

Admission Control and Scheduling Algorithms Based on ACO and PSO Heuristic for Optimizing Cost in Cloud Computing

Ha Nguyen Hoang, Son Le Van, Han Nguyen Maue
and Cuong Phan Nhat Bien

Abstract Scheduling problem for user requests in cloud computing environment is NP-complete. This problem is usually solved by using heuristic methods in order to reduce to polynomial complexity. In this paper, heuristic ACO (Ant Colony Optimization) and PSO (Particle Swarm Optimization) are used to propose algorithms admission control, then building a scheduling based on the overlapping time between requests. The goal of this paper is (1) to minimize the total cost of the system, (2) satisfy QoS (Quality of Service) constraints for users, and (3) provide the greatest returned profit for SaaS providers. These algorithms are set up and run a complete test on CloudSim, the experimental results are compared with a sequential and EDF algorithms.

Keywords Admission control · Scheduling algorithms · Qos constraint · Resource allocation

1 Introduction

Cloud computing is the development of distributed computing, parallel computing and grid computing [1]. Resources in cloud computing including virtual machines, storage space, network bandwidth, etc., in which virtual machines have speed,

H.N. Hoang (✉) · H.N. Maue · C.P.N. Bien
Hue University of Sciences, Hue, Vietnam
e-mail: nhha76@gmail.com

H.N. Maue
e-mail: nmhan2009@gmail.com

C.P.N. Bien
e-mail: pnbckhmtpy2@gmail.com

S. Le Van
Da Nang University of Education, Da Nang, Vietnam
e-mail: levansupham2004@yahoo.com

bandwidth and costs vary. Users must pay for the entire time the rent for each virtual machine, although they may not use up this time.

The previous studies mainly focus on scheduling for user requests on multi-processor system with a fixed number of processors. For scalable computing, the virtual machines (VMs) are rented, and can be scaled up to any number. This creates some fundamental changes in the problem: (1) the requests can be done in parallel, its deadline and budget can be always guaranteed; (2) the number of VMs ready to serve is huge, so at every moment one can choose the number and type of VMs appropriately. (3) VMs can be rented with a fixed cost in a certain period of time. If the user does not use up the amount of time, other users can take advantage of it.

Generally, the admission control and scheduling request with parameters such as arrival time, deadline, budget, workload, and penalty rate, etc. is an NP-complete problem [2]. Therefore, to give an optimal solution one must often do exhaustive search while complexity is exponential, so this method cannot be applied. To overcome this disadvantage, we study the heuristic methods to produce a near optimal solution such as ACO method [1, 3, 4], PSO method [5–7]. After that, we take advantage of the remaining period of this request to implement for the next request with the purpose of greatest benefit for service providers.

In cloud computing environment, users rent through Internet and pay a fee for use. Therefore, the scheduling algorithms based on constraints are often used. In this case, the user's parameters such as time, users' service fees, providers' service fees, reliability, etc., are given priority when scheduling. Lee [8] made scheduling model for the requests on the cloud computing environment with the goal of bringing the highest profit for the service provider but looking in detail at the two participating elements of budget and deadline requirements. The studies [9, 10] focus on the scheduling requests for power savings on data center. The recent study by Ramkumar [11] of schedule in real-time requests used for priority queues mapped into resource requests but focused to solve scheduling tasks quickly satisfy most of the requests deadline regardless of cost and its budget. Irugurala [12] make scheduling algorithm with the objective to bring the highest return for SaaS providers but considering between the two types of costs: the cost of initializing virtual machine and the fee of virtual machine which are used to select resources. The studies in [1, 7], using the heuristic PSO, ACO to propose scheduling algorithms but only focus on the minimal makespan of the tasks.

In this paper, ACACO and ACPSO algorithms are proposed with the goal of are making the smallest cost for the system and combining with these algorithms for proposing Mprofit algorithm to bring big profits to SaaS providers.

The article includes: building system model (Sect. 2), building algorithm, introducing three ACACO, ACPSO and Mprofit algorithms then simulating, evaluating between the algorithms (Sect. 3) and conclusions (Sect. 4).

2 System Model

Systems in cloud computing environment consist of the following components [13]: Users, SaaS, PaaS and IaaS providers. Users send requests to the SaaS provider. PaaS providers use component admission control here to analyze the QoS parameters and to decide acceptance or rejection of the request based on the user's abilities, the availability and cost of virtual machines. If the request is accepted, the scheduling component is responsible for locating the resources for the user's requests.

Users send N service requests $\{t_1, t_2, \dots, t_N\}$ to SaaS providers [13], each request t_i ($a_i, d_i, b_i, \alpha_i, w_i, in_i, out_i$) includes the following constraints: a_i : arrival time of request; d_i : deadline of request; b_i (budget): the maximum cost users will pay for the services; α_i (penalty rate): a ratio of compensation to the user if the SaaS provider does not provide timely; w_i (workload): how many MI (million instruction) are required to meet the request; Size of input and output file: in_i and out_i .

In cloud computing environment with Y IaaS providers $\{x_1, x_2, \dots, x_Y\}$, each IaaS provider provides M virtual machines $\{vm_1, vm_2, \dots, vm_m\}$ for SaaS providers and is responsible for coordinating the VMs which runs on the their physical resources, each virtual machine vm_{jx} ($p_{jx}, s_{jx}, Dtp_{jx}, Dts_{jx}$) of the provider x attributes includes: p_{jx} (price): pricing depends on per hour that SaaS providers must pay for IaaS providers using VMs; s_{jx} : processor speed of virtual machines (MIPS); Dtp_{jx} : the price SaaS providers must pay to transport data from resource provider to user's computer; Dts_{jx} : data transporting speed depends on network performance

In this session, we focus on PaaS provider model. All IaaS's resource providers are not related to one another, can be executed in parallel and are represented by R . We set schedule for N requests independently not to follow any particular order of priority (non-preemptive) on Y providers. The requirements are denoted $npmtn$. The aim is to find the minimum cost but still satisfying deadline and budget of the requests, it means that C_{min} must be found. So the model is $R \mid npmtn \mid C_{min}$

Call C_{ijx} the cost of executing the request i on the virtual machine j of the resource provider x . Mean while C_{ijx} costs include costs:

- The cost of executing request (CP_{ijx}):

$$CP_{ijx} = p_{jx} * \frac{w_i}{s_{jx}} \quad (1)$$

- The cost of data transmission (CTD_{ijx}):

$$CTD_{ijx} = Dtp_{jx} * \frac{in_i + out_i}{Dts_{jx}} \quad (2)$$

- 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 β_{ijx} :

$$CR_{ijx} = \alpha_i * \beta_{ijx} \quad (3)$$

Let T_{ijx} is the time to process the request i on the virtual machine j of resource providers x . T_{ijx} is determined as follows:

$$T_{ijx} = \frac{w_i}{s_{jx}} + \frac{in_i + out_i}{Dts_{jx}} + \beta_{ijx} \quad (4)$$

Therein: $\frac{w_i}{s_{jx}}$: time to process the requests; $\frac{in_i + out_i}{Dts_{jx}}$: time to transfer data; β_{ijx} : exceeded time deadline.

The goal of the paper is to construct algorithms to find the virtual machine in the data center to minimize the cost, such as:

$$\text{Max}_{1 \leq i \leq N} \left(\sum_{x=1}^Y \sum_{j=1}^M (b_i - C_{ijx}) \right) \quad (5)$$

- For the profit of SaaS provider, the cost of request i must satisfy the requests of its budget that is:

$$C_{ijx} < b_i \quad (6)$$

- To satisfy the constraints of user, the execution time of request i must meet the deadline itself:

$$T_{ijx} \leq d_i + \beta_{ijx} \quad (7)$$

Thus, to achieve the proposed goals (5), it must satisfy two constraints (6) and (7).

3 Construction of Algorithm

3.1 ACACO Algorithm

The ACO heuristic and system model (session 2) are used to make a scheduling algorithm with the objective of making the total cost to the minimum but still meet

the budget and deadline of the requests. To apply the ACO, one must determine the minimum function F , heuristic information η_i , pheromone update and probability P [3, 14]. We construct minima function F and heuristic information η_i to find the best IaaS provider as follows:

$$F = \text{Max}(C_{jx}), \quad j = 1 \dots M, x = 1 \dots Y \quad (8)$$

$$\eta_i = \frac{1}{C_{jx}}, \quad i = 1 \dots N, j = 1 \dots M, x = 1 \dots Y \quad (9)$$

Every ant starts from the resource provider IaaS and requests resources randomly. At each iteration of the ants; find the minima function and pheromone update as follows:

$$\tau_{ijx} = \rho * \tau_{ijx} + \Delta\tau_{ijx} \quad (10)$$

Therein: $\Delta\tau_{ijx} = \frac{1-\rho}{F_k}$; with F_k is a minima function of the ant k ; $\Delta\tau_{ijx}$: was added to the pheromone; ρ is the evaporation rate is determined in the range (0,1).

According to [4] first request is done and it selects providers randomly. The next request will be processing and it selects the next provider with the probability:

$$P_{ij} = \frac{\tau_{ij} * \eta_{ij}}{\sum_{j=1}^M \tau_{ijx} * \eta_{ijx}} \quad (11)$$

In this paper, we consider on multiple providers, each provider offers multiple virtual machines, so the probability to select the next request t_i on the virtual machine j of the resource provider x is defined as follows:

$$P_{ijx} = \frac{\tau_{ijx} * \eta_{ijx}}{\sum_{x=1}^Y \sum_{j=1}^M \tau_{ijx} * \eta_{ijx}} \quad (12)$$

Therein: η_{ijx} : heuristic information, τ_{ijx} pheromone rate left when moving

ACACO algorithm

Input: $\rho = 0.05$, $\tau_{ijx} = 0.01$, number of ant $k = 10$; T, X, VMx : The set of user requests, the set of IaaS providers and the set of VMs.

2Output: The scheduling list S contains all approved requests by SaaS provider, each request i is mapped to the virtual machine j of provider x .

Description algorithm:

```

FOR EACH  $t_i$  in  $T$  DO
  FOR EACH ant  $k$  DO
    FOR EACH  $x$  in  $X$  DO
      Calculating heuristic information for request  $t_i$  on virtual machines  $vm_{ix}$  as the
      formula (8);
      Find the value of current pheromone  $\tau_{ijx}$ ;
      Calculated cost and execution time as the formula (1), (2), (3), (4);
      Pheromone update as the formula (10);
      Calculate the probability for request  $t_i$  map into virtual machine  $vm_{ix}$  as the
      formula (12);
    END FOR
  END FOR
  From the probability on virtual machines  $vm_{ix}$ , find the virtual machine which has
  the highest probability, but the  $cost \leq b_i$  and processing time  $\leq d_i + \beta_{ijx}$  if found
  then  $S = S + \{t_i \rightarrow vm_{ix}\}$  else inform the users that the request has been reject;
END FOR

```

3.2 ACPSO Algorithm

Eberhart and Kennedy introduced PSO algorithm based on the experience of swarm in 1995 [5]. It simulates the social behavior of birds or fish searching for food. In every generation, each particle can change its position from time to time and find local optimal solution ($Pbest$) in D-dimension search space. Then $Pbest$ is compared with global optimization solution ($Gbest$) of the swarm to update the value for $Gbest$. Based on $Gbest$, the best optimal solution is found. To apply the PSO algorithm, one must determine the position, velocity, $Pbest$ and $Gbest$ [5–7]. Each particle is based on current velocity and distance from $Pbest$ to $Gbest$ to change position and speed as follows [6]:

$$v_x^{j+1} = \omega * v_x^j + c_1 * r_1 * (Pbest_x - pos_x^j) + c_2 * r_2 * (Gbest - pos_x^j) \quad (13)$$

$$pos_x^{j+1} = pos_x^j + v_x^{j+1} \quad (14)$$

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 ; $Pbest_x$: local best position of particle x ; $Gbest$: global best position of the entire swarm.

We call P the population consisting of Y particles: $P = \{X_1, \dots, X_Y\}$. Each particle X_k ($k = 1 \dots Y$) searches food in M_k dimension space and is defined: $X_k = \{pos_{ix}^1, \dots, pos_{ix}^{M_k}\}$, where in pos_{ix}^j is the position of the particle x at the loop i in dimension j ($j = 1 \dots M_k$).

Velocity V_k of particle X_k is shown as follows: $V_k = \{v_{ix}^1, \dots, v_{ix}^{M_k}\}$, wherein v_{ix}^j is the velocity of the particle x at the loop i in dimension j .

In the model at Sect. 2, we have N requests of users and Y providers, each particle X_k ($k = 1 \dots Y$) loop M_k times to find resources for the request t_i ($i = 1 \dots N$), each particle is equivalent with each providers. 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_k , and the value of v_{ix}^j is taken from $-M_k$ to M_k randomly, wherein M_k is the number of VMs in each provider x . To achieve the objective of SaaS providers as in formula (6) we construct fitness function for particle x to select virtual machine j for request i as follows:

$$f(pos_{ix}^j) = \frac{1}{C_{ijx}} \quad (15)$$

Therein, C_{ijx} is defined as in formula (1)–(3).

Based on the fitness function in formula (15), the local optimal position of particle x is as follows:

$$pb_{ix}^{j+1} = \begin{cases} p_{ix}^{j+1} & \text{if } f(pos_{ix}^{j+1}) \geq f(pos_{ix}^j) \text{ and } C_{ijx} \leq b_i \text{ and } T_{ijx} \leq d_i + \beta_{ijx} \\ pb_{ix}^j & \text{Other cases} \end{cases} \quad (16)$$

Therein, $C_{ijx} \leq b_i$ and $T_{ijx} \leq d_i + \beta_{ijx}$ as formula (6) and (7)

Local best position of particle x ($Pbest_x$) and global best position ($Gbest$) are calculated as:

$$Pbest_x = \max_{1 \leq j \leq M_x} (pb_{ix}^j) \quad (17)$$

$$Gbest = \max_{1 \leq x \leq Y} (Pbest_x) \quad (18)$$

ACPSO algorithm

Input: T, P, VM_j : The set of user requests, the set of IaaS providers, the set of VMs.

Output: The scheduling list S contains all approved requests by SaaS provider, each request $t_i \in S$ is mapped to the virtual machine j of provider x .

Description algorithm:

Initialization: pos_{kx}^j is a random number from 1 to M_k ; v_{kx}^j is a random number from $-M_k$ to M_k ; $Pbest_x = pos_{1x}^1$; $Gbest = \max_{1 \leq x \leq Y} (Pbest_x)$;

FOR EACH t_i **in** T **DO**

FOR EACH X_k **in** P **DO** {

Calculate fitness function as formula (15);

Calculate $Pbest_i$ as formula (16) and (17);

Calculate $Gbest_i$ as formula (18);

END FOR

Based on $Gbest_i$, find the virtual machine which has the $cost < b_i$ and processing time $\leq d_i + \beta_{ijx}$, if found then $S = S + \{t_i \rightarrow vm_{jx}\}$ else inform the users that the request has been reject;

Update the position and velocity of the particles as formula (13),(14);

END FOR

3.3 Mprofit Algorithm

Each virtual machine of IaaS providers is hired for hours and SaaS vendors must pay a fixed fee for the rental hours, if they do not use all their one-hour of hiring time, they also have to pay for a whole hour. This promotes a demand for effective positioning of costs for requests. Each vendor x can accept multiple requests, the advantage of validity period of the lease within 1 h of request is taken in the same vendor to provide the highest return for SaaS providers. The period of validity within an hired hour is called as the advance time of both requests and defines the set T_i including every request of the same provider with request t_i and put the advantage on request t_i . All these requests can share the same virtual machine.

$$T_i = \{t_l | d_l \geq d_i \text{ and } a_l < d_i\} \quad (19)$$

After identifying set T_i , the overlapping time will be calculated. t_{iljx} is defined as the effective time to calculate the request t_l after completing the request of t_i on virtual machine j of resource provider x . The value of t_{iljx} depends on the speed of virtual machines, arrival time, deadline, and workload of t_i and t_j . t_{iljx} is calculated as follows:

$$t_{iljx} = \begin{cases} \min(D - U_{il}, d_l - a_l) & \text{if } a_l - a_i \geq \frac{w_i}{s_{jx}} \\ D - U_{il} & \text{if } a_l - a_i < \frac{w_i}{s_{jx}} \text{ and } d_l - a_i \geq D \\ d_l - (a_i + U_{il}) & \text{if } a_l - a_i < \frac{w_i}{s_{jx}} \text{ and } d_l - a_i < D \end{cases} \quad (20)$$

Therein $U_{il} = \frac{w_i}{s_{jx}} + \max(a_l - d_i, 0)$, s_{jx} the speed of virtual machine is mapped to request t_i

Mprofit algorithm

Input: S is the set of request which has been accepted by the SaaS provider, and the output of ACACO and ACPSO algorithms.

Output: An optimal schedule ST to map the request to virtual machine.

Description algorithm:

Sort all request in S accordingly to the provider, then all requests of the same provider will be in the same group;

FOR EACH provider x in S DO

PUSH(t_i); // Save t_i into the stack, t_i is the first request of the provider x

$ST = ST + \{t_i\}$; $S = S - \{t_i\}$;

FOR EACH request t_i in the provider x DO

$t_i = \text{POP}()$; // Take t_i from stack

Find T_i and calculate t_{iljx} for the requests in T_i as in formula (19), (20);

Find $\max(t_{iljx})$, t_i has the largest overlapping time as the next request;

Base on $\max(t_{iljx})$ to find w_i reload all request status of t_i ;

PUSH(t_i);

$ST = ST + \{t_i\}$; $S = S - \{t_i\}$;

END FOR

END FOR

Base on ST to produce the mapped schedule onto the request of resources;

3.4 Correctness of the Algorithms

- Stutzle and Dorigo [3], Clerc [15] proved the convergence of the ACO and PSO algorithms which ensure the convergence of proposed algorithm ACACO and ACPSO.
- ACACO and ACPSO algorithms map the request i into the virtual machine j of the provider x based on the probability and fitness function as in formula (12) and (15). Therefore, the smaller the cost of virtual machine is, the greater the heuristic information and the greater fitness function are. This results in more probability of selecting low-cost VMs.
- The resource rental period is D-minute, therefore request t_i completes its task on itself with the lesser time than D-minute, but to pay the fee of D-minute. Mprofit algorithm takes advantage of this effective time interval to process the next request, which makes the costs of the entire system reduced, according to the objectives of SaaS providers

3.5 Simulation and Evaluation of the Algorithms

The algorithms are installed in NetBean 7.1.1, JDK 6, CloudSim 2.1 tools package [16] with the following parameters: Use 4 Datacenter, 10 physical hosts, 150 virtual machines. The parameters of users and resource providers are identified:

On the user side: the arrival time to be taken at random from 1 to 500, deadline is generated randomly between (d_l , d_u) minutes and the different values of d_l and d_u are limited from 10 to 1500, deadline must be greater than arrival time. Workload is taken at random from 8×10^4 to 10^5 MI, based on the workload the required budget is estimated, the remaining parameters are taken as implicit in CloudSim.

On the resource provider's side: the researchers simulate upon four resources providers; each resource provider has a number of virtual machines, costs, speed, 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 defined as follows: speed is taken at random from 10^3 to 5×10^3 MIPS corresponding with the costs which are real numbers taken at random from 0.001 to 0.01, other parameters of the virtual machine as a virtual machine initialization time, bandwidth, etc. took default values of CloudSim.

The values in simulation is the result of the 5 tests and the average results are obtained. In the simulation, we compare the total cost on the requests which were accepted by the SaaS provider, these requests must be present in the output of the algorithms. In contrast when compared to the benefit of providers, we calculate on all requests are accepted.

3.5.1 Analyze the Total Cost and Total Profit as Fixed Requests

Figures 1 and 2 show the total cost and total profit of algorithms; using 150 VMs and 1000 requests. Simulation results show that the total cost is often lower and

Fig. 1 The total cost of the algorithms as fixed requests

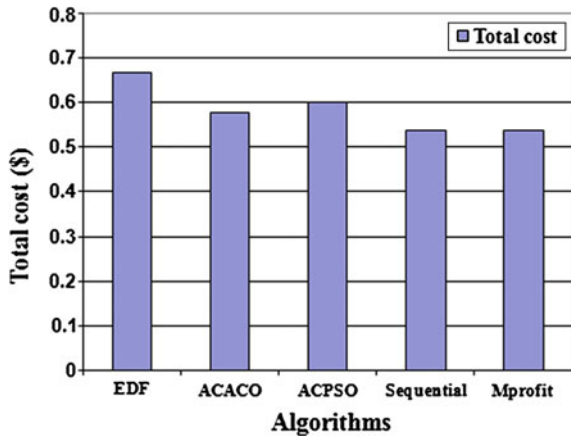
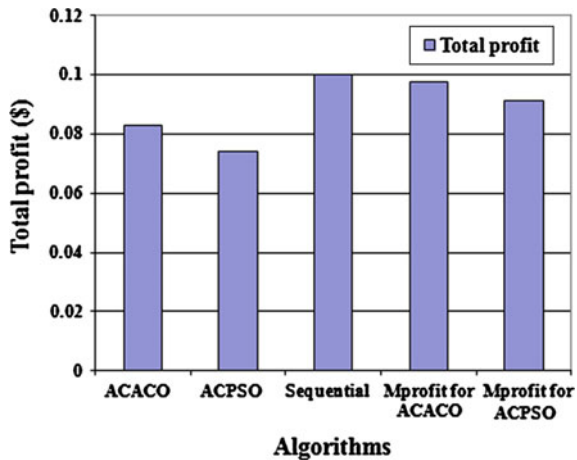


Fig. 2 The total profit of the algorithms as fixed requests



total profit of Mprofit algorithm is often higher than in remaining algorithms. The reason is that the ACACO and ACP SO algorithms are responsible for admission control, accepting or rejecting the user requests, the requests is accepted will be mapped to VMs which have low cost, then Mprofit algorithm continues taking advantage of the overlapping time of the requests in the same IaaS provider, which leads to the total of processing fee reduction as Fig. 1.

In contrast, sequential algorithm does not consider the overlapping period between requests, but uses exhaustive algorithm to find 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 take a huge amount of time to make a schedule. As EDF algorithms only consider the ratio used: $U = \sum_{i=1}^m \frac{C_i}{T_i} \leq 1$ (where C_i is the execution time and T_i corresponded deadline) [17, 18] to map the request to the resource, thus EDF algorithm only ensures the request to complete before the deadline, regardless of the cost of the request. So, we do not consider the EDF algorithm when comparing the benefits of SaaS providers.

The results in Fig. 2 shows the total profit given by SaaS providers of sequential algorithm and Mprofit algorithm are nearly equal. The sequential algorithm uses exhaustive algorithm to find the best resources, whereas Mprofit algorithm takes advantage of the period that is not used up to implement for the next request.

3.5.2 Analyze the Total Cost and Total Profit as Fixed Number of VMs and Changing Number of Requests

This section presents the results of the total cost, total profit when change the number of requests from 1000 to 5000 and also maintaining the fixed number of VMs as 150, as shown in Figs. 3 and 4. Sequential algorithm uses the exhaustive algorithm to find the resource, therefore the larger the requests are, the more time

Fig. 3 The total cost of the algorithms when requests change

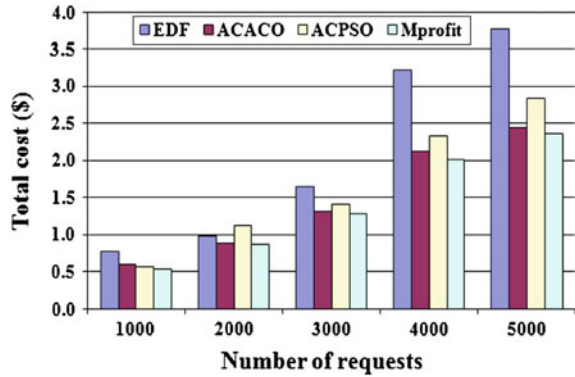
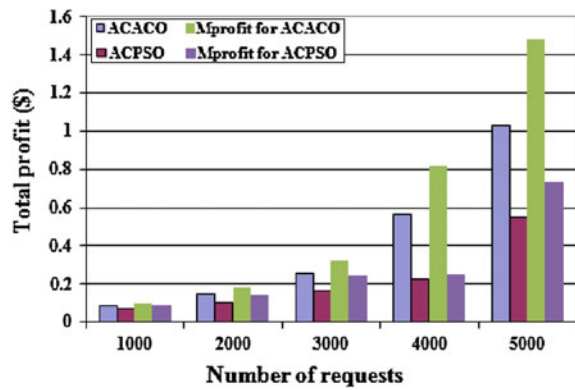


Fig. 4 The total profit of the algorithms when requests change



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 Mprofit algorithm would be able to use all of such rental time. This will lead to the total cost of Mprofit algorithm is much smaller than EDF, ACP SO and ACACO algorithms, as shown in Fig. 3. Although the cost of ACP SO algorithm is higher than ACACO algorithm but ACP SO load balancing and faster than ACACO. When the input data of Mprofit algorithm is outputs of ACACO (Mprofit for ACACO) will give the biggest benefit to the SaaS provider, as shown in Fig. 4.

4 Conclusions

The article focuses on researching the admission control algorithm and scheduling for users' requests with QoS constraints, in each request, the researchers check all factors such as arrive time, cost, deadline, budget, word-load, rates of penalization,

input and output file sizes; each virtual machine has speed, cost and different bandwidth. This paper proposes ACACO, ACP SO algorithms to admission control and find the resources with low cost in order to bring the lowest cost to users. In combining with Mprofit algorithm, the Mprofit algorithm is proposed to use up all the time that the requests rented for bringing the most profit to SaaS provider. According to the analysis and strategically experimental results of the same samples and using some CloudSim simulations show that ACACO, ACP SO and Mprofit algorithms have impressive improvement of cost compared to the sequential and EDF algorithms.

References

1. Li, K., Xu, G., Zhao, G., Dong, Y.: Cloud task scheduling based on load balancing ant colony optimization, In: 2011 Sixth Annual Sixth Annual Chinagrid, pp. 3–9. IEEE (2011)
2. Brucker, P.: Scheduling Algorithms, 5th edn., Springer, Berlin (2007)
3. Stutzle, T., Dorigo, M.: A Short convergence proof for a class of ant colony optimization algorithms. *Trans. Evol. Comp.* **6**, 358–365 (2002)
4. Kousalya, K., Balasubramanie, P.: An enhanced ant algorithm for grid scheduling problem. *IJCSNS* **8**, 262–271 (2008)
5. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948. IEEE (1995)
6. Bergh, F.V.D.: An analysis of particle swarm optimizers, Doctoral Dissertation, University of Pretoria Pretoria, South Africa, South Africa (2002)
7. Gomathi, B., Krishnasamy, K.: Task scheduling algorithm based on hybrid particle swarm optimization in cloud computing environment. *JATIT* **55**, 33–38 (2013)
8. Lee, Y.C., Wang, C., Zomaya, A.Y., Zhou, B.B: Profit-driven service request scheduling in clouds. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 15–24. IEEE (2010)
9. Mao, M., Li, J.: Cloud auto-scaling with deadline and budget constraints. In: Grid Computing, 11th IEEE/ACM International Conference, pp. 41–48. IEEE (2010)
10. Kim, K.H., Beloglazov, A., Buyya, R.: Power-aware provisioning of cloud resources for real-time services. In: Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science, pp. 1–6. ACM (2009)
11. Ramkumar, N., Nivethitha, S.: Efficient Resource utilization algorithm for service request scheduling in cloud. *IJET* **5**, 1321–1327 (2013)
12. Irugurala, S., Chatrapati, K.S.: Various scheduling algorithms for resource allocation in cloud computing. *IJES* **5**, 16–24 (2013)
13. Wu, L., Garg, S.K., Buyya, R.: SLA-based admission control for a Software-as-a-Service provider in cloud computing environments. *JCSS* **78**, 1280–1299 (2012)
14. Dorigo, M., Stutzle, T.: Ant Colony Optimization, A Bradford Book. The MIT Press, Cambridge, Massachusetts (2004)
15. Clerc, M.: The swarm and the queen: towards a deterministic and adaptive particle swarm. In: Proceedings of the 1999 Congress on Evolutionary Computation, CEC 99, vol 48, pp. 1951–1957. IEEE (1999)
16. Buyya, R., Ranjan, R.: 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)

17. Burns, A., Davis, R.I., Wang, P., Zhang, F.: Partitioned EDF scheduling for multiprocessors using a $C = D$ scheme. Department of computer science. *J. Real-Time Syst.* **48**, 3–33 (2012)
18. Gupta, G., Kumawat, V., Laxmi, P.R., Singh, D., Jain, V., Singh, R.: A simulation of priority based earliest deadline first scheduling for cloud computing system. In: 2014 First International Conference on Networks & Soft Computing (ICNSC), pp. 35–39. IEEE (2014)

Recent Developments in Intelligent Information and
Database Systems

Król, D.; Madeyski, L.; NGUYEN, N.-T. (Eds.)

2016, XII, 468 p. 145 illus., 91 illus. in color., Hardcover

ISBN: 978-3-319-31276-7