of the other request

d) Each user rents a virtual machine for a period of time. If the user does not use up the amount of time, other users can take advantage of it.

Generally, Scheduling problem on cloud computing with parameters such as workload, budget, deadline, ... is an NP-complete problem [1]. Therefore, one can not use the exhaustive method to find the optimal solution because the search time is too large. Nowadays, researchers often use heuristic methods to find near-optimal solutions such as: greedy method EDF ((Earliest Deadline First)) [4][19], ACO (Ant Colony Optimization) method [2] [20], techniques optimized fuzzy bees [3], ...

On cloud computing, users rent resources on datacenters and pay for cost over time. . Therefore, the scheduling problem based on constraint Quality of Service (QoS) is often used. In this case, the user's parameters such as arrival time, budget, deadline, etc., are given priority when scheduling. Jzau-Sheng Lin and colleagues [5] has proposed a scheduling model for cloud computing with the goal of bringing the highest profit for the SaaS provider but only interested in the deadlines and budget of the requirements. The study [6] focuses on the scheduling requirements for power savings on IaaS provider. Ramkumar N [7] has proposed a real-time scheduling algorithm but focused to solve scheduling tasks quickly satisfy most of the requirements deadline regardless of cost and its budget. Swarupa Irugurala, Dr.K.Shahu Chatrapati [8] has proposed a scheduling algorithm but is concerned only with the cost of virtual machine initialization and the cost of using the virtual machine. Gomathi B. and Medhat A. Tawfeek base on PSO (Particle Swarm Optimization), ACO to propose scheduling algorithms but only concerned with the execution time of the system.

Professor Rajkumar Buyya and colleagues [20], [21]have proposed scheduling algorithm works on cloud computing foward the system performance. The authors have combined with parallel processing, heuristics and PSO to handle the large volume of work. The proposed study and build the model of optimize non-linear workflows, minimize retrieve data for workloads which are requiring a large volume of data on cloud computing. These studies use a heuristic algorithm to provide PSO algorithm for faster convergence and computing time is less than the existing algorithm. However, these algorithms applies only to classes of problems with large volumes, the data transmission time is larger than the calculation time, not interested in the user's QoS constraints.

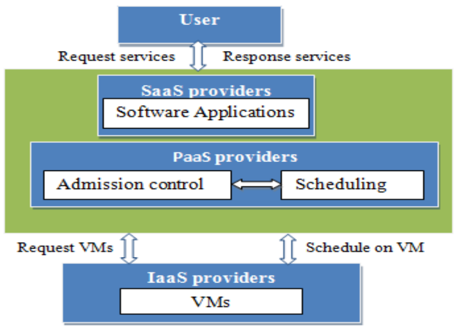

Fig.1: IaaS, Paas and SaaS Providers

The studie [22] are using the heuristic of the pack to provide optimal schedule of implementation time, not interested in the user's QoS constraints. To optimize the execution time, the studies [18], [23] are using the heuristic ACO, GA to provide a schedule with the aim of giving time to complete the smallest system but still satisfy the user's QoS constraints. The present study focuses on time constraints, not concerned about the cost of the system.

In this paper, the virtual machines on the data center are used to map the requirements aiming at making real-time implementation of the system minimal but still meeting deadlines and budgets requirements. SAPSO and MPSOalgorithmsare proposed.The goal of these algorithms is: (1) optimizing the time for the user, (2) providing the greatest benefits for SaaS providers, (3) satisfying for the constraints of QoS (Quality of Service) of the user

The paper includes: section 2, system model;section 3, two algorithms are proposed: SAPSO and MPSO then simulating, evaluating between the algorithms and conclusions [section 4].

2. System Model

Cloud computing consists of four main components [11]: User, Software as a Service (SaaS) providers, Platform as a Service (PaaS) providers and Infrastructure as a Service (IaaS) providers. Users send requests to use the attached software to their QoS requirements to the SaaS provider. PaaS providers analyze the QoS parameters, thereby deciding whether to accept or reject user requests. If the request is accepted, the scheduler will search for resources on the IaaS provider such as Figure 1.

2.1. User model

Users send *N* service requests: $T = \{ t_1, t_2, \dots, t_N \}$ to SaaS vendors, $\forall t_i \in T$ is a 7-tuble$< a_i, d_i, b_i, \alpha_i, w_i, in_i, out_i >$Where

a_i: arrival time of request.

d_i (deadline): longest time users need to wait for the results.

b_i (budget): the cost of the user paid to the provider.

α_i (penalty rate): penalty rate if the supplier fails to deliver on time

w_i (workload): workload of the request.

in_i and out_i: size of input and output file

2.2. SaaS providers model

SaaS providers lease resources from the IaaS provider and its leasing software as services for users. The goal of SaaS provider is how to minimize the cost of using resources from the IaaS providers to bring the highest profit to them.

2.3. IaaS provider model

Let $X = \{ 1, 2, \dots, Y \}$ is set suppliers, each IaaS provider $x \in X$ provides M_x virtual machines: $VM_x = \{ vm_{1x}, vm_{2x}, \dots, vm_{M_x x} \}$ for SaaS providers, $\forall vm_{jx} \in VM_x$ is a5-tuble $< t_{jx}, p_{jx}, s_{jx}, Dtp_{jx}, Dts_{jx} >$includes [11]:

Initialization time t_{jx}: how long it takes to deploy one virtual machine.

Price p_{jx}: pricing depends on per hour that SaaS must pay for IaaS using VMs

s_{jx}: processor speed of virtual machines

Dtp_{jx}: the price SaaS provider must pay to transport data from resource provider client.

Dts_{jx}: data transporting speed depends on network performance.

2.4. PaaS provider model

PaaS provider model is is a 3-tuble $<R \mid npmtn \mid T_{min}>$. All IaaSprovider are not interdependent, can be executed in parallel and are denoted by *R*. We set schedule for *N* requests independently not to follow any particular order of priority (non-preemptive) on *Y* providers. These requirements are denoted by*npmtn*. The aim is to find the minimum total completed time for requirements but still satisfying deadline and budget of the requirements, it means that T_{min} must be found.

Call$T_{ijx}$is the time to process the request$t_i \in T$ on $vm_{jx} \in VM_x$. Based on user model and IaaS providers model (session 2.1, 2.3), then time T_{ijx} is calculated as follows:

$$T_{ijx} = CT_{ijx} + DT_{ijx} + TI_{ijx} + \beta_{ijx} (1)$$

Where- CT_{ijx}: time to process the requests depends on the workload $w_i$of the request t_i and the speed$s_{jx}$of virtual machinevm_{jx}, $CT_{ijx}$is calculated as follows:

$$CT_{ijx} = \frac{w_i}{s_{jx}} \quad (2)$$

- DT_{ijx}: time to transfer data including time to send data to and retrieve data from resource providers depend on the size of the input file size $in_i$and output file size $out_i$of the request t_i, data transfer speed $DT_{ijx}$of virtual machine vm_{jx},DT_{ijx} is calculated as follows:

$$DT_{ijx} = \frac{in_i + out_i}{Dts_{jx}} \quad (3)$$

- TI_{ijx}: virtual machine initialization time is given.

-β_{ijx}: exceeded time deadline of the $t_i \in T$on virtual machine $vm_{jx} \in VM_x$.

Call $CT_{ijx}$the cost of executing the request$t_i \in T$ on the $vm_{jx} \in VM_x$. Based on user model and IaaS providers model (session 2.1, 2.3), then CT_{ijx}include cos$CT_{ijx} = CP_{ijx} + CDT_{ijx} + CI_{ijx} + CR_{ijx}$

Where the cost of processing request (CP_{ijx}) depends on the price ofvirtual machinevm_{jx} (p_{jx}), speed of virtual machine vm_{jx} (s_{jx}) and workload of request t_i (w_i). CP_{ijx} is calculated as follows:

$$CP_{ijx} = p_{jx} \times \frac{w_i}{s_{jx}} \quad (4)$$

The cost of data transmission (CDT_{ijx}) includes the cost of sending data to and retrieve data from resource providers depend on the size of the input file$in_i$and output file $out_i$of the request$t_i \in T$, data transfer speeds(Dts_{jx})and prices to transfer data Dtp_{jx} from the virtual machine $vm_{jx} \in VM_x$ to user computers. CDT_{ijx} is calculated as follows:

$$CDT_{ijx} = Dtp_{jx} \times \frac{in_i + out_i}{Dts_x} \quad (5)$$

Costs initialized(CI_{ijx})of virtual machine depends on the initialization time $t_{ijx}$and pricep_{ijx} of the request $t_i \in T$ on virtual machinevm_{jx}. CI_{ijx} is calculated as follows:

$$CI_{ijx} = t_{ijx} \times p_{ijx} \quad (6)$$

Costs of the SaaS provider must be returned to the users if not meeting the deadline (CR_{ijx}), depending on the penalty rate (α_i) and exceeded time deadline of request t_i on on virtual machine vm_{jx} (β_{ijx}). CR_{ijx} is calculated as follows:

$$CR_{ijx} = \alpha_i \times \beta_{ijx} \quad (7)$$

The objective of the article is find the virtual machine in the IaaS provider to minimize the time of completion, such as:

$$\sum_{i=1}^{N}(f_{time}(t_i)) \rightarrow \min(6)$$

where

$$f_{time}(t_i) = \min_{x=1...Y,j=1...M_x}\{T_{ijx}\} \quad (9)$$

The cost of request $t_i \in T$ is less than budget that is:

$$C_{ijx} < b_i(10)$$

The execution time (T_{ijx}) of request $t_i \in T$must less than the deadline itself:

$$T_{ijx} \leq d_i + \beta_{ijx} \quad (11)$$

Thus, To achieve the objective of article (8), two constraints (10) and (11) must be satisfied .

3. Construction of Algorithm

3.1. SAPSO algorithm

Based on the experience of swarm, Kennedy and Eberhart has proposedheuristic PSO [12]. heuristic PSO simulates the social behavior of birds or fish searching for food. Heuristic PSO is a common algorithm framework for solving optimal problems. So to apply PSO to the scheduling problem we need to define the parameters: velocity, *Pbest* and *Gbest* [12], [13],[14]. Bergh F. V. D and Clerc, M based on current velocity and distance from *Pbest* to *Gbest* to change position and speed as follows [14], [15]:

$$v_x^{j+1} = K \times (\omega \times v_x^j + c_1 \times r_1 \times \left(Pbest_x - pos_x^{j+1}\right) + c_2 \times r_2 \times \left(Gbest - post_x^{j+1}\right)) \quad (12)$$

where

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4 \times \varphi}\right|}, \varphi = c_1 \times z_1 + c_2 \times z_2, \varphi > 4$$

$$pos_x^{j+1} = pos_x^j + v_x^{j+1} \quad (13)$$

Therein:pos_x^j: position of particle *x* in dimension *j*;pos_x^{j+1}: position of particle *x* in dimension *j+1*; ω: inertia weight (value: 0.1 … 0.9);c_1,c_2: acceleration coefficient (value: 1..2); r_1, r_2: random number between 0 and 1;v_x^j: velocity of particle *x* in dimension *j*; v_x^{j+1}: velocity of particle *x* in dimension *j+1*; ; $Pbest_x$: local best position of particle *x*; *Gbest* : global best position of the entire swarm.

Let $P = \{1, 2, ..., Y\}$ is set consisting of *Y* particles. Each particle $x \in P$ willloop N times to searches food inM_x dimension space and is determined:

$$POS_{ix} = \{pos_{ix}^1, \quad pos_{ix}^2, \quad ..., \quad pos_{ix}^{M_x}\} \quad (14)$$

$$V_{ix} = \{v_{ix}^1, \quad v_{ix}^2, \quad ..., v_{ix}^{M_x}\} \quad (15)$$

where in pos_{ix}^j is the position at the loop *i* (i=1 ... N) in dimension *j* (*j*= *1* ... M_x) of the particle *x;* v_{ix}^j is the velocity at the loop *i* in dimension *j*of the particle *x*.

In section 2, we have *Y* providersand*N* requests of users, each particle is equivalent with each providers, each particle $x \in P$loop$M_x$times to find resources for the request $t_i \in T$, The value of $pos_{ix}^j$is virtual machine *j* of provider *x*, which is mapped to request t_i. This value is taken from 1 to M_x, and the value of $v_{ix}^j$is taken from $-M_x$to $M_x$randomly, wherein $M_x$is the number of virtual machines in each provider *x*. To achieve the goal of the problem as in formula (8), We need to determine the fitness function for particle *x* to select virtual machine *j* for request $t_i \in T$. Fitness function is calculated as follows:

$$f\left(pos_{ix}^j\right) = \frac{1}{T_{ijx}} \quad (16)$$

where $T_{ijx}$is calculated as in formula (1), (2), (3) (section 2).

After calculating the fitness function for particle *x*, We can determine the local optimal position of particle *x* is as follows:

$$pb_{ix}^{j+1} = \begin{cases} pb_{ix}^{j+1}, & if\ f\left(pos_{ix}^{j+1}\right) \geq f\left(pos_{ix}^j\right)\ and\ C_{ijx} \leq b_i\ and\ T_{ijx} \leq d_i + \beta_{ijx} \\ pb_{ix}^j, & Other\ cases \end{cases} \quad (17)$$

where $C_{ijx} \leq b_i\ and\ T_{ijx} \leq d_i + \beta_{ijx}$as constraints (10) and (11) in session 2.4

Local best position of particle *x* (*Pbest*$_x$) and global best position (*Gbest*) are determined as:

$$Pbest_x = \min_{x=1\ldots Y,\ j=1\ldots M\_x} \left(pb_{ix}^j\right) \quad (18)$$

$$Gbest = \max_{x=1\ldots Y} \left(Pbest_x\right) \quad (19)$$

SAPSO algorithm

Input:

$T = \{t_1, t_2, \ldots, t_N\}$, $\forall t_i \in T$is a 7-tuble $< a_i, d_i, b_i, \alpha_i, w_i, in_i, out_i >$;

$X = \{1, 2, \ldots, Y\}$, $\forall x \in X$, $VM_x = \{vm_{1x}, vm_{2x}, \ldots, vm_{M_x x}\}$, $vm_{jx} \in VM_x$ is a 5-tuble $< t_{jx}, p_{jx}, s_{jx}, Dtp_{jx}, Dts_{jx} >$

Output:

$S = \{t_i \rightarrow vm_{jx}, t_i \in T, vm_{jx} \in VM_x\}$ *where* $t_i \rightarrow vm_{jx}$:each request $t_i \in T$ is mapped to the $vm_{jx} \in VM_x$

Description algorithm:

```
1. Initialization:
    $pos_{ix}^j$ is a random number from 1 to $M_x$;
    $v_{ix}^j$ is a random number from $-M_x$x to Mx;
    $Pbest_x = pos_{1x}^1$;
    $Gbest_x = \max_{x=1\ldots Y} Pbest_x$
2. Create Y thread that executes concurrently:
    $TH := \{th_1, th_2, \ldots, th_Y\}, th_x \in TH$ search resources in $x \in X$.
3.FOR EACH$t_i \in T$DO
```

```
4.    FOR EACH $th_x \in TH$ DO{
5.        Calculate fitness function as formula (16):
              $f(pos_{ix}^{j}) = \frac{1}{T_{ijx}}$
6.        Calculate $Pbest_j$ as formula (18):
              $Pbest_x = \min_{x=1 ... Y,\ j=1 ... M\_x} (pb_{ix}^{j})$
7.        Calculate $Gbest_i$ as formula (19):
              $Gbest = \max_{x=1 ... Y} (Pbest_x)$
8.    END FOR
9.    Based on $Gbest_i$, search the virtual machine which has cost less than budget: $cost < b_i$ and
      processing time $\le d_i + \beta_{ijx}$ , if found then $S := S \cup \{t_i \to vm_{ijx}\}$ else the request has
      been reject;
10.   Update the position and velocity of the particles as formula (12),(13):
          $v_x^{j+1} = K \times (\omega \times v_x^j + c_1 \times r_1 \times (Pbest_x - pos_x^{j+1}) + c_2 \times r_2 \times (Gbest - post_x^{j+1}))$
          $pos_x^{j+1} = pos_x^j + v_x^{j+1}$
11.END FOR
```

3.2. MPSO algorithm

Provider resources are rented hourly, the user has to pay the hourly fee. If user do not use all their one-hour of hiring time, user also have to pay for a whole hour. At each point, each vendor has a lot of users rent resources. So if the user does not use up the amount of time, other users can take advantage of it. We call T_i is the set of requirements, this set includes requirements, which can take advantage time of request t_i. The requirements in T_i can share a virtual machine. The set T_i is defined as follows:

$$T_i = \{t_l | d_l \ge d_i \text{ and } a_l < d_i\} \qquad (20)$$

Call t_{iljx} is effective time to calculate the request t_l after completing the request of $t_i \in T$ on $vm_{jx} \in VM_x$. The value of t_{iljx} depends on the workload, arrival time, speed of virtual machines and deadline of request $t_i \in T$ and request $t_j \in T$. t_{iljx} is calculated as follows:

$$t_{iljx} = \begin{cases} \min(D - U_{il},\ d_l - a_l) & \text{if } a_l - a_i \ge \frac{w_i}{s_{jx}} \\ D - U_{il} & \text{if } a_l - a_i < \frac{w_i}{s_{jx}} \text{ and } d_l - a_i \ge D \\ d_l - (a_i + U_{il}) & \text{if } a_l - a_i < \frac{w_i}{s_{jx}} \text{ and } d_l - a_i < D \end{cases} \qquad (21)$$

where

$U_{il} = \frac{w_l}{s_{jx}} + \max(a_l - d_i, 0)$, s_{jx} the speed of $vm_{jx} \in VM_x$ is mapped to request $t_j \in T$.

MPSO algorithm

Input:

S is output of SAPSO algorithms. Mapping in S contains the requests, these requests are accepted by the supplier

Output:

$ST = \{ t_i \rightarrow vm_{jx}, t_i \in T, vm_{jx} \in VM_x \}$ *where* $t_i \rightarrow vm_{jx}$: each request $t_i \in T$ is mapped to the $vm_{jx} \in VM_x$

Description algorithm:

```
1. Sort all request in S accordingly to the provider, then all requests of the same  provider will be in the
       same group;
2. FOR EACH provider x in S DO
3.PUSH($t_i$);// Save $t_i$ into the stack, $t_i$ is the first request of the provider x
4.$ST := ST \cup \{ t_i \}$;; $S := S \setminus \{ t_i \}$;
5.FOR EACH request $t_i$ in the provider x DO
6.      $t_i$=POP(); // Take $t_i$ from stack
7.      Find $T_i$ and calculate $t_{iljx}$for the requests in $T_i$ as in formula (20), (21):
          $T_i = \{t_l | d_l \geq d_i \text{ and } a_l < d_i \}$
          $t_{iljx} = \begin{cases} \min(D - U_{il}, d_l - a_l) & \text{if } a_l - a_i \geq \frac{w_i}{s_{jx}} \\ D - U_{il} & \text{if } a_l - a_i < \frac{w_i}{s_{jx}} \text{ and } d_l - a_i \geq D \\ d_l - (a_i + U_{il}) & \text{if } a_l - a_i < \frac{w_i}{s_{jx}} \text{ and } d_l - a_i < D \end{cases}$
9.      Find max($t_{iljx}$),$t_l$ has the largest overlapping time as the next request;
10.     Base on max($t_{iljx}$)  to find $w_l$ reload all request status of $t_l$;
11.     PUSH($t_l$);
12.     $ST := ST \cup \{ t_l \}$;$S := S \setminus \{ t_l \}$;
13.END FOR
14.END FOR
15.Base on ST to produce the mapped schedule onto the request of resources;
```

3.3. Correctness of the algorithms

M.Clerc [15] added the rate of convergence K to update the velocity for particle and has proved the correctness of the PSO algorithm. This ensures the correctness of the algorithm SAPSO.

The input of the algorithm SAPSO is Y independent provider, so we can create the Y thread which performed concurrently, each thread$th_x \in TH$ is an particle which searched on the set virtual machine of the corresponding provider. This will reduce the complexity of the algorithm and optimize the time schedule.

Each particle (Thread) just focuses on mappings request $t_i \in T$ into $vm_{jx} \in VM_x$ based on the fitness function as formula (16). So, when the lower $T_{ijx}$is, the higher the fitness function is, the probability of request $t_i \in T$choose $vm_{jx} \in VM_x$ is more. Therefore, when the required mapping into the virtual machine has a low time to implement, it will make the total execution time of all systems are reduced.

The resource rental provider with time is D-minute, therefore if user do not use all their D-minute of hiring time, user also have to pay for a whole D-minute. MPSOalgorithm takes advantage of this request to process the next request. The goal of the MPSO algorithm is to reduce the cost of the system, bringing the most benefit to the SaaS provider

3.4. Simulation and evaluation of the algorithms

This article uses NetBean 7.1.1, JDK 6 and CloudSim 2.1 (Simulation tool on cloud computing) [16] to install algorithms. In CloudSim 2.1, we use 4 Datacenter, 10 physical hosts, 150 virtual machines. Other parameters (users and resource providers) on CloudSim 2.1 are defined as follows:

User service side: Workload is a random number from $8*10^4$ MI to 10^5MI, after determining the workload, we can determine the corresponding budget for the requirements. The arrival time is a random number from 1 to 500, deadline is a random number between (d_l,d_u) minutes and the different values of d_l and d_u are limited from 10 to 1500, deadline is random but must be greater than the arrival time. Other parameters of user service are taken as implicit in CloudSim.

On the IaaS provider's side: In simulation, we use four IaaS providers; each IaaS provider has a number of different virtual machines, each virtual machines has speed, costs, and different bandwidth. In simulation installation, the Vm class of CloudSim is inherited to create a virtual machines with the parameters of speed and cost are calculated as follows: speed is a random number from from 10^3 to $5*10^3$ MIPS corresponding with the costs which are real numbers is a random number from from 0.001 to 0.01, other parameters of the virtual machine as bandwidth, initialization time of virtual machine, data center, brokers, etc. ... are taken as implicit in CloudSim.

Results in simulation is the result of the 10 tests and the average results are obtained.

3.4.1. Analyze the total cost, time and profit as fixed requests.

Figures 2,3 and 4 show the total time, cost and total profit of four algorithms: EDF, SAPSO, MPSO and Sequential, using 150 VMs and 1000 requests. Simulation results show that the total execution time of algorithms SAPSO and MPSO are almost as same as Sequential algorithms (as Figure 2), but the cost of MPSO is always lower than EDF, SAPSO and Sequential algorithms. Because the SAPSO algorithm is responsible for admission control, accepting its satisfied deadline and budget requests, so after the SAPSO algorithm has done processing, the accepted requests are got with less time consuming, this requesting set is the input data of MPSO scheduling algorithm. MPSO scheduling algorithm continues taking advantage of the advanced time interval of requests onto IaaS resource provider, which lead to the total of cost reduction as Figure 3 and total of profit increase as Figure 4.

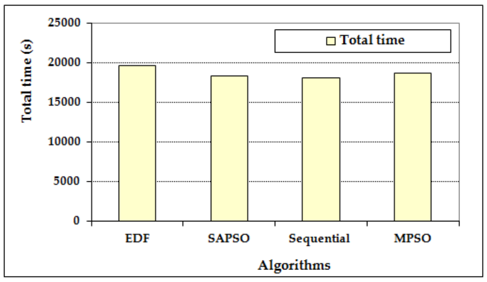

Fig 2. The total time of the algorithms when using 1000 requests.

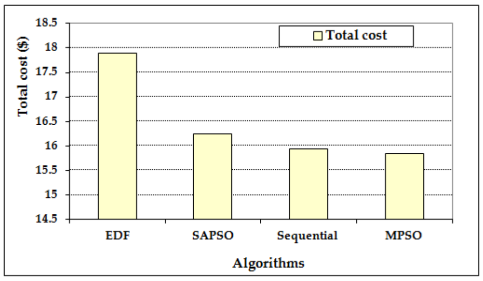

Fig 3. The total cost of the algorithms as fixed requests.

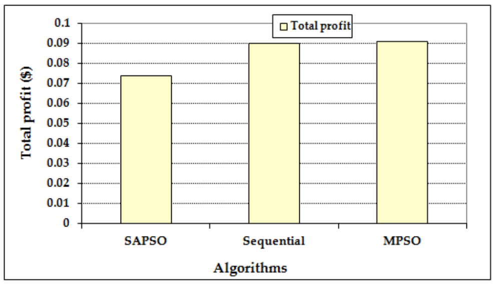

Fig 4. The total profit of the algorithms when using 1000 requests

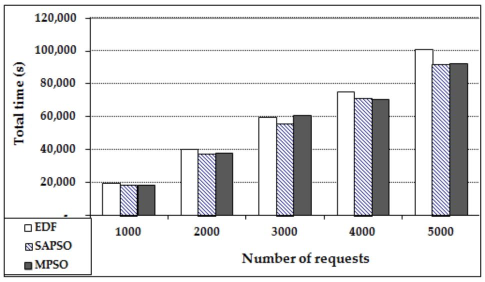

Fig 5. The total time of the algorithms when requests change.

Sequential algorithm does not take advantage of the time between requests, but uses exhaustive method to search the resource, so there will be many cases that the request can’t use all the rental time, this will make the cost of the sequential algorithm increase and takes a very long time to schedule. For EDF algorithms, thisalgorithms only focus the ratio used: (where Ci is the execution time,Tiis deadline, m is is the number of virtual machines) [17] to map the request to the resource, so EDF algorithm but not focus in the budget of request. The advantage of the EDF algorithm is that all requests are always completed before the deadline,Therefore, to consider for the total profit for supplier, the paper doesn’t consider to EDF algorithm.

3.4.2. Analyze total cost, total execution time and total profit of requests as a fixed number of virtual machines and changing number of requests.

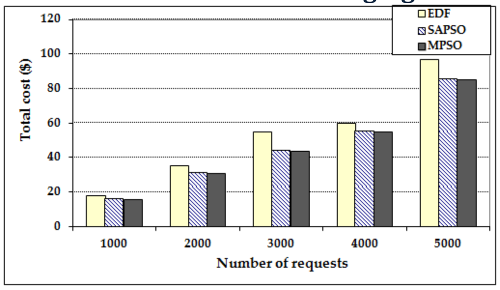

Fig 6. The total cost of the algorithms when requests change

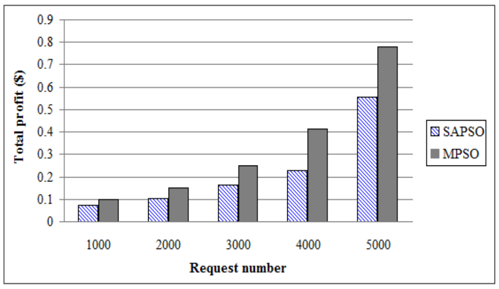

Fig 7. The total profit of the algorithms when requests change

This section presents the results of the total time, total cost and total profit when changing the number of requests from 1000 to 5000 and maintaining the fixed number of virtual machines as 150 VMs of the algorithms, as shown in Figure 5, Figure 6 and Figure 7.

Sequential algorithm uses the exhaustive method to search the resource, therefore the larger the requests are, the more time used for scheduling the complexity of algorithm will be exponential, so the sequential algorithm is not considered in this section.

When the number of requests increases, it will have many requests that can’t use all the rental time, while MPSO algorithm would be able to use all of such rental time. This will lead to the total cost of MPSO algorithm is much smaller than EDF and SAPSO algorithms as shown in Figure 6.And the total Profit gives to suppliers MPSO algorithm higher than SAPSO algorithm as Figure 7.

As shown in Figure 5, when the number of requests gets larger, the total of processing time will increase, the total processing time of both SAPSO and MPSO are nearly equal, while EDF algorithm only considers to use the ratio of U to find resources, not considering in finding the best resources, so that the total execution time of the EDF is often greater than the total execution time of SAPSO and MPSO.

4. Conclusion

In this article, we focus on the issue of admission control and the schedule for the requirements of users toward of multi-objective optimization. Based on the model of users and providers of IaaS of R.Buyya. The articles made the models of PaaS provider.From there, they apply the heuristic PSO to make calculations fitness function, the local best position of each particle and the global best position of the entire swarm. To optimize time-to schedule, optimize the execution time for the request and bring benefits to the suppliers , the articles proposed SAPSO and MPSO algorithm. These algorithms used parallel processing techniques and heuristic PSO to identify the resources with high-speed; to leverage overlapping period between the request to save costs.

After analyzing and evaluating simulation results using the the same samples and using same simulations tool show that SAPSO and MPSO algorithms have impressive improvement of time and cost when compared with some other scheduling algorithms.

References

[1]. O. M. Elzeki, M. Z. Reshad, M. A. Elsoud: Improved Max-Min Algorithm in Cloud Computing, International Journal of Computer Applications v.50, no.12, pp. 0975 – 8887, 2012.DOI: 10.5120/7823-1009

[2]. Liyun Zuo,Lei Shu, Shoubin Dong, A Multi-Objective Optimization Scheduling Method Based on the Ant Colony Algorithm in Cloud Computing, IEEE, vol 3, pp. 2687 – 2699, 2015. DOI: 10.1109/ACCESS.2015.2508940

[3]. Jzau-Sheng Lin and Shou-Hung Wu, Fuzzy Artificial Bee Colony System with Cooling Schedule for the Segmentation of Medical Images by Using of Spatial Information, Research Journal of Applied Sciences, Engineering and Technology, vol.4, no.17, pp. 2973-2980, 2012.

[4]. A.Burns, R.I. Davis, P. Wang and F. Zhang, Partitioned EDF Scheduling for Multiprocessors using a C=D Scheme. Department of Computer Science, Journal Real-Time Sysstem, vol.48, no.1, pp. 3-33, 2012.DOI: 10.1007/s11241-011-9126-9

[5]. Jianguang Deng, Yuelong Zhao, Huaqiang Yuan, A Service Revenue-oriented Task Scheduling Model of Cloud Computing, Journal of Information & Computational Science, vol.10, no.10, pp.3153-3161, 2013.DOI: 10.12733/jics20101954

[6]. Mao, Ming and Li, Jie and Humphrey, Marty, Cloud auto-scaling with deadline and budget constraints, Grid Computing (GRID), 11th IEEE/ACM International Conference, pp. 41-48, 2010.DOI: 10.1109/GRID.2010.5697966

[7]. Ramkumar N, Nivethitha S, Efficient Resource Utilization Algorithm (ERUA) for Service Request Scheduling in Cloud, International Journal of Engineering and Technology, vol. 5, no. 2, pp.1321-1327, 2013.

[8]. Swarupa Irugurala, Dr.K.Shahu Chatrapati, Various Scheduling Algorithms for Resource Allocation In Cloud Computing. The International Journal of Engineering and Science (IJES), pp. 16-24, 2013. DOI: 10.1080/02564602.2014.890837

[9]. Gomathi B., Krishnasamy, K.: Task scheduling algorithm based on hybrid particle swarm optimization in cloud computing environment, JATIT, vol. 55, pp. 33-38 , 2013.

[10]. Medhat A. Tawfeek, Ashraf El-Sisi , Arabi E. keshk, Fawzy A. Torkey, Cloud Task Scheduling Based on Ant Colony Optimization, ICCES, 2016. DOI: 10.1109/ICCES.2013.6707172

[11]. Wu, L., Garg S. K., Buyya, R.: SLA-based admission control for a Software-as-a-Service provider in Cloud computing environments, JCSS, vol. 78, pp. 1280–1299, 2012. DOI: 10.1016/j.jcss.2011.12.014

[12]. Kennedy, J., Eberhart, R.: Particle Swarm Optimization, Proceedings of IEEE International Conference on Neural Networks . pp. 1942–1948. IEEE, 1995. DOI: 10.1109/ICNN.1995.488968

[13]. Gomathi B., Krishnasamy, K.: Task scheduling algorithm based on hybrid particle swarm optimization in cloud computing environment, JATIT, vol. 55, pp. 33-38, 2013.

[14]. Bergh F.V.D.: An analysis of particle swarm optimizers, Doctoral Dissertation, University of Pretoria Pretoria, South Africa, South Africa, 2002.

[15]. Clerc, M.: The swarm and the queen: Towards a deterministic and adaptive particle swarm, In Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, pp. vol 48, 1951-1957. IEEE, 1999.DOI: 10.1109/CEC.1999.785513

[16]. Buyya, R., Ranjan: Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities, In Proceedings of the 7th High Performance Computing and Simulation, pp. 1-11. IEEE, 2009.DOI: 10.1109/HPCSIM.2009.5192685

[17]. Swarupa Irugurala, Dr.K.Shahu Chatrapati, Various Scheduling Algorithms for Resource Allocation In Cloud Computing. The International Journal of Engineering and Science (IJES), pp. 16-24, 2013.

[18]. Li, K., Xu, G., Zhao, G., Dong, Y., Cloud Task scheduling based on Load Balancing Ant Colony Optimization, In Chinagrid, 2011 Sixth Annual Sixth Annual, pp. 3-9. IEEE, 2011.DOI: 10.1109/ChinaGrid.2011.17

[19]. Łukasz Kruk, John Lehoczky, Kavita Ramanan and Steven Shreve, Heavy traffic analysis for EDF queues with reneging, The Annals of Applied Probability. vol. 21, no. 2, pp. 484–545, 2011.DOI:10.1214/10-AAP681

[20]. S. Pandey and R. Buyya. Scheduling workflow applications based on multi-source parallel data retrieval in distributed computing networks. Comput. J., 55(11):1288 - 1308, 2012.DOI: 10.1093/comjnl/bxr128

[21]. S. Pandey, L. Wu, S. M. Guru, and R. Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications, pages 400 - 407. IEEE Computer Society, 2010.DOI: 10.1109/AINA.2010.31

[22]. JiayinLi, MeikangQiu, ZhongMing, GangQuan, XiaoQin, and ZonghuaGu. Online optimization for scheduling preemptable tasks on iaas cloud systems. J. Parallel Distrib.Comput., 72(5):666 - 677, 2012. DOI: 10.1016/j.jpdc.2012.02.002

[23]. M. Frincu and C. Craciun. Multi-objective metaheuristics for scheduling applications with high availability requirements and cost constraints in multicloud environments. In Proceedings of the 2011 Fourth IEEE International Conference on Utility and Cloud Computing, pages 267- 274. IEEE Computer Society, 2011. DOI: DOI: 10.1109/UCC.2011.43.

[24]. Sharma, S. A Review on Cloud Computing Security and Privacy in Service Oriented Architecture (SOA). International Journal of Machine Learning and Networked Collaborative Engineering, 1(02), 81-92,2018. DOI: 10.30991/IJMLNO

Author's Biography

Nguyen Hoang Ha : Nguyen Hoang Ha, born in 1976 in Vietnam. Working in the Faculty of Information Technology, Hue University of Sciences.

1995-1999 Bachelor of Science (B.S) in Science (majoring in COMPUTER SCIENCE), Hue University of Sciences.

2003-2005 Master of Science (M.S) in Computer Science, Hue University of Sciences.

2012-2017 Doctor of Science (D.S) in Computer Science, Hue University of Sciences.

Nguyen Hoang Nguyen: Nguyen Hoang Nguyen born in 1983 in Vietnam.

2002-2006 Bachelor of Science (B.S) in Science (majoring in COMPUTER SCIENCE), Hue University of Sciences.

2010-2012 Master of Science (M.S) in Computer Science, Hue University of Sciences.

How to Cite

Ha, Nguyen Hoang, Nguyen, and Nguyen Hoang,, "A Scheduling Algorithm based on PSO Heuristic in Cloud Computing," *International Journal of Machine Learning and Networked Collaborative Engineering*, Vol. 02, No. 1, 2018, pp.15-26. doi: https://doi.org/10.30991/IJMLNCE.2018v02i01.003 .