International Journal of Engineering & Technology, 7 (4) (2018) 2088-2094

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: 10.14419/ijet.v7i4.16486
Research paper

GALO:A New Intelligent Task Scheduling Algorithm in Cloud Computing Environment

Omer K. Jasim Mohammad *

Director of Computer Center University of Fallujah
**Corresponding author E-mail: Omerk.jasim@uofallujah.edu.iq*

Abstract

Cloud communication technology is Internet-based computing, where shared resources, software, information, are provided to computers and devices on-demand. They guarantee a way to share distributed resources and services that belong to different organizations through virtualization technology. Cloud has announced a modern idea by deploying one application which offers variety and a lot of services to a number of cloud-users at the same time, however, it suffers from scheduling and workload problems. This paper proposed a cloud computing task scheduling algorithm based on greedy algorithm and Antlion Optimizer algorithm. The main goal of this algorithm is to reduce the make-span and the total cost of the tasks and execution time. This paper suggested the objective share-search function of the make-span and costs of the tasks in order to improve the initialization of the pheromone, the greedy algorithm and the pheromone update method in the antlion algorithm. Also, this context illustrates the analytical study between almost used scheduling algorithms and the proposed algorithm. Theoretically, the proposed algorithm provides a more flexible and guarantee a solution to solve the problem of task scheduling in the cloud computing environment.

Keywords: *Cloud Computing; Task Scheduling; Intelligent Search; Resource Allocation; Antlion Algorithm; The Greedy Algorithm.*

1. Introduction

Recently, the science of computing is categorized according to their usage pattern, distributed computing, grid computing, utility computing and cloud computing are famous examples of categories. The growing of computing leads to growing in high- speed of networks broadband and the growth of the Internet changed the network communication mechanism [1]. Accordingly, the new trends of distributed computing technology require integration between distributed computing systems and networking communication systems. Such categories represent the result of such integration which in turn changing the entire computing paradigm toward a new trend in internet computing. Cloud computing is a specialized form of distributed and internet computing, and it takes grid computing style when the dynamic connection service, and the virtual resource's service are available through the Internet [1], [2].

Nowadays, cloud computing has become the fast spread in the field of internet computing family and it acts a hot research topic in the different fields such as medical, education, library, etc., also, it shows a new trend of deployment applications and services via virtualization technology and internet. Cloud computing consists of a number of shared resources, such resources vary according to the cost of executing tasks and type of scheduling resources. Intuitively, anything provided by the cloud is provided as a service whether on the level of hardware or on the level software, all such services completely depend on the task scheduling and resources availability [3], [4].

Task scheduling plays a key role to improve flexibility and reliability of services in the cloud. The main reason behind tasks scheduling can be abbreviated in finding out a complete and best sequence for the tasks and executed it in the best time prelude to arrival to a satisfactory result.

Lately, a lot of algorithms have been proposed for solving task scheduling in cloud and green computing. Swarm optimization, greedy, antlion, deferential algorithms are famous examples for such algorithms. The task scheduling process strives to distribute the load in equal proportions across virtual machines (VMs) depending on resources capacity so that each resource is not overload or underutilized in a cloud system [5]. Moreover, two types of scheduling in a cloud environment such as a dynamic scheduling and Static scheduling. A dynamic scheduling is to rebalance the system through running when detecting overloaded VMs while a static used to balance the system from the start by scheduling. Generally, static load balancing is preferable as it avoids VM migration costs and provides better execution time and quality of service (QoS) [6].

This paper proposed a new type of scheduling optimizing algorithm annotated as (GALO) in which, a hybrid algorithm is used in the scheduling processes. GALO a hybrid between heuristic greedy search and Antlion to optimize scheduling based on the workload profile in virtualized environments and it makes an equal resource allocation of VMs and adapts these allocations based on the estimated cost.

The paper structure is organized as follows: Section 2 surveys the existing studies of task scheduling algorithm in cloud computing. Section 3 discusses the task scheduling concept in cloud computing environment. Section 4 proposes the GALO algorithm and discusses in detail its main parameters in the proposed algorithm. While, Section 5 provides the analytical analysis for the proposed algorithm and finally, Section 6 presents the conclusions and future directions.

2. Existing studies

Recently, numerous papers have been published related to the task scheduling and load balancing in cloud computing. However, no holistic or fruitful solution for this critical topic has been forthcoming yet. Although improvements are achieved in several areas, most of this improvement focus on the cloud task scheduling area especially, task resource requirements, CPU, memory, execution time and execution cost.

For example, Zhifeng et al. [5] introduce a Greedy Particle Swarm Optimization (G&PSO) algorithm to solve the task scheduling problem. They are using a greedy algorithm to quickly solve the initial particle value of a particle swarm optimization algorithm derived from a virtual machine-based cloud platform. The obtained experimental results showed that the G&PSO exhibits better performance like faster convergence rate, stronger local and global search capabilities, and a more balanced workload on each virtual machine. Also, they have concluded that the G&PSO algorithm demonstrates improved virtual machine efficiency and resource utilization compared with the traditional particle swarm optimization algorithm. However, the authors don't explain the cost estimation for the proposed algorithm.

Qiang Guoa [6] presented a new task scheduling algorithm based on ant colony algorithm in order to minimize the makespan and the total cost of the tasks. He defined a load balance function by established the objective function of the make-span and costs of the tasks, which in turn, improved the initialization of the pheromone, the heuristic function and the pheromone update method. In this study, the author used the CloudSim as a cloud platform to carry out the proposed algorithm, also, he compared the obtained results with algorithms of ACO and Min-Min. Finally, the results show that the algorithm is more efficient than the other two algorithms in make-span, costs, and system load balancing.

Ji Lia et al [7] proposed Greedy-Based Algorithm (GB) in cloud computing in order to approve that the greedy algorithm almost selects a local optimum. This study classified tasks based on QoS and categories then selected the appropriate branch of the function and computed the justice evaluation. The authors compared their study with other algorithms and they are concluded that GB decreases the completion time of submitted jobs and increases the user satisfaction. But, they aren't showing in details the experimental environment of their work and they aren't discussed in details the obtained results.

Junwei Ge1al. [8] Illustrated an improved task scheduling based on a differential evolution algorithm for cloud computing. This paper presented the scheduling by classified the task scheduling according to the model of the fitness function and used improved optimization calculation of the fitness function. Two steps support the evolutionary algorithm to improve the calculation according to the evolution of a generation of dynamic selection strategy through dynamic mutation strategy to ensure the global and local search ability. The experimental results showed that the improved differential evolution algorithm can reduce the cloud computing task execution time and user cost saving, good implementation of the optimal scheduling of cloud computing tasks. However, such an algorithm isn't compatible with all environments of cloud computing due to lack of heavy workloads.

Abdul et al. [9] announced an efficient task scheduling algorithm, which is presented divisible task scheduling algorithm regarding network bandwidth. They are proposed task-scheduling algorithm by using a nonlinear programming model for divisible task scheduling, which assigns the correct number of tasks to each virtual machine. Also, this study allocated the workflow based on the availability of network bandwidth. However, this paper focused on space required for the server and network bandwidth, not more.

Suraj, P. [10] suggested a particle swarm optimization based heuristic to schedule applications to cloud resources that optimizes computation cost and data transmission cost. The PSO based mapping algorithm has much lower cost as compared to another algorithm called (Best Resource Selection) based mapping. It is used for a workflow application through differing its computation and communication costs. The results show that PSO can achieve cost savings and good distribution of the workload onto resources. Table 1 reveals the methods or algorithms are used in surveyed studies like factor consideration, cloud environment used, features and weakness, and future enhancement of proposed ideas.

Table 1: Summary of Existing Studies

Authors/Ref	Cloud environment	Algorithm| hybrid or new	Factor consideration	Features	Future Enhancement
Zhifeng et al. [5]	Open Stack	Greedy Particle Swarm Optimization /hybrid	Performance, time	completing the task in given time with highly response	Needing to support task scheduling for real applications
Qiang Guoa [6]	CloudSim	Ant Colony algorithm/ new	CPU, cost	Giving a cost saving and balancing the workload on resources	Needing to be extended to work with intelligent applications in cloud env.
Ji Lia et al [7]	Hyper-V	Greedy-Based Algorithm (GB)/new	QoS, time	Reflects the QoS requirements for each task user	Needing to solve more complex dynamic factors like dynamically changing cloud
Junwei Ge1al. [8]	Java Environment	differential evolution	task execution time and cost saving	Optimizing cost and time for each task in mathematical mode.	None
Abdul et al. [9]	CloudSim	task-scheduling algorithm	availability of network bandwidth, search	Minimizing the execution cost and optimize the QoS	Needing to modify to support QoS in order to scalable with different dimensions
Suraj, P.[10]	CloudSim	particle swarm optimization	cost savings and good distribution of the workload	Enhancing the resources cost and time utilization.	Needing to support scheduling workflows in multi-tier.

From above, many researchers and studies attempted to solve the task scheduling problem, however, no holistic solution for this matter. Furthermore, the scheduling process in cloud computing means to allocate the task to each VM under cloud environment regarding resource availability and power of computing of the VM. Each VM in the cloud data center has a special quota which includes hardware resources like storage, CPU, bandwidth, etc. Also, the inferences show that the two basic layers for VM-task scheduling: (i) Job scheduler means the assigning task to VM , (ii) VM scheduler acts the partitioning the host in to a number of VMs. Hence, this study attempts to solve the task-scheduling to virtual machines and arranges the waiting for nullity tasks into single VM.

3. Task scheduling in cloud computing

This section explains the concept of task scheduling and the selected scheduling algorithms.

3.1. Concept

The concept of cloud computing is closely related with the concept of task scheduling process, such process refers to share the computing-tasks among different users according to certain rules of available resource and then use under a given cloud condition (i.e. allocation of $\boldsymbol{t}$ independent tasks assigned to $\boldsymbol{m}$ virtual Machine) see figure 1 . Generally, all computing-tasks are completely depend on the power and availability of cloud-resources pooling. Till now, there isn't standard task scheduling in the cloud world, however, the management of cloud resources and cloud task scheduling are the keys of the cloud technology.

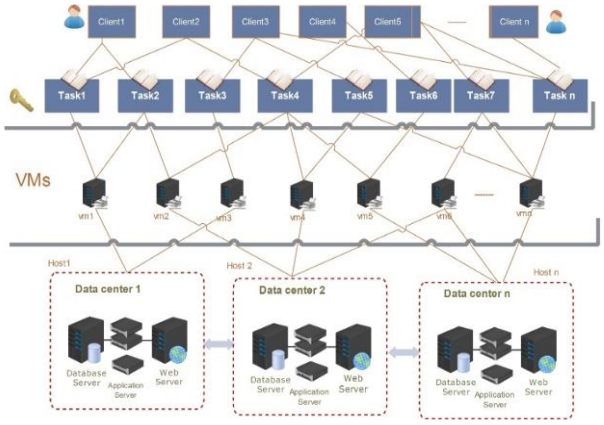

Fig. 1: Task Scheduling Concept.

Mathematically, tasks set as $task = \{t_0, t_1, t_2, \dots, t_{tot}\}$, tot=$|task|$ denotes the whole number of tasks in the task set. Each task (t_i) has a formal description can be denoted as $t_i = \{t_{id}, t_{length}, t_{pesnumber}, t_{size}, t_{outputfile}\}$, also, VMs set $vm = \{vm_0, vm_1, vm_2, vm_{tot}\}$, each vm_i includes $\{vm_{id}, vm_{ip}, vm_{size}, vm_{RAM}, vm_{BW}, vm_{vmm}\}$. According to $Eq1.$, the execution time for each task represent the execution time of ti on vmj [11], [12].

$$E_t(i,j) = \frac{t_{length(i,j)}}{vm_{comp(i,j)}} \quad (1)$$

Where $t_{length(i,j)}$ represents the length of execute-instruction task, $vm_{comp(i,j)}$ indicates vm completion time and it calculates the efficiency of the vm according to $Eq.2$.

$$vm_{comp(i,j)} = vm_{pesNumber(i,j)} * vm_{mips(i,j)} \quad (2)$$

$vm_{pesNumber(i,j)}$: indicates the no. of processors under each vm
$vm_{mips(i,j)}$: The processing efficiency of vm_j.
With the scheduling process, the best suitable resource for execution task or assign VMs to tasks must be achieved. Currently, static and dynamic scheduling process are well-known scheduling types in the cloud environment, with static scheduling a complete guide and information about the structure of tasks and mapping of resources are available before the execution time. While a dynamic scheduling describes the current states of system and VMs in order to obtain scheduling decision. Table 2 includes the collection of abbreviations have been used in this paper.

Table 2: Key of Abbreviations Used

Abbreviations	Key
t_{id},t_{length}, $t_{pesnumber}$, t_{size}, $t_{outputfile}$	Task proprieties (id, length, no. of processors, size, result)
$vm_{id}, vm_{ip}, vm_{size}, vm_{RAM}, vm_{BW}, vm_{vmm}$	Virtual machine proprieties (id, ip, size, memory, bandwidth, management)
E_t	Execution time
$vm_{comp(i,j)}$	completion time for each VM
$vm_{pesNumber(i,j)}$	No. of processors under each VM
$vm_{mips(i,j)}$	Processing efficiency
RW_i	random a walk of ant_i
a	Minimum value of walk
NRW_i^k	Number of random walk
LB^k	lower bound in k^{th}iteration
UB	upper bound
AL_j	position of ant lion
D	decreasing factor
$Reco$	Recommendation configuration

3.2. Nominated algorithms

Recently, a lot of algorithms have been proposed for solving optimization problems and task scheduling in cloud and green computing. Swarm optimization, greedy, antlion, and others algorithms are famous examples of such algorithms. This paper focuses on two scheduling algorithms, the first one belong to the heuristic search algorithm (greedy), while another algorithm depends on an intelligent optimizer search (Antlion optimizer algorithm).
Greedy is an algorithm usually works on optimization problems with two main features, the first one is a greedy-choice property (GCP) while the second is a greedy optimal substructure (GOS). GCP acts the process for accessing to the optimum state by selecting a local optimum point however GOS establishes when an optimal solution to the problem contains an optimal solution to subproblems like dynamic programming environment [12]. In other words, the greedy method takes the style straightforward no backtracking style, thus, its do not always find an optimal solution but it makes the selection that seems the finest at the moment like follow:

$for \leftarrow 1\ to\ n\ do$

It means, select an element for xi that "looks" best at the moment. Also, in some cases [11 - 13], the greedy algorithm follows the best local choice in order to lead to a globally optimal solution [15]. For more details, Figure 2 illustrates the pseudo code for the greedy algorithm.

Initialize: set the value current solution and current location to 0.
Input: the value of the current state
Calculate the approximately optimal value (a).
Output: the optimal solution

Begin
$for \leftarrow 1\ to\ n\ do$
Begin
- Select an optimal solution and put in X;
- Compare between the current solution and obtained solution X;
- If feasible
- Set the value of new solution by union (solution, x);

End;
Update a value of the current location with a new obtained location.
Return solution;

End;

Fig. 2: Pseudo-Code for the Greedy Algorithm.

As shown Figure2, two basic fundamental criteria have been used by the greedy algorithm, the objective function (represent the optimal value either maximum or minimum) and the set of constraints (limitations on the needed solutions).

While the Antlion optimizer (ALO) [14, 15] is a new proposed scheduling optimizer algorithm that is used to solve the optimization problems. Hence, a new cloud scheduler based on ALO presented by Cristian Mateos and et.al [16]. Generally, ALO depends on creating a random walk of an ant i and scales it within the boundaries of an ant lion$j$. $Eq.3\ to\ 8$ Show that the any random walk depending on n number of iterations i's calculated by adding either a step forward or backward randomly for each iteration.

$$RW_i = \left[RW_i^1, \dots RW_i^k, \dots RW_i^n\right] \quad (3)$$

$$such\ that\ RW_i^k = RW_i^{k-1} \pm 1\ , RW_i^1 = 0, \quad (4)$$

$n =\ is\ number\ of\ iterations\ , RWi =$
$random\ a\ walk\ of\ ant_i, and\ k =$
$is\ number\ of\ back|forward$

$Eq.4$ Describes the type of work, when the walk in k^{th} iterations,

$$NRW_i^k = \frac{(RW_i^k - a)}{(b-a)} \quad (4)$$

Where a is the min value in RW_i; b is the max value in RW_i.

Furthermore, the search space process has boundaries to look for obtained solutions within. $Eq.5$ shows that the ALO has such boundaries, however, it decrease gradually in each iteration using the following equation:

$$LB^k = \frac{LB}{D}\ ;\ UB^k = \frac{UB}{D} \quad (5)$$

Where LB is the initial lower bound, LB^k is the lower bound in k^{th}iteration, UB is the initial Upper bound, UB^k is the upper bound in k^{th} iteration and D is the decreasing factor depend on the iteration number.

Regarding the bounds for the k[th] search space, ant lion j bounds can be calculated as shown in Eq. 6:

$$\mathrm{LB_j^k} = \mathrm{AL_j} + \mathrm{LB^k} \quad (6)$$

Where $\mathrm{LB_j^k}$ the lower is bound around ant lion j and $\mathrm{AL_j}$ is the position of ant lionj.

In each iteration, the ALO algorithm calculates the new position of ant i by the average of its scaled random walks towards the corresponding ant lion j and towards the elite ant lion using, see Eq. 7:

$$\mathrm{Ant_i} = \frac{(\mathrm{NRW_{i,j}^k + NRW_{i,e}^k})(\mathrm{UB^k - LB^k}) + \mathrm{LB_j^k + LB_e^k}}{2} \quad (7)$$

Where $\mathrm{NRW_{i,j}^k}$ is the normalized random walk of anti towards the selected ant $\mathrm{lion_j}$and $\mathrm{NRW_{i,e}^k}$ is the normalized random walk of anti towards the elite ant lion. For more details, Figure 3 illustrates the pseudo code for ALO algorithm.

Input: the value of current state (list of tasks and VMs)
Calculate the workload of each VM (fitness of antlions).
Output: the best solution for tasks allocation on VMs .

1) Initialize:
Best_Solution=null; Current_solution= cost;
cost=c; the cost of the path between task and VM;
Begin
2) Place the wlk ants on the starting VM randomly;
for a=1 to wlk do
While (ants_trip don't end)
Every ant selects the VM for the next task based on Eq. (3) & Eq.(4);
End while;
3) Update search boundaries LB, UB using Eq.(5)
4) For a=1 to k do
Update the position of ant using Eq. (7)
Calculate the length of the ant tour using Eq.(5);
Check a new obtained B-Solution;
5) Apply a new value (local update)
6) Apply global update for VM-TASKS
7) If (C_solution< Best_Solution)
Go to (step2);
Else
Return Best_Solution;
End if
8) End;

Fig. 3: Pseudo-code for ALO algorithm [15].

As shown in Figure 3, ALO algorithm completely depends on two scheduling assignments loops. Such loops represent the type of ant walk through cloud-VMs and the ant trip ends when ant complete all searching process and return the best solution. Also, the algorithm needs $O(n^2 + n)$ as a cost of execution that is stemmed from Figure 3.

The main reasons behind the selection of Greedy and ALO coming from the following:

1) A greedy algorithm is directly affecting on the task scheduling and the load balancing process in the cloud data center.
2) ALO handle task scheduling using a cost function that is dependent on minimizing make-span and task processing time.
3) ALO obtains higher quality schedules for heterogeneous computing.
4) ALO makes full use of the instance-based heuristic information, which is discovered to be essential for scheduling problems.

4. GALO: Proposed Algorithm

A lot of daily services offered by cloud computing technology such as remote processing, file storage, software deployment, H/W infrastructure and many others, which in turn revealed an imbalance in the loads on cloud datacenter. Due to the heavy request for such services which are available and sharing among the cloud-users, it's essential to continuously enhance the performance and reliability of the cloud environment. The concept of load balancing is a very important aspect in the virtual environment of cloud computing, it helps to exploit the resources efficiently, saves time as it overcomes the VM-overloading, and provides the users with the high quality of services.

Therefore, this paper surveyed and explored the load balancing algorithms from traditional to heuristic, intelligent and Metaheuristic. All algorithms attempt to arrange and solve the task scheduling in the cloud, hence, this context proposes a new hybrid algorithm that is a response to present an optimal solution for scheduling.

A hybrid of the greedy search and Antlion optimizer, GALO, is proposed as a new algorithm to overcome the local states and scheduling task to global ones. This algorithm is called Greedy Antlion optimizer (GALO). Such as others load balancing algorithms, GALO required many computation operations however it is succeeded to enhance the result in VM configurations in numerous cases. Figure 4 shows the idea of the GALO algorithm and illustrates the main idea of the proposed algorithm. GALO uses ALO algorithm to handle task scheduling depend on an expected execution time and adjust the shared parameter of the greedy to reduce the situations in which the greedy algorithm gets trapped into local optima.

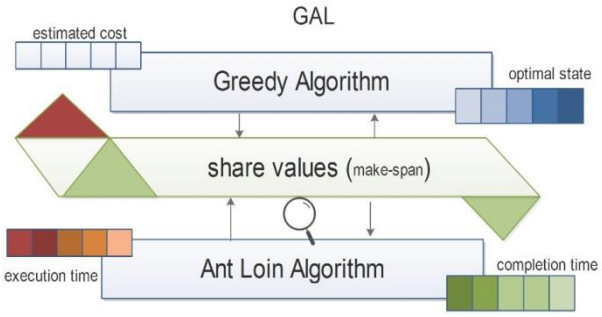

Fig. 4: GALO Wok Mechanism.

GALO algorithm is implemented in two modes, they are interacted to find the recommended arrangement for cloud-VMs, which are illustrated as follows:

i) Cost Mode: the greedy algorithm enumerates resource allocations for the VMs based on the estimated cost of the given workloads. The enumeration process used to evaluate the performance share VM according to the estimated execution time, while, the total of estimated costs is calculated using given workloads under candidate VMs.

ii) Time execution Mode: - ALO algorithm handles task scheduling depend on an estimated execution time. The handling process achieved by manipulating the scheduling problem through VMs in a cloud environment. The processing time is used to represent the processing time expected for every task on every resource; this is done before cloud scheduling is started. Accordingly, Eq.8 shows the main computing parameters for completion time that completely depends on task constitution (instructions and data) which in turn to specify the resources allocation features.

$$C = ET + R \quad (8)$$

Where ET is the expected execution time of task T and R denotes the expected time which resource will become ready to execute a task after finishing the execution of all tasks assigned to it. GALO assumes inter-task communication and presupposes processing times of every task of every resource where one resource is used by each task. Substantially, in each iteration, a share function of resources is de-allocated from the VM which affect to performance and allocated to another VM. Furthermore, the GALO steps are listed below:

Step1:- Assigning equal resource allocation for all cloud-VMs (1/N of each resource is allocated to each VM).
Step2:- Computing a total estimated cost for workloads under VMs configuration. Each VM adjusted using share's value using Eq. 2.
Step3:- Operating in repetitions style, the position of each particle in Eq. 2 (share's value) evaluates by running the greedy algorithm in order to determine an optimal state.
Step4:- Updating the share's value of VM and assigning sample task to VM.
Step5:- Computing the execution time for each task using ALO, after achieving the search on cloud-VMs.
Step6:- According to step4 and step 5, ALO algorithm allows the greedy algorithm to escape from the trap of the local optimal solution to a globally optimal solution.
Step7:- After the repetitions terminate, VM configuration R is set to the configuration of the optimal value and optimal execution time.
Step 8:- Finally, GALO reveals optimal sharing resources and reveals an optimal execution time and cost.
Moreover, Figure 5 described the pseudo code of the proposed algorithm.

Input: $R = [\frac{1}{N}, \frac{1}{N}, \dots, \frac{1}{N}]$ //Initial resources for each cloud-VM
$W_{vm} = [W_{vm1}, W_{vm2}, \dots, W_{vmn}]$ // workloads for cloud datacenter
$L_{vm} = [L_{vm1}, L_{vm}2, L_{vm3}, .., L_{vmn}]$ // location of each VM through cloud infrastructure
$T_{vm} = [t_{vm1}, t_{vm2}, t_{vm3}, .., t_{vmn}]$ // tasks
Calculation: compute the approximate of an optimal value for workload and execution time.
Output: $Reco = [r_{ct1}, r_{ct2}, \dots, r_{ctn}]$ //Recommendation configuration for each VM.

1) Initialize:
- Set the value of estimated cost for cloud-VM (resources)
- Set the value of current location (vm)

2) Begin
3) Compute the workload for each vm ($W_{vm}, T_{vm}(length), L_{vm}$)
4) Compute the estimation cost (W_{vm},L_{vm}, share) // resource needed for default configuration.
5) Compute the estimation of execution time (share, T_{vm}) // relation between share parameters (CPU and memory with task length)
6) For i = 1 to N do // begin with equal resources for all workloads
- Flag = false;
- Initial_cost = initial_config * Cost(Wi, Ri); // Initial cost under an initial configuration for each vm;
- Assign tasks to cloud-VMs (T_{vm}, L_{vm});
- Update the search boundaries initial_config using Eq. (3)

7) Repeat
- Max_Diff = 0; Max_Gain j = 0 Min_Loss j = ∞
- For j = 1 to R do // size of resources
- For i = 1 to N do // number of VMs

New_cost = initial_config * Initial_cost (Wi, [t_{vmnew} +share jnew,….., R])
availability= initial_config - New_cost ;
if (availability > Max_Gain j) then
Max_Gain j = availability ;
Else
Max_Gain j= New_cost ;
End if
- Loss= New_cost -confi ;
- if (loss < Min_Loss j) then

Min_Loss j = New_cost - initial_config ;
End if
- if (Max_Gain j – Min_Loss j > Max_Diff) then

Max_Diff = Max_Gainj – Min_Lossj;
Compute LB according Eq. 6;
Return(rvm);
Endif

8) Update the position of task according to share parameters using Eq. (7)

Calculate execution time(et)
If (et <initial time)
Et=initial time;
Return Et
Return New_cost;
Else
Flage =ture;
Return(rvm);
Endif

9) Until done;

Fig. 5: Pseudo Code for GALO Algorithm.

Figure 5 explains that the GALO algorithm needs $O(n^2)$ as a cost of execution time that is stemmed from the figure 5 and it takes four primary requested inputs ($R, W_{vm}, L_{vm}, T_{vm}$) and produce one global output, it's a recommendation configuration (Reco). The computations inside the GALO algorithm are terminated when the value of cloud-VMs configuration reach to optimal state, it means, when the task assignment and processing complete in optimal environment.

Practically, every cloud-user sends tasks through the internet or intranet, then task scheduler utilizes the GALO, to be assigned to the appropriate VMs to complete the processing operation. Then, the greedy algorithm makes a decision for increasing and decreasing the allocated resources to VMs based on the estimated cost of the given tasks. In the end, the greedy search algorithm gives a report of the recommended configuration for all virtual machines regrading to estimated execution time that is concluded by ALO. After that, the results return to the cloud-users, each user can produce any number of tasks form any device. Nevertheless, every task has a different number of instructions, processing time, data size, location and output result. So, the scheduler shares the estimated optimum VM regarding task needs based on estimated execution time.

From above, GALO runs in two stages:-

i) VM allocating: a greedy algorithm provides a minimum strategy for the ALO algorithm to select an optimum VM in searching round.
ii) Task prioritizing and distribution: ALO employed to generate task sequences and a new heuristic value is obtained to find better task priorities and speed up the searching process.

Finally, GALO successively assigns tasks regarding their significances and simultaneously looks for the most suitable VM for each task. Iteratively, ALO reveals a high-quality scheduling result to cloud-user after complete the search process.

5. GALO analytical analysis

In this section, the proposed algorithm is analyzed based on the load balancing, cost, and the execution time.

5.1. Load balancing

The aim of this paper is to conclude that the GALO algorithm fragments the shared resource, CPU, into equal allocations when the workloads are identical or un-identical assignments - i.e., the GALO algorithm is efficient in detecting the workloads, which reflects the fair distribution of the shared resource.

Each workload consists of a random combination of between a number of tasks and task contents. Generally, Each VM runs one workload and each algorithm starts with one VM and increases by 1 till reaches heavy workload and heavy make-span. Due to dynamic allocation and searching, GALO promises an acceptance estimated cost, lower than the estimated cost obtained by the greedy or Ant Lion algorithm separately.

In a nutshell, the greedy algorithm achieves the greatest result when it local optimum workloads and cannot improve the VMs configuration [17]. However, GALO suggests an optimal configuration for cloud-VM by using the make-span share parameters.

Inevitably, the advent of intelligent dynamic scheduling algorithm (GALO) helps to solve the problem of load balancing in a cloud environment by the possibility of a combination between GALO algorithm and any profiling technique for random workloads proprieties, which in turn the continuous perception for the intensively of workloads. Also, GALO used to measure the randomness of the dynamic tasks variation by periodically implementing it and periodically changing the resource allocation in each time interval.

Such perception aides the cloud owner to allocate appropriate resources to incoming tasks, Consequently, The owner can arrange the workloads over multiple pools based on the intensively of workloads.

According to logical analysis, GALO promises to achieve better allocations in terms of total cost at the runtime and it is acceptable for obtaining an optimal configuration for cloud-VMs.

5.2. Cost

One of the most important criteria in the cloud computing environment is a cost of resource scheduling. Hence, all studies classified the optimal scheduling algorithm according to reduce the processing power and the cost (consuming resources) [17], [18]. GALO is showed to find an effect of varying the shared parameter to explore an optimal resource allocation for different tasks at the same time. GALO promises to get best results with a cost of resources, because of, the proposed algorithm used to find the value of the share parameter and escaping from local optima.

Scientifically, GALO considered an intelligent heuristic algorithm, so, it works fine through the dynamic search on share parameters (processor, memory) for each VM under a cloud datacenter. Also, it also works for cost optimization at the cloud provider, while rescheduling and reconfigure for tasks and VMs in order to make space for a newly arrived query. However, GALO algorithm does not work when all the VM are super-busy.

5.3. Execution time

All proposed algorithms are good in some features however none of them achieve 100% efficiency and not able to give very less execution time GALO algorithm promises to present an efficient solution for speed up the execution time for assign tasks by achieving an intelligent heuristic search. The efficient solution is coming from the working mechanism (metaheuristic method) of proposed algorithm that is summarized in two steps: (i) calculate the total execution time for the problem resource comparing to the make span of the task solution (heuristic searching into task prioritizing), (ii) attempt to move or exchange set of tasks from the loaded VM to another resource that has the minimum make span (greedy strategy into machine allocating).

Substantially, the search operation is executed on each overweight VM and continues until there is no more enhancement in the configuration value, which in turn reach to best task scheduling with minimum execution time.

Finally, through the implementation phase, the belief will be in increasing the expansion of throughput through high and frequent access to cloud-VMs, which in turn to help the cloud system to achieve the equally work and data transferred without effect on the performance. Theoretically, the performances of the cloud environment heavily rely on the effectiveness of the heuristic search indicators.

6. Conclusions and Future Directions

This paper proposed a new cloud task scheduling algorithm, which integrates and deploys both the greedy based local heuristic solution and ant lion optimizer algorithm in a hybrid style aiming to (i) provide an ideal task assignment (load balancing) via optimal VM-configuration, (ii) optimize the performance in cloud system through improving the execution time, and (iii) reduce the make-span for global tasks processing. Moreover, this paper innovates a new intelligent heuristic search scheduling algorithm by applying GALO algorithm, such an algorithm performs a priority-based searching style in to yield task sequences and it performs low-complexity by greedy allocation.

A contribution of this paper can summarize in developing of scheduling process in cloud environment from the transition probability to produce searching paths during scheduling, which in turn, improving the scheduling quality (quality of services) in metrics of make span, speed up, and frequency of better results.

Our attempt enjoys certain advantages when compared with the others, especially with respect to the makespan and task scheduling. It can be considered as the first cloud environment that integrates both

the intelligent search and heuristic mechanisms. The preliminary study of the proposed model shows promising results where it shows the availability and the reliability of the task scheduling and processing can be easily achieved. In the future, an algorithm will be developed and simulated in the real cloud environment that includes at least 20 VMs in order to produce practical results and provide a deep interpretation and explanation.

Acknowledgement

I acknowledge university of Fallujah for support me to complete this scientific work and thanks to my colleagues in the computer center.

References

[1] Kirit J. Modi; Debabrata Paul Chowdhury; Sanjay Garg, Automatic cloud service monitoring and management with prediction-based service provisioning, International Journal of Cloud Computing, vol. 7, issue 1, 2018.

[2] Shieny, J. A Survey on Cloud Computing: Architectures, Data Storage, Services, Security and Applications-manager's Journal on Cloud Computing, vol. 4, issue 2, pp 30-38, 2017.

[3] Nagamani H. Shahapure and Jayarekha P, "Load Balancing with Optimal Cost Scheduling Algorithm", 2014 International Conference on the computation of Power, Energy, Information and Communication (ICCPEIC), 16-17 April 2014, Chennai, India.

[4] Kaveh Khorramnejad, Lilatul Ferdouse, Ling Guan and Alagan Anpalagan, Performance of integrated workload scheduling and prefetching in multimedia mobile cloud computing, Journal of Cloud Computing, Vol. 7, issue 13, 2018. https://doi.org/10.1186/s13677-018-0115-6.

[5] Zhifeng Zhong, Kun Chen, Xiaojun Zhai, and Shuange Zhou , "Virtual Machine-Based Task Scheduling Algorithm in a Cloud Computing Environment, TSINGHUA SCIENCE AND TECHNOLOGY, Volume 21, Number 6,pp 660-667, 2016. https://doi.org/10.1109/TST.2016.7787008.

[6] Qiang Guoa, "Task Scheduling Based on Ant Colony Optimization in Cloud Environment", 2017 5th International Conference on Computer-Aided Design, Manufacturing, Modeling and Simulation (CDMMS 2017), AIP Conf. Proc. 1834, 040039-1–040039-11; https://doi.org/10.1063/1.4981635.

[7] Ji Lia; Longhua Fenga, Shenglong Fang, "An Greedy-Based Job Scheduling Algorithm in Cloud Computing", JOURNAL OF SOFTWARE, VOL. 9, NO. 4, 2014.

[8] Junwei Ge1, Qian He, and Yiqiu Fang, "Cloud computing task scheduling strategy based on improved differential evolution algorithm", AIP Conference Proceedings 1834, 040038, 2017.

[9] Abdul Razaque, Nikhileshwara Reddy Vennapusa, Nisargkumar Soni, Guna Sree Janapati, khilesh Reddy Vangala "Task Scheduling in Cloud Computing", 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT), 29 April 2016, Farmingdale, NY, USA.

[10] Suraj, P., A Particle Swarm Optimization (PSO)-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments. 2010.

[11] Asif Mohammad, Ashish Kumar, Lal Shri Vratt Singh, "A Greedy Approach for Optimizing the Problems of Task Scheduling and Allocation of Cloud Resources in Cloud Environment, Volume: 03 Issue: 09, Sep-2016.

[12] Radhya Sahal, Sherif M. Khattab, Fatma A. Omara, GPSO: An Improved Search Algorithm for Resource Allocation in Cloud Databases, 2013 ACS International Conference on Computer Systems and Applications (AICCSA), 27-30 May 2013, Ifrane, Morocco.

[13] Gamal F. Elhady and Medhat A. Tawfeek, Comparative Study into Swarm Intelligence Algorithms for Dynamic Tasks Scheduling in Cloud Computing, 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS'15), Dec. 2015, Cairo, Egypt.

[14] Melika Mani, Omid Bozorg-Haddad and Xuefeng Chu, Antlion Optimizer (ALO) Algorithm, Advanced Optimization by Nature-Inspired Algorithms pp 105-116, 2017.

[15] Seyedali Mirjalili, "The Antlion Optimizer", Advances in Engineering Software, 83 (2015) 80–98. https://doi.org/10.1016/j.advengsoft.2015.01.010.

[16] Cristian Mateos, Elina Pacini, Carlos García Garino, An ACO-inspired Algorithm for Minimizing Weighted Flowtime in Cloud-based Parameter Sweep Experiments, Advances in Engineering Software, Volume: 03 Issue: 09, Sep-2014.

[17] Harshadkumar B. Prajapati, Vipul A. Shah, "Scheduling in Grid Computing Environment", 2014 Fourth International Conference on Advanced Computing & Communication Technologies, 8-9 Feb. 2014, Rohtak, India. https://doi.org/10.1109/ACCT.2014.32.

[18] Raja Manish Singh, Sanchita Paul, Abhishek Kumar, Task Scheduling in Cloud Computing: Review, International Journal of Computer Science and Information Technologies, Vol. 5 (6), 2014, 7940-79443.