IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Revue d'Intelligence Artificielle
Vol. 34, No. 4, August, 2020, pp. 479-485
Journal homepage: http://iieta.org/journals/ria

# GAP: Hybrid Task Scheduling Algorithm for Cloud

Bhupesh Kumar Dewangan[1*], Anurag Jain[2], Tanupriya Choudhury[1]

[1] Department of Informatics, School of Computer Science, University of Petroleum and Energy Studies, Dehradun 248007, India
[2] Department of Virtualization, School of Computer Science, University of Petroleum and Energy Studies, Dehradun 248007, India

Corresponding Author Email: b.dewangan@ddn.upes.ac.in

https://doi.org/10.18280/ria.340413

**Received:** 9 March 2020
**Accepted:** 16 July 2020

***Keywords:***
*resource scheduling, completion time, cost, VM utilization, optimization algorithm*

**ABSTRACT**

Resource optimization is cost effective process in cloud. The efficiency of load balancing completely depends on how the infrastructure is utilizing. As per the current study, the resource optimization techniques are very costly and taking more convergence time to execute the task and load distribution among different virtual machines (VM). The objective of this paper is to develop a hybrid optimization algorithm to find the best virtual machine based on their fitness values and schedule different task to the fittest VM so that each task should get complete on time, and system can utilize the VM as well. The proposed algorithm is hybrid version of genetic (GA), ant-colony (Aco), and particle-swarm (Pso) algorithms, which is implemented and tested in amazon web service and compared with existing algorithms based on VM utilization, completion time, and cost. The proposed hybrid system genetic-aco-pso based algorithm (GAP) perform utmost while comparing with the existing systems.

## 1. INTRODUCTION

In a distributed environment, resource management is implemented under load balancing to minimize the cost, satisfaction of service level agreement (SLA), and efficient utilization of resources. Work found in literature does not meet the efficient resource scheduling parameters like server utilization, quality of service, and cost-effective workload balancing in minimum time collectively. The demand of users and the number of users is never certain. Therefore, there must be some mechanism to handle this growing and shrinking demand of users in such a manner so that it will not impact the quality of service, and also resources must be utilized inefficient manner. In this paper, authors have proposed a load balancing approach by merging the best features of Genetic [1] Particle Swarm Optimization [2], and Ant Colony Optimization [3] approach to ensure even load distribution, efficient usage of resources and satisfaction of service level agreement. Authors have also ensured that limitations of a genetic, ant colony and particle swarm approach are not inherited in the proposed hybrid approach (GAP). The structure of this paper is as follows: In section 2, authors have discussed the different scheduling algorithms to achieve load balancing in the cloud environment. Problem definition and methodology of the proposed new approach are discussed in sections 3 and 4 respectively. Details of the simulation environment, result, and their analysis are given in section 5. It is followed by the conclusion given in section 6.

## 2. TASK SCHEDULING IN CLOUD

A real-time scheduling of task in cloud can be optimized through many nature and bioinspired optimization algorithms like Honey Bee, Particle Swarm, Ant Colony, Ant Lion, Grey Wolf, Genetic and other evolutionary patterns. In this research work, the best features of Genetic, Particle-swarm, and Ant-colony have been used to propose a novel idea. Genetic-Algorithm by Holland [4] in 1992 has given a new approach to search by simulating the concept of human evolution. He used the concept of crossover, mutation and recombination to find the best option among a pool of feasible options. At the same time, this concept has also discarded those options, which are lying below a certain level. Based on this concept, many modified and extended approaches are now used in optimization problems. Another bioinspired optimization approach is Ant-colony, it is based on the forging nature of a group of ants. Ants are very social and they collectively find the best path to find foods. This behavior of ant finding the best food in the shortest path is also used in many optimization problems. In continuation of this, another bioinspired approach is the particle-swarm approach. Optimization problems based upon this approach simulate the behavior of swarm which hunting fish and searching food. The fitness function of particles [5] can be given by the formula given in Eq. (1).

$$f(i) = \frac{1}{\text{ExecT}} \tag{1}$$

where, *i* is several particles and belongs to [1, S], *S* is the total size of particles, and ExecT is task completion time. And completion-time by the task can be defined as:

$$ExecT = \sum_{n=1}^{k} \text{VM(m, n)} \tag{2}$$

The comparative analysis of the above-studied algorithms is presented in Table 1.

**Table 1.** Comparative analysis of algorithms

| Algorithm | Time Complexity | Space Complexity |
|---|---|---|
| Genetic-Algorithm | O(g(nm+n) | O(gnm) |
| Ant-Colony | $O(n^2)$ | $O(n^2)$ |
| Particle-Swarm | $O(n^2)$ | $O(n^2)$ |
| GAP | $nO(\log_n)$ | $nO(\log_n)$ |

Recent advancements of resource-management in the cloud:

In past decades, many new approaches have been developed by cloud researchers with some specific parameters. In 2020, a recent survey conducted in cloud resource management technique by author Dewangan et at. [6] it is categorized as follows:

2.1.1 Cost-aware approach

This parameter has been considered to schedule the resources at a minimal cost. This results in a low operational cost for both service providers and cloud users [7].

2.1.2 QoS-aware approach

Quality of service attracts to the cloud users to opt cloud services. The satisfaction ratio decides the user's trust [8].

2.1.3 SLA-aware approach

Service level agreement approach is one of the key factors to enhance user's trust [9].

2.1.4 Energy-aware approach

In this approach, the authors considered the power consumption rate, they provided novel approaches to reduce energy consumption, which is directly connected with cost [10].

2.1.5 Auction-based approach

Cloud user can approach the service as per demand, in this concept, the service provider does auctions, and users can opt for the required services [11].

2.1.6 Nature-bio-inspired approach

It is based on nature and bio-inspired algorithm to find the optimal resource for scheduling [12].

2.1.7 Profit-based approach

Focused on the profit of both cloud users and service providers [13].

2.1.8 Fault-tolerant approach

Avoid the unnecessary failures of scheduling the job and load balancing approach.

2.1.9 Optimization approach

This approach is used to optimizes several parameters as desire [14].

2.1.10 Autonomic computing

New era in resource management, using some self-characteristics for load balancing [15].

The outcome of this extensive study is presented in Figure 1. In this figure, the different parameters used in all recent approaches which are developed for the load balancing approach have been presented based on its utilization. Based on this review, the resource-utilization, cost, SLA, and execution time parameters are widely used.

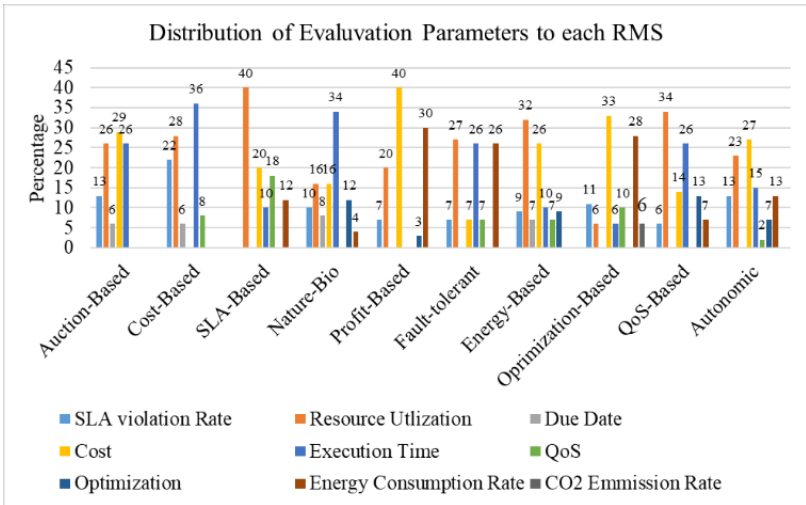


**Figure 1.** Extensive analysis of cloud resource management approaches based on parameters

## 3. PROBLEM DEFINITION

It is managing a problem diagnosed in a preceding algorithm or previous work interconnected to the task scheduling troubles. The sum of troubles that might be recognized is as: Local optima, Premature convergence [16], Parameter tuning, Stochastic procedures have trouble-based overall performance [17], and PSO algorithm does not have strong global seek capabilities.

### 3.1 Local optima

In done mathematics and laptop technological know-how, a nearby maximum appropriate of an optimization dilemma is a most suitable answer (either maximal or minimal) within a neighboring set of candidate solutions. This is in assessment to a worldwide most fine that is the choicest solution among all viable solutions, not the ones in a specific community of values. While the characteristic to be optimized is non-prevent,

it may be possible to appoint calculus to discover nearby optima. If the primary spinoff exists anywhere, it can be equated to zero; if the character has an unbounded domain, for a factor to be a local premiere it's miles critical that it fulfill this equation. Then the second one by-product check gives enough condition for the factor to be a local highest or nearby minimal [18].

### 3.2 Premature convergence

Premature convergence' [19] is common because of the loss of diversity.

3.2.1 Diversity

The degree of the amount varies i.e. wide range of diverse solutions [20] in the population, and the way one-of-a-type they're (distance between possibility answers).

3.2.2 Loss of variety

After the population converges, it will become very uniform (all answers resemble the tremendous one [21].

3.2.3 Motives

Too strong selective pressure in the direction of an awesome solution. An excessive amount of exploitation of current building blocks from the contemporary populace (e.g. via recombining them, or mutating them only barely) [22].

## 4. GAP: METHODOLOGY

The fitness value by GAP is given by the following condition:

$$fitness(i) = \frac{1}{\mathrm{x}} \tag{3}$$

where, x is total time need to complete the task, and i lies [0,1], and x can be obtained through:

$$x = \sum_{n=1}^{k} \mathrm{VM(m, n)} \tag{4}$$

where, $VM_{(m,n)}$ is the total time in the $n^{th}$ task to execute the $m^{th}$ virtual machine, where K is the total task allocated to the virtual machine. Then evaluate the value of pbest and gbest that is local/particle best and global best, by using the above equations.

$$pbestni(tn+1) = \begin{cases} pbestn(tn), & if\ f(pni(tn+1)) \le f(pni(tn)) \\ pni(tn+1), & if\ f(pni(tn+1) \ge f(pni(tn)) \end{cases} \tag{5}$$

$$gbestni(tn) = \begin{cases} \max\left(pbestn(tn)\right), if\ f(max(pni(tn))) \ge f(gbest(tn)) \\ else\ gbest(tn) \end{cases} \tag{6}$$

After finding the 2 high-quality values, the particle updates its speed and positions with the following equations:

$$Vi(tn+1) = w.Vi(tn) + C1.R1.(pbestni(tn) - pi(tn) + C2.R2.(pbestni(tn) - pi(tn) \tag{7}$$

$$pni(tn+1) = pni(tn) + Vi(tn) \tag{8}$$

where, *tn* is total iteration, *w* is weight, C1 and C2 are train and learn facts, and generally C1= C2 =2, R1 and R2 is a random value within [0, 1], f(pni (tn)) denotes particles fitness function, f(pni (tn+1)) denotes as fitness function during next iteration.

In each iteration the particles rang is specific to ($1 \le pni(tn) \le M$) then calculate its velocity then calculate and update the position of particles, then terminate if criteria are satisfied and show the gbest optimal value. If criteria are not satisfied then again go back to the second process and calculate the fitness value and execute the whole loop and repeat the process to optimize solution achieve.

### 4.1 Task initiation

The tasks are submitted to the cloud service provider for pertaining services. The task initiation process is presented through algorithm-01.

**Algorithm-01**

**Start**
Submit taks T←$T_1$, $T_2$, $T_3$, … $T_n$
***For*** each T ***do***
  **for** each T ***do***
    Calculate the priority value
    **If** priority≤ threshold ***then***
      Swap T with consecutive task
    ***End if***
  ***Enf for***
***End for***
**End**

### 4.2 Service monitor

Initially, Service Monitor has collected the information from Task Initiation to monitor continuously the task value based on priority (exaction time in this research) as shown in Algorithm-02: Service Monitor (SM).

**Algorithm-02**

**Start**
Set Taks T←$T_1$, $T_2$, $T_3$, … $T_n$
***for*** each T ***do***
  **if** ExecT($t_1$) ≤ ExceT($t_n$) ***then***
    Start Swap T with consecutive task
  ***End if***
  Alert
***End for***
**End**

A proposed hybrid method is a novel approach based on GA, ACO, and PSO to find the optimal solutions. Therefore the objective function of the proposed method is:

$$min.f(ExecT) \tag{9}$$

Subjected to: 0 ≤ ExecT ≤ 1.

### 4.3 GAP scheduling

The scheduling approach is presented in Algorithm 3.

**Algorithm-03**

**Start**
The pseudo code of the procedure is as follows
***For*** initiate each particles ***do***
    **For** repeat the following for each particle ***do***
        Calculate the fitness score of each particle
        **If** there is any improvement in the futness score relative to best fitness score(pbest) in the history ***then***
            update new value as the latest pbest
        ***end if***
    ***end for***
***end for***

Select the particle having best fitness score of all the particle as gbest

***For*** every particle ***do***
    Compute particle velocity according to equation (7)
***End for***

***If*** crossover particle ≥ 0.5 ***then***
    Apply one point crossover and update position of particle
    Evaluvate the newly generated crossover particle
***End if***

Repeat until minimum error or maximum iteration criterion is not attained.
***For*** every particle ***do***
    Arrange particles (VM's) in order
    ***If*** particles value ≤ threshold ***then***
        Separate from group
    ***End if***
    Produce Optimal particles (VM's)
***End for***

**End**

In the above algorithm, each particle is a virtual machine in this research.

### 4.4 Service executer and scheduling

The task initialized in task initiation are monitored and managed by service monitor, and ready to be scheduled with the virtual machine. The virtual machine is processed through GAP algorithm which is presented in algorithm 3, and the optimal VM is obtained for scheduling. The service executer executes the task and virtual machine and based on the priority value virtual machine assigned to the tasks through a round-robin algorithm.

## 5. RESULTS AND ANALYSIS

The proposed algorithm is initialized with 10 virtual machines and 600 tasks. The virtual machines, which have different weight values as per CPU, and RAM value in the current state. The following analysis has been done to evaluate proposed research work.

### 5.1 Waiting time

Waiting time represents the time for which task remains in the queue. Higher waiting time indicates the poor performance of the system in terms of load balancing and resource management. The average waiting time of tasks is judged when there are 100, 200, 300, 400, 500, and 600 tasks in the system. From Table 2, it can be concluded that the proposed approach GAP is performing better in the context of waiting time relative to its parent approach. This has also shown graphically in Figure 2. The waiting time is calculated through the following equation:

$$waiting\ time = turnaround\ time - burst\ time \quad (10)$$

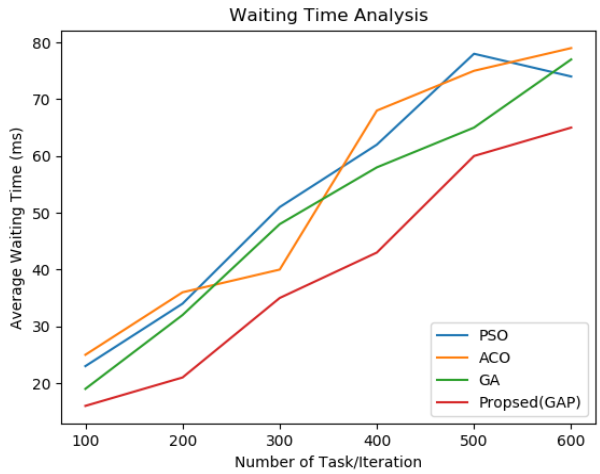


**Figure 2.** Waiting time (sec)

Table 2 shows the test results of the waiting time.

**Table 2.** Comparative analysis of waiting time (sec.)

| Tasks | GAP | PSO | ACO | GA |
|---|---|---|---|---|
| 100 | 16 | 23 | 26 | 18 |
| 200 | 21 | 34 | 35 | 32 |
| 300 | 36 | 50 | 37 | 45 |
| 400 | 43 | 64 | 65 | 54 |
| 500 | 60 | 70 | 72 | 62 |
| 600 | 65 | 75 | 79 | 54 |

### 5.2 Execution time

Execution time represents the time taken by the virtual machine to process the task. Less execution time represents that task is mapped with a suitable virtual machine in terms of its resource requirement. The average execution time of tasks is judged when there are 100, 200, 300, 400, 500, and 600 tasks in the system. From Table 3, it can be concluded that the proposed approach GAP is performing significantly better in the context of execution time relative to its parent approach. This has also shown graphically in Figure 3. Table 3 shows the execution time. The execution time is calculated through the following equations:

$$execution\ time = exit\ time - arrival\ time \quad (11)$$

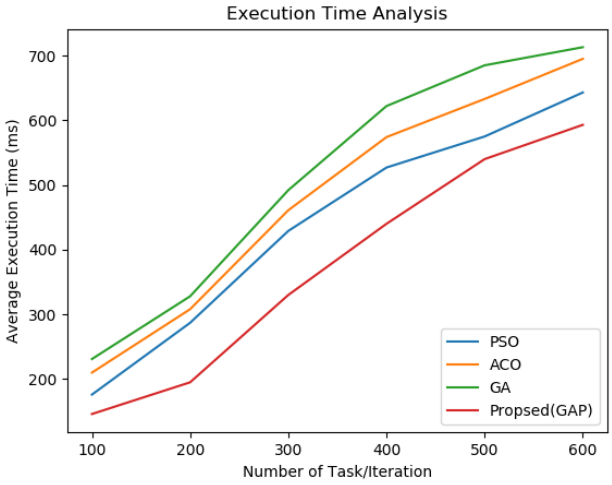


**Figure 3.** Execution time

**Table 3.** Comparative analysis of execution time (sec.)

| Tasks | GAP | PSO | ACO | GA |
|---|---|---|---|---|
| 100 | 146 | 176 | 209 | 232 |
| 200 | 195 | 287 | 309 | 328 |
| 300 | 329 | 430 | 464 | 495 |
| 400 | 440 | 525 | 577 | 626 |
| 500 | 540 | 578 | 636 | 688 |
| 600 | 593 | 647 | 695 | 716 |

### 5.3 Completion time

Completion time represents the sum of response time, waiting time, and execution time. Less completion time represents that the load balancing approach is performing better at all fronts. The average completion time of tasks is judged when there are 100, 200, 300, 400, 500, and 600 tasks in the system. From Table 4, it can be concluded that the proposed approach GAP is performing significantly better in the context of completion time relative to its parent approach. This has also shown graphically in Figure 4.

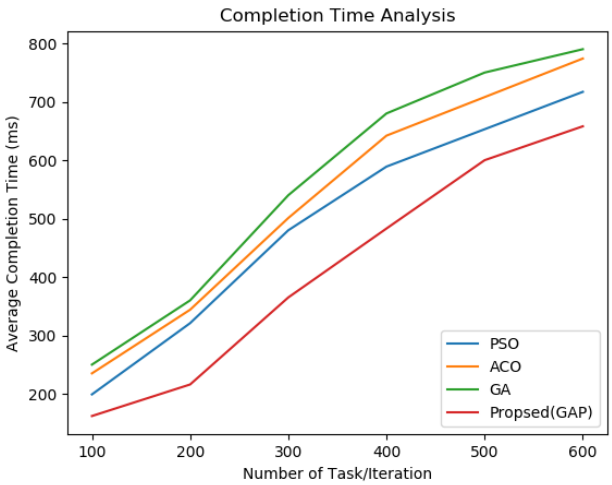


**Figure 4.** Completion time

While Table 4 shows the completion time of each algorithm. The completion time is calculated through the following equation:

$$completion\ time = execution\ tiwaiting\ time \qquad (12)$$

**Table 4.** Comparative analysis of completion time (sec.)

| Tasks | GAP | PSO | ACO | GA |
|---|---|---|---|---|
| 100 | 162 | 199 | 235 | 250 |
| 200 | 216 | 321 | 344 | 360 |
| 300 | 365 | 480 | 501 | 540 |
| 400 | 483 | 589 | 642 | 680 |
| 500 | 600 | 653 | 708 | 750 |
| 600 | 658 | 717 | 774 | 790 |

All three graphs shown in Figures 2, 3, and 4 are demonstrating that the proposed approach (GAP) is outshining the other three on the scale of waiting time, execution time, and completion time. This has happened due to better mapping of tasks with resources. Also, the proposed approach GAP map different tasks to different resources based on their need. This is resulting in less execution time of the proposed approach relative to the other three approaches. Moreover, better performance of the proposed approach GAP on the scale of waiting time and execution time is also resulting in better completion time relative to the other three approaches

### 5.4 Cost analysis

Cost is estimated based upon the time for which resources are used by the task. As it has analyzed from Figures 4, 5, and 6 that the proposed approach GAP is taking less waiting, execution, and completion time, this implies that it will also use the resources for less time. This has reflected in Table 5 for 100, 200, 300, 400, 500, and 600 tasks in the system. It has reflected graphically in Figure 5.

**Table 5.** Comparative analysis of cost ($)

| Tasks | GAP | PSO | ACO | GA |
|---|---|---|---|---|
| 100 | 4.8 | 5.1 | 5.9 | 7 |
| 200 | 6.4 | 6.9 | 7 | 10 |
| 300 | 10 | 12 | 13 | 15 |
| 400 | 14 | 15 | 17 | 19 |
| 500 | 16 | 19 | 21 | 22 |
| 600 | 19 | 21 | 23 | 24 |

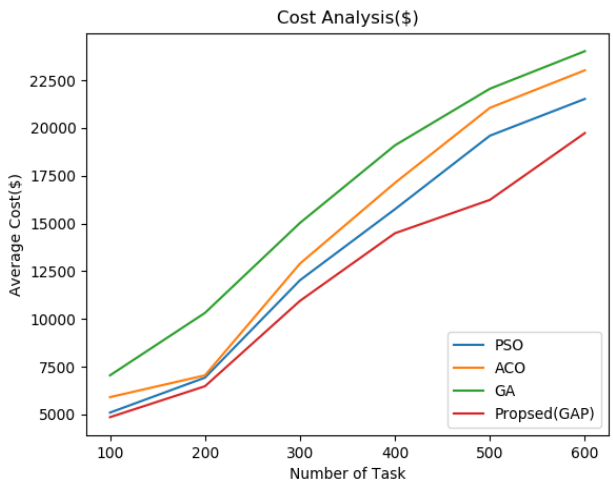


**Figure 5.** Cost analysis

### 5.5 VM utilization

In the proposed approach GAP, while mapping the task with a VM, the fitness value of VM is considered. The fitness value

is calculated based upon the present load and resources VM has. So in this manner, it ensures that all VM is used efficiently and no VM is overloaded. This results in better utilization of VM. This has reflected through the data shown in Table 6 for 100, 200, 300, 400, 500, and 600 tasks in the system. It has reflected graphically in Figure 6. Table 6 shows VM utilization.

**Table 6.** Comparative analysis of VM utilization (sec.)

| VM Id | GAP | PSO | ACO | GA |
|---|---|---|---|---|
| 1 | 5 | 4 | 3 | 2 |
| 2 | 5 | 4 | 4 | 3 |
| 3 | 4 | 4 | 3 | 3 |
| 4 | 5 | 3 | 4 | 4 |
| 5 | 5 | 4 | 5 | 4 |
| 6 | 6 | 5 | 4 | 5 |
| 7 | 6 | 4 | 5 | 4 |
| 8 | 4 | 4 | 4 | 5 |
| 9 | 4 | 4 | 5 | 5 |
| 10 | 6 | 5 | 5 | 5 |

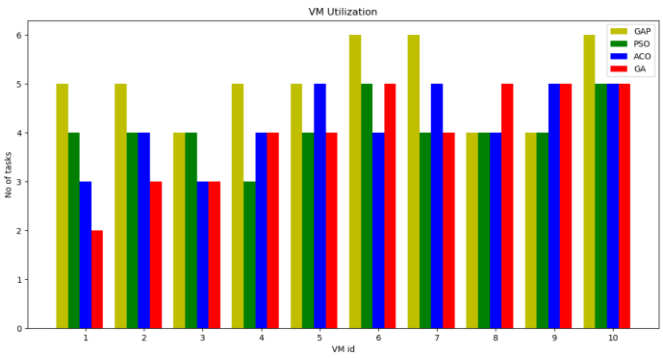


**Figure 6.** VM utilization

### 5.6 Throughput analysis

Throughput represents the number of tasks completed per unit time. Lesser the sum of response time, waiting time, and execution time, more will be the throughput. Now due to better performance shown on the scale of waiting time and completion time, it is obvious that the proposed approach GAP will also show better performance on the scale of throughput. This has reflected through the data shown in Table 7 for 100, 200, 300, 400, 500, and 600 tasks in the system. It has reflected graphically in Figure 7. While Table 7 shows throughput analysis of each algorithm.

It shows the comparison of different approaches for different number of tasks on the scale of throughput. Better performance on the scale of waiting time, completion time, execution time, resource utilization, cost and throughput results in satisfaction of service level agreement. It also creates a win-win situation for customer as well as cloud service provider.

**Table 7.** Comparative analysis of throughput (sec.)

| Tasks | GAP | PSO | ACO | GA |
|---|---|---|---|---|
| 100 | 0.6 | 0.5 | 0.4 | 0.4 |
| 200 | 0.9 | 0.6 | 0.6 | 0.6 |
| 300 | 0.8 | 0.6 | 0.6 | 0.6 |
| 400 | 0.8 | 0.7 | 0.6 | 0.6 |
| 500 | 0.8 | 0.8 | 0.7 | 0.7 |
| 600 | 0.9 | 0.8 | 0.8 | 0.8 |

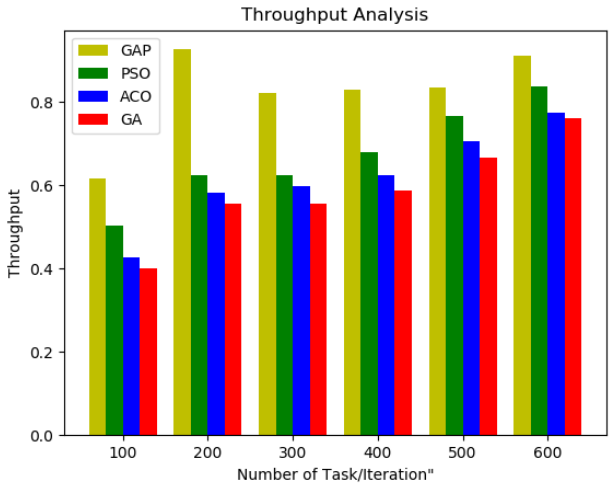


**Figure 7.** Throughput analysis

By embedding the proposed load balancing approach GAP in hardware like IP server, we can improve the performance and availability of the system. This will provide the capability to handle sudden traffic burst.

## 6. CONCLUSION

In cloud environment, satisfaction of SLA is prime objective. It can be achieved by providing services in minimum time in an efficient manner on a lowest cost by utilizing the resources in efficient manner. This will create a win-win situation to not only cloud user but also for cloud service provider too. In this paper, authors have proposed a hybrid load balancing approach for cloud environment by incorporating the best features of genetic, ant colony and particle swarm optimization algorithms. Authors have tested and compared the proposed algorithm GAP with existing GA, ACO & PSO algorithms in cloud environment developed using 10 virtual machines created in Amazon Web Service environment. It has found that proposed algorithm GAP has outperformed than GA, ACO & PSO based load-balancing algorithm on all the popular parameters.

## REFERENCES


[1] Ma, J., Li, W., Fu, T., Yan, L., Hu, G. (2016). A novel dynamic task scheduling algorithm based on improved genetic algorithm in cloud computing. Wireless Communications, Networking and Applications, Springer, New Delhi, pp. 829-835. https://doi.org/10.1007/978-81-322-2580-5_75

[2] Jana, B., Chakraborty, M., Mandal, T. (2019). A task scheduling technique based on particle swarm optimization algorithm in cloud environment. In Soft Computing: Theories and Applications, 742: 525-536. https://doi.org/10.1007/978-981-13-0589-4_49

[3] Azad, P., Navimipour, N.J. (2017). An energy-aware task scheduling in the cloud computing using a hybrid cultural and ant colony optimization algorithm. International Journal of Cloud Applications and Computing (IJCAC), 7(4): 20-40. https://doi.org/10.4018/IJCAC.2017100102

[4] Holland, J.H. (1992). Genetic algorithms. Scientific American, 267(1): 66-73.

[5] Engelbrecht, A.P. (2013). Particle swarm optimization: Global best or local best? 2013 BRICS congress on computational intelligence and 11th Brazilian Congress on Computational Intelligence, Ipojuca, pp. 124-135. https://doi.org/10.1109/BRICS-CCI-CBIC.2013.31

[6] Dewangan, B.K., Agarwal, A., Choudhury, T., Pasricha, A., Chandra Satapathy, S. (2020). Extensive review of cloud resource management techniques in industry 4.0: Issue and challenges. Software: Practice and Experience, 1-20. https://doi.org/10.1002/spe.2810

[7] Dewangan, B.K., Agarwal, A., Choudhury, T., Pasricha, A. (2020). Cloud resource optimization system based on time and cost. International Journal of Mathematical, Engineering and Management Sciences, 5(4): 758-768. https://doi.org/10.33889/IJMEMS.2020.5.4.060

[8] Naseri, A., Navimipour, N.J. (2019). A new agent-based method for QoS-aware cloud service composition using particle swarm optimization algorithm. Journal of Ambient Intelligence and Humanized Computing, 10(5): 1851-1864. https://doi.org/10.1007/s12652-018-0773-8

[9] Dewangan, B.K., Agarwal, A., Pasricha, A., Chandra Satapathy, S. (2019). Sla-based autonomic cloud resource management framework by antlion optimization algorithm. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 8(4): 119-123.

[10] Dewangan, B.K., Agarwal, A., Venkatadri, M., Pasricha, A. (2019). Energy-aware autonomic resource scheduling framework for cloud. International Journal of Mathematical, Engineering and Management Sciences, 4(1): 41-55. https://dx.doi.org/10.33889/IJMEMS.2019.4.1-004

[11] Zaman, S., Grosu, D. (2013). A combinatorial auction-based mechanism for dynamic VM provisioning and allocation in clouds. IEEE Transactions on Cloud Computing, 1(2): 129-141. https://doi.org/10.1109/TCC.2013.9

[12] Dewangan, B.K., Agarwal, A., Venkatadri, M., Pasricha, A. (2019). Design of self-management aware autonomic resource scheduling scheme in cloud. International Journal of Computer Information Systems and Industrial Management Applications, 11: 170-177.

[13] Dewangan, B.K., Agarwal, A., Pasricha, A. (2016). Credential and security issues of cloud service models. 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), Dehradun, pp. 888-892. https://doi.org/10.1109/NGCT.2016.7877536

[14] Dewangan, B.K., Agarwal, A., Venkatadri, M., Pasricha, A. (2019). Self-characteristics based energy-efficient resource scheduling for cloud. Procedia Computer Science, 152: 204-211. https://doi.org/10.1016/j.procs.2019.05.044

[15] Dewangan, B.K., Agarwal, A., Venkatadri, M., Pasricha, A. (2018). Autonomic cloud resource management. In 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC), Solan Himachal Pradesh, India, pp. 138-143. https://doi.org/10.1109/PDGC.2018.8745977

[16] Zhong, Z., Chen, K., Zhai, X., Zhou, S. (2016). Virtual machine-based task scheduling algorithm in a cloud computing environment. Tsinghua Science and Technology, 21(6): 660-667. https://doi.org/10.1109/TST.2016.7787008

[17] Xu, G. (2013). An adaptive parameter tuning of particle swarm optimization algorithm. Applied Mathematics and Computation, 219(9): 4560-4569. https://doi.org/10.1016/j.amc.2012.10.067

[18] Sivaram, M., Batri, K., Amin Salih, M., Porkodi, V. (2019). Exploiting the local optima in genetic algorithm using tabu search. Indian Journal of Science and Technology, 12(1): 9. https://doi.org/10.17485/ijst/2019/v12i1/139577

[19] Zhao, D., Zhang, H., Pan, J. (2019). Solving optimization of a mine gas sensor layout based on a hybrid GA-DBPSO algorithm. IEEE Sensors Journal, 19(15): 6400-6409. https://doi.org/10.1109/JSEN.2019.2909277

[20] Neumann, A., Gao, W., Wagner, M., Neumann, F. (2019). Evolutionary diversity optimization using multi-objective indicators. In Proceedings of the Genetic and Evolutionary Computation Conference, pp. 837-845. https://doi.org/10.1145/3321707.3321796

[21] Bharathan, G., Fernandez, T.T., Ams, M., Woodward, R.I., Hudson, D.D., Fuerbach, A. (2019). Optimized laser-written ZBLAN fiber Bragg gratings with high reflectivity and low loss. Optics Letters, 44(2): 423-426. https://doi.org/10.1364/OL.44.000423

[22] Raju, M., Gupta, M.K., Bhanot, N., Sharma, V.S. (2019). A hybrid PSO–BFO evolutionary algorithm for optimization of fused deposition modelling process parameters. Journal of Intelligent Manufacturing, 30(7): 2743-2758. https://doi.org/10.1007/s10845-018-1420-0