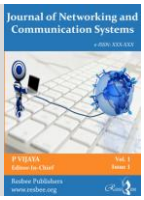
Resbee Publishers

Journal of Networking and Communication Systems

Received 7 September, Revised 19 November, Accepted 27 December



Job Scheduling in Cloud Environment Using Lion Algorithm

Brammya G

Department of Applied Electronics,
Arunachala college of engineering
Nagercoil, Tamil Nadu, India
brammyagece@gmail.com

Angelin Deepa T

Department of Computer Science and Engineering,
Bethlahem College of Engineering
Ulaganvillai, Tamil Nadu, India
Deepalibin1992@gmail.com

Abstract: Cloud computing is an emerging technology in the field of service oriented computing and software engineering. Security is a critical issue in cloud environment as it possess huge amount of data. In this circumstance, scheduling has become a challengeable mechanism and to utilize the resources in a secure manner we proposed a new metaheuristic algorithm called Lion algorithm (LA). In this paper, the scheduling problems are submitted to solve by eleven algorithms namely Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Optimisation with Differential Evolution (DEopt), Firefly Algorithm (FA), LA, Artificial Bee Colony (ABC), Glowworm Swarm Optimization (GSO), Bacterial Foraging Optimization (BFO), Gravitational Search Algorithm (GSA), Ant Lion Optimizer (ALO) and Cuckoo Search (CS) algorithm. Then the performances of the entire algorithms are compared with LA in terms of its cost function. The final result of convergence analysis and statistical analysis reported that LA has better convergence property and it possess better statistical metric values such as best, worst, mean, median and standard deviation than all the ten algorithms.

Keywords:*Cloud computing, metaheuristic, GA, PSO, DEopt, FA, LA, ABC, GSO, BFO, GSA, ALO and CS*

1. Introduction

In the emerging world, cloud computing is an unavoidable technology and it holds huge amount of different types of data, so it is called as Heterogeneous system [6]. This system works in a good manner when the server performance and resource utilization has improved; also it performs well by minimizing the processing cost, processing time and completion time. Thus, it is necessary to schedule the tasks in the cloud [6] and is termed as Task scheduling. Generally, scheduling is known to be NP-complete problem, which refers to a set of regulations to arrange the work to be performed by the system [7]. Many scientific fields such as Astronomy, Meteorology, Bio-informatics, Environmental science and Geological science has deal with large scale of data, however to handle such huge data workflow scheduling has been used in the cloud environment [8].

In addition to the optimal resource utilization, security has become another critical concern for a wide range of application on cloud computing [10-13]. Unfortunately, since distributed computing permits an enormous users to utilize a broad spectrum of unchecked third-part applications, both clients and applications can be the sources of security threats to cloud environment[15] [9]. A risk-based approach to the establishment of a security program that accepts appropriate controls will ensure that all users can be secured and that data can be privileged, have probity and be accessible [14].

Various scheduling mechanism has been employed in cloud computing to increase the efficiency of the system and it is compulsory to employ security services to protect security-critical workflow applications that are implementing on clouds from being attacked [1]. Commonly, the scheduling mechanism is carried out in two ways: Job Scheduling and Workflow scheduling. Most of the task scheduling problem has been solved by using the rule based algorithms [16] [17] as it is easy to implement, but it is inappropriate for large-scale applications (workflow scheduling). And hence some metaheuristic techniques have mostly applied to solve scheduling problems in grid and cloud computing. The most commonly used heuristic algorithms are Genetic Algorithm (GA) [18-21], Particle Swarm Optimization (PSO) [22-25], and Ant Colony Optimization (ACO) [26-29]. Owing to the better performance of Lion algorithm (LA) [30] over the existing methods, a modified version of LA has been used in our proposed work.

2. Literature Review

2.1 Related Works

First, confirm that you have the correct template for your paper size. This template has been tailored for output on the A4 paper size. If you are using US letter-sized paper, please close this file and download the file "MSW_USltr_format".

In 2016, Zhongjin et al [1] developed a security and cost aware scheduling (SCAS) algorithm which consider the heterogeneous task with data-intensive, memory-intensive or computation-intensive characteristics for scientific workflow applications. Mainly, SCAS algorithm is employed to reduce the cost of execution at which it meet the deadline and risk rate constraints. For their work, they utilized the coding approach of PSO to acquire the solution for multi-constraint and multi-dimensional optimization problem which occur in workflow scheduling.

In 2015, Xiaomin Zhu et al [2] designed an agent-based dynamic scheduling algorithm named ANGEL on the basis of bidirectional announcement-bidding mechanism. They also designed two selection strategies, MAX strategy and P strategy to determine the contractors. And finally, an investigation has made for the dynamic scaling up method which was used by ANGEL to enhance the schedulability, priority, scalability in virtualized cloud environment.

In 2016, He Hua et al [3] originate a PSO-based Adaptive Multi-objective Task Scheduling (AMTS) to acquire quasi-optimal solutions for task scheduling problem in cloud computing. Their goal is to obtain optimal task completion time, minimize energy consumption, average cost and minimize resource utilization. SPV has been established to convert the continuous position values of AMTS algorithm to a discrete task permutation. They concluded the experiment by stated as PSO-based AMTS algorithm is an effective scheduling algorithm to obtain better quasi-optimal solution.

In 2015, Ehab Nabiel Alkhanak et al [4] have discussed various workflow scheduling (WFS) challenges that affects the execution cost of WFS. Later, they determined some objectives to identify the cost-aware challenges of workflow scheduling in the cloud environment. The objective is to investigate state-of-the-art-cost-aware WFS approaches based on QoS, system architecture and system functionality and to establish relevant taxonomies of WFS challenges and finally to identify the correlation between the cost-aware WFS challenges and profitability.

In 2015, Chunling Cheng, Jun Li and Ying Wang [5] have developed an energy-saving task scheduling algorithm for cloud computing on the basis of vacation queuing model. Also they proposed a task scheduling algorithm to minimize the energy consumption. The experimental results showed that the energy consumed while performing a particular task can be reduced using this algorithm in cloud computing environment.

3. Work Flow Application Model For Cloud System

3.1 Work Flow Architecture

Workflow scheduling is one of the important issues in cloud computing environment which map the workflow operations to the virtual machines (VM) [32]. It is an NP- hard optimization problem as it is difficult to obtain an optimal schedule. As there are enormous virtual machines present in the cloud, a large number of user tasks have to be scheduled by taking various scheduling schemes and factors into account. The main aim of workflow scheduling is to reduce the makespan by allocating the tasks to the virtual machines in a proper manner [4]. Generally, workflow is classified into two types, simple and scientific workflow [9]. Workflow applications are usually employed to handle large-scale data. The complexities of large-scale problems are reduced by workflow application. Fig. 1 shows the workflow architecture, in which workflow scheduling process is carried out in various stages.

In the first stage, a high level workflow (abstract workflow) is constructed with the software components and the data that required for execution without the detail of resources. The second stage, called mapping stage optimized resource allocation by mapping the high level workflow to the physical resource and it provides a concrete workflow. Then, the mapped workflow is scheduled on the available resource and submitted for execution. After executing the scheduled workflow, its performance has monitored and the outcomes are gathered depending upon the needs. Thus, an efficient scheduling can enhance the performance of cloud computing [8].

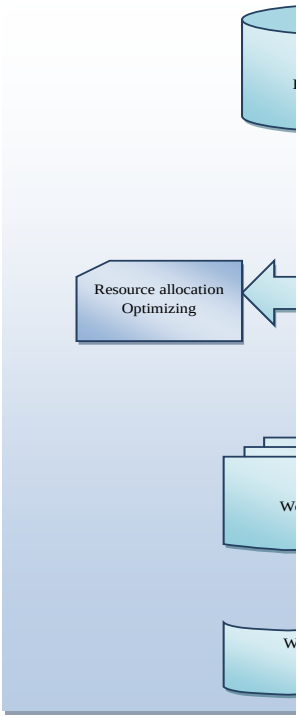


Fig. 1. *Workflow architecture of cloud computing environment.*

3.2 System Model

In general, scheduling is defined as the process of task/operation mapping by a pack of jobs, or a set of machines to a specific time period. Because of its complexity, scheduling is considered as a main problem in cloud computing environment. Scheduling becomes more complex when it is subjected to handle very large scale data. Thus the computer application which operates a large volume of data and allows most of its processing time to I/O, then the data manipulation is considered as data-intensive. Workflow simplifies the data-intensive application in which the application is decomposed into smaller task and it is processed in order to acquire the desired result. The basic data-intensive application is constructed by a set of machine or computing unit that are connected directly or indirectly with a set of data resource in which database is connected with the server using a connection link which represents the corresponding bandwidth as shown in Fig. 3. The central processing unit (CPU) count, memory and storage space constitute the computing unit capacity. The speed of each computing unit (Processing speed) is expressed as the number of cycles per unit time.

Let N_m be the total computing units, and then each computing unit or machine can be denoted as $\{CM_1, CM_2, ..., CM_{N_m}\}$. The security rank that provides computing service for each computing unit $CM_{i\in(1,2..N_m)}$ is represented as CS_{r,CM_i} . If the data-intensive application consists of N_r number of data resources, then it is denoted as $\{DR_1, DR_2, ..., DR_{N_r}\}$, then each data resource $DR_{i\in(1,2,..N_r)}$ can provide the data service with the security rank DS_{r_i} . The security rank of the computing units and the data resources are commonly termed as Sd_{TO}. A job consists of a set of task/operation that has to be performing in a machine with processing constraints [31]. Fig. 2 shows the processing constraints in the cloud.

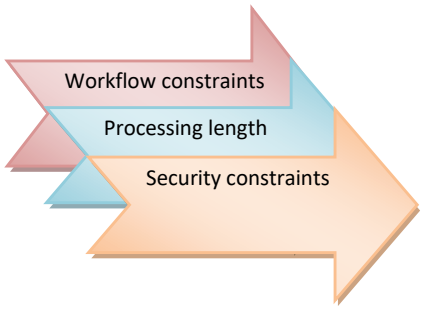


Fig. 2. *Processing constraints in cloud environment*

Arranging certain task to a specific application is referred as workflow and each task can be performed after the completion of the previous task in a sequence order is the workflow constraints. The number of cycles that required for the accomplishment of an operation is termed as processing length. The main theme in cloud computing are security constraints and that is explained briefly next.

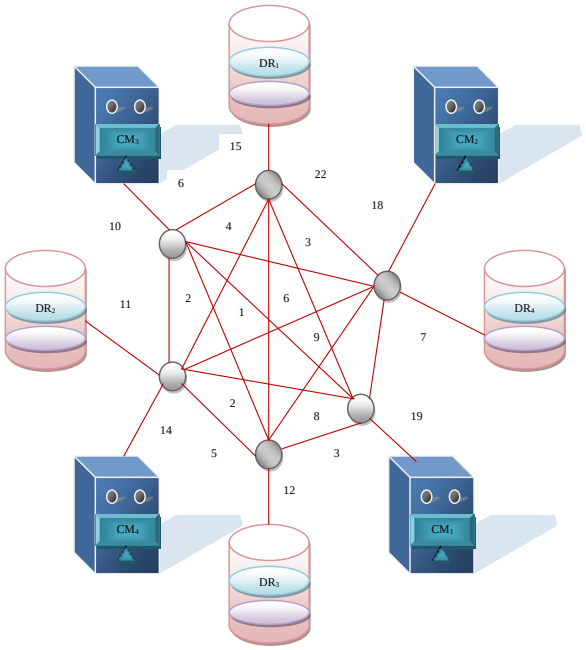


Fig. 3. Data-Intensive Computing Environment with four computing units and four data resources

4. Security Constraints

Let us consider a workflow application which consists of j jobs $\{JB_1, JB_2, ..., JB_j\}$ and the q^{th} job consists of a set of task or operation $\{TO_{q,1}, TO_{q,2}, ..., TO_{q,p}\}$. Then to reduce the complexity, all the task or operation are combined together as $\{TO_1, TO_2, ..., TO_m\}$. The computing service has the security demand CS_{d,TO_i} and the data service security demand of the operation $TO_i (i = 1, ..., m)$ is represented as DS_{d,TO_i} . Both the security ranks are combined together and termed as Sr_{CM} .The three security modes [31] that are available in job scheduling process are given in Fig. 4.

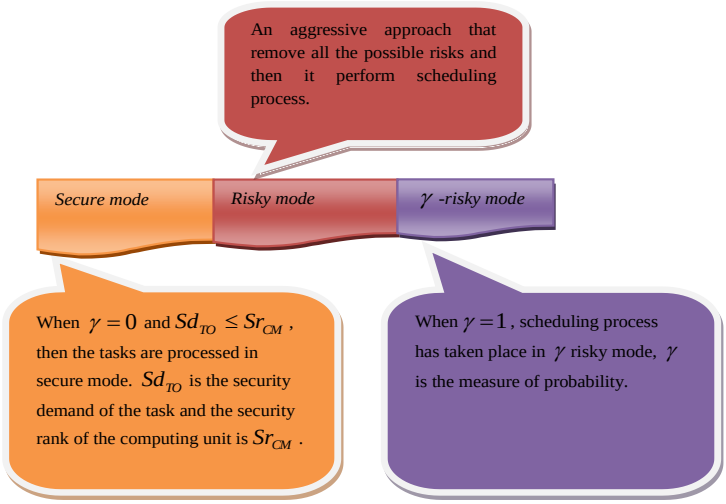


Fig. 4. Security mode for job scheduling in data-intensive compuing environment

As secure mode is very expensive to achieve and so risk mode and γ-risky mode is commonly employed for scheduling the job in cloud environment. Also the security levels are approached by a fuzzy or qualitative scale of five levels, namely very high, high, medium, low and very low [31].

In the security constraint model, the probability of risk is given in eq. (1)

$$\text{Prob}_{\text{risk}} = \begin{cases} 0 & \text{if} \quad \text{Sd}_{\text{TO}} \quad \text{Sr}_{\text{CM}} \le 0 \\ 1 \quad e^{\,0.5(\text{Sd}_{\text{TO}} \quad \text{Sr}_{\text{CM}})} & \text{if} \quad 0 < \text{Sd}_{\text{TO}} \quad \text{Sr}_{\text{CM}} \le 1 \\ 1 \quad e^{\,1.5(\text{Sd}_{\text{TO}} \quad \text{Sr}_{\text{CM}})} & \text{if} \quad 1 < \text{Sd}_{\text{TO}} \quad \text{Sr}_{\text{CM}} \le 2 \\ 1 & \text{if} \quad 2 < \text{Sd}_{\text{TO}} \quad \text{Sr}_{\text{CM}} \le 5 \end{cases} \tag{1}$$

A machine is said to be safe when it satisfy the condition, $\text{Sd}_{\text{TO}} \le \text{Sr}_{\text{CM}}$ or $Sd_{TO} \quad Sr_{CM} \le 0$ and thus its risk probability has become zero. When the scheduled operation is allocated to a machine with the condition $\text{Sd}_{\text{TO}} > \text{Sr}_{\text{CM}}$ then its risk probability is less than 50% and also the machine will process the task while it met the condition $0 < \text{Sd}_{\text{TO}} \quad \text{Sr}_{\text{CM}} \le 1$. When $1 < \text{Sd}_{\text{TO}} \quad \text{Sr}_{\text{CM}} \le 2$ the scheduled tasks will get delay but it will be executed before the deadline. When $2 < \text{Sd}_{\text{TO}} \quad \text{Sr}_{\text{CM}} \le 5$, the task or operation cannot be completed and hence it has to be rescheduled.

5. Secure Work Flow Scheduling Using Lion Algorithm

5.1 Scheduling Model

Let $\{P_{s1}, P_{s2}, ..., P_{sN_m}\}$ be the processing speed and $\{p_{l1}, p_{l2}, ..., p_{lm}\}$ be the processing length of the workflow application, and it will be executed by the computing unit, which is then subjected to a set of security constraint $S_c = \{\text{Sd}_{\text{TO}}, \text{Sr}_{\text{CM}}\}$.

Generally, workflow applications are represented by the Directed Acyclic Graphs (DAGs) as $G\{\text{TO}, D\}$ in which vertices, $\text{TO} = \{TO_1, TO_2, ..., TO_m\}$ denotes the individual task of the workflow and edge is denoted as $D \in (TO_i, TO_q)$, which implies that the task TO_q cannot initialize its process until the task TO_i accomplished its process and send an acknowledgement to TO_q about the completion of task. Let $\text{pre}(TO_i)$ be the predecessor of the task TO_i and $\text{suc}(TO_i)$ be the successor of the task TO_i. When the successor task got an acknowledgement from its predecessor, it starts to perform as it shows the machine availability. A task can be performed by a machine at a time and it cannot be re-processing after its completion.

The relations between the tasks or operations are commonly related by a matrix called Flow Matrix (FM). Usually, the flow matrix $f_m = \lfloor F_{i,q} \rfloor$ is designed as, if (TO_i, TO_q) exists in the graph, then $F_{i,q}$ carries its weight and if it doesn't exist, then $F_{i,q}$ is assigning as -1. However, the retrieval matrix $r_m = \lfloor R_{i,q} \rfloor$ is used to determine the data resource dependencies of tasks, the retrieval time for the task TO_i is indicated by $R_{i,q}$ that performs the data retrieval from the data resource DR_q. Additionally, the throughput rates are evaluated by the two metrics $X = \lfloor x_{i,q} \rfloor$ and $Y = \lfloor y_{i,q} \rfloor$, in which $x_{i,q}$ represents the

connection link capacity between the computing units CM_i and CM_q, and $y_{i,q}$ is the connection link capacity between the computing unit CM_i and the data resource DR_q. The overall completion time for each task is calculated by the sum of the time taken for retrieve data and the time taken to input data, and the execution time of an operation by the available machine.

Let us consider a possible solution, $V = \{V_1, V_2, .., V_m\}$, in which V_i is the serial number of the computing unit to which the task TO_i is assigned, then the completion time $C_{time(TO_i)}$ of the task TO_i on the machine CM_{V_i} can be determine using the eq. (2)

$$C_{time(TO_i)} = \sum_{\substack{u=1 \\ F_{u,i} \neq 1}}^{m} F_{u,i} x_{V_u, V_i} + \sum_{g=1}^{d} R_{i,z} y_{V_i, g} + p_{l_i} / P_{sV_i} \tag{2}$$

The maximum or peak completion time is referred as makespan $MS_{C_{max}}$ is computed using the eq. (3),

$$MS_{C_{t\_max}} = \max\left\{ \sum C_{time(CM_i)} \right\} \tag{3}$$

The total completion time of the solution is calculated using the eq. (4),

$$C_{time\_total} = \sum_{i=1}^{c} \left(\sum C_{time(CM_i)} \right) \tag{4}$$

The above eq. (3) & eq. (4) are considered as the performance criteria for job scheduling problem in cloud computing environment. The executing time gets reduced by minimizing C_{time_total} and by minimizing $MS_{C_{max}}$ the execution time will increase. By doing so, no task/operation requires too long time to execute and to achieve an equal balance between these two facts, a weighted aggregation is introduced as in eq. (5),

$$F = wt_1 \{ MS_{C_{t\_max}} \} + wt_2 \{ C_{time\_total} \} \tag{5}$$

In the eq. (5), wt_1 and wt_2 are non-negative weights and the summation of both weights should be unity, it can be achieved by fixing or adjusting the values.

The main objective of the scheduling problem is to find the task to be performing and allocate certain task to certain machine in certain order by satisfying the security constraints and finally to minimize F.

6. Proposed Heuristics

In the proposed method, the job scheduling problem has been solved using an algorithm called Lion Algorithm (LA) in the data-intensive computing environment.

Lion Algorithm*:* On the basis of the natural behavior of Lion, the Lion algorithm [30] was originated in the year 2012. According to LA, a strong solution (territorial lion) will defeat a random solution (nomadic lion), and so the weak solution (weak lion or cubs) may disappear from the solution pool. The solution that was obtained from the successful solution is stronger than the solution that was obtained from the defeated solution as the lion succeeded in the territorial takeover and territorial defense that causes due to the lack of success of some solutions (laggard lion).

The basic functions of LA algorithm and its pseudo code are discussed below,

Pride Generation: Let us initialize the arbitrary solutions L^{male}, L^{female} and L_1^{nomad} of L^{male} and L^{female} of the pride, all are referred as Q. The elements of the arbitrary solutions are $L^{male}(x)$, $L^{female}(x)$ and $L_1^{nomad}(x)$ respectively, which represents arbitrarily selected locations.

Fitness Evaluation: The fitness function of the arbitrary solutions are determined as $f(L^{male})$, $f(L^{female})$ and $f(L_1^{nomad})$ using the eqn. (6). Eventually, the initialization has been done as $ft_{ref} = f(L^{male})$ and the generation counter $G_c = 0$, which is applied in the termination stage. For the future progress, L^{male} and L^{female} are saved.

Fertility evaluation: The algorithm for the fertility evaluation is illustrated in the following pseudocode:

Process: Fertility Evaluation
Input : $L^{male}, L^{female}, ft_{ref}, r_L$ and r_S
Output : $L^{male}, L^{female}, ft_{ref}, r_L$ and r_S
// L^{male} Evaluation
If $ft_{ref} \le f\left(L^{male}\right)$
    $r_L \leftarrow r_L + 1$
else
    Reset r_L
    $ft_{ref} \leftarrow f\left(L^{male}\right)$
End if
// L^{female}Evaluation
If r_S is not tolerable
    Set fe_{uc} and fe_{gc} to zero
    Do
        Calculate $L^{female+}$
        $fe_{gc} \leftarrow fe_{gc} + 1$
        If $f\left(L^{female+}\right) < f\left(L^{female}\right)$
            $fe_{uc} \leftarrow 1$
            $L^{female} \leftarrow L^{female+}$
            Reset r_S
        End if
    Until fe_{gc} reaches $fe_{gc}^{\max}$
End if

In the fertility evaluation stage, the fertility of both L^{male} and L^{female} are verified and evaluated to overcome the problem of convergence at the local optima. Here, $L^{female+}$ is the updated female lion and ft_{ref}, fe_{gc} and fe_{uc} are the reference fitness, female generation count and female update count respectively. At beginning, the sterility rate r_S and r_L are initialized as zero and finally it get the determined value. It is essential, to verify the tolerance level of the sterility rate, whether it reaches $r_S^{\max}$. The value of $r_S^{\max}$, $L_r^{\max}$ and $fe_{gc}^{\max}$ and all LA elements are evaluated as per the guidelines given in [30]**Mating:** In LA, the mating process of L^{male} and L^{female} has undergone by crossover and mutation operation as in the evolutionary optimization process. Here, the crossover operation is processed on the basis of the littering rate of lion to generate L^{cubs} and L^{cubs} undergoes the mutation process with the probability $P_{mutation}$ and hence an equal number of new cubs L^{new} are generated and that are located in the cub pool. At the end of the two operations, the male cub L^{m_cub} and the female cub L^{f_cub} are extracted from the L^{cubs} based on the fitness function.

Cub growth: The local solution search function is referred as cub growth in which L^{m_cub} and L^{f_cub} are permitted for a random mutation with a rate g_r. However, if the mutated L^{m_cub} and L^{f_cub} is effective than the old L^{m_cub} and L^{f_cub} then they the old ones are replaced by the mutated L^{m_cub} and L^{f_cub}. Then, I_{cub} is incremented by one at each iteration of cub's growth towards maturity and finally the effective local solution of both L^{m_cub} and L^{f_cub} has been searched with $g_r < 0.2$ and it is not necessary to be equal to the mutation rate r_M.

Territorial Defense: The territorial defense directs LA algorithm to find the search space and it is presented in the sequence order of nomad coalition, survival fight, pride and the nomad coalition updates. The territorial defense level of LA is illustrated in the following pseudocode

Process : Territorial Defense
Get nomad coalition
  Select l_{e_nomad}
  if l_{e_nomad} wins
    $l^{male} \leftarrow l_{e\_nomad}$
    Remove l_{e_nomad} from nomad world
    Kill l^{m_cub} and l^{f_cub}
    Reset age (cubs)
    Defense result $\leftarrow 1$
  Else
    Update nomad coalition
    Defense result $\leftarrow 0$
  End if

In this level, two nomadic lions are introduced in which L_1^{nomad} is initialized in the first step and, L_2^{nomad} has been initialized on the basis of laggardness rate r_L. If L^{male} is not laggard, then L_2^{nomad} is initialized as similar as L_1^{nomad}, or else L_2^{nomad} is initially given as the updated model of L^{male} by the mutation process with the rate of $1 \quad r_M$. On the basis of the pride and the nomad coalition, territorial fight arises between the nomadic lions. Moreover, winner take approach has been used and thus, the won nomadic lion gets occupied within the coalition in the territorial defense.

The nomadic lion L_{e_nomad} is chosen from the survival fight if it met the coalition criteria [30]. The pride gets updated, when $L^{male}$is exchanged by L_{e_nomad} and the nomad coalition gets updated if L_{e_nomad} is defeated. Finally, the update process has been done by choosing only one L^{nomad} with K^{nomad}, greater than or equal to the exponential of unity.

Territorial takeover: The process of supplying territory to the matured L^{m_cub} and L^{f_cub} when it becomes stronger than L^{male} and L^{female} is referred as territorial takeover. The Territorial takeover stage is illustrated in the following pseudocode

Process: Territorial Takeover
If $f\left(L^{male}\right) > f(L^{m\_cub})$
  $L^{male} = L^{m\_cub}$
End if
$L^{old} = L^{female}$
If $f\left(L^{female}\right) > f\left(L^{f\_cub}\right)$
  $L^{female} = L^{f\_cub}$
End if
If $L^{female} \neq L^{old}$
  Clear r_S
End if

In the territorial takeover stage, the process begins to start only if $I_{cub} \geq I_{\max}$, otherwise the cub starts to grow. When the effectiveness of the female cub L^{f_cub} is discovered then L^{f_cub}occupies the position. Mostly, this kind of L^{f_cub} will be fertile. And hence r_S reoccupies the zero position and $I_{\max}$ is proportional to the cub maturity. Thus, one generation is considered as completed and then N_{gen} is incremented by one.

Termination: LA algorithm has reached the termination stage when it achieves maximum number of fitness evaluations. Once it gets terminated, L^{male}has returned as the optimal scheduling of job in the cloud computing environment.

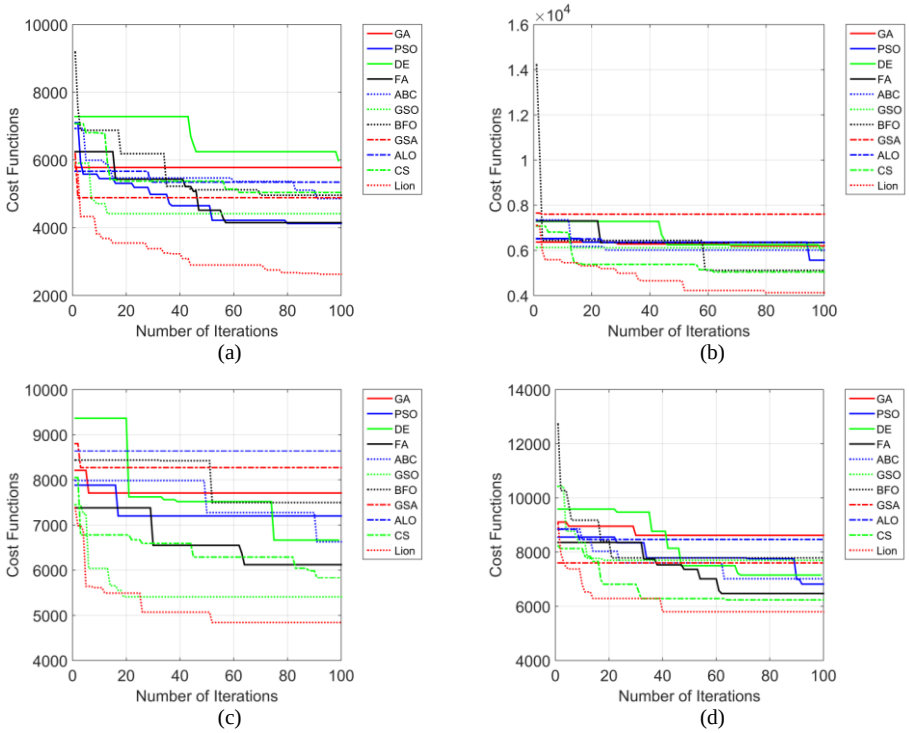
7. Simulation Results

7.1 Simulation Setup

The simulation and the implementation of LA algorithm for solving job scheduling problem in cloud environment have been done in MATLAB R2015a. Then the results are compared with other algorithms (GA, PSO, DEopt, FA, ABC, GSO, BFO, GSA, ALO and CS) in terms of convergence and cost statics to measure the effectiveness of the proposed LA. The scheduling problems have been simulated in ten cases; a set of $JB_{j\in\{1,2...10\}}$ jobs and $CM_{N_m\in\{1,2...10\}}$ computing units/machines constitute each case. Let the scheduling problem be case 1: $\{500,50\}$, case 2: $\{1000,100\}$, case 3: $\{1500,150\}$, case 4: $\{2000,200\}$, case 5: $\{2500,250\}$, case 6: $\{3000,300\}$, case 7: $\{3500,350\}$, case 8: $\{4000,400\}$, case 9: $\{4500,450\}$ and case 10: $\{5000,500\}$. Then the experiment has been run for five times over 100 iterations with different random inputs for each algorithm.

7.2 Convergence Analysis

The convergence behaviour of all the eleven algorithms on solving the entire ten cases is illustrated in Fig. 5. It can be seen from Fig. 5 that, on solving case 1 problem ALO initialized with best solution whereas LA initialized with worst solution but it converges earlier than all algorithms and obtain the best solution with 37.65% better than PSO. In case 2, though LA initialized with worst solution, it obtain the best solution with early convergence and its performance is 18% better than CS. Both LA and FA initialized the best solution, but only LA acquires the best solution for case 3 problem and it performs 26.30% better than GSO. In case 4, GSA initialized the best solution, but LA obtains the best solution with early convergence and hence LA performs 1.80% than CS. Though LA converges later, it performs 11.97% better than GSO to obtain the best solution for case 5 problem. The case 6 problem was solved by all algorithms, in which ALO initialized the best solution and CS converges earlier and LA obtain the desired solution with 19.23% better than ABC. LA has good convergence property and it obtain the best solution for case 7 problem, also it performs 1.5% better than PSO. In case 8, PSO initialized the best solution whereas GSA converges earlier, but LA obtains the best solution than all algorithms and it performs 1.49% better than PSO. LA performs 3.84% better than GSO and it obtain the best solution for case 9 problem than all the algorithms. The best solution is obtained by LA as it converges earlier than all the algorithms and its performance is 6.49% better than GSO.



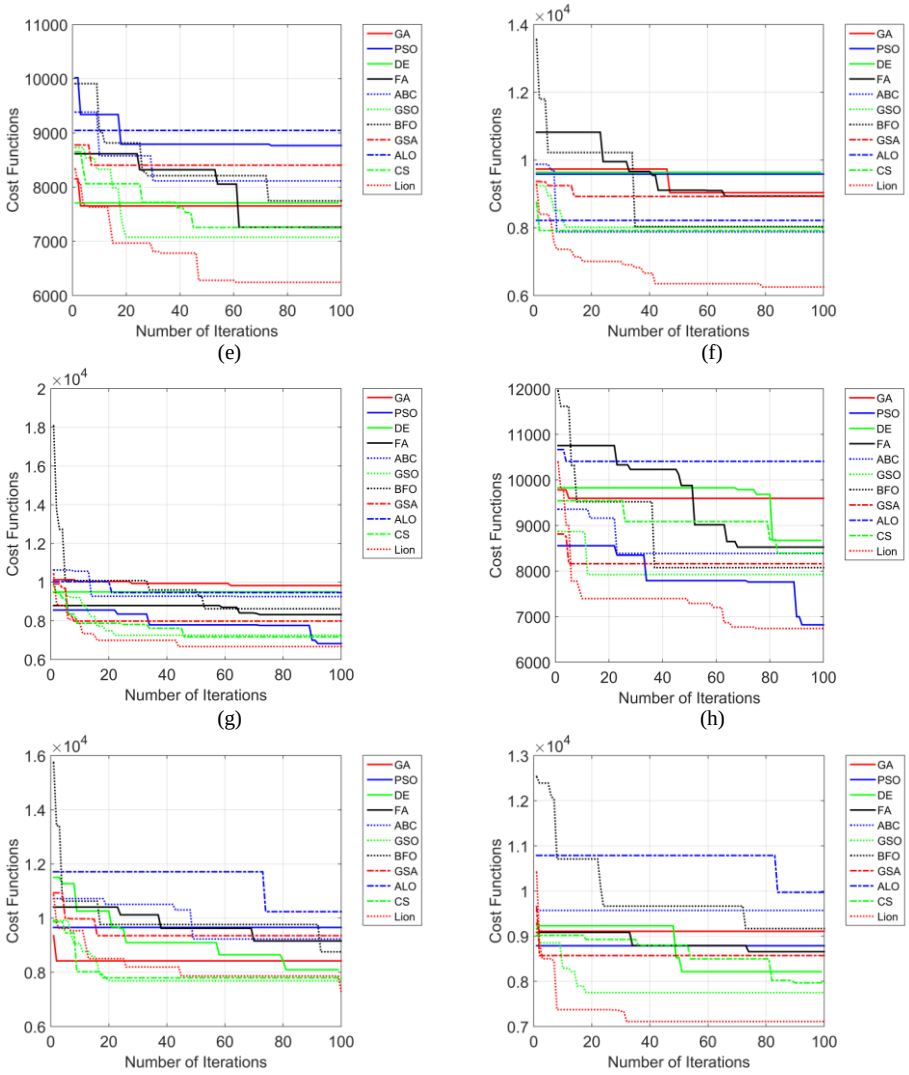


Fig. 5. *Convergence behaviour of eleven algorithms for (a) case1 (b) case 2 (c) case 3 (d) case 4 (e) case 5 (f) case 6 (g) case 7 (h) case 8 (i) case 9 (j) case 10*

7.3 Statistical Analysis

In this section, the statistical information of the entire eleven algorithms on solving all the eleven cases are analyzed and the results are tabulated in Table I, Table II, Table III, Table IV, Table V, Table VI, Table VII, Table VIII, Table IX and Table X. The statistical analysis of all the eleven algorithms in terms of cost function has reported that the mean value of LA on solving case 1, case 2, case 3, case 4, case 5, case 6, case 7, case 8, case 9 and case 10 are 99.93% better than PSO, 0.21% better than CS, 0.81% better than CS, 13.71% better than CS, 8.30% better than GSO, 2.86% better than GSO, 3.43% better than PSO, 7.18% better than PSO, 6.52% better than CS and 6.12% better than CS respectively.

Table 1. *Cost Statistical Report for case 1*

1	Best	Worst	Mean	Median	STD_DEV
GA	3610.5	5981.5	5164.6	5682.4	985.6
PSO	3322.8	2021.2	2.2663	89.26	74.235
DEopt	5684.6	11.804	2.3608	5248.2	5.2789
FA	3475.8	4815.3	4282.7	4352.9	517.11
LA	3228.8	3952.5	3623.4	3651.5	267.63
ABC	3832.2	4861.1	4434.3	4456.1	376.06
GSO	3691.7	4848.3	4377.5	4414.5	434.59
BFO	5125.6	8499.8	6707.2	6670.1	1246.6
GSA	4888.1	6023.5	5665.2	5814.7	448.74
ALO	5764.6	21930	11330	8693.1	6271.5
CS	5503.1	208.67	102.38	94	75.676

Table 2. *Cost Statistical report for Case 2*

2	Best	Worst	Mean	Median	STD_DEV
GA	5744	6956	6233.1	6200	445.89
PSO	5261.3	6654.9	6079.7	6360.3	627.56
DEopt	4271.4	7652.5	6131.4	6216.6	1221.5
FA	4851.4	6345.7	5520.7	5490.6	550.03
LA	4512.6	5310.9	4718.2	4578	337.85
ABC	5271.8	6328.9	5719.4	5615	444.69
GSO	5089.1	6112.7	5488.5	5336.4	425.89
BFO	6318.6	9291.9	7814	7335.7	1262.7
GSA	5735.2	7600.7	6613.1	6719.3	832.67
ALO	6241.1	20970	9683	7151.1	6333.4

Table 3. *Cost Statistical report for Case 3*

3	Best	Worst	Mean	Median	STD_DEV
GA	6844.9	8231.6	7485.3	7520.8	536.58
PSO	6423.8	8184.4	7133.9	7200.5	678.4
DEopt	6069.6	7189.4	6678.5	6682.1	401.12
FA	5373.7	6610.7	5923.2	5859.1	472.18
LA	5405	6285.7	5746.9	5610.7	357.24
ABC	6041.1	7228.2	6736.9	6773.3	451.5
GSO	5408.6	6439.9	6055.3	6066.5	404.67
BFO	7047.8	9147.4	7927.7	8068.6	848.69
GSA	7472.4	8272.9	7910	7891.3	362.76
ALO	7733.6	20223	12464	11979	4978.2

Table 4. *Cost Statistical report for Case 4*

4	Best	Worst	Mean	Median	STD_DEV
GA	6609.5	9762	8459.5	8621.2	1204.2
PSO	6654.8	8029.5	7334.3	7329.4	605.31
DEopt	6652.3	10035	7805.3	7323.6	1319.2
FA	6248.4	7062.6	6672.8	6763.7	316.77
LA	7017	7814.5	7548.9	7670.6	310.3
ABC	7017	7814.5	7548.9	7670.6	310.3
GSO	7095.1	7698.9	7283	7214.5	245.34
BFO	8676.4	10170	9324.8	9114.3	638.23
GSA	7485.6	8701.5	8037.3	8016.3	513.67
ALO	7823.8	1.5412e+07	5.8594e+06	28027	8.0222e+06
CS	6236.5	6884.9	6513.6	6470.3	273.63

Table 5. *Cost Statistical report for Case 5*

5	Best	Worst	Mean	Median	STD_DEV
GA	7161.6	8282.9	7837.6	7924.5	448.83
PSO	8095.8	8767.3	8372.9	8249	313.73
DEopt	7588.8	9618.5	8277.9	7914.5	837.77
FA	6222.2	8359.6	7252.6	7258.4	810.44
LA	5470	6927	6248	6383.8	535.99
ABC	7205.2	8357.8	7784.7	7644.4	454.44
GSO	5660.5	7389.6	6813.6	7074.6	675.12
BFO	8376.5	11673	9646	9101.3	1352
GSA	7799.6	8405.1	8082.1	8046.6	217.32
ALO	8800	24559	13871	9047.5	7184.8
CS	6642.1	7414.5	6988.3	6844.2	327.24

Table 6. *Cost Statistical report for Case 6*

GA	9036.5	10075	9672.6	9654	431.62
PSO	7826	9584.3	8444.3	8247.3	675.4
DEopt	7622.3	10372	9207	9637.7	1109.6
FA	6631.7	8935.1	8112.8	8228.4	913.72
LA	6829.5	8042.9	7327	7160.8	494.31
ABC	7878.3	9038.1	8466.7	8636.7	459.13
GSO	6916.2	8014.3	7543.3	7541.8	427.03
BFO	8736.7	11952	10055	9786.8	1250
GSA	7972	8999.1	8655.6	8802.9	413.94
ALO	8216.9	13944	10217	9594	2177.6
CS	6458.6	8154.3	7728.2	7972.5	716.91
GA	9036.5	10075	9672.6	9654	431.62

Table 7. *Cost Statistical report for Case 7*

7	Best	Worst	Mean	Median	STD_DEV
GA	8208.9	9820.6	8929	8780.7	659.91
PSO	7025.6	4915	7416.3	7219.5	300.56
DEopt	7499.2	10197	8902.8	9282.1	1103.7
FA	7419.7	9105.3	8407.6	8408.9	635.41
LA	6948.4	7340.1	7161.2	7179.6	149.64
ABC	8766.4	9501.1	9057.7	8931.2	308.85
GSO	7140.1	8015.4	7492.4	7329.1	365.93
BFO	8559	12226	10309	10471	1382.1
GSA	7427.9	9096.5	8359	8585.1	656.24
ALO	8211.3	5.2017e+05	1.8235e+05	24043	2.3703e+05
CS	7166.6	8103.1	7586.4	7582.5	420.14

Table 8. *Cost Statistical report for Case 8*

8	Best	Worst	Mean	Median	STD_DEV
GA	7700.9	9594.5	8777	8624.8	774.43
PSO	6900.2	7111.5	7502.6	7250.9	250.36
DEopt	8181.2	9390.5	8701.7	8672.2	436.78
FA	7797.7	9404	8356.5	8150.6	648.51
LA	6696.9	7360.3	6962.8	6943.7	266.38
ABC	8202.4	9219.1	8700.2	8809.3	407.14
GSO	7923.7	8910.5	8385.8	8337.7	469.56
BFO	8514.9	10724	9327.2	8960.2	940.94
GSA	7595.2	8187.7	8045.3	8160.2	254.03
ALO	10873	1.4368e+06	4.8483e+05	17920	6.6702e+05
CS	6986.7	8394.3	7607.2	7264.7	725.93

Table 9. *Cost Statistical report for Case 9*

9	Best	Worst	Mean	Median	STD_DEV
GA	8415.6	10164	9156.9	9197.4	756.47
PSO	8882.5	10365	9570.3	9656.1	555.26
DEopt	8095.7	10284	9198.9	9048.7	873.76
FA	8567.3	11088	9295	8927.8	1025.8
LA	7252	8471.8	7854.3	7974.4	459.25
ABC	8949.3	9982	9418.8	9421.6	381.61
GSO	7682.1	9392.7	8514.1	8388.1	673.52
BFO	9298.7	11615	10055	9729.8	906.07
GSA	8283	9794.6	9080.2	9352.2	618.92
ALO	10006	23312	13139	10499	5726
CS	7796.3	9237	8402.2	8344.3	547.48

Table 10. *Cost Statistical report for Case 10*

10	Best	Worst	Mean	Median	STD_DEV
GA	8957.2	9535.3	9269.8	9233.4	252.88
PSO	8486.5	10276	9348.4	9231	759.16
DEopt	7938.3	10642	9205.8	9242.6	1147.6
FA	8213.8	9011.7	8689	8764.8	295.2
LA	7152.6	10119	7879.5	7311.5	1266.2
ABC	8756.1	9570.5	9110.2	9073.9	303.09
GSO	7751	11113	8686.1	7966.7	1425.2
BFO	10061	11694	10895	11054	621.89
GSA	8571.1	9487.2	9039.8	9073.3	327.64
ALO	8968.9	21252	14731	15662	4930
CS	7966.9	9109.6	8393.9	8244.8	451.12

8. Conclusion

In this paper, job scheduling problem of ten cases was solved by eleven algorithms namely, GA, PSO, DEopt, FA, LA, ABC, GSO, BFO, GSA, ALO and CS. Then the performance of the proposed LA and the ten algorithms are monitored and analyzed based on its cost function. The final results obtained from the convergence analysis and the cost statistical analysis shows that LA provides better convergence property than all the ten algorithms and it possess better mean value in each cases, i.e., 99.93% better than PSO, 0.21% better than CS, 0.81% better than CS, 13.71% better than CS, 8.30% better than GSO, 2.86% better than GSO, 3.43% better than PSO, 7.18% better than PSO, 6.52% better than CS respectively.

References

[1] Zhongjin Lia, Jidong Ge, Hongji Yang, Liguo Huang, Haiyang Hu, Hao Hu and Bin Luo, “A Security and Cost Aware Scheduling Algorithm for Heterogeneous Tasks of Scientific Workflow in Clouds,” Future Generation computer systems, January 2016.

[2] Xiaomin Zhu, Chao Chen, Laurence T. Yang and Yang Xiang, "ANGEL: Agent-Based Scheduling for Real-Time Tasks in Virtualized Clouds," IEEE Transactions on Computers, vol. 64, no. 12, pp. 3389-3403, 2015.

[3] He Hua, Xu Guangquan, Pang Shanchen and Zhao Zenghua, "AMTS: Adaptive Multi-Objective Task Scheduling Strategy in Cloud Computing," Strategies and Schemes, pp. 162-171, April 2016.

[4] Ehab Nabiel Alkhanak, Sai Peck Lee and Saif Ur Rehman Khan, "Cost-aware challenges for workflow scheduling approaches in cloud computing environments: taxonomy and opportunities," Future Generation Computer Systems, vol. 50, pp. 3-21, September 2015.

[5] Chunling Cheng, Jun Li and Ying Wang, "An Energy-Saving Task Scheduling Strategy Based on Vacation Queuing Theory in Cloud Computing," TSINGHUA science and technology, vol. 20, no.1, pp. 28-39, February 2015.

[6] Lipsa Tripathy and Rasmi Ranjan Patra, "Scheduling in cloud computing," International Journal on Cloud Computing: Services and Architecture (IJCCSA), vol. 4, no. 5, pp. 21-27, October 2014.

[7] Ayed Salman, Imtiaz Ahmad and Sabah Al-Madani, "Particle swarm optimization for task assignment problem," Microprocessors and Microsystems, vol. 26, pp. 363-371, 2002.

[8] Saima Gulzar Ahmad, Chee Sun Liew, Ehsan Ullah Munir, Tan Fong Ang and Samee U. Khan, " A Hybrid Genetic Algorithm for Optimization of Scheduling Workflow Applications in Heterogeneous Computing Systems," Journal of Parellel and Distributed Computing, vol. 87, pp. 80-90, January 2016.

[9] Zhongjin Lia, Jidong Ge, Hongji Yang, Liguo Huang, Haiyang Hu, Hao Hu and Bin Luo, “A Security and Cost Aware Scheduling Algorithm for Heterogeneous Tasks of Scientific Workflow in Clouds,” Future Generation computer systems, January 2016.L.F. Zeng, B. Veeravalli, X.R. Li, SABA: a security-aware and budget-aware workflow scheduling strategy in clouds, Journal of Parallel and Distributed Computing 75 (2015) 141-151, 2015.

[10] V. Chang, “The business intelligence as a service in the cloud,” Future Generation Computer Systems, vol. 37, pp. 512-534, 2014.

[11] V. Chang, “Towards a big data system disaster recovery in a private cloud,” Ad Hoc Networks, vol. 35, pp. 65-82, 2015.

[12] V. Chang, R.J. Walters and G.B. Wills, “Organisational sustainability modelling-an emerging service and analytics model for evaluating cloud computing adoption with two case studies,” International Journal of Information Management, vol. 36, no. 1, pp. 167-179, 2016.

[13] V. Chang, Y.H. Kuo and M. Ramachandran, “Cloud computing adoption framework: A security framework for business clouds,” Future Generation Computer Systems, vol. 57, pp. 24-41, 2016.

[14] W. Yurcik, X. Meng, G. Koenig and J. Greenseid, “Cluster security as a unique problem with emergent properties,” Fifth LCI International Conference on Linux Clusters: The HPC Revolution 2004, May 2004.

[15] V. Chang, Y.H. Kuo, M. Ramachandran, Cloud computing adoption framework: A security framework for business clouds, Future Generation Computer Systems 57 (2016) 24-41.
[16] Upendra Bhoi and P.N. Ramanuj, "Enhanced Max-min Task Scheduling Algorithm in Cloud Computing," International Journal of Application or Innovation in Engineering and Management, vol. 4, no. 2, pp. 259-264, 2013.
[17] Ehsan ullah Munir, Jian Zhong li, and S. Shi, "QOS sufferage Heuristic for Independent Task Scheduling in Grid," Journal of Information Technology, vol. 6, no. 8, pp. 1166-1170, 2007
[18] L. Wu, Y.J. Wang, and C.K. Yan, "Performance comparison of energy aware task scheduling with GA and CRO algorithms in cloud environment," Applied Mechanics and Materials, pp. 204-208, 2014.
[19] C. Zhao.,et al, "Independent Tasks Scheduling Based on Genetic Algorithm in Cloud Computing," 5th International Conference on Wireless Communications, Networking and Mobile Computing, pp. 5548-5551, 2009.
[20] Fei Taoa and L.Z. Ying Fengb and T.W. Liaoc, "CLPS-GA: A case library and Pareto solution-based hybrid genetic algorithm for energy aware cloud service scheduling," Applied Soft Computing, vol. 19, pp. 264 279, 2014.
[21] Zhu, Y. and P. Liu, "Multi-dimensional constrained cloud computing task scheduling mechanism based on genetic algorithm," International Journal of Online Engineering, vol. 9(SPECIALISSUE.6): pp. 15-18, 2013.
[22] Meihong Wang and Wenhua Zeng, "A comparison of four popular heuristics for task scheduling problem in computational grid," The 6th International Conference on Wireless Communications Networking and Mobile Computing, pp. 1-4, 2010.
[23] Zhanghui Liu and Xiaoli Wang, "A PSO-based algorithm for load balancing in virtual machines of cloud computing environment," Advances in Swarm Intelligence, vol. 7331, pp. 142-147, 2012.
[24] Juan Wang, Fei Li and Luqiao Zhang, "Apply PSO into cloud storage task scheduling with QoS preference awareness," Tongxin Xuebao/Journal on Communications, vol. 35, no. 3, pp. 231-238, 2014.
[25] L.Z. Guo, Y.J. Wang, S.G. Zhao, C.Y. Jiang, "Particle swarm optimization embedded in variable neighborhood search for task scheduling in cloud computing," Journal of Donghua University (English Edition), vol. 30, no. 2 pp. 145-152, 2013.
[26] Xue, S., et al, "An ACO-LB algorithm for task scheduling in the cloud environment," Journal of Software, vol. 9, no. 2, pp. 466-473, 2014.
[27] Xue, S., J. Zhang, and X. Xu, "An improved algorithm based on ACO for cloud service PDTs scheduling," Advances in Information Sciences and Service Sciences, vol. 18, no. 4, pp. 340-348, 2012.
[28] Tong, Z., et al, "H2ACO: An optimization approach to scheduling tasks with availability constraint in heterogeneous systems.," Journal of Internet Technology, vol. 15, no. 1, pp. 115-124, 2014.
[29] Sun, W., et al, "PACO: A period ACO based scheduling algorithm in cloud computing," Proceedings of International Conference on Cloud Computing and Big Data, CLOUDCOM-ASIA 2013.
[30] B.R Rajakumar, "Lion Algorithm for Standard and Large Scale Bilinear System Identification: A Global Optimization based on Lion's Social Behavior," IEEE Congress on Evolutionary Computation (CEC), pp. 2116-2123, July 2014.
[31] Hongbo Liu, Ajith Abraham, Vaclav Snasel and Sean McLoone, "Swarm scheduling approaches for work-flow applications with security constraints in distributed data-intensive computing environments," Information Sciences, vol. 192, pp. 228-243. June 2012.
[32] Mohammad Masdari, Sima ValiKardan, Zahra Shahi and Sonay Imani Azar, "Towards workflow scheduling in cloud computing : A comprehensive analysis," Journal of Network and Computer Applications, vol. 66, pp. 64-82, May 2016.