ENPP: Extended Non-preemptive PP-aware Scheduling for Real-time Cloud Services

Fereshteh Hoseini[1], Mostafa Ghobaei Arani[1], Alireza Taghizadeh[2]
[1] Department of Computer Engineering, Mahallat Branch, Islamic Azad University, Mahallat, Iran
[2] Department of Computer Engineering, Parand Branch, Islamic Azad University, Tehran, Iran

Article Info

Article history:

Received Mar 23, 2016
Revised Jul 6, 2016
Accepted Jul 24, 2016

Keyword:

Cloud computing
Non-exclusiveonline algorithms
Real-time scheduling
Real-time systems

ABSTRACT

By increasing the use of cloud services and the number of requests to processing tasks with minimum time and costs, the resource allocation and scheduling, especially in real-time applications become more challenging. The problem of resource scheduling, is one of the most important scheduling problems in the area of NP-hard problems. In this paper, we propose an efficient algorithm is proposed to schedule real-time cloud services by considering the resource constraints. The simulation results show that the proposed algorithm shorten the processing time of tasks and decrease the number of canceled tasks.

Corresponding Author:

Fereshteh Hoseini,
Department of Computer Engineering,
Mahallat Branch, Islamic Azad University,
Mahallat, Iran.
Email: fereshteh.hossini@gmail.com

1. INTRODUCTION

The world of computing has changed dramatically since the appearanceof Internet. The computing on a singleprocessor has given its place to parallel computing and finally thedistributedcomputing, and in particular, the cloud computing. The cloud computing is a new generation of data centers with virtualized nodes having a set of resources that are provided dynamically and according touser's demand. The cloud computing is a set of applications, system hardware and software provided as services [1]-[3]. The cloud providers should guarantee offering these services as requested.

The cloud computing technology provides the services in three forms of software as a service (SaaS), platform as a service (PaaS) and infrastructures a service (IaaS) to the users. The IaaS layer provides the virtual infrastructure including the processor, memory and network to support execution various operating systems. The PaaS layer provides the traditional services such as operating systems using the resources provided by IaaS. The SaaS layer provides the application program which the end-users can write, develop, and execute programs in the cloud environment [4].

The cloud computing as distributed computing iscomposedof a many resourcesand requests with a purpose to share resources as services on the internet platform. The resources such as memory and processor are expensive and the optimum use of them is considered as an infinite challenge. Hence, the scheduling of the tasks in the cloud computing is a very important issue which attempts to determine an optimum resource allocation [5]. The services provided in the cloud environment are essentially based on real-time software's. Not providing the required services in the specified time, it might have no dire consequences but ultimately leads to dissatisfaction of the customers [6],[7].

In this paper after evaluating the problems of the existing algorithms, it isattempted to provide a solution using an improvednon-exclusiveonline scheduling algorithm.The experimental results show that the proposed method has better execution compared to the methods such as earliest deadline first (EDF), functional cumulative scheduling and the other scheduling methods aware of the profit and penalty that are based on the same model.

The rest of the paper is organized as follows: In the second section, the relevant tasks are reviewed, and then the proposed algorithm is introduced in the third section.In the fourth section, the algorithm performance is evaluated and finally in the fifth section, the conclusion and recommendations are provided.

2. RELATED WORKS

The real-time scheduling algorithms can be divided into two categories of static and dynamic. In the static mode, before the onset of the system, the scheduling decisions are made but in the dynamic mode, the scheduling decisions carried out at the time of system execution [8],[9]. The static algorithms have no use in the cloud computing. The rest of this section introduce a few samples of dynamic algorithms.

Li et al. [10] proposed the use of proactive EDF. In this method, scheduling performs the tasks based on the priority of them. These priorities are not a good representative to show the necessity to perform a task, because the necessity of a task is determined publicly, and according to other tasks.

Kumar et al. [11] propose a multi-stage algorithm based on the old EDF where the user has selected the VMs and pay the costs as amazon model.

Jenson et al. [12], for the first time in order to overcome the deficiencies in the previous algorithms, raised another criterion named TUF, in which the soft real-time system timeconstraints are specified accurately. In fact, TUFs are a generalized model in deferral period which determine the efficiency of a task due to the distance of task to deferral period.

Yu et al. [13], propose a task model that considered both profit and penalty. According to this model, the task is related to two TUF, one profit TUF and the other penalty TUF. The system (determined by the profit function) considers the profit, if the task was completedaccording to deadline, and give penalty (determined by penalty function) if the task violated deadline or was removed before the deadline. In this task, the negative values are used for the penalty, and as a result, both TUFs were used in a single TUF.

Santhosh et al. [14] providedthe exclusive scheduling of the online real-time services with task transmission. In this type of algorithm, if the deadline was violated, task is transferred to another virtual machine, which results in the improved overall system efficiency and maximization of the general use.

Deniziak et al. [15] proposed the real-time scheduling algorithm using the evolutionary genetics programming. This method is operated based on the worst mode. The worst modeis the time that all the programs started simultaneously, this assumption is corresponding to the simultaneous creation of the requests. All the duties are scheduled on a constant order and activated in a specific time frame.

3. PROPOSED APPROACH

One method of the modern dynamic algorithms is that, for each task, two types of profit and penalty functions are considered.By adding upprofit and penaltyfor eachtask, the expected benefits are obtained then the tasks are sorted in the order of preference. In the exclusive methods, when the new task is entered with high priority, it can not be performed until the execution task is not finished, but in the non-exclusive methods, by entering a task with higher priority, the resources are taken from the present task and given to task with higher priority. These methods aside from their advantages, have challenges such as increase in the response time, in the case the number of requests was increased.

The proposed algorithm, i.e., Extended of Non-preemptive PP-aware scheduling (ENPP), which is based on non-exclusive scheduling, attempts to consider the best type of prioritization for eachtask, so that the tasks with the highest penalty are canceled in the minimum time.

3.1. Proposed Algorithm

The extended of non-exclusive online scheduling policy with the purpose to maximize the overall system efficiency. These policies includes the sorting tasks in the queue, and change in the acceptance test at the time of the entering a new task during scheduling. Once the tasks areaccepted to enter the queue, the problem is to how an appropriate scheduling decision is adopted for maximum benefit. In this algorithm, the scheduling decisions are made at following points:

- Task has been successfully completed before deadline
- Reached to the critical time of the execution task

The critical time is the time when task execution is at the expense of system. In fact, when a task is performed too late, it has no longer any profit for the system. Any longer execution time reduces the profit even if the task was completed before the deadline. This is shown with $t_{critical}$, which can be obtained according to Equation (1):

$$t_{critical} = \inf\{t + t_0 : \rho(t, t_0) > \rho_{max}\} \tag{1}$$

At any moment t, the expected profit and penalty to perform or cancel the task is different. In this mode, it is better that the decision to cancel or continue the task is adopted based on the factors logically. So the ratio between the probable losses against the expected profit, is an index to measure the task processing risk, which is called risk factor and shown with p, and obtained according to the following equations. First, all possible modes are evaluated according to Equation (2):

I. If(t.+B$\le D < t_0 + w \quad and \quad t_0 + t \le t_0 + B \rightarrow$

$$\int_{t.+B}^{D} * = \frac{a}{g}\left[\frac{(t.+B)^z}{z} - D(t.+B) + \frac{D_2}{Z}\right]$$

II. if(t_0+B$\le D < t_0 + w, t_0 + t > t_0 + B) \rightarrow$

$$\int_{t_0+B}^{D} * = \frac{a}{g}\left[\frac{(t.+B)^z}{z} - D(t_0 + B) + \frac{D_2}{Z}\right]$$

III. D$\ge t_0$+w,t+t$\le$t+$B \rightarrow$

$$\int_{t.+B}^{t.+w} * = \frac{a}{g}\left[\frac{(t.+B)^z}{z} - D(t_0 + B) + \frac{(t.+w)^z}{z}\right] \tag{2}$$

IV. D$\ge t_0$+w,t+B<t_0+t<t+w$\rightarrow$

$$\int_{t.+B}^{t.+w} * = \frac{a}{g}\left[\frac{(t.+t)^z}{z} - D(w - B) - \frac{(t.+w)^z}{z}\right]$$

V. else$\rightarrow \int = 0$

The task is accepted when the Equation (3) is established

$$W+R<D \tag{3}$$

And the risk factor obtained by Equation (4), is not more than the system maximum risk factor

$$\rho(t, t_0) = \begin{cases} \text{if } (t_0 + t \le B + t_0, B + t_0 \le D < t_0 + w) \\ \qquad \rho = \frac{a_l(t+t_0-R)(w+t_0-D)}{a_g\left[\frac{t_0+B}{t} - D(t_0+B) + \frac{D^t}{t}\right]} \\ \text{if } \quad (D \ge t_0 + w, t_0 + t \le t_0 + B \ )\rho = 0 \\ \qquad \text{else} \qquad \rho = +\infty \end{cases} \tag{4}$$

In general, a high risk in the system, can lead togreater profitsas well aslosses. Different service providers tolerate various risk levels, and only the tasks with risk level lower than the tolerated risk or maximum risk factor, shown with p_{max}, are accepted and executed, meaning that according to Equation (1), the task is canceled.

The scheduling method choose a task with the highest expected profit, and only executed until the critical time and the moment of exceeding the system tolerable risk will be removed or canceled, meaning the moment that leads to loss. Therefore, at any point of the t_s scheduling, besides choosing the task for the execution, it is tested that if the current choice of the risk would increase other tasks or not, and remove those tasks as soon as possible.

If a request was removed at the time of release, there is no profit or penalty for the service provider, when the request was accepted for the process, there is the possibility of profit and penalty. The profits are decreased overtime and the costs are increased. The overall system efficiency is obtained by the overall profit, and shown according to Equation (5):

$$P(t)=\begin{cases} 0 & t \le r \\ G(t) & r < t < D \\ -L(t) & r < t \le D \end{cases} \tag{5}$$

where, r (task releasing time), D (completion deadline), G(t) (task profit), L(t) (task penalty) are task execution time and random variable between the best and worst execution time and determined by the probability density function.

The scheduling algorithm is sorting the queue based on the profit density according to Equation (6), and each time a new task is entered the queue, the sorting is done again to predict which task has the higher profit at the shortest time and achieve higher profit for the system.

$$A(t_0) = \frac{G(t_0+e)}{e} \tag{6}$$

There are two positions where the new request might be accepted:
1) System contains sufficient resources
2) New request had more profits than other accepted requests in the system

In fact, the proposed algorithm is areal-time scheduling algorithm by having time-dependent set of requests in order to maximize the system overall profit. T_i is the time when T_i is established or removed, meaning $max\Sigma P_i(t_i)$.

Suppose that a set of tasks, $M = \{T_1, T_2, T_3, \dots\dots\dots\dots\dots . T_4\}$ are the requests for entering into the system, and the request recently entered into the system is shown with T.

$$T_n = \{r, e, B, W, G(t), L(t), D, f(c)\}$$

A(t0) is considered as the profit density of the pending requests of the system at the time t0, $\rho(t, t_0)$is the risk factor, E(Gi(t)) is the expected profit of Ti, D is the completion deadline and B is the best time and W is the worst time for execution, and f(c) is the probability density function of task execution time. The general process of the proposed algorithm is shown in Algorithm 1:

Algorithm 1: The proposed algorithm pseudo-code (ENPP)

```
1.  While $M(T_n) \neq \emptyset$
2.  $E_H=0; T_H=T_1; t_{action}=inf;$
3.  **While** $T_i$
    a. Calculate $E(G_i(T))$ at scheduling point $t_s$
    b. **If**$E(G_i(T)) > E_H$**then**
    c. $E_H=E(G_i(T)); T_H=T_i$ ;
    d. End if
4.  **Sort request $M(T_n)$ on order A(t0)**
5.  End while
6.  **Calculate** $T_H$ is $t_{critical}$ using
7.
    $t_{critical} = \inf\{t + t_0 : \rho(t, t_0) > \rho_{max}\}$
8.  $t_{action}=\min\{t_{critical}, D_H\}$;
9.  **if (**$t_s + B_H + B_i > D_i$ or $\rho_i(B_i, t_s + B_H) > \rho_{max}$)
10. using
11. $\rho(t,t_0)\begin{cases} \text{if } (t_0 + t \le B + t_0,\ B + t_0 \le D < t_0 + w) \\ \quad \rho = \frac{a_l(t+t_0-R)(w+t_0-D)}{a_g\left[\frac{t_0+B}{t} - D(t_0+B) + \frac{D^t}{t}\right]} \\ \text{if } (D \ge t_0 + w,\ t_0 + t \le t_0 + B) \quad \rho = 0 \\ \text{else} \quad \rho = +\infty \end{cases}$
12. then
13. **Remove $T_i$**
14. **Execute** $T_H$ to min{ $T_H$ finishing time ($t_f$) ‹$t_{action}$};
```

```
15. If $T_h$ does not finish $att_{action}$ then
16. Abort $T_h$ ;
17. Remove $T_h$ from
18. $t_s = t_{action;}$
19. else
20. $t_s = t_f$ ;
21. End
End
```

4. PERFORMANCE EVALUATION

In this section, the efficiency of the proposed method is evaluated using experiments. The proposed algorithms are compared against EDF [11] algorithms, and non-exclusive online scheduling aware of profit and penalty algorithms called NPP [12]. In this paper, the single sequence of the real-time random tasks are evaluated, where is defined using the parameters in Table 1.

Table 1. Parameters Used in the Proposed Method

No	Parameter	
1	[Bi,Wi]	Best and worst execution time
2	Di	Relative deadline
3	e	Task execution time
4	r	Task release time
5	Fi(T)	Probability density function for execution time
6	Gi(t)	Profit TUF shows the task cumulative profit at the completion time t
7	Li(t)	Penalty TUF shows the penalty caused by task rejection at the time t

- ✓ Suppose that before deadline Gi(t) is a non-ascending single-mode function, which means $G(t_i) \geq G(t_j)\ if\ \ t_i \leq\ t_j$ and $G_i(t) = 0$ if $t \geq D_i$
- ✓ Suppose that before deadline Li (t) is a non-descending one-aspect function, which means $L(t_i) \leq L(t_j)\ \ \ if\ \ \ t_i\ \ \leq\ \ t_j$ and the task are rejected immediately after the deadline.

The comparative evaluation focuses on two aspects: the system profit, and the number of removed tasks and completion time for computing the system profit, total profit, and penalty of all the tasks obtained according to Equation (7):

$$max \Sigma P_i(t_i) \tag{7}$$

In order to find the number of removed tasks, the sum of tasks canceled at each step obtained through the algorithm, the canceled tasks didn't scheduled due to having scheduling penalty. In order to obtain the task completion time, the sum of times lasted until all the tasks are scheduled are considered.

The method is in such that the algorithm is evaluated with 10 tasks. Then, the number of input loads is increased until the number of tasks reached to 100, and test the algorithm with a set of large tasks, and compare the system profit and the number of removed tasks as well as completion time of the proposed algorithm to two other algorithms.

In the tests, it is assumed that g and l are linear functions and time-dependent tasks are created randomly. In these tests, it is assumed that the amount of $G_i(t)$ and $L_i(t)$ is obtained using the Equations (8) and (9):

$$G(t) = \begin{cases} 0 & t < r + e \quad or \quad t > D \\ -a_g(t - D)r + e & t \leq D \end{cases} \tag{8}$$

$$\mathrm{L(t)} = \begin{cases} 0 & \mathrm{t < r} \\ \mathrm{a_l(t - r)r \leq t \leq D} \end{cases} \tag{9}$$

The proposed algorithm is in fact the developed form of non-exclusive online scheduling algorithm with same changes.

The proposed algorithm in the loop is simulated to the number of 100 loads in a time deadline between 10-100 seconds. The proposed algorithm is based on the number of tasks applied per second. Three scenarios are defined for this algorithm, the defined scenarios are removed based on three criteria, the system profit, the number of removed tasks, and completion time, and then the results have been analyzed. Table 2 shows the proposed scenarios.

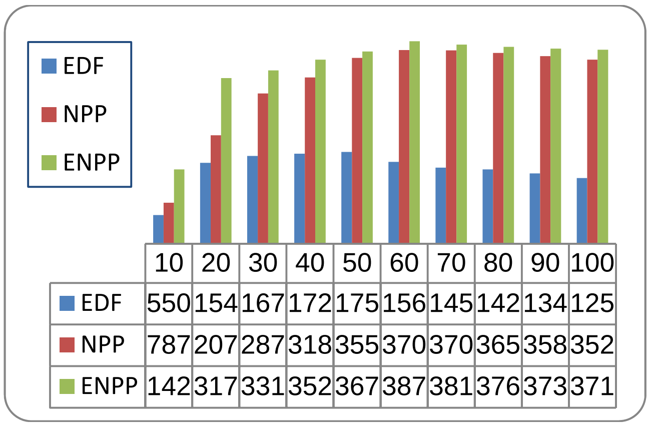
Table 2. Evaluated Scenarios

Scenario	Description	Target
First scenario	Considering workload from 10 to 100 tasks	Overall system profit
Two scenario	Considering workload from 10 to 100 tasks	Number of removed tasks
Third scenario	Considering workload from 10 to 100 tasks	Tasks scheduling completion time
Fourth scenario	Considering workload from 10 to 100 tasks	Correlation coefficient impact measurement

4.1. First Scenario

In the first scenario, the total profit obtained from tasks in the three mentioned methods has been analyzed. These three methods repeated with increase in the number of tasks from 10 to 100, the results have been shown in Figure 1.

According to Figure 1, it is understood that at the beginning, the system efficiency is raisedby the increasing ofload to the maximum extent and then it starts tofall. With low load, majority of tasks meetthe deadline and the system provide better profit, because it scheduled more tasks completely. However, by increasing the system load, some tasks start to miss the deadline and hence, the system encounters penalty. More increase in the system load causes more 5 be deadline violation and penalty and lower profit.



	10	20	30	40	50	60	70	80	90	100
🟦 EDF	550	154	167	172	175	156	145	142	134	125
🟥 NPP	787	207	287	318	355	370	370	365	358	352
🟩 ENPP	142	317	331	352	367	387	381	376	373	371

Figure 1. Comparison of Final Profit in ENPP, NPP, EDF

As shown in Figure 1, by raisingthe number of tasks, the profit is decreased. The reason isasthe time demand of tasks is increased, the higher competition causes many tasks don't be completed on time and be profitable. However, when the system workload is low, most of the requests finish quickly and cause more profit. In overall, the proposed method still makes more profit in many modes than other methods.

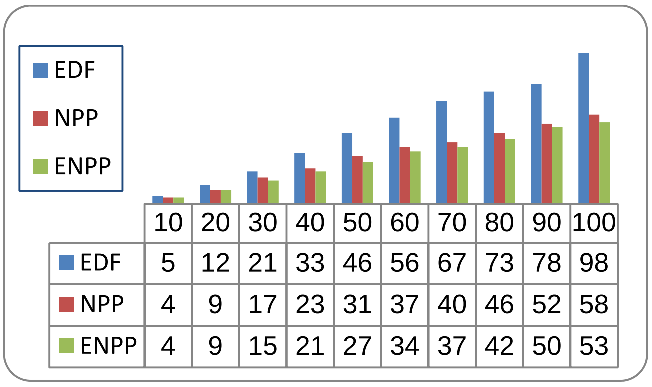
It is also understood that, EDF has lower efficiency, because it give more priority to the tasks with earlier deadline, regardless of the probable profit. Hence, even withlower demand density, EDF postpones the tasks completion with a high probability compared to other scheduling methods.

4.2. Second Scenario

One of the objectives in the proposed algorithm is to minimize the number of removed tasks. Thisevaluating experimenthas been conducted under various workloads for all the algorithms, and the results have been shown in Figure 2.

According to the results, it can be concluded that regarding request removal rate, since EDF gives higher priority to the tasks with earlier deadline, while the system workload is low, it has the lowest removal rate among all the methods. By increasing the number of tasks, this algorithm is not capable of scheduling and removes the large number of tasks. However, in high workloads, the ENPP algorithm has better performance.

Both NPP and ENPP remove the tasks in three steps. When the task is applied, the amount of profit and penalty is computed, and it is removed if the penalty was high. The next step, is the time when task correlation coefficient is higher than the system maximum correlation coefficient, and the final step of removing is the time when the task is reached to its critical point, which is when the task scheduling is at the expense of system and although it is not reached to the deadline, but the task execution is not in favor of the system. Since in ENPP method, the correlation coefficient is obtained with more efficient formula, the number of tasks removed in this method is lower than the other algorithms.



	10	20	30	40	50	60	70	80	90	100
🟦 EDF	5	12	21	33	46	56	67	73	78	98
🟥 NPP	4	9	17	23	31	37	40	46	52	58
🟩 ENPP	4	9	15	21	27	34	37	42	50	53

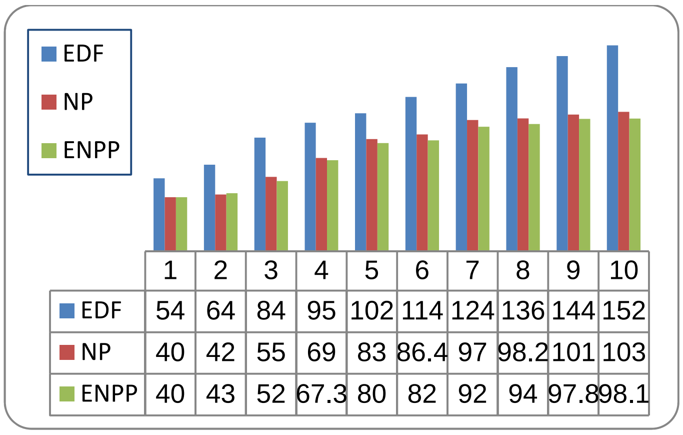
Figure 2. Comparison of Tasks Removalrate in ENPP, NPP, EDF

Figure 2 shows the request removalrate and it is observed that, when the number of tasks is low, the system is hardly removes the tasks. But with increase in the demand, the competition is enhanced and hence, the removal rate is increased even in the best algorithms.

4.3. Third Scenario

In this scenario, the completion time of the users aremeasured and evaluated in all mentioned methods. The experiments are repeated by increasing the number of tasks from 10 to 100 tasks.

According to Figure 3, in all methods the completion times become longer as the number of tasks increases. But for each number of tasks, the proposed method shows shorter completion time comparing other methods. The reason is that in the new sorting method, the tasks with higherprofit are available sooner to the scheduling. They also are filtered according to the new correlation coefficient, which makes them scheduled in shorter time.



	1	2	3	4	5	6	7	8	9	10
🟦 EDF	54	64	84	95	102	114	124	136	144	152
🟥 NP	40	42	55	69	83	86.4	97	98.2	101	103
🟩 ENPP	40	43	52	67.3	80	82	92	94	97.8	98.1

Figure 3. Comparison of Tasks Completion Time In ENPP, NPP, EDF

It is also observed that while that the workload is low, the proposed algorithm shows performance at the level of non-exclusive online algorithm, but with higher workload, it's performance is better.

4.4. Fourth Scenario

In this experiment, the effect of risk factor on the efficiency of the proposed algorithm is evaluated. To do this the risk factor $p_{max,}$ was raised from 1 to 10 and the number of produced tasks was equal to 100. The experiment was repeated for one hundred times and the general interest of the experiment is obtained.

As shown in Figure 4, the system profit is declinedby higher risk factor, for example in p_{max}=2 compared to p_{max}=10, the interest is increased 1.2 times. The risk factor p_{max} decided the system speed response, the lower coefficient means a faster response.Hence, in the system under high workload, the lower p_{max}should be used to achieve better efficiency. Although, the experimental results show that a lower risk factor should be selected for the system, identification of appropriate risk factor for each system is a complex and important issue that required further investigation.

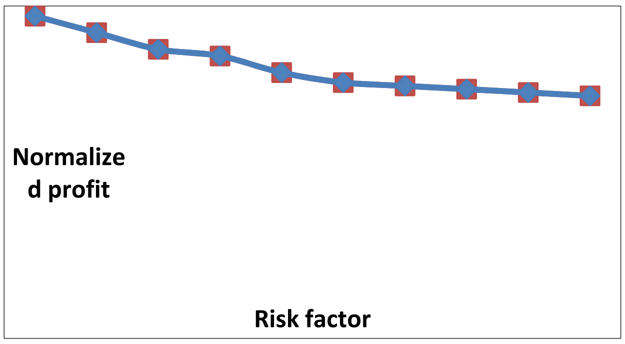


Figure 4. Effect of Correlation Coefficient on ENPP

4. CONCLUSION

A scheduler aimed to find a way to allocate the tasks to the limited resources properly. The increasing number of available resources and the computing requests at the minimum time and costs in cloud computing have emerged the resource allocation and scheduling as serious challenges. When the user requests are real-time, the scheduling problem become more obvious. Many algorithms have been provided for the real-time scheduling with their strengths and weakness. In this paper, due tothe evaluation of the problems in each of the existing algorithms, a new method named ENPP has been provided and regarding the system overall efficiency, completion time, and the number of the removed tasks, it was compared to algorithm EDF and non-exclusive online algorithm aware of the profit and penalty. According to the experimental results, the proposed method reduces the completion time and increases the system overall profit. The future studies could be on the algorithms with lower removal rate as well as the ability to determine the access level and scheduling of each resource for the user.

REFERENCES

[1] K. Mogouie, *et al.*, "A Novel Approach for Optimization Auto-Scaling in Cloud Computing Environment," *International Journal of Modern Education and Computer Science*, vol/issue: 7(8), pp. 9, 2015.

[2] H. G. Tani and C. E. Amrani, "Cloud Computing CPU Allocation and Scheduling Algorithms using CloudSim Simulator," *International Journal of Electrical and Computer Engineering (IJECE)*, vol/issue: 6(4), 2016.

[3] B. B. G. Abadi and M. G. Arani, "Resource Management of IaaS Providers in Cloud Federation," *International Journal of Grid and Distributed Computing*, vol/issue: 8(5), pp. 327-336, 2015.

[4] H. Ghiasi and M. G. Arani, "Smart Virtual Machine Placement Using Learning Automata to Reduce Power Consumption in Cloud Data Centers."

[5] M. G. Arani, *et al.*, "An autonomic approach for resource provisioning of cloud services," *Cluster Computing*, pp. 1-20, 2016.

[6] S. Kato, *et al.*, "A loadable real-time scheduler suite for multicore platforms," *Technical Report CMU-ECE-TR09-12, Tech. Rep.*, 2009.

[7] M. G. Arani and M. Shamsi, "An Extended Approach for Efficient Data Storage in Cloud Computing Environment," *International Journal of Computer Network and Information Security*, vol/issue: 7(8), pp. 30, 2015.

[8] H. Sun, *et al.*, "Research and simulation of task scheduling algorithm in cloud computing," *Indonesian Journal of Electrical Engineering and Computer Science*, vol/issue: 11(11), pp. 6664-6672, 2013.

[9] S. Pal and P. K. Pattnaik, "A Simulation-based Approach to Optimize the Execution Time and Minimization of Average Waiting Time Using Queuing Model in Cloud Computing Environment," *International Journal of Electrical and Computer Engineering (IJECE)*, vol/issue: 6(2), pp. 743-750, 2016.

[10] P. Li, “Utility accrual real-time scheduling: Models and algorithms,” *Doctoral dissertation*, Virginia Polytechnic Institute and State University, 2004.
[11] K. Kumar, *et al.*, “Resource Allocation For Real-Time Tasks Using,” *School of Electrical and Computer*, pp. 1-5, 2011.
[12] M. Jensen, *et al.*, “On technical security issues in cloud computing,” in *2009 IEEE International Conference on Cloud Computing,* pp. 109-116, 2009.
[13] S. Li, *et al.*, “Profit and penalty aware scheduling for real-time online services,” *IEEE Transactions on industrial informatics*, vol/issue: 8(1), pp. 78-89, 2012.
[14] R. Santhosh and T. Ravichandran, “Pre-emptive scheduling of on-line real time services with task migration for cloud computing,” in *Pattern Recognition, Informatics and Mobile Engineering (PRIME), 2013 International Conference on,* pp. 271-276, 2013.
[15] S. Deniziak, *et al.*, “Cost optimization of real-time cloud applications using developmental genetic programming,” in *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on,* pp. 774-779, 2014.

BIOGRAPHIES OF AUTHORS

Fereshteh Hoseini received the B.S.C degree in Information Technology from azad University of Arak, Iran in 2004, and M.S.C degree from Azad University of mahallat, Iran in 2015, respectively. Her research interests include Cloud Computing, Distributed Systems and Software Engineering

Mostafa Ghobaei Arani received the B.S.C degree in Software Engineering from IAU Kashan, Iran in 2009, and M.S.C degree from Azad University of Tehran, Iran in 2011, respectively. He is a PhD Candidate in Islamic Azad University, Science and Research Branch, Tehran, Iran. His research interests include Grid Computing, Cloud Computing, Pervasive Computing, Distributed Systems and Software Development.

Alireza Taghizadeh received his BS and MS in Computer Engineering from Iran University of Science and Technology (IUST) and Sciences and Research Branch of Azad University (IAU), Tehran, Iran. He obtained his PhD in the area of Network and Communication from University Sains Malaysia (USM) in 2013. He was formerly with Iran Telecom. Research Center (ITRC) as a Senior Research Engineer in IP-based core networks. He is currently a lecturer in Islamic Azad University of Parand. His research interests include Cloud computing, IP mobility, handover modeling and evaluation.