Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2022.025292
Article

Tech Science Press

Multi-Agent with Multi Objective-Based Optimized Resource Allocation on Inter-Cloud

J. Arravinth* **and D. Manjula**

Department of Computer Science and Engineering, College of Engineering, Guindy, Anna University, Chennai, 600025, Tamilnadu, India
*Corresponding Author: J. Arravinth. Email: arravinth@rediffmail.com
Received: 19 November 2021; Accepted: 31 December 2021

Abstract: Cloud computing is the ability to provide new technologies and standard cloud services. One of the essential features of cloud computing is the provision of "unlimited" computer resources to users on demand. However, single cloud resources are generally limited and may not be able to cope with the sudden rise in user needs. Therefore, the inter-cloud concept is introduced to support resource sharing between clouds. In this system, each cloud can tap the resources of other clouds when there are not enough resources to meet the demands of the consumer. In cloud computing, allocating the available resources of service nodes to on-demand tasks is an important concern. To achieve this, in this paper, multi-agent with multi-objective optimized resource allocation on inter-cloud is proposed. The proposed algorithm is a combination of adaptive tree seed optimization (ATSO) and multi-agent. The proposed approach consists of four agents namely, user interface agent, monitoring agent, scheduler agent, and Executer agent. Initially, the user agent collects the task from the users, and the monitoring agent reports the resource list to the scheduler agent. Based on the resource list, the scheduler agent schedules the task with the help of ATSO. Finally, the executer agent, allocate the task to the resources. The performance of the proposed approach is evaluated using makespan, cost, and resource utilization.

Keywords: Resource allocation; adaptive tree seed optimization; multi-agent; inter-cloud; scheduling and task

1 Introduction

Cloud computing gives adaptable and cost-powerful services for endeavors, associations, and people running computational and information concentrated applications. The quick execution of cloud computing has brought about significant investment and quick development. Consistent exploration of cloud computing has turned into an incredible motor for the improvement of artificial intelligence [1]. Through the cloud computing stages, clients can present their resource requests to cloud service providers (CSPs). The CSPs then give the clients their necessary resources as a virtual machine in return. A viable virtual machine (VM) resource allocation should not just convey adaptable services to fulfill different client requirements in the pursuit of expanding the CSP's benefit [2,3], yet additionally save the energy

utilization of the PMs utilized for running clients' applications in the pursuit of diminishing the CSP's cost. As a rule, communication cost needs more energy utilization than computation cost [4]. Lately, to diminish the communication cost, numerous commitments have been made in proposing pertinent communication plans for multi-agent agreement issues [5,6]. An agent is a computer system that is fit for settling on choices freely, doing activities independently, and communicating with different agents through participation, coordination, and arrangement [7].

A multi-agent system (MAS) is a distributed artificial intelligence system made out of various independent agents equipped for communicating and working together with one another to accomplish shared objectives [8]. Multi-agent learning is a space of study that arrangements with the hypothetical test of seeing how agents learn and adjust their behavior within the sight of different agents that are also learning [9]. Task and resource allocation among heterogeneous agents by arrangement is a focal issue in multi-agent systems because it seriously influences the presentation of the whole system. In any case, suitable allocation via independent agents in a dispersed environment is a challenging one [10]. Scheduling and resource allocation in a multi-project environment are pervasive in present-day task and operations management [11]. In any case, the expanding interest for additional resources in different settings puts a substantial interest on the cloud data centers [12]. Resource Allocation is the most common way of relegating accessible resources to the required cloud applications over the web [13]. It expects to distinguish the small part of a restricted resource that an agent can use to accomplish ideal worldwide targets. This prompts limiting resource battle, absence of resources, resource fracture, over-provisioning, and under-provisioning [14,15].

The main objective of the proposed approach is to optimally allocate the task to the resources using ATSO and multi-agents in inter-cloud. In this paper, four agents are incorporated with the resource allocation process namely, user interface agent, monitor agent, scheduler agent, and executer agent. For the resource allocation phase, ATSO algorithm is used. For the optimal resource allocation process, the multi-objective function is designed. The designed function is a combination of Execution time, cost, resource utilization, and skewness. And also all the tasks are must complete before the deadline. The main contribution of the proposed approach is listed below;

- The multi-objective function is designed to obtain the optimal resource allocation on inter-cloud.
- For resource allocation,an ATSO algorithm is proposed. The ATSO algorithm is a hybridization of Tree Seed Optimization (TSO) and GA parameters namely crossover and mutation.
- The performance of the proposed approach is analyzed in terms of efficiency metrics such as makespan, cost, and resource utilization.

2 Literature Survey

Many of the researchers had developed multi-agent-based resource allocation on inter-cloud. Among them some of the works are analyzed here; Xiangqiang et al. [16] have provided a step-by-step multi-agent optimization (HMAO) algorithm for resource allocation. It is a combination of the GA and Multi-Agent Optimization Algorithm (MOA). An advanced GA was proposed to detect the service terminal and to reduce bandwidth cost using the MAO algorithm.

Moreover, Ligade et al. [17] have suggested a resource allocation model using the Sunflower Whale Optimization Method (SFWOA). The proposed algorithm was obtained by combining the hunting technique and the behavior of the humpback whale and the strange behavior of the sunflower. The task was assigned to the low-cost virtual machine using the proposed optimization algorithm, taking into account working time and deadlines. A modified honey bee-inspired algorithm for resource allocation was proposed by Sharma et al. [18]. The utilization of their strategy for dispersing jobs to more than one

organization guarantees that assets are not underutilized. Judith et al. [19] explored a wide range of dual-mode endpoint search control techniques to solve resource allocation problems in multi-agent systems with dynamically linked agents.

They specifically addressed issues related to resource allocation, depending on resource requirements and real-time system dynamics. Similarly, Li et al. [20] studied a multi-agent system with quantized communications to solve distributed optimization problems. That method could reduce communication costs in terms of frequency and amount of transmitted data. Since it was not accurately obtained from the system, there was a bias between the resulting solution and the optimal solution for distributed optimization with a fixed quantizer. Therefore, they proposed a multi-agent system with a quantizer density that changes over time to obtain the optimal solution.

Wanyuan et al. [21] proposed a multi-agent-based distributed resource allocation approach to reduce the energy costs of cloud systems. This method consists of two complementary mechanisms. (1) An auction-based VM allocation mechanism configured to determine which VM the agent assigns to which PM, and (2) a negotiation-based VM integration mechanism designed to replace the assigned virtual machine to save energy costs and deal with system dynamics.

In [22], optimal and energy-efficient scheduling in a public cloud environment was proposed. To improve the efficiency of this method, a load balancing algorithm was introduced. To prove the efficiency speed, time, energy, and security level was utilized. Moreover, improved particle swarm optimization (IPSO) algorithm-based task schedule was presented in [23]. The experiment result showed that the performance of the approach was better when it was compared with the existing task scheduling approaches.

3 Task Flow Model

The main objective of this paper is to effectively schedule the task on VM based on resource availability. Let us consider the task "T" which has a different deadline, start time, and Execution time. Here, four tasks are considered as $\{T_1, T_2, .T_3, T_4\}$ and the deadline for these tasks is represented as $\{d_1, d_2, .d_3, d_4\}$ with the start time of $\{w_1, w_2, .w_3, w_4\}$ and the Execution time of$\{r_1, r_2, r_3, r_4\}$. Once the cloud receives the application from the user, the tasks T1, T2, T3, and T4 are allocated to the VM using the proposed ATSOA.

However, allocation is based on the cost, makespan, and resource utilization of the VM. The main factors to consider when assigning tasks to a VM are cost, makespan, and deadline.The task with minimum cost value is initially allocated to the VM on the cloud. Once the task enters into the cloud for the scheduling process, the proposed optimization algorithm checks the deadline and Execution time of the task and the task with low cost is allocated to the VM to effectively perform the resource allocation. Let us consider three different applications as A_1, A_2 and$A_3$and each application has a different number of tasks. The different tasks are given in Tab. 1 with their start time, execution time, and deadline.

Table 1: Sample value for deadline and execution time of each task

Application	Application (A_1)			Application (A_2)		Application (A_3)	
Tasks	T_1	T_2	T_3	T_4	T_5	T_6	T_7
Deadline	4	3	6	7	5	9	8
Execution time	2	2	3	2	2	4	4
Start time	1	1	1	1	1	1	1

In Tab. 1, the start time of the entire task Ti presented in the entire application is set as "1". Here, the tasks T_1, T_2 and T_3 is under the application A_1, the task T_4 and T_5 are under the application A_2 and the tasks T_6 and T_7 are under the application A_3, respectively. The specified Execution time and deadline-based resource allocation is given in Tab. 2.

Table 2: Sample resource allocation scheme based on deadline and execution time

Virtual machine Time Slot	VM_1	VM_2	VM_3
Y_1	T_2	T_5	T_7
Y_2	T_2	T_5	T_7
Y_3	T_1	T_4	T_7
Y_4	T_1	T_4	T_7
Y_5	T_3	T_6	–
Y_6	T_3	T_6	–
Y_7	T_3	T_6	–
Y_8	–	T_6	–

Task T is allocated to VM based on the minimum cost associated with the task and tasks are allocated to VM within the Execution time and deadline. For example, consider the three virtual machines such as VM_1, VM_2, and VM_3 and assume VM_1 has the minimum cost then VM_2 and VM_3. So that, initially, the tasks are assigned to VM_1 then VM_2 and VM_3 respectively. To perform the resource allocation, we assign the value of each task between 0–1. The task values are given in Tab. 3.

Table 3: Sample task value

Application	A_1			A_2		A_3	
Task	T_1	T_2	T_3	T_4	T_5	T_6	T_7
Task value	0.5	0.3	0.6	0.7	0.5	0.8	0.9

Let us assume an eight-time slot and three VM for the tasks to be allocated. The tasks T_1, T2, and T_3 under application A1 have values of 0.5, 0.3, and 0.6. Among the tasks T_1, T_2, and T_3, task T_2 has the minimum value hence, task T_2 is allocated to VM_1, as VM_1 has minimum cost. As the runtime of T_2 is '2' and the deadline is '2', the task T_2 executes in VM_1 at time slots Y_1 and Y_2. Then, task T_1 is verified with T_3 and selects task T_1 for resource allocation as it has a minimum value of 0.4. The runtime and the deadline of $T_1$is '2' and '5', so T_2 is allocated to VM_1 at the time slot Y_3 and Y_4, and then, the task T_3 is assigned to VM_1 at the time slot Y_5,Y_6, and Y_7 since the run time of T_3 is '3'. Task T_4 and T_5 under application A_2 have the value of 0.7 and 0.5 while comparing T_4 and T_5, task T_5 has the minimum value so it is allocated to VM_2 at the time slot Y_1 and Y_2 since the runtime of T_5 is '2' T_2 uses two-time slots. Task T_4 is allocated to VM_2 at time slots Y_3 and Y_4. Similarly, the remaining tasks T_6 and T_7 are allocated to VM based on the minimal value.

4 Multi-Objective Function Design

Our proposed approach is designed based on a multi-objective function whose objectives are to trade between makespan, cost, and resource utilization. The proposed system should minimize the makespan and cost while it tries to maximize resource utilization. The multi-objective function is designed using Eq. (1).

$$Fitness = \min\left(Makespan,\ Cost,\ {}^{1}/_{\mathrm{Re}\,source\,utilization}\right) \tag{1}$$

Makespan: Makespan is calculated based on the time it takes to complete the workflow. Due to the structure of the directed acyclic graph (DAG), it is possible to calculate the time of completion of the exit task of a solution makespan. Consider the start time T^{start} and end time T^{end} of the task t_a. Task t_a start time depends on all its parental task completion time and data transfer time between parent and ti. The start time is calculated using Eq. (2).

$$T^{start}(t_a) = \begin{cases} 0 & t_a = t_{entry} \\ \max\limits_{t_a \in Parent(t_a)} \left\{ T^{com}(t_b) + T^{trm}_{a,b} \right\} & otherwise \end{cases} \tag{2}$$

where $T^{com}(t_a)$ represent the completion time of a task t_a. The completion time is calculated using Eq. (3).

$$T^{com}(t_a) = T^{start}(t_b) + T^{exe}_{b,k} \tag{3}$$

Once we find out the $T^{start}(t_a)$ and $T^{com}(t_a)$, the makespan of workflow is calculated using Eq. 4.

$$Makespan = T^{com}(t_{exit}) \tag{4}$$

Cost: In cloud computing, users are charged based pay-per-use system, which means they have to pay based on how long they have been using the service. The overall cost of the task t_a is calculated based on three costs such as processing cost, cost of data transfer, and the cost of storage. The cost of the execution task t_i is calculated using Eq. (5).

$$\cos t^{PC}_a = T^{exe}_{a,k} \times C^{PC}_k \tag{5}$$

where;

$T^{exe}_{a,k} \rightarrow$ The execution time of a task t_a

$C^{PC}_k \rightarrow$ Processing cost

The data transfer price among task t_a and their child is evaluated using Eq. (6)

$$Cost^{BW}_a = \sum_{t_b \in T: t_a \in Parent(t_b)} T^{trm}_{b,a} \times C^{BW}_k \tag{6}$$

where; C^{BW}_k demonstrate the cost of using bandwidth. The storage cost of the task t_a is evaluated using Eq. (7).

$$Cost^{S}_a = \left(T^{exe}_{a,k} + \max_{t_b \in Parent(t_b)} T^{trn}_{a,b} \right) \times C^{s}_k \tag{7}$$

$C^{s}_k \rightarrow$ The instance type P_k storage cost.

The total cost is calculated using Eq. (8).

$$Cost = \sum_{t_i \in T} Cost_i^{PC} + Cost_i^{BW} + Cost_i^{S} \tag{8}$$

Resource utilization: The resource utilization VM_k is the same as the number of tasks that the VM has executed up to its finishing time. The VM_k utilization is calculated using Eq. (9).

$$U_k = \frac{\sum_{t_a \in A} MI(t_a)}{\sum_{t_a \in A} \left(T_{a,k}^{exe} + \max_{t_b \in Parent(t_a)} T_{a,b}^{trm} \right)} \tag{9}$$

where A represents the group of task and $MI(t_a)$ is the size of the task t_a in million instructions. The average utilization is calculated using Eq. (10).

$$Utilization = \frac{\sum_{VM_k \in \vartheta} U_k}{|\vartheta|} \tag{10}$$

where, ϑ is the set of instances and $|\vartheta|$ shows the cardinality of it.

5 The Proposed Model Plan

The main objective of the proposed methodology is allocating the resources to incoming tasks. To provide the resources to users without any delay, in this paper inter-cloud system is presented. Inter-cloud is an interconnected global "cloud of clouds" that allows each cloud to utilize the resources of other clouds, making the interactions between cloud partners more complex because the resources between clouds are distributed and controlled by different clouds. Today, agent-based cloud computing approaches have been developed to track resources between clouds. An agent is a computer company that can make decisions independently and communicates with other agents. Therefore, in this paper, multi-agent with multi-objective-based optimized resource allocation on inter-cloud is proposed. The overall structure of the proposed approach is given in Fig. 1.

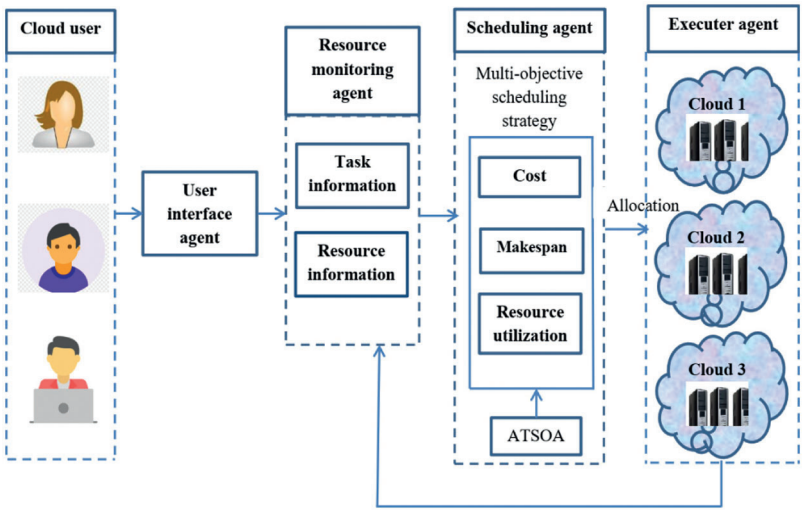

Figure 1: Overall structure of multi-agent-based resource allocation

5.1 User Interface Agent

Initially, the user agent collects the request from the different cloud users. The task contains the information of (request rate, type, size, etc.) Then, the collected requests are directly transferred to the monitor agent. The user interface agent algorithm is given below;

```
Interface agent (receive a request from the user)
{
Input = receive a request from the user;
Output: entire user request
Generate a request-id for each task;
Then;
Call resource monitoring agent (αi)}
```

5.2 Resource Monitor Agent

Monitor agent must collect the request from the user agent and resources information from the data center. The monitor agent is denoted as $\boldsymbol{\alpha_i}$. The agent collects the information of resource load rate (CPU load and memory). This agent checks that the available resources are there in the corresponding cloud. If it does not mean, it collects the resources from inter-cloud. The agent collects the information of resource load rate (CPU load and memory). The resource monitor agent algorithm is presented below;

```
Resource monitor agent (receive a task from users and collect the resource list)
{
Input = users task and resource list
Output: details of task and resource
Generate a resource table;
Then;
Call scheduler agent (γi)}
```

5.3 Scheduler Agent

In this section, the monitor agent transmits the collected information about a user requests and resource load information to the scheduler agent (γ_i). The scheduler agent, optimally allocates the task to resources while considering, time, cost, and resource utilization. For resource allocation, in this paper, the ATSO algorithm is presented. The TSO was derived from the relationship between trees and seeds by the population-based evolutionary method. When the seed formation process takes place in the TSO, the level improvement of each dimension of the seed is calculated separately. In the tree-seed method, each tree represents a parent individually and each seed represents a child created from one parent tree. To enhance the TSO algorithm, the GA operators namely, crossover and mutations are adapted with TSO. The scheduler agent algorithm is presented below;

scheduler agent (receive resource table from monitor agent)
{
Input= Resource table
Output: Scheduled task
Schedule the suitable task with resources
Then;
Call Executer agent (Ei)}

The step by step process of ATSO algorithm based resource allocation is listed below;

Step 1: Solution Encoding: Solution encoding is an initial step for performing the resource allocation strategy optimally in the cloud. In this section, initially, the parameter used in this paper is initialized. The parameters are task count, number of clouds, number of VM, CPU capacity, population size, maximum iteration, ATSO parameters. Then, the solutions are randomly generated. In solution encoding, the virtual machines are assigned with a task, which has low-cost price and the task with high-cost price are assigned at last. Based on the values associated with each task T, the solution vector is designed. The solution format is given in Fig. 2.

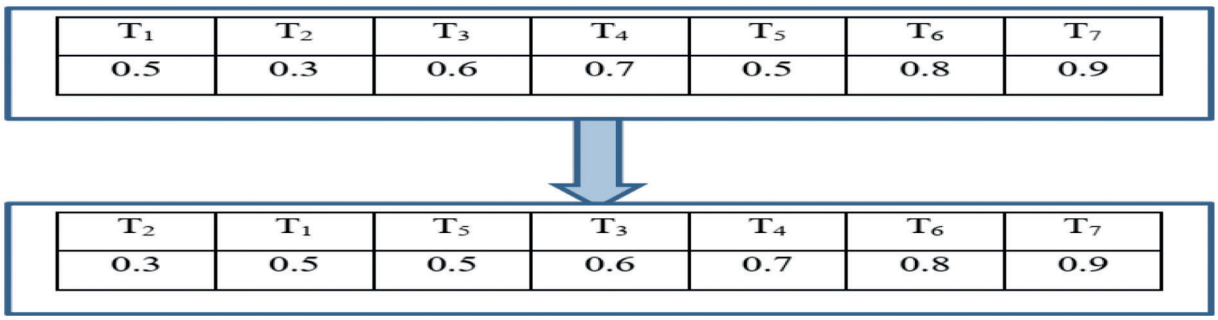

Figure 2: Solution encoding

Step 2: Fitness Evaluation: The parameters Execution time; cost, resource utilization, and skewness are used for calculating fitness function. The fitness is evaluated for each solution. The fitness function is used to obtain the optimal solution. In this paper, the minimization function is used for fitness. The fitness function is computed using Eq. (11).

$$Fitness = \min\left(Makespan,\ Cost,\ {}^{1}/_{\text{Resource utilization}}\right) \tag{11}$$

*Step 3: Updation Using TSOA:*once the fitness is evaluated, the solutions are updated using the TSO algorithm. The TSO is updatedits position based on Search Tendency (ST) value. If the ST value is lesser than the random value R_{ij} means, then the value of seed is updated their position using Eq. (12).

$$S_{i,j} = T_{i,j} + \alpha_{i,j}\left(B_j - T_{r,j}\right) \tag{12}$$

Otherwise, the value of seed is updated by the following equation;

$$S_{i,j} = T_{i,j} + \alpha_{i,j}\left(T_{i,j} - T_{r,j}\right) \tag{13}$$

Step 4: Crossover operation: After TSO updation, the solutions are again updated using the Crossover operator. The crossover operation is used to generate a new set of solutions from the old solutions. The crossover operation is given in Fig. 3.

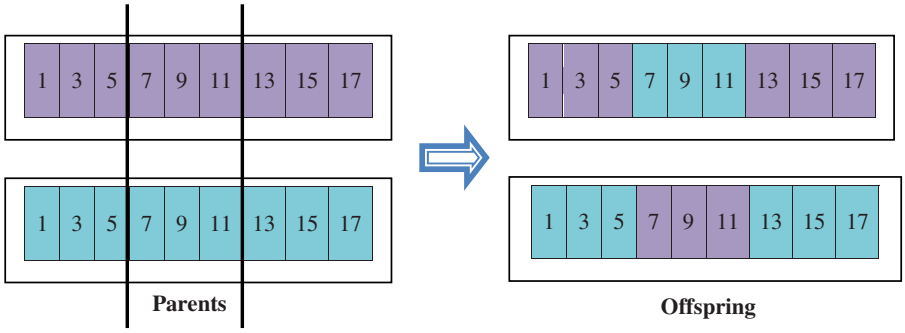

Figure 3: Single point crossover

*Step 5: Mutation Operation:*After the crossover, the mutation is performed. Mutations are the formation of new offspring from single parents and maintain the diversity of each chromosome shown in the figure. There is the possibility of randomly changing a child's genotype. This gene performance is better than that of aging parents. There are two methods of mutation, random and alternative. Mutation operation is given in Fig. 4.

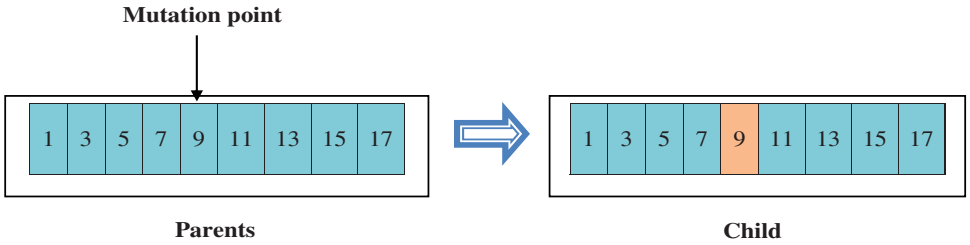

Figure 4: Mutation

*Step 6: Termination Criteria:*The algorithm stops its performance when the best fitness value is selected. Once optimal fitness is achieved, the resource allocation is addressed. The flowchart of ATSO is given in Fig. 5.

5.4 Executer Agent

After the scheduling process, the scheduled tasks are given to the executer agent (E_i). Here, the executor allocates the task to each VM. The allocation should minimize the cost, computation time and increase resource utilization. The resources allocated are based on the obtained optimal solution.

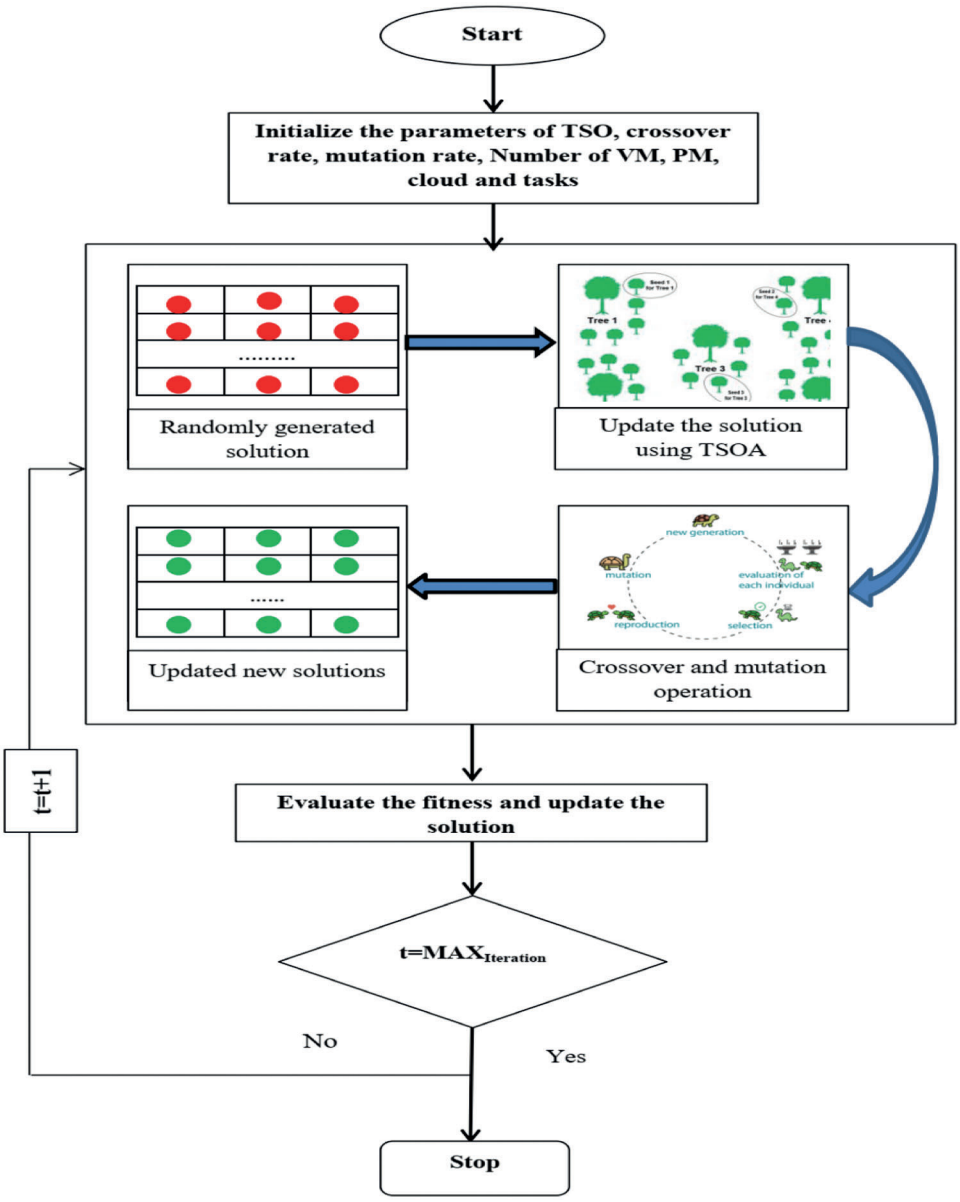

Figure 5: Flowchart of proposed ATSO algorithm

6 Results and Discussion

This section discusses the results and discussion of the proposed adaptive tree seed optimization approach for resource allocation in cloud computing. The proposed resource allocation is done on an Intel Core i5 processor, and a computer with 6GB of memory using the Windows 10 operating system. Simulation of the proposed method is implemented in JAVA. For experimentation analysis, three types of workflows are used such as Montage, CyberShake, and LIGO. The structure of workflow is presented in Fig. 6. The VM specifications are given in Tab. 4. The Amazon EC2 VM instance specification is given in Tab. 5 and Google Compute Engine VM instance specifications are given in Tab. 6.

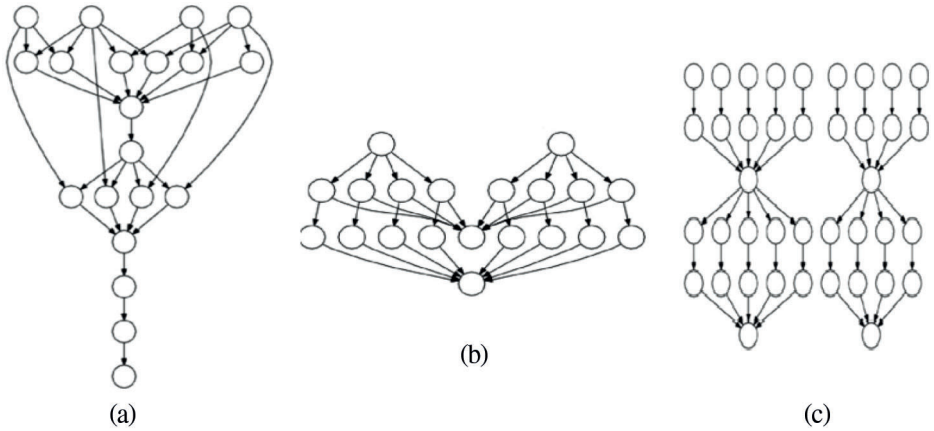

Figure 6: Workflow structure (a) Montage, (b) CyberShake, and (c) LIGO

Table 4: VM specifications

Type	"Processing capacity"	"Processing cores"	"RAM (MB)"	"Bandwidth (Mbps)"	"Storage"
A	1000	1	512	512	512
B	2000	2	1024	1024	1024
C	3000	4	2048	2048	2048

Table 5: Amazon EC2 VM instance specification

Instance type	"Core speed"	"Processing cores"	"RAM (GB)"	"Storage" (GB)	"Cost per hour" ($)
m1.small	1	1	1.6	150	0.5
m1.large	5	3	7.6	900	0.22
m1.xlarge	9	5	16	1710	0.51
c1.medium	6	3	1.8	370	0.29
c1.xlarge	21	9	7.2	1710	1.20

Table 6: Google Compute Engine VM instance specification

Instance type	"Core speed"	"Processing cores"	RAM (GB)	"Storage" (GB)	"Cost per hour" ($)
m1.small	1	1	1.9	160	0.7
m1.large	5	3	7.6	920	0.39
m1.xlarge	9	5	15.5	1720	0.52
c1.medium	6	3	1.9	350	0.4
c1.xlarge	21	9	7.5	1720	1.30

6.1 Experimental Results

In this section, the experimental results obtained from the proposed approach are analyzed. The main objective of the proposed approach is to optimally allocate the task to the resources by using multi-agents and ATSO. For scheduling the task, the multi-objective function is utilized.

6.1.1 Experimental Results Based on Makespan

Makespan is an important parameter for scheduling. The makespan is changes based on the workflow scheduling model. The result obtained by various workflow models is analyzed in this section.

In Fig. 7a, the performance of the proposed approach is analyzed using makespan. When analyzing Fig. 7a, the proposed method attained the makespan of 46 s for small instances, the 50 s for medium instances and 55 s for large instances type.The results demonstrate that the makespan of the proposed approach is lower than the TSO, PSO, and GA-based task allocation. This is due to the adaptation of TSO. Besides, the TSOA algorithm has better results compared to the other two methods. The proposed ATSO is accomplished by ducking entanglement in local optimization by removing solutions within a given radius of optimal points and replacing them with new random solutions. In Fig. 7b, a Cybershake workflow-based experimental result is analyzed. The makespan is an important parameter. The makespan is varied based on task flow. According to Fig. 7b, the proposed method takes minimum makespan compared to other methods. In Fig. 7c, LIGO workflow-based experimental results are analyzed. Here, the experimental analysis is carried out based on three instances. Compared to three instances, a large instance takes the maximum makespan. When analyzing Fig. 7c, the proposed method is taken a maximum makespan of 700 s. From Fig. 7, the proposed method is taken a maximum makespan of 55 s, 120 s, and 700 s for Montage, cyberShake, and LIGO respectively.

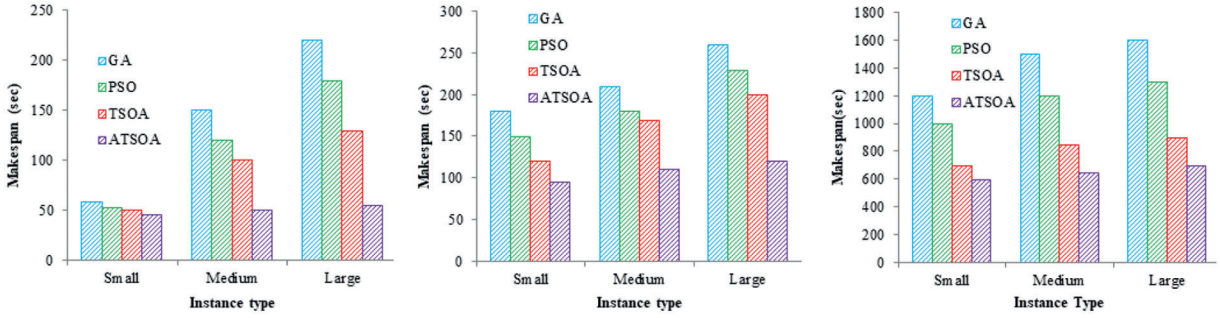

Figure 7: Comparative analysis based on Makespan (7a) Montage, (7b) CyberShake, and (7c) LIGO

6.1.2 Experimental Results Based on Cost

In this section, we compare the performance of the proposed approach based on normalized cost. The cost is varied for each task and virtual machine. The normalized cost of the workflow execution is given in Eq. (14).

$$NC = \frac{Cost_w}{Cost_w^{chp}} \tag{14}$$

where; $Cost_w$ represent the overall cost of the workflow and $Cost_w^{chp}$ represent the cost of all tasks present in the workflow on the cheapest instance respectively. The experimental results obtained by three workflows are explained below;

In Fig. 8a, the performance of the proposed approach is analyzed based on the normalized cost for Montage workflow. When analyzing Fig. 8a, the outcome depicts that the NC is most similar to TSOA

based scheduling. But our proposed ATSO little bit better than TSO. This is due to genetic operators or adapted with TSOA. In Fig. 8b, a Comparative analysis based on the normalized cost for CyberShake is discussed. When analyzing Fig. 8b, the proposed approach taken 15$ for scheduling cybershake workflow using small instance, 29$ for scheduling cybershake workflow using medium instance, and 34$ for scheduling cybershake workflow using large instance. Cost is varied based on instance. Here, two clouds are integrated for resource allocation. In Fig. 8c, the performance of the proposed approach is analyzed based on normalized cost. Here also proposed approach attained better results compared to the exiting approach.

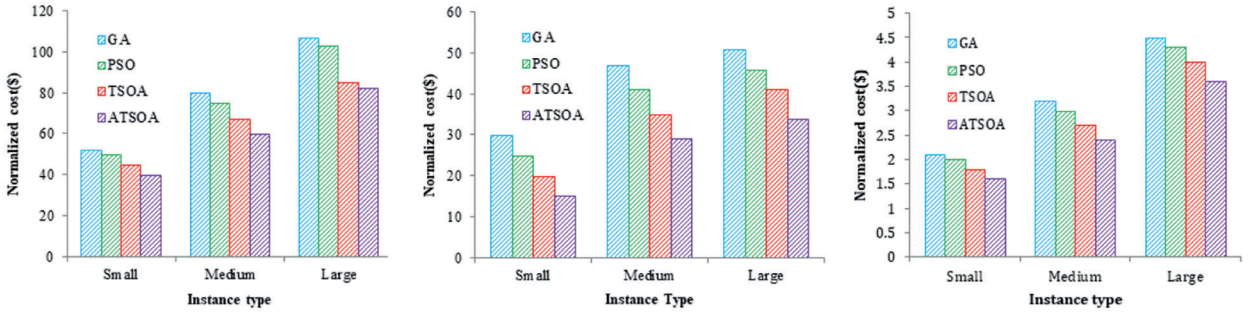

Figure 8: Comparative analysis based on normalized cost (8a) montage, (8b) CyberShake, and (8c) LIGO

6.1.3 Experimental Results Based on Resource Utilization

In this section, the performance of the proposed approach is analyzed based on resource utilization. In this paper, the multi-objective function is designed based on three parameters namely, makespan, cost, and resource utilization.

In Fig. 9a, the efficiency of the presented approach is analyzed based on resource utilization using a montage workflow model. A good resource allocation system should have maximum resource utilization. When analyzing Fig. 9a, the proposed approach utilized more resources compared to other methods for scheduling the montage workflow models. This indicates, the proposed ATSO is much suitable for the scheduling process. Similarly, in Figs. 9b and 9c also our proposed approach attained the better results. Compared to the four methods, GA-based scheduling got work performance output. Overall, from the test results, it is clear that our approach works best in reducing makespan and cost as well as increasing the utilization of resources.

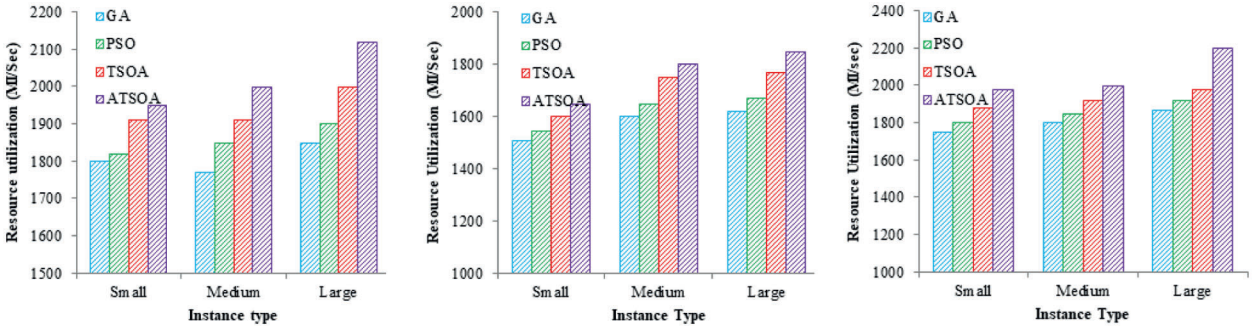

Figure 9: Comparative analysis based on Resource utilization (9a) Montage, (9b) Cyber Shake, and (9c) LIGO

7 Conclusion

An efficient resource allocation on inter-cloud using multi-agent and ATSO algorithm has been proposed in this paper. Resource allocation based on the proposed optimization effectively allocates resources in the cloud without reducing the efficiency of the model system. To allocate the task, the multi-objective fitness function has been developed which is based on cost, makespan, and resource utilization.The fitness with minimum value has been considered as the best solution. The proposed optimization algorithm is used the behavior of tee and seeds growth. The efficiency of the proposed approach has been analyzed based on various metrics. The proposed approach attained the minimum makespan of 46s, cost of 40$ for small instance of montage workflow. Overall, from the test results, it is clear that our approach works best in reducing makespan and cost as well as increasing the utilization of resources. T In the future, we will apply resource allocation on real-time applications and we will use a machine learning algorithm for resource allocation.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Lin, D. Cui, Z. Peng, Q. Li and J. He, "A two-stage framework for the multi-user multi-data center job scheduling and resource allocation," *Institute of Electrical and Electronic Engineers Access*, vol. 8, pp. 197863–197874, 2020.

[2] J. Cao, K. Hwang, K. Li and A. Zomaya, "Optimal multiserver configuration for profit maximization in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1087–1096, 2013.

[3] S. Chaisiri, B. Lee and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 164–177, 2012.

[4] P. Yi, Y. Hong and F. Liu, "Initialization-free distributed algorithms for optimal resource allocation with feasibility constraints and application to economic dispatch of power systems," *Automatica*, vol. 74, no. 1, pp. 259–269, 2016.

[5] G. Seyboth, D. Dimarogonas and K. Johansson, "Event-based broadcasting for multi-agent average consensus," *Automatica*, vol. 49, no. 1, pp. 245–252, 2013.

[6] D. Ye and X. Yang, "Distributed event-triggered consensus for nonlinear multi-agent systems subject to cyber attacks," *Information Sciences*, vol. 473, no. 1, pp. 178–189, 2019.

[7] K. Sim, "Agent-based approaches for intelligent intercloud resource allocation," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 442–455, 2019.

[8] S. Adhau and M. Mittal, "A multiagent based system for resource allocation and scheduling of distributed projects," *International Journal of Modeling and Optimization*, vol. 2, no. 4, pp. 524–528, 2012.

[9] N. Barbalios and P. Tzionas, "A robust approach for multi-agent natural resource allocation based on stochastic optimization algorithms," *Applied Soft Computing*, vol. 18, no. December, pp. 12–24, 2014.

[10] Y. Ishihara and T. Sugawara, "Multi-agent task allocation based on the learning of managers and local preference selections," *Procedia Computer Science*, vol. 176, pp. 675–684, 2020.

[11] F. Li, Z. Xu and H. Li, "A multi-agent based cooperative approach to decentralized multi-project scheduling and resource allocation," *Computers & Industrial Engineering*, vol. 151, no. 1, pp. 106961, 2021.

[12] M. Masdari and M. Zangakani, "Efficient task and workflow scheduling in inter-cloud environments: Challenges and opportunities," *The Journal of Supercomputing*, vol. 76, no. 1, pp. 499–535, 2019.

[13] M. Katyal and A. Mishra, "Application of selective algorithm for effective resource provisioning in cloud computing environment," *International Journal on Cloud Computing: Services and Architecture*, vol. 4, no. 1, pp. 1–10, 2014.

[14] Z. Xiao, W. Song and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, 2013.

[15] S. Pankaj, P. Kumar and T. Singh, "Resource allocation strategies in cloud computing," *International Journal of Computer Science & Communication Networks*, vol. 5, no. 6, pp. 358–363, 2015.

[16] G. Xiangqiang, R. Liuand and A. Kaushik, "Hierarchical multi-agent optimization for resource allocation in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 3, pp. 692–707, 2002.

[17] S. Ligade and R. Udayakumar, "Sunflower whale optimization algorithm for resource allocation strategy in cloud computing platform," *Wireless Personal Communications*, vol. 116, no. 4, pp. 3061–3080, 2021.

[18] A. Sharma, K. Upreti and B. Vargis, "Experimental performance analysis of load balancing of tasks using honey bee inspired algorithm for resource allocation in cloud environment," *Materials Today: Proceedings*, 2020.

[19] O. Judith and M. Guay, "Distributed extremum seeking control of multi-agent systems with unknown dynamics for optimal resource allocation," *Neurocomputing*, vol. 381, no. 3, pp. 217–226, 2020.

[20] K. Li, Q. Liu and Z. Zeng, "Quantized event-triggered communication based multi-agent system for distributed resource allocation optimization," *International Journal of Information Science*, vol. 577, pp. 336–352, 2021.

[21] W. Wanyuan, Y. Jiang and W. Wu, "Multiagent-based resource allocation for energy minimization in cloud computing systems," *IEEE Transactions on Systems Man and Cybernetics: Systems*, vol. 47, no. 2, pp. 205–220, 2016.

[22] S. Muthurajkumar, M. Vijayalakshmi, A. Kannan and S. Ganapathy, "Optimal and energy efficient scheduling techniques for resource management in public cloud networks," *National Academy Science Letters*, vol. 41, no. 4, pp. 219–223, 2018.

[23] B. P. Kavin, S. Ganapathy and A. Karman, "An intelligent task scheduling approach for cloud using IPSO and A* search algorithm," in *2018 Eleventh Int. Conf. on Contemporary Computing*, India, pp. 1–5, 2018.