International Journal of Parallel Programming (2020) 48:549–565
https://doi.org/10.1007/s10766-018-0590-x

Resource Allocation in Cloud Computing Using SFLA and Cuckoo Search Hybridization

P. Durgadevi[1] · **S. Srinivasan**[2]

Received: 7 May 2018 / Accepted: 30 July 2018 / Published online: 25 August 2018

Abstract

The 'cloud computing' technology is requisite for modern technology. It has a notable facet called Resource Allocation. This given paper proposes Hybridized Optimization algorithm that is the combination of 'Shuffled Frog Leaping Algorithm' (SFLA) and 'Cuckoo Search' (CS) Algorithm. This technique overcomes the limitations of the existing works like HABCCS algorithm, GTS algorithm task, krill herd algorithm, also combines the advantages of SFLA and CS. In this method, SFLA section performs the preceding steps; initializing the request size, generating requests, and estimate fitness value of SFLA, sorting, dividing and evaluating the requests of user. The SFLA encompasses the advantage of higher speed convergence and easier implementation, with the capacity of having global optimization and are utilized widely in numerous areas. Then, CS algorithm executes operations like initializing, generating, evaluate fitness function, modification and then evaluating the new solutions. The CS algorithms possess the advantage of easier evaluation and it is utilized in complex situations. In this given system, the request speed, sizes are evaluated. Those evaluations are utilized in allocating the resources on the server-side. Less computed times are consumed in this technique. An experimental outcome displays that the approach performs well in contrasting with other related approaches.

Keywords Cloud computing · Shuffled frog leaping algorithm (SFLA) · Cuckoo search (CS) algorithm · Resource allocation

✉ P. Durgadevi
durgadevi@rmkcet.ac.in

S. Srinivasan
ssn.cse@rmd.ac.in

1 R.M.K. College of Engineering and Technology, Puduvoyal, India

2 R.M.D. Engineering College, Gummidipoondi, India

1 Introduction

A computing method centered on the Internet termed Cloud Computing (CC), taking pros of the storage resources of cloud servers, attracts millions of users [1], and shares the hardware resources. It is a striking computing model as it lets for the provision of resource on-demand [2]. Information is delivered centered on the necessities of computers and other equipment. It is mainly the upsurge in Internet-related services, use, and delivery model. Internet proffers the virtualized and also dynamically scalable resources [3]. Services are proffered by the Internet to the consumers as Something-as-a-Service. The 3 service models are 'Infrastructure as a Service' (IaaSmodel), 'Platform as a Service' (PaaSmodel) together with 'Software as a Service' (SaaS model) [4]. In CC, auction stands as a technique of selling clouds resources on a public forum via competitive bidding. Generally, a cloud auction requires cloud users to bid aimed at their required resources [5].

In this cloud environment, 2 actors play a notable role. One is cloud users and the other is cloud providers. The providers possess numerous computing resources on the huge data centers and on a 'pay-per-use' method; they rent those resources to the users to augment the revenue by accomplishing enriched resource usage. Resources also remain as demand for the cloud users. Applications with dynamic nature are predicted by them. The cloud users have applications with changing loads, rent the resources from the providers and activate their applications with least expenses. Every user desires multiple resources intended for a definite task or cloudlet that intensify the performance and finished on time [6].

This computing is successfully utilized by organizations as it offers extensive solutions along with pros to business say, increase flexibility, scalability, agility, reduces costs and higher efficiencies [7]. It is utilized and applied in organizations and it aids to enhance revenue every year.

Resource management encompasses the diversified stages of workloads and resources as of workload submission to final workload execution. In Cloud, this management comprises of 2 stages namely: (1) resource scheduling, (2) resource provisioning [8]. RA is a notable aspect of the grid and also distributed computing [9].

RA that implements a utility-centered technique is requisite at diversified levels of utility-based grid computing. Also, utility-based computing becomes widespread in end-user applications together with enterprise applications. A utility cloud proffers services to users having diversified resource needs (Fig. 1). The distribution of services in the cloud is grounded on auction-based models, combinatorial models, and other economic models, which are assured to augment the cloud owner's revenue. Those economical models enhance user demand and intensify revenue by the provision of resources [10].

Several latest research-works proposed energy-aware RA computing methods intended for distributed computing. Numerous surveys of RA of CC are reported [11]. However, they not concentrated on the core difficulties of energy efficient RA in clouds. The former works fail in clearly addressing the energy effectual resource management issue from the perspective of application engineering.

The cloud user gets good quality services from their provider with a reasonable cost. The cost and quality of the services are contingent on their source allocation

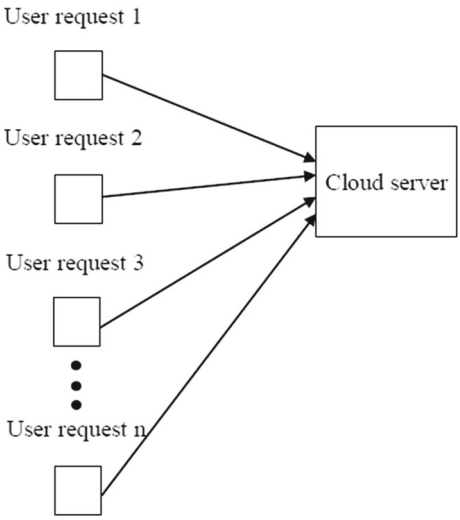

Fig. 1 Resource allocation request

process in the specific service environment. The providers dispense the resources to the clients in an optimum way. There are copious RA models that are utilized in the cloud environment. Those models utilize specific algorithms and also methods for this reason. The succeeding part shows the survey done on the former works of RA models in the cloud environment. It is chiefly concentrated on RA methodologies [12].

Therefore, evaluation of RA application in CC is proposed. This mechanism is appropriate for the well performed RA contingent on the utilization of user request. This RA diminishes execution time additionally power consumption [13].

A fresh technique to increase the system performance through a hybrid algorithm is apprised in the proceeding sections. The existing techniques were observed to debase the performance.

The draft structure of this paper is systematized as Sect. 2 surveys the associated works regarding the method proposed. In Sect. 3, a brief illustration of the proposed methodology is proffered, Sect. 4, explore the Investigational outcome and Sect. 5 deduces the paper.

2 Literature Survey

Sharma and Guddeti [14] proffered a Euclidean distance centered multiple objective RA in the appearance of virtual machines (VMs) and VM migration policy at the data centre. Additionally, the distribution of 'VMs' to 'PMs' (Physical Machines) was done by the suggested hybridized approach of Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) inferred as HGAPSO. The suggested algorithm centered RA and VM migration saved the energy usage and diminished the resource wastage in addition evades SLA violations in the cloud data center. To check the suggested HGAPSO algorithms' and also VM migration techniques' performance in concerning energy usage, resources usage together with SLA violation, conducted several experiments in

heterogeneous based and homogeneous based data center environments. The suggested work was as well compared with the bound and branch based exact algorithm. The experimental outcome displayed the dominance of VM migration together with the HGAPSO techniques over exact algorithm in concerning with optimum resources usage, energy efficacy, together with SLA violation.

Kayalvili and Selvam [15] presented an RA method in CC for handling the cloud resource and dynamic configuration for all sorts of hardware resources that were encompassed in virtualization technology to deliver services to users with VM as the fundamental unit, a central role was performed by virtualization technology. The need of using VM was to comprehend the ideal outcome by varying the placement and the layout of all the VM. The dispersion of the cloud resources to that of the user-centered on the request was an NP-Hard issue. Heuristic methodologies were employed for optimization of RA. The SFLA encompasses the pros of higher speed convergence and easier implementation with the capacity of having global optimization and were utilized widely in several areas. The GA was the iterative stochastic optimization grounded methods that were centered on the natural selection principles in tandem with their evolution. For this work, the hybridized SFLA-GA was utilized for attaining the distribution of optimum resources in the CC environment.

Pillai and Rao [16] proffered an RA mechanism in the cloud, centered on the coalition formation principles and the game theory's uncertainty principle. They contrasted the outcomes of applying this mechanism with prevailing RA methods that were deployed on the cloud. It displayed that this method of RA by coalition-formation of the machines on the cloud direct to better resource usage and also higher request satisfaction.

Mireslami et al. [17] proffered a runtime friendly and cost effectual algorithm that diminished the deployment cost whilst concerning the QoS performance needs. This algorithm proffered an optimum choice, from customers' view, for using web applications on the cloud arena. The multiple targeted optimization algorithms diminished cost and increased the QoS performance instantaneously. The recommended algorithm was validated by several experiments on diverse workload situations utilized in 2 distinct cloud providers. The outcomes displayed that the recommended algorithm determined the optimum combination of cloud resources that delivered a stabilized tradeoff betwixt QoS performance and deployment cost in comparatively low runtime.

Zheng et al. [18] proffered a hybridized energy-aware RA approach to assist requestors in acquiring satisfied and energy-effectual manufacturing services. The problem on the energy-aware RA in CMfg was initially summarized. Then a localized selection strategy grounded on fuzzy similarity degree was employed to attain suitable candidate services. Multiple targeted mathematical designs aimed at the energy-aware services configuration were recognized and the non-dominated sorting 'genetic algorithm' was employed to perform the combined optimization process. Additionally, a technique aimed at order preference via similarity to a perfect solution was employed to find the optimum composite services. Finally, a normal case study was exemplified to confirm the recommended approach's effectiveness.

Chen et al. [19] presented an outline that optimized and evaluated RA strategies quantitatively. By utilizing the statistic model checker called UPPAAL-SMC together with the supervised learning methods, this outline: (1) performed difficult QoS queries

on RA instances concerning resource variations; (2) quantitative and qualitative contrasts amongst RA strategies were done; (3) assisted the tuning of parameters to expand the complete QoS; and (4) supported the quick optimization of the complete workflow QoS under resource variations and customer needs. The outcomes validated that the automatic framework supported the 'Service Level Agreement' (SLA) and workflow RA optimization effectually.

Di et al. [20] analyzed in detail the suggested optimal algorithm diminishing task implementation length with dividable resources as well as payment budget. Cloud users were capable to formulate an assortment of tasks centered upon 'off-the-shelf' web services. Experiments demonstrated that the task execution lengths on this algorithm were always near to their hypothetical optimal values, also in competitive circumstances with restricted available resources. Also a higher-level of fair treatment on the RA was observed among all tasks.

3 Flowchart for Propose System

This given paper proposes Hybridized Optimization algorithm which is a combination of SFLA and CS Algorithm. In this method, SFLA section performs preceding steps; initializing the request size, generating requests, and estimating fitness value of SFLA, then sorting, dividing and evaluating the requests of the user and modifying the request position and finally evaluating the new solution. Then shuffle the request and the resources are finally assigned to the CC server grounded on the shuffled requests.

Finally, the hybridized optimization algorithm diminishes the waiting time of the user request. The requests are then perfectly allocated on the cloud server. Diagram for the technique which is proposed is established in Fig. 2.

3.1 User Request

In the primary phase, the requests are conveyed to the cloud environment from the users. Then the users are assigned as,

$$UR_q(x) = \left\{UR_{q1}, UR_{q2}, UR_{q3}, \ldots, UR_{qn}\right\} \tag{1}$$

where $UR_q(x)$ = Number of requests, x = {1, 2, 3,…,n}. The requests are gathered from the users for every particular time interval by the real-time request queue. The evaluation and also the 'fitness functions' are conferred below,

3.2 Analyze Request

In the secondary phase, the user request is analyzed one by one contingent on the request speed, weight, and energy. After that, the optimization algorithm process starts as below.

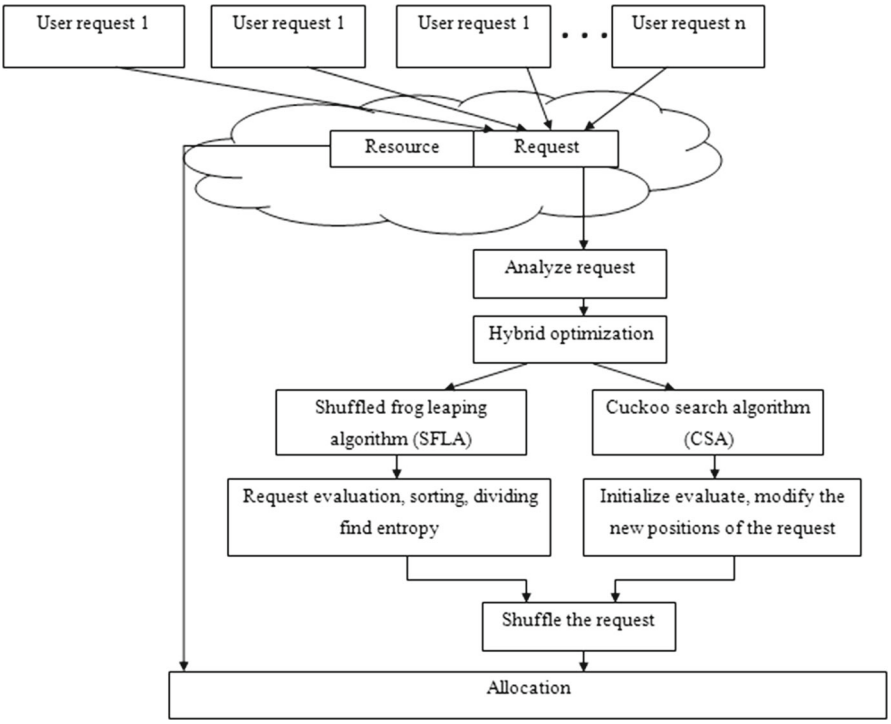

Fig. 2 Architecture of the proposed system

3.3 Hybrid Optimization

Hybrid Optimization is the amalgamation of SFLA and CSA. Hybrid SFLA-CSA used for attaining the allotment of optimal resources in the CC. It gives better optimum solution compared to the existing systems.

3.3.1 Shuffled Frog Leaping Algorithm

The SFLA stands as a meta-heuristic optimization method that is centered upon observing, emulating, and also modeling the activities of a collection of frogs hunting for the maximum available food location. SFLA was initially developed by Eusuff and Lansey in 2003, can well be employed to solve numerous intricate optimization problems that are nonlinear, non-differentiable, and also multi-modal. The SFLA unites the genetic-based memetic algorithm's advantages with that of the social behavior-centered PSO algorithm.

The individual frogs encompass ideas inside each memeplex that can well be defiled by the other frogs' ideas. Subsequent to a defined number of 'memetic evolution' phases, ideas proceed betwixt memeplexes in the shuffling process. The 'local search' together with the 'shuffling process' continues till the defined convergence norms are fulfilled.

The primary step of SFLA is that the populace with P frogs is randomly created within the possible search space. The location for i th frog is signified as,

$$X_i = (X_{i1}, X_{i2}, \ldots, X_{iD}) \tag{2}$$

where X_i signifies the frog group, D signifies the variables. Then, Conferring to the fitness, the frogs get organized in descendent order. Subsequently, the whole populace is split into m subsets conferred to as memeplexes where each contains n frogs,

$$(i.e.,\ P = m \times n) \tag{3}$$

where m is the subset conferred and n is the number of frogs. The approaches of this partitioning are as follows:

The 1st memeplex is taken by the 1st frog; the second one is conquered by the 2nd frog, where the mth memeplex is conquered by the mth frog, and the $(m + 1)$th frog return to the 1st memeplex, simultaneously.

In all memeplex, the locations of frogs comprising the 'best-fitness' together with 'worst-fitness' are acknowledged as,

$$X_b = bestfitness \tag{4}$$

$$X_w = worstfitness \tag{5}$$

Also, the location of frogs comprising the best global-fitness is acknowledged as.

$$X_g = globalfitness \tag{6}$$

.

Then, within every memeplex, a course identical to the PSO algorithm is implemented to enhance only the frog having the worst fitness (not all frogs) in every single cycle. Consequently, the location of the frog that had the worst fitness bounds to the location of the best frog, as below:

$$D_i = rand \times (X_b - X_w) \tag{7}$$

where i=1 to m, then D_i stands as the number of variables, r means random frogs. Accept D_i if it is $D_{\min}$ and $D_{\max}$, if not set to minimal or maximal limits of D_i. The new position is computed by

$$X_w^{new} = X_w^{old} + D_i; \left(D_{\min} < D_m < D_{\max}\right). \tag{8}$$

Again compute fitness of this frog. If the fitness of X_w^{new} is above the fitness of X_w^{old} after that accept the X_w^{new}. Else arbitrarily generate the new frog in place of X_w within the acceptable frog limits.

If these total processes yield a better solution, it substitutes the worst frog. Or else, repeat the calculations in (7) and (8) and replace X_b by X_g. If it has no development,

a new solution is arbitrarily created within the possible space to substitute it. The calculations continued for specific iterations. Consequently, SFLA instantaneously implements an independent local search on every memeplex utilizing a course identical to the PSO algorithm. After continuous memetic evolutionary phases within every memeplex, the results of such memeplexes are substituted into the fresh population shuffling process.

The process called shuffling stimulates the global information interchange amongst the frogs. Then, the populace is sorted in sequence of declining performance value and updates the population finest frog's position; the frog group repartitioned into memeplexes, furthermore progress the evolution within every memeplex till the conversion situations are satisfied. Typically, the convergence criterion is illustrated as below:

The comparative changes in the global frog fitness within the successive shuffling iterations are less considering the pre-specified tolerance. The supreme shuffling iterations are attained. Finally, the solution criterion is,

$$\left[\left|X_w^{new}\right| - \left|X_w^{old}\right|\right] < \in \tag{9}$$

where X_w^{new} is the new fitness and X_w^{old} is the old fitness and $\in$ is the convergence tolerance.

3.3.2 Cuckoo Searching Algorithm

This algorithm is grounded on the cuckoo bird's breeding behavior. It encompasses three basic optimization rules.

1. Each of the cuckoos at an instance lays a solitary egg and puts it at an arbitrarily selected host nest.
2. The finest nests having highest-quality eggs will proceed to the subsequent generations.
3. The available number of host's nests is set, and the host bird may perhaps find out the unfamiliar egg that belongs to a cuckoo with a probability $P_a \in [0, 1]$ and builds a new solution.

CS utilizes a balanced blend of a local and a global explorative random walk, which is managed via a switching parameter P_a. A 'greedy strategy' is utilized subsequent to every random walk, to select better solutions as of the present and new produced solutions as per their fitness values.

Performed the global random walk by utilizing Levy flights as follows,

$$X_i^{t+1} = X_i^t + \alpha \oplus Levy(\lambda) \tag{10}$$

where X_i^{t+1} is the new nest with the high-fitness value, X_i^t is the nest wherein the cuckoo primarily lives. Where α ($\alpha > 0$) stands as the step size connected to the optimization problem scale, and also the product $\oplus$ indicates the entry-wise multiplication. Levy

flights fundamentally offer random walks, the random steps of which are drawn as of a Levy distribution aimed at large steps:

$$Levy \sim t^{-\lambda}, (1 < \lambda \leq 3) \tag{11}$$

where t stands as the time of completing the task and λ means random walk in addition to random steps that encompass an infinite variance and also mean.

$$X_i^{t+1} = X_i^t + \alpha \oplus t^{-\lambda}. \tag{12}$$

After Levy flights random walk, CS persists to produce new solutions regarding biased/selective random walk which utilize a crossover operator. In consideration of the probability of cuckoos being discovered, make a fresh solution using a crossover operator:

$$X_i^{'} = \left\{ \begin{array}{l} X_i + r \cdot (X_{r1} - X_{r2}), if(rand[0, 1] > P_a) \\ X_i \, otherwise \end{array} \right\} \tag{13}$$

where $r1$, $r2$ are mutually different random integers; r denotes the scaling factor that stands as a uniformly distributed arbitrary number on the interim [0, 1]. The next generation solution is selected from X_i and $X_i^{'}$ as per their fitness values. At the ending of each iteration process, the finest solution attained so far is updated.

3.3.3 Sfla-csa

The proposed work is employed for evading the knapsack problem and increasing the speed for allocating request on the resource. The SFLA-CSA algorithm architecture appears in Fig. 3.

The Entropy formula is utilized to compute the request weight, speed, and sizes. This entropy calculation is mainly used for utilizing the resource request. The generalized entropy method is stated as below step,

$$E = -\sum_{i=1}^{m} X_i \log X_i \tag{14}$$

where E is the entropy of the request, X_i is the user request from each set. By means of employing the request, the entropy value is found by employing the Eq. (14). For every request, the entropy value is calculated individually and the whole values are summed to get the better solution. Then, the entire user requests are utilized, and finally allocating the resource (Fig. 4).

Algorithmic Description

Step 1 Initialize the population and the initial $X_i(t)$ value is initially lived in the nest
Step 2 Produce the population randomly on the nest (i.e., UR)

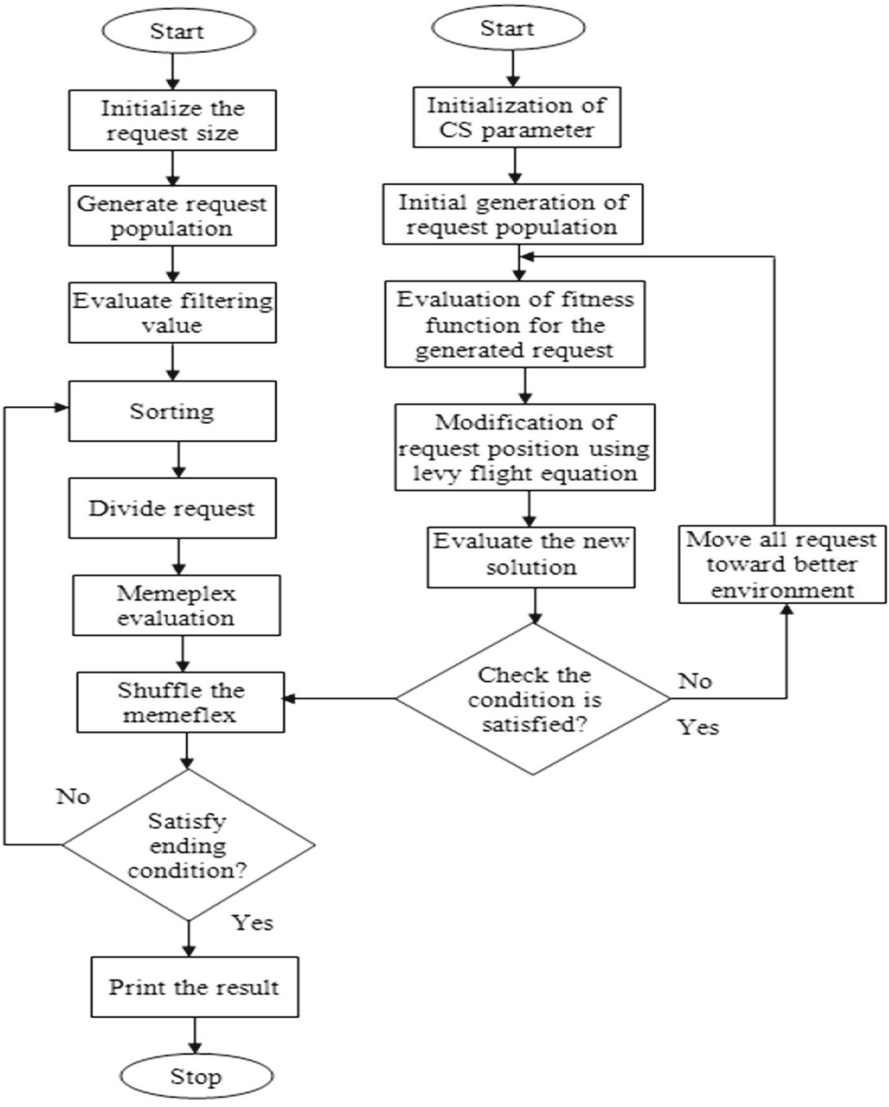

Fig. 3 SFLA-CSA algorithm

Step 3 Then assess the fitness values of each (X_i)

Step 4 Find an entropy value using the Eq. (14), calculate the entropy values utilizing the request. Find an entropy value for each request individually and add the whole value you will obtain the correct solution

Step 5 Then the while do condition in t time max generation of cuckoo nest ought to be calculated

Step 6 Generate cuckoo randomly utilizing *Levy flights* equation, evaluate the random request and replace the nest

Begin

1. Initialize the population size, number of memplex m, total frogs $P = M \times n$
2. Generate required population (X_i), i=1 to m by random generation.
3. Evaluate the fitness values for each X_i
4. Calculate the entropy using the eggs
5. Determine the best nest with the best fitness value.
6. **while** $t \leq$ max generation do

for i= 1,2,…..m do

Generate cuckoo based on Levy flights

Evaluate the fitness value $F_i = f(X_i)$

Choose the random nest X_j

if ($F_j > F_i$) then

Replace X_i with X_j

End if

End for

Abandon a fraction P_a of the worst nests that it moves towards to build nest in better environment

Keep the best nest with quality solution.

Rank the nest an d find the best one

End while

End

Fig. 4 Pseudo code for the hybrid optimization SFLA-CSA algorithm

Step 7 After the modification to the new nest, it gives a better solution
Step 8 Then, check the condition, if it is fulfilled then move it to the shuffled method otherwise, move the nest again to a better environment
Step 9 Condition satisfied, the memeplex is moved
Step 10 Print the result.

4 Results and Discussion

The proposed work is utilized to lessen the knapsack issues in RA. Moreover, their time and capacity of processing elements also get reduced. Furthermore it is executed in JAVA with CloudSim and utilizing the database is the yardstick for basic scheduling troubles.

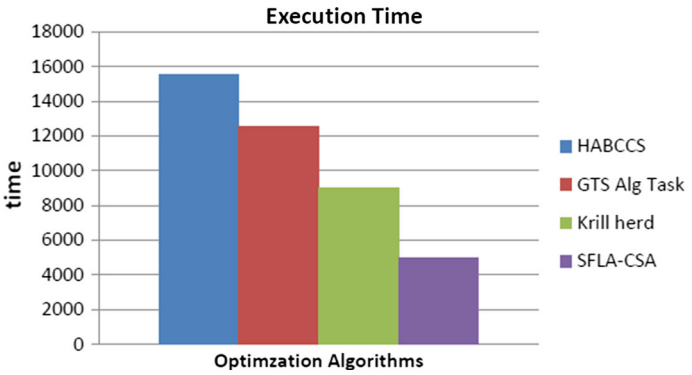

Fig. 5 Comparison analysis of proposed system with existing system for resource execution time

4.1 Execution Time

Comparing with other existing systems, the executing time is small for the proposed system. The SFLA-CSA shows better execution time compared to the HABCCS algorithm, GTS algorithm task, krill herd algorithm and SFLA-CSA. The executing time is computed as,

$$E_T = E(t) - F(t) \tag{15}$$

where E_T is the computational time, $E(t)$ is the ending time of the process, $F(t)$ is the beginning time of the process.

A comparison on the executing time between the existing and the proposed one is exhibited in Fig. 5. In Fig. 5 the names of different optimization algorithms are taken alongside the horizontal axis additionally the executing time is taken alongside the vertical.

4.2 Throughput

Throughput alludes to the quantity of information transported as of one site to a disparate one in a specified quantity of time. And it is utilized to gauge the performances of hard drives, RAM, and Internet and also network connections. The proposed system's throughput should be bigger than the existing system. The throughput is estimated as

$$T_t = \frac{I_t}{t} \tag{16}$$

where I_t is the information transported and t means the quantity of time. The throughput of different algorithms is compared and exhibited in Fig. 6. In Fig. 6 the name of the different algorithms is taken in the horizontals axis additionally the amount is taken in the vertical axis.

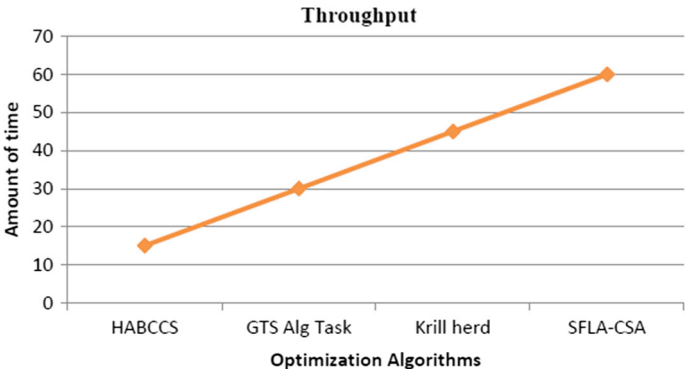

Fig. 6 Comparison analysis of proposed system with existing system for resource throughput

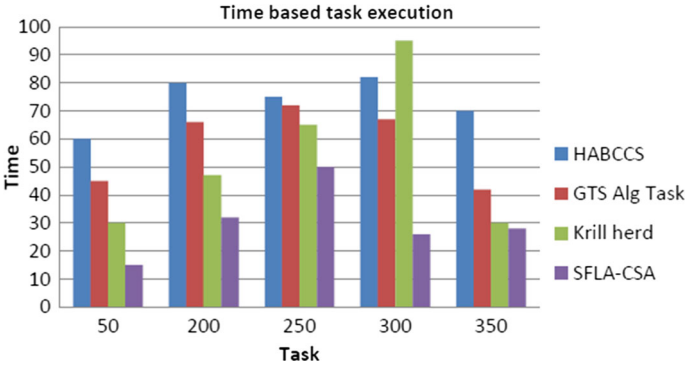

Fig. 7 Comparison analysis of proposed system with existing system for resource throughput

4.3 Time Based Task Execution

Time-based task execution means to schedules the task aimed at execution, time-based or else event-centered triggers are set on a task which starts its execution. In the proposed system, the time-based task execution is contrasted with the prevailing system. The SFLA-CSA optimization will provide a better result. A comparison on the time taken for execution of task is displayed in Fig. 7. In Fig. 7 the task is taken along the horizontals axis and also the time is taken along the verticals axis.

4.4 Turnaround Time

It stands as the total time utilized for execution betwixt the submission of a task and the return of the entire outcome to the consumer. It may differ for miscellaneous programming languages contingent on the developer of the software or the program. It

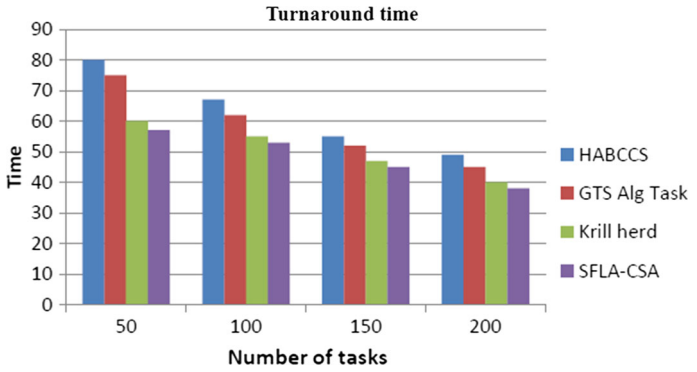

Fig. 8 Comparison analysis of proposed system with existing system for resource turnaround time

manages the total time taken for a program to give the requisite output to the consumer after the program begins.

$$T(t)_{Avg} = C(t) - A(t) \tag{17}$$

where $T(t)_{Avg}$ means 'average turnaround time', $C(t)$ means completion time of the task then $A(t)$ means arrival time. The proposed work, when correlated with the existing system, gives a better outcome. A comparison on this is exhibited in Fig. 8. In Fig. 8 the number of tasks is taken along the horizontals axis besides the time is taken in the verticals axis.

4.5 Waiting Time

It means the time taken between the process request and consummation of the request the waiting time is computed as below,

$$W(t)_{Avg} = T(t)_{Avg} - B(t) \tag{18}$$

where $W(t)_{Avg}$ means 'average waiting time', $T(t)_{Avg}$ means 'average turnaround time', then $B(t)$ means 'burst time'. The proposed work is better than the existing system. A comparison on the waiting time is given in Fig. 9. In Fig. 9 the number of tasks is taken along the horizontals axis besides the waiting time is taken along the verticals axis.

4.6 Allocation Mechanism

Utilizing this Mechanism, the existing system and the proposed system are compared here. Overall, the hybrid SFLA-CSA optimization algorithm is better in allocating resource in the cloud environment. A comparison on this mechanism is exhibited in Fig. 10. In Fig. 10 the optimization algorithm is taken in the horizontals axis additionally the allocation percentage is taken along the verticals axis.

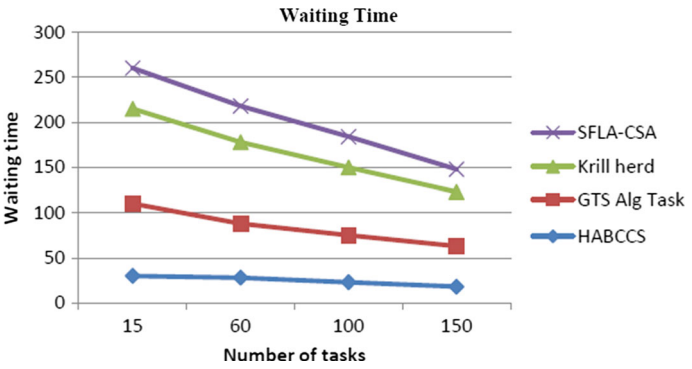

Fig. 9 Comparison analysis of proposed system with existing system for request waiting time

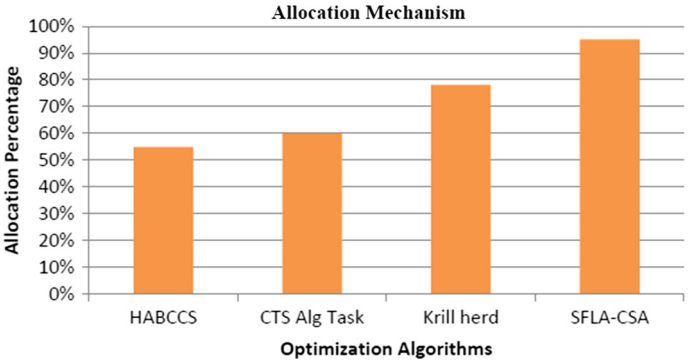

Fig. 10 Comparison analysis of proposed system with existing system for RA mechanism

Table 1 Overall comparison of results

Optimization algorithm	Execution time	Throughput	Allocation percentage
HABCCS	15,900	15	55
GTS Alg Task	12,200	30	60
Krill herd	9000	45	79
SFLA-CSA	5000	60	95

The overall results are tabulated on the subsequent table. Table 1 enables to correlate the performance related metrics.

5 Conclusion

Here, SFLA-CSA algorithm is formulated to diminish the knapsack issue for RA mechanism on CC environment. In the proposed system, using hybrid SFLA and cuckoo searching algorithm optimization algorithms, the optimization issue is solved.

Execution of the proposed RA is done on the working platform of JAVA with CloudSim. The proposed work and the prevailing system are compared. The execution time for the proposed work was observed to be 5000 ms, the throughput was 60 s, the time taken for task execution was low, the 'turnaround time' was lower the allocation percentage was 95. The existing system like HABCSS, CTS Alg task, krill herd showed high execution time, throughput, turnaround time and percentage of allocation these may results in stern degradation in the overall performance of the system which is evident from the experimental results. The experiment showed the dominance of the proposed works over the prevailing methodologies and waiting time, allocation mechanisms stand better to the prevailing system. This work could be ameliorated by improving the proposed work's performance.

References

1. Xiaoying, T., Dan, H., Yuchun, G., Changjia, C.: Dynamic resource allocation in cloud download service. J. China Univ. Posts Telecommun. **24**(5), 53–59 (2017)
2. Pradhan, P., Prafulla, B.K., Ray, B.N.B.: Modified round robin algorithm for resource allocation in cloud computing. Procedia Comput. Sci. **85**, 878–890 (2016)
3. Mingxin, W.: Research on improvement of task scheduling algorithm in cloud computing. Appl. Math. Inf. Sci. **9**(1), 507–516 (2015)
4. Lee, H.M., Jeong, Y.S., Jang, H.J.: Performance analysis based resource allocation for green cloud computing. J. Supercomput. **69**(3), 1013–1026 (2014)
5. Madni, S.H.H., Latiff, M.S.A., Coulibaly, Y.: Recent advancements in resource allocation techniques for cloud computing environment: a systematic review. Clust. Comput. **20**(3), 2489–2533 (2017)
6. Kumar, N., Saxena, S.: A preference-based resource allocation in cloud computing systems. Procedia Comput. Sci. **57**, 104–111 (2015)
7. Xue, C.T.S., Xin, F.T.W.: benefits and challenges of the adoption of cloud computing in business. Int. J. Cloud Comput. Serv. Arch. (IJCCSA) **6**(6), 1–15 (2016)
8. Singh, S., Chana, I.: A survey on resource scheduling in cloud computing: issues and challenges. J. Grid Comput. **14**(2), 217–264 (2016)
9. Ergu, D., Kou, G., Peng, Y., Shi, Y., Shi, Yu.: The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment. J. Supercomput. **64**(3), 835–848 (2013)
10. Kolhar, M., Abd El-atty, S.M., Rahmath, M.: Storage allocation scheme for virtual instances of cloud computing. Neural Comput. Appl. **28**(6), 1397–1404 (2017)
11. Hameed, A., Khoshkbarforoushha, A., Ranjan, R., Jayaraman, P.P., Kolodziej, J., Balaji, P., Zeadally, S., et al.: A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. Computing **98**(7), 751–774 (2016)
12. Sudeepa, R., Guruprasad, H.S.: Resource allocation in cloud computing. Int. J. Mod. Commun. Technol. Res. **2**(4), 19–21 (2014)
13. Zuo, X., Zhang, G., Tan, W.: Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud. IEEE Trans. Autom. Sci. Eng. **11**(2), 564–573 (2014)
14. Sharma, N., Guddeti, R.M.: Multi-objective energy efficient virtual machines allocation at the cloud data center. IEEE Trans. Serv. Comput. (2016). https://doi.org/10.1186/s13677-017-0086-z
15. Kayalvili, S., Selvam, M.: Hybrid SFLA-GA algorithm for an optimal resource allocation in cloud. Clust. Comput. (2018). https://doi.org/10.1007/s10586-018-2011-8
16. Pillai, P.S., Rao, S.: Resource allocation in cloud computing using the uncertainty principle of game theory. IEEE Syst. J. **10**(2), 637–648 (2016)
17. Mireslami, S., Rakai, L., Far, B.H., Wang, M.: Simultaneous cost and QoS optimization for cloud resource allocation. IEEE Trans. Netw. Serv. Manag. **14**(3), 676–689 (2017)
18. Zheng, H., Feng, Y., Tan, J.: A hybrid energy-aware resource allocation approach in cloud manufacturing environment. IEEE Access **5**, 12648–12656 (2017)

19. Chen, M., Huang, S., Fu, X., Liu, X., He, J.: Statistical model checking-based evaluation and optimization for cloud workflow resource allocation. IEEE Trans. Cloud Comput. (2016). https://doi.org/10.1109/TCC.2016.2586067
20. Di, S., Wang, C.L., Cappello, F.: Adaptive algorithm for minimizing cloud task length with prediction errors. IEEE Trans. Cloud Comput. **2**(2), 194–207 (2014)