I.J. Intelligent Systems and Applications, **2016, 11, 61-69**
Published Online November 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2016.11.07

Hybrid Algorithm Based on Swarm Intelligence Techniques for Dynamic Tasks Scheduling in Cloud Computing

Medhat A. Tawfeek
Faculty of Computers and Information's, Menofia University / Computer Science Department, Menofia, Egypt
E-mail: medhattaw@yahoo.com

Gamal F. Elhady
Faculty of Computers and Information's, Menofia University / Computer Science Department, Menofia, Egypt
E-mail: gamal.farouk@ci.menofia.edu.eg

***Abstract*—**Cloud computing has its characteristics along with some important issues that should be handled to improve the performance and increase the efficiency of the cloud platform. These issues are related to resources management, fault tolerance, and security. The purpose of this research is to handle the resource management problem, which is to allocate and schedule virtual machines of cloud computing in a way that help providers to reduce makespan time of tasks. In this paper, a hybrid algorithm for dynamic tasks scheduling over cloud's virtual machines is introduced. This hybrid algorithm merges the behaviors of three effective techniques from the swarm intelligence techniques that are used to find a near optimal solution to difficult combinatorial problems. It exploits the advantages of ant colony behavior, the behavior of particle swarm and honeybee foraging behavior. Experimental results reinforce the strength of the proposed hybrid algorithm. They also prove that the proposed hybrid algorithm is the best and outperformed ant colony optimization, particle swarm optimization, artificial bee colony and other known algorithms.

***Index Terms*—**Cloud Computing, Task Scheduling, Ant Colony Optimization, Particle Swarm Optimization, Artificial Bee Colony, Makespan; CloudSim.

I. INTRODUCTION

Cloud computing is catching more attention because it is the only one of its kind, and it has many unique merits that can be utilized to ease services execution. Scalability of cloud resources lets a flexible provisioning of resources and supplies on demand computing infrastructure for applications [1]. The propagation of cloud as a general-purpose computing wakes up awareness of the requirement for versatile management methods. So, the success of cloud services is based on the power of cloud management algorithms [2]. On one hand, cloud computing allows users to access services that remain in a remote data centers, other than local computers. Data-centers are the main computing infrastructures that supply many kinds of services via scaling capacity. The cloud provider accumulates a large number of hosts or servers in a data center where each host may run one or multiple virtual machines (VMs). On the other hand, cloud providers shall present easy and fast application deployment to cloud users and improve resources utilization [3]. One of the main technologies that let cloud computing to be possible is the virtualization. Virtualization technology has simplified the hard resource consolidation. Cloud providers can earn the benefits of consolidation regarding reduced management costs and allow multiple users to share computing, storage and networking infrastructure provided by the service provider. So the use of virtualization in the cloud is essential because the servers can be sliced up for users as virtual cloud instances in the form of individual VMs. The VMs may include processors running at different speeds, memory and storage that deal with various storage systems at different locations. Moreover, applications can be carried out independently without needing for any particular configuration [4]. In cloud computing, VMs need to be allocated and scheduled in a way that providers can realize high VMs utilization. The right tasks scheduler over VMs shall enforce the scheduling manner to the changing environment and the types of tasks. The user application consists of multiple tasks that need allocation over VMs. The task scheduler handles assigning preferred VMs to the submitted tasks so that the overall VMs are utilized effectively. Such a scheduling decision becomes more uneasy in the cloud because its environment is heterogeneous and frequently mutates [2]. Therefore, cloud scheduling strategies that are based on Swarm Intelligence (SI) techniques, for example, Ant Colony Optimization (ACO), Artificial Fish Swarm (AFS), Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) are preferable [5]. SI is based on the studying of the combined behaviors raised from interactions between individuals and the environment to solve very hard optimization problems. They offer excellent performance and prove its capabilities for handling scheduling problems in cloud computing.

Moreover, they are very flexible to design and implement [4].

In this paper, a new hybrid algorithm is introduced to find the near-optimal VMs allocation for dynamic cloud tasks to minimize the makespan that is the finishing time of the last job or maximum completion time. This hybrid algorithm merges the behaviors of ACO, ABC and PSO techniques. The simulation based experiments using CloudSim in [6], studies the performance of proposed hybrid algorithm compared to ACO in [7], PSO in [8], and ABC in [9]. The results from experiments show that the proposed hybrid algorithm can obtain better VMs utilization and remarkably outperforms the compared methods on the basis of makespan and degree of imbalance. The rest of this paper is organized as follows. Section II covers background that outlines the basics of ACO, PSO and ABC. Section III describes the commonly related work in cloud task scheduling. The cloud task scheduling based on the proposed hybrid algorithm is detailed in section IV. The implementation and simulation results are investigated in section V. The conclusion to this paper is invoked in section VI.

II. Background

Cloud computing can be rated a natural evolution from grid computing by delivering computing resources as services to users remotely [1]. The fundamental features of a cloud computing are scalability to meet user requests, providing multiple service levels and dynamic configuration of services on demand [3]. In cloud computing, it is paramount to schedule tasks on its suitable resources, and the capacities of different VMs need to be taken into account when a user sends a service request. The purpose of scheduling is finding an optimal mapping from a finite set of objects. An easy scheduling problem aims to one with a small number of the objects so that it can be simply worked out by enumerations. On the contrary, a hard scheduling problem if its purpose is optimization needs heuristic and approximation methods. Enumeration is not workable for cloud scheduling problems because only a few cases of these problems have solvable algorithms in polynomial time [10]. The direction is finding near-optimal solutions that are acceptable to achieve accuracy and time. Heuristic is considered a near-optimal algorithm to find good solutions quickly. It iteratively enhances a candidate solution concerning a particular measure of quality but does not guarantee to find the optimal solution [11]. Heuristic or metaheuristic algorithms obtained much popularity because they supply acceptable solutions in a suitable time for solving hard problems in many fields. Many new algorithms from metaheuristics algorithms depend on swarm intelligence (SI) [4]. Examples of techniques in which SI is inspired are bees colonies, ants colonies, fish schools and birds flocks where the whole group of individuals does the desired task that may not be performed individually. Recently, several researchers have proposed algorithms based on ACO, PSO and ABC for scheduling problems in distributed environments such as grids and clouds [4].

A. The Ant Colony Optimization (ACO)

The main idea of ACO is to simulate natural behavior of ant colonies. Fig. 1 presents the pseudo-code of ACO [11]. The algorithm mainly contains two iterated steps: solution construction and pheromone update.

- Solution construction: - the construction of solutions is done according to a probabilistic transition rule that depends on pheromone trails and heuristic information.
- Pheromone update: - the update of the pheromone is performed using the generated solutions. A pheromone updating rule carried out in two phases: evaporation phase where the pheromone trail is lowered automatically and reinforcement phase where a positive value is added [7].

```
Initialize the pheromone trails.
Repeat
    For each ant Do
        Solution construction using the pheromone trail;
        Update the pheromone trails:
            Evaporation;
            Reinforcement;
Until stopping criteria
Output: Best solution found or a set of solutions.
```

Fig.1. Pseudo code of basic ACO

B. Particle Swarm Optimization (PSO)

The PSO algorithm works as a simulation by modifying the position of each particle depending on its velocity using the global best position and the best position of the particle [12]. Over time, the particles go together around right solution. Fig. 2 shows the pseudo code of PSO algorithm [11]. The velocity value is computed due to how far a particle is from the target by (1).

$$V_i(t+1) = V_i(t) + U_1C_1 \times (pb_i - X_i(t)) + U_2C_2 \times (gb - X_i(t)) \tag{1}$$

Where, V_i $(t+1)$ represents the new velocity of a particle and V_i (t) represents its current velocity. U_1 and U_2 are two random variables in the range [0, 1]. The constants $C_1$and C_2 represent the learning factors. The x-vector records the current position of the particle in the search space. Each particle keeps track of its coordinates in the solution space which are associated with the best solution (fitness) that has achieved so far by that particle. This value is called personal or particle best (pb_i). Another best value that is tracked by the PSO is the best value obtained so far by any particle in the neighborhood of that particle. This value is called global best (gb). After updating the velocity of each particle, each particle will moves to the new position in the decision space [8], using (2).

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{2}$$

```
Random initialization of the whole swarm
Repeat
   Evaluate object function f(xi)
   For all particles i
       Update velocities by (1)
      Move to the new position by (2)
      If f (xi) < f (pBesti) Then pBesti = xi
      If f (xi) < f (qBest) Then qBest = xi
   EndFor
Until stopping criteria
Output: Best solution found or a set of solutions.
```

Fig.2. Pseudo code of basic PSO

C. The Artificial Bee Colony(ABC)

The ABC algorithm that is based on the clever foraging behavior of honey bee swarm is an optimization algorithm. Fig. 3 shows the pseudo code for the ABC algorithm [11]. The algorithm begins with scout bees that scan the search space randomly. The quality of visited sites by these bees is then rated. After that, sites that have the highest fitness are selected for a neighborhood search. Then, the algorithm continues searching for the selected sites by assigning more active bees to search in the neighborhood of these sites [9]. The colony will contain after iteration two groups for its new population, the solutions from each selected sites and the scout bees assigned to generate random solutions. This process is repeated until meeting a given stopping criterion [9].

```
Random initialization of the whole colony of bees
Evaluate the fitness of the population of bees
Repeat
     Select sites for neighborhood search
     Determine the patch size
     Recruit bees for selected sites and evaluated their fitness
     Select the representative bee from each patch
    Assign remaining bees to search randomly and evaluate
their fitness
Until stopping criteria
Output: Best solution found or a set of solutions.
```

Fig.3. Pseudo code of basic ABC

III. Related Work

A cloud task scheduling based on ACO algorithm is implemented in [7]. The main objective of this algorithm is minimizing the makespan of a given tasks set. It handles all the tasks requests according to different VMs available in a cloud. Another ACO scheduling was proposed to handling job scheduling within a cloud in [13]. It maximizes the throughput of the heterogeneous computing system. In this algorithm, the modification was done in basic pheromone updating function to give better resource utilization. The algorithms in [7] and [13] depend on the fact that each task is executed with different speed on a different processor. This information is exploited to save information about which processors are suitable for each job. Therefore, the pheromone value is used by the scheduler to determine the desirable assigning of a particular task into a specific Virtual Machine (VM). The modified ant colony optimization algorithm (MACO) is proposed in [14]. The objective of this modification is to enhance the performance of the basic ant colony optimization algorithm and enhance the execution time of the tasks. This approach introduces self-adapting criteria for control parameters of the basic ant colony optimization. The Max–Min Ant System (MMAS) in [15] was proposed to control the pheromone amount. In this method, Local Search (LS) technique has been implemented to select the swap that reduces makespan. Load balancing of nodes using ACO proposed in [16] is used for achieving load balancing. In this algorithm, an ant can move in two directions: forward and backward. Such as an ant searches for the food is called forward direction and return to the nest is the backward direction. This behavior is helpful for balanced the node quickly. The results of this algorithm provide better utilization of resources but consume more power. Another disadvantage is that it has a high network overhead. Cloud task scheduling based on Load balancing Ant Colony Optimization (LBACO) which is used to find the best VMs allocation for each task dynamically was proposed in [17]. It works on minimization of makespan of tasks that is distributed among VMs. The MACOLB algorithm that is the MACO for load balancing has been proposed in [18]. The main goal of MACOLB is to balance the load and to try to minimize the makespan of a given tasks set. The load balancing factor in MACOLB is related to the task finishing rate. It is proposed to make the finishing rate of VMs being similar, and the ability of the load balancing will be increased. The PSO for tasks scheduling in the cloud has been proposed in [8]. It simulates the behavior of particle swarm. This algorithm is used to find the near-optimal VMs allocation for tasks in the dynamic cloud system to minimize the makespan of tasks. A PSO to schedule jobs in a cloud that considers both computations of job costs and job data transfer costs has been proposed in [19]. It dynamically enhances the main cost of a job-resource-mapping.

Cloud task scheduling based on artificial bee colony algorithm has been proposed in [9]. It simulates the behavior of foraging bees to the cloud scheduling problem. It tracks the overall best solution with high quality related to makespan by any of the bees. The high quality of solution means the small time of solution makespan and low quality means large solution makespan. Honey Bee Behavior inspired Load Balancing (HBB-LB) has been proposed in [20]. The main goal of HBB-LB is to achieve well-balanced load across VMs and minimize the makespan. The VMs are grouped depending their loads in three sets overloaded VMs, under loaded VMs and balanced VMs. Each set contains the number of VMs. HBB-LB removes jobs from an overloaded VMs and makes the decision to place the removed jobs in one of the under loaded VMs. A job works as a honey bee and the VMs with low load are considered as the food sources for honey bees. A Join-Idle-Queue in [21] and Join-shortest-queue (JSQ) in [22] are proposed for dynamically scalable web services. These algorithms apply load balancing with distributed dispatchers by assigning jobs to idle processors firstly to reduce average queue length at each processor. Stochastic hill climbing that is used for allocation of incoming jobs to cloud VMs is proposed in [23]. This algorithm is simply a loop that continuously moves in the direction of increasing value, which is uphill. It stops when it reaches a "peak" where no neighbor has a higher value.

IV. Cloud Scheduling based the Proposed Hybrid Algorithm

The pseudo code of the proposed hybrid procedure is shown in Fig. 4. The hybrid algorithm paves the way for finding the near-optimal resource allocation for dynamic tasks in the cloud to minimize the makespan of tasks. It manipulates the overall best-founded solution by any member (ant, bee or particle) at any iteration. In an initialization phase of the proposed ABC algorithm, the parameters are initialized. *Number_of_BeesAntsParticles* variable represents all the members that are the total number of bees, particles, and ants. Each bee and each particle generate a random solution. The solution is represented as an array of VM's IDs that represents the order of VMs. The first task will be allocated to the first ID in the matrix of VM's IDs; the second task will be allocated to the second ID and so on. Then each *CommonBoard* generates a random solution. The hybrid algorithm uses four *CommonBoard*.

```
Input: List of Cloudlet (Tasks) and List of VMs
Output: the best solution for tasks allocation on VMs

Steps:
 1. Initialize:
    Set value of parameters Number_of_BeesAntsParticles, Number_of_Bees, Number_of_Ants, Number_of_Particles,
    Number_of_CommonBoard, Max_Number_of_Stagnation, tmax.
    Set V_Max.
    Set an initial value $\tau ij(t)=c$ for each path between tasks and VMs.
    Set t=1.
    Set BSolution=null.
 2. Generate Random Solution for each Bee.
 3. Generate Random Solution for each Particle.
 4. Generate Random Solution for each CommonBoard.
 5. Check to update BSolution
 6. For k :=1 to Number_of_BeesAntsParticles
    IF k is a bee
    Perform Bees()
    ElseIF k is an ant
    Perform Ants()
    Else
    Perform Particles()
    End IF
 7. Apply global pheromone update.
 8. Increment t by one.
 9. If (t < tmax)
       Goto step 4
       Else
       Print Bsolution.
       End If
 10. Stop
```

Fig 4. Pseudo code of hybrid procedure.

The first CommonBoard registers and shares the best-founded solution by any bees with other members like ants and particles. The second CommonBoard registers and shares the best-founded solution by any ants with other members like bees and particles. The third CommonBoard registers and shares the best-founded solution by any particles with other members like bees and ants. The fourth CommonBoard registers and shares a randomly generated solution with all members like bees, particles, and ants. These four CommonBoards arms the hybrid algorithm by sharing all the possible best-founded solutions from any members. The iterative phase simulates the technique of hybrid procedure. The members are iterated using for loop and each member type is handled by the suitable module from Bees(), Ants() and Particles() modules. The *BSolution* variable is checked iteratively to hold the overall best-founded solution.

A. The Bee() module

The Bees() module is presented in Fig. 5. In this module, the bee firstly gains a neighbor solution proportional to its current solution. Logically, each solution has some neighbor. The natural neighbor solution relative to a current solution is a replacement of the current solution where two, three or some adjacent VM's IDs have been swapped. If the neighbor solution is better than the current solution, this bee will accept the better neighbor. After that, the *BSolution* will be checked and a MentionAdvert() module is called. *Max_Number_of_Stagnation* is a threshold value used to prevent the bee from stagnation solution. If so, the current bee's selects another area for searching by selecting one solution from 2-th, 3-th, or 4-th *CommonBoard* randomly.

```
Bees()
 1. Generate a neighbor Solution.
 2. If (a neighbor solution quality > current bee solution quality)
      bee accepts the better neighbor solution
      Reset Number_of_Stagnation to zero
      Check to update BSolution
      Perform MentionAdvert ()
        Else
           Increase Number_of_Stagnation by one
      End If
 3. If (Number_of_Stagnation >Max_Number_of_Stagnation )
      Select solution from 2-th,3-th, or 4-th CommonBoard randomly.
      Reset Number_of_Stagnation to zero.
      End If
 4. Return
```

Fig.5. Pseudo code of Bees()

B. The MentionAdvert()

The MentionAdvert() module is presented in Fig. 6. This module simulates the action of how the members of hybrid algorithm contact or share the solutions with each other's. The mentioned member's solution is compared against the solution of its commonboard. If the solution of mentioned member is preferable, the commonboard will modify its solution by accepting the mentioned solution.

```
MentionAdvert()
 1. If (the mentioned item is Bee)
     If (a mentioned solution quality > 1-th CommonBoard solution quality)
         1-th CommonBoard accepts the solution of mentioned item
     End if
    Else If (mentioned item is Ant)
     If (a mentioned solution quality > 2-th CommonBoard solution quality)
         2-th CommonBoard accepts the solution of mentioned item
     End if
    Else
     If (a mentioned solution quality > 3-th CommonBoard solution quality)
         3-th CommonBoard accepts the solution of mentioned item
     End if
    End If
 2. Generate Random Solution for 4-th CommonBoard.
Return
```

Fig.6. Pseudo code of MentionAdvert()

C. The Ants()

The Ants() module is presented in Fig. 7. The ant selects the starting VM for the first task randomly.

```
Ants()
 1. Place this Ant on the starting VM randomly.
 2. Do ant_trip while ScoutAnt does not end its trips
        Ant chooses the VM for the next task according to (3).
 End Do
 3. Check to update the Bsolution.
 4. Perform MentionAdvert()
 5. Apply local pheromone update.
Return
```

Fig.7. Pseudo code of *Ants()*

After that, it constructs a solution by moving from one VM to another until completing a solution. The ant chooses VM_j for next $task_i$ by probabilistic transition rule that is computed by (3).

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\times[\eta_{ij}]^{\beta}}{\sum_{s\in allowed_k}[\tau_{is}(t)]^{\alpha}\times[\eta_{is}]^{\beta}} & if\ j\in allowed_k\\ 0, & otherwise\end{cases}\quad(3)$$

Where, $\tau_{ij}(t)$ shows the pheromone concentration on the path between $task_i$ and VM_j. The symbol $allowed_k$ express the allowed VMs for ant_k. The term η_{ij} represents the visibility or heuristic information that represents the expected execution time for $task_i$ on VM_j. Finally, the two parameters α and β control the weight of the pheromone and the visibility information respectively. After the completion of a tour of ant, the *BSolution* will be checked and the MentionAdvert() module is called. The ant also lays a small quantity of pheromone on each edge (i,j) that this ant traveled through. This process is called local pheromone update.

D. The Particles()

The Particles() module is presented in Fig. 8. This module demonstrates the manner of particles to solve a problem.

```
Particles()
 1. Calculate Particle Velocity according to Eq. (4).
 2. Use Velocity to update Particle Data
 3. If (obtained solution quality > current solution quality)
    Particle accepts the obtained solution
    Reset Number_of_Stagnation to zero
    Check to update BSolution
    Perform MentionAdvert()
    Else
    Increase Number_of_Stagnation by one
    End If
 4. If (Number_of_Stagnation > Max_Number_of_Stagnation)
    Select solution from 1-th, 3-th, or 4-th CommonBoard randomly
    Reset Number_of_Stagnation to zero
    End If
Return
```

Fig 8. Pseudo code of Particles()

The velocity is computed considering the *BSolution*. This form defines velocity as the measure of how the current particle is far from *BSolution*. The velocity of each particle in PSO is computed by (4).

$$V=\frac{V\_Max\times BSolution}{pBest}\quad(4)$$

Where, V is the computed velocity of the particle, and *pBest* variable represents the best fitness value of particle's solution. It means that the particles far from the *BSolution* would carry out an effort to follow with the other particles by flying faster toward the *BSolution*. Once the velocity of the particle has been obtained, the modifying solution is done by swapping some of VM's IDs. Particles are pushed towards the *BSolution* by copying pieces from the *BSolution* solution.

V. Implementation and Experimental Results

The Cloud computing has become the fast spread in the field of computing, research and industry in the last few years. Task scheduling is the most significant matter in the cloud computing. In this paper, cloud task scheduling depends on a hybrid approach has been proposed for allocation the incoming tasks to the available VMs considering into account the makespan to minimize the VMs consumption and achieve user satisfaction. In this section present the evaluation and experimental results.

A. Parameters Setting of Cloud Simulator

Table 1. Parameters setting of cloudSIM

Entity Type	Parameters	Value
Tasks (cloudlets)	Length of task	1000-50000
	Total number of tasks	100-1000
Virtual Machines	Total number of VMs	25-50
	MIPS	500-2000
	VM memory(RAM)	128-2048
	Bandwidth	500-1000
	Number of PEs requirement	1-4
Datacenters	Number of Datacenters	5
	Number of Hosts	10

Simulation is a technique that simulates the behavior of a specific system by actually playing back observations from this system. The researcher has used CloudSim for implementing the experiments in a simulated cloud environment because CloudSim can be used to simulate data centers, host, service brokers, scheduling and allocation policies of a large scaled cloud platform [6]. The parameters setting of cloud simulator is depicted in Table 1.

B. Parameters evaluation and setting of hybrid algorithm

The control parameters of hybrid algorithm (*Number_of_BeesAntsParticles, Number_of_Bees, Number_of_Ants, Number_of_Particles, Number_of_CommonBoard, Max_Number_of_Stagnation, V_MAX, α, β,* t_{max}) are sensitive and must be fine-tuned.

Several values for each parameter were tested while all the others were held constant. Table 2 shows the suitable values settings of the proposed hybrid algorithm parameters that are experimentally determined. The parameter settings of the proposed hybrid algorithm were determined to be *Number_of_BeesAntsParticles=50, Number_of_Bees=25, Number_of_Ants=5, Number_of_Particles=20, Number_of_CommonBoard=4, Max_Number_of_Stagnation=10, V_MAX=5, α=0.2, β=0.8,* t_{max}*=100.*

Table 2. Selected Parameters of Hybrid Algorithm

Parameter	Vsalue
Number_of_BeesAntsParticles	50
Number_of_Bees	25
Number_of_Ants	5
Number_of_Particles	20
Number_of_CommonBoard	4
Max_Number_of_Stagnation	10
V_MAX	5
α	0.2
β	0.8
t_{max}	100

C. Implementation results of Hybrid, ABC, PSO and ACO

The cloud task scheduling algorithms to be compared in the experiments of this subsection include ABC algorithm in [9], PSO in [8], ACO in [7] and the proposed hybrid algorithm. The parameter settings of ABC algorithm are as follows. *Number_of_Bees*=100, *Number_of_*active=75, *Number_of_Scout*=15, *Number_of_Inactive*=10, *Max_number_of_Vists*=70, *Prob_Mistake*= 0.1, *Prob_Presuasion*=0.90 and t_{max}=100 as in [9]. The parameter settings of PSO algorithm are as follows. *Number_of_particles*=100, *V_MAX*=8 and t_{max}=100 as in [8]. The parameter settings of ACO algorithm are as follows. m(number of ants) = 10, t_{max}= 100, α = 0.3, β= 1, Q(adaptive parameter)=100 and ρ = 0.4 as in [7]. In the following experiments, the makespan of each algorithm with different tasks set is computed. The average makespan of the hybrid algorithm, ABC, PSO, and ACO are shown in Fig. 9. It can be seen from this figure, when the quantity of tasks is increased, the hybrid algorithm takes less time than ABC, PSO, and ACO algorithms. This indicates that the proposed algorithms take less time to execute than other methods because the proposed hybrid algorithm has intelligently different concepts for exploring the search space. Beside that strategies cooperation from ABC, PSO and ACO are used to share and accumulate information to find efficiently good solutions.

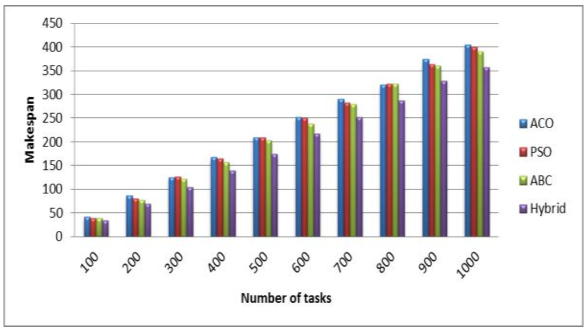

Fig.9. Average makespan of ACO, PSO, ABC and Hybrid

The explanation of how the proposed hybrid algorithms outperform ABC, PSO and ACO is as follows. The hybrid algorithms model the behavior of honey bees, swarm fly and ant colony in a mixed algorithm to solve the cloud task scheduling problem. It tracks the overall best solution that is associated with the makespan length, founded by any member. At the same time, it saves the best solution founded by a similar group of members and shares them with other two groups by feeding information in an external *CommonBoards* and simultaneously incorporates continuous updating of pheromone. There are different activities for the groups of similar members. There is a group of bees that simulate the skillful foraging behavior of honey bee swarm. There is a group of particles that simulate the efficient foraging behavior of swarm fly over an environment following the best members of the swarm and directing their movement toward right areas from their environment. There is a group of ants that simulate the effective foraging behavior of ants that try to search for the abundant food sources. The bees continue aggregating neighbor solutions from a particular area until this field is consumed. After that, they check the *CommonBoards* for selecting another abundant area. The particles advance the position of each particle successively based on its velocity using the global best-known solution and the best solution known to a particle. The ants exploit a particular kind of chemical pheromone to communicate with each other and to contact with bees and particles. They go ahead to construct the solution by sensing pheromone on the allowed paths. After any ant completes its tour, it lays a quantity of pheromone called local pheromone updating. The global pheromone updating reinforces pheromone on the edges belonging to the best-founded solution by any member (ants, bees or particles). These are the reasons why hybrid algorithm outperforms other algorithms. The degree of imbalance (DoI) measures the imbalance among VMs. The small value of DoI tells that the load of the system is more balanced and efficient [17]. Three different methods can measure DoI. The first method measures the difference between the maximum and minimum completion time of VMs that is defined as in (5).

$$DI_1 = AT_{\max} - AT_{\min} \tag{5}$$

Where, AT_{max} and AT_{min} represent the simulated maximum and minimum completion time of VMs [18]. The DI_1 of the hybrid algorithm, ABC, PSO and ACO is shown in Fig. 10.

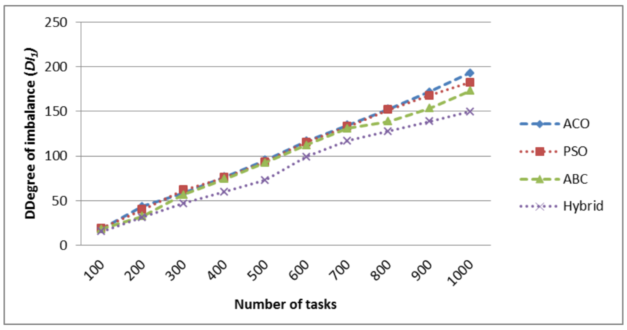

Fig.10. DI_1 of ACO, PSO, ABC, and Hybrid

The second method measures the degree of imbalance, as in (6).

$$DI_2 = \frac{T_{\max} - T_{\min}}{T_{avg}} \tag{6}$$

Where, T_{max}, T_{min} and T_{avg} are the maximum, minimum and average completion time of VMs respectively [9]. The DI_2 of the hybrid algorithm, ABC, PSO and ACO is shown in Fig. 11.

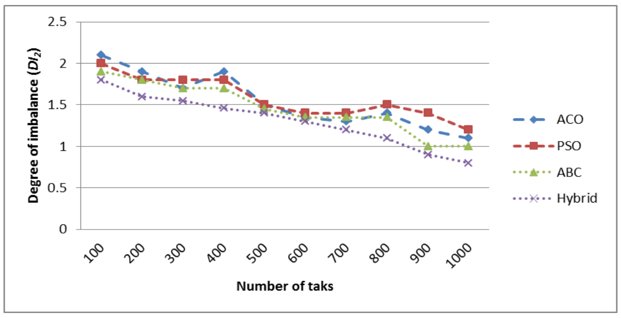

Fig.11. DI_2 of ACO, PSO, ABC and Hybrid

The third method that measures the degree of imbalance using standard deviation is given by (7).

$$DI_3 = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(x_j - \bar{x})^2} \quad for\ all\ j \in VM\ list \tag{7}$$

Where, DI_3 is the standard deviation, N is the number of VMs, x_j is the completion time of VM_j and $\bar{x}$ is the average completion time of all VMs. If the value of DI_3 is small, it means that the differences of loads are small and the load on VMs is more balanced [8].

The DI_3 of the hybrid algorithm, ABC, PSO and ACO is shown in Fig. 12.

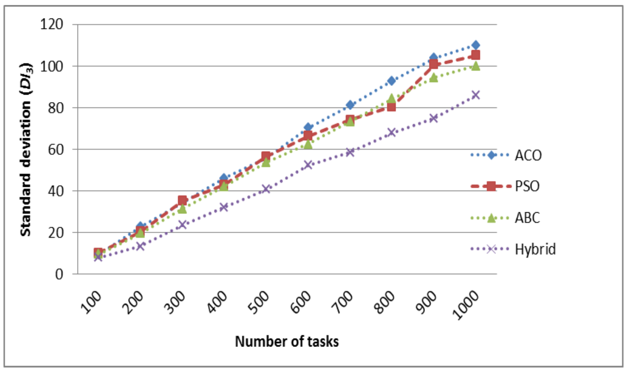

Fig.12. DI_3 of ACO, PSO, ABC and Hybrid

It can be seen from Fig. 10 – Fig. 12 that the hybrid can achieve better load balancing than ACO, PSO and ABC algorithms. The VMs in a data center have a different amount of processing powers. The proposed hybrid algorithm searches for a solution that assign tasks firstly to the most powerful VM and then to the lowest that trying to balance the load of the VMs.

D. Implementation results of Hybrid, HBB-LB, LBACO, Hill Climbing and JSQ

In the following experiments, the average makespan and DoI of proposed hybrid, HBB-LB in [20], LBACO in [17], Hill Climbing in [23] and the JSQ in [22] algorithms with different tasks set are compared. The average makespan of these algorithms is shown in Fig. 13. It can be seen that, the hybrid algorithm takes less time than other algorithms but with tasks less than 300 and more than 800 tasks the HBB-LB weakly outperforms the hybrid. The HBB-LB algorithm has an added cost for network overhead by the high number of migrated tasks. This indicates that the hybrid algorithm is more suitable than HBB-LB algorithm.

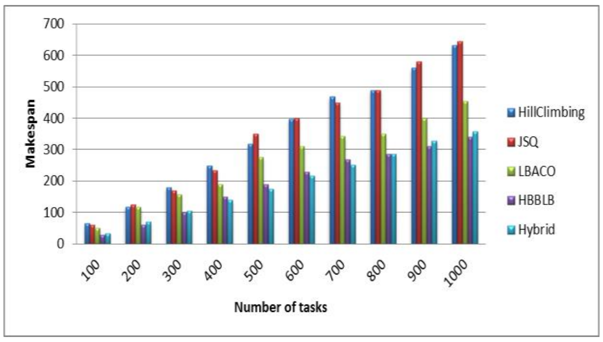

Fig.13. Average makespan of HBB-lB, HillClimbing, LBACO, JSQ and Hybrid

The DI_1, DI_2 and standard deviation of each algorithm are shown in Fig. 14, Fig. 15 and Fig. 16 respectively. It can be seen from these figures that the hybrid can achieve better load balance than hill climbing, LBACO, HBB-LB algorithms.

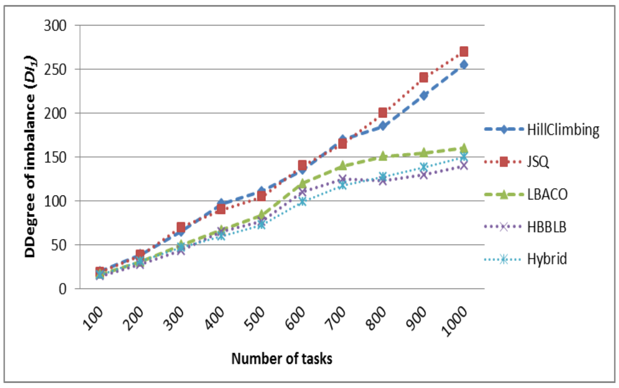

Fig.14. DI_1 of HBB-lB, HillClimbing, LBACO, JSQ and Hybrid

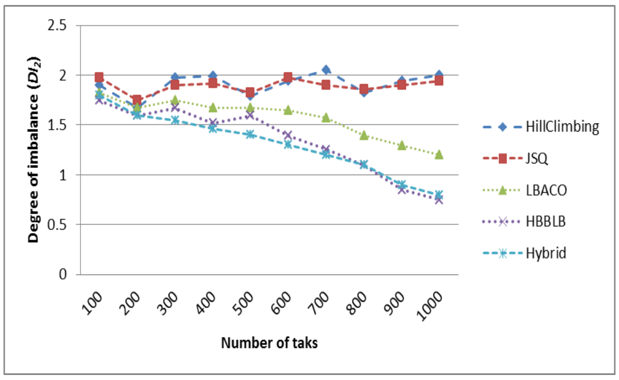

Fig.15. DI_2 of HBB-lB, HillClimbing, LBACO, JSQ and Hybrid

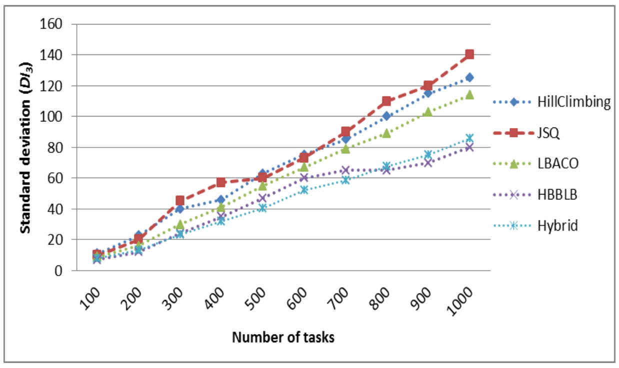

Fig.16. DI_3 of HBB-lB, HillClimbing, LBACO, JSQ and Hybrid

The explanation of how hill climbing, LBACO, HBB-LB, JSQ and hybrid algorithms give the above results, is as follows.

The hybrid algorithm takes less time to execute tasks and achieve better load balance because it depends on the benefits from cooperation between different types of swarm members. These members are searching for a solution that makes the task finishing rate at different resource being similar so the ability of the load balancing will be improved. This is the reason why the hybrid algorithm is preferable one.

In HBB-LB algorithm, VMs are grouped based on their loads in three sets: overloaded VMs, under loaded VMs and balanced VMs. Each set contains the number of VMs. Tasks removed from an overloaded VM have to a make decision to get placed in one of the under loaded VMs. The HBB-LB algorithm removes overloads on server and balances load amongst servers but it has a high network overhead by tasks migration. When the numbers of tasks increase, the effectiveness of the HBB-LB algorithm will appear. This is the reason why the HBB-LB algorithm outperforms the hybrid when the numbers of tasks being more than 800 tasks.

Hill climbing algorithm maps assignments to a set of assignments by making minor changes to the original assignment. It stops when it reaches a "peak" where no neighbor has a higher value. The LBACO uses load balancing factor and doesn't use self-adapting criteria for the basic ant colony optimization control parameters. So the LBACO with its load balancing factor is better than hill climbing and LBACO without migration will be less efficient than HBB-LB. JSQ have bad performances because it depends only on the status of the queue. It depends only on the number of waited tasks on the queue. It doesn't care with a load of each task.

The final conclusion is that the hybrid algorithm is more suitable than other algorithms because it reduces makespan times. The HBB-LB algorithm balances the system after scheduling process using tasks migration so the HBB-LB algorithm has another cost for tasks migration and migration time.

VI. CONCLUSIONS

In this paper, a hybrid algorithm for handling cloud tasks scheduling has been proposed. The behaviors of an ant colony, particle swarm and honeybee are mixed in the proposed hybrid algorithm. After that, an evaluation of the proposed hybrid algorithm compared to artificial bee colony, ant colony optimization, particle swarm optimization and other known algorithms has been performed. Firstly the best values of parameters for the hybrid algorithm, experimentally determined. Then the algorithms in applications with different sets of tasks evaluated. Simulation results prove that proposed hybrid algorithm is the better, achieves high resource utilization and significantly outperforms the compared algorithms on the basis of makespan and degree of imbalance.

REFERENCES

[1] R. Buyya, J. Broberg, and A. M. Goscinski, "Cloud Computing Principles and Paradigms", Wiley Publishing, 2011.

[2] A. N. Toosi, R. N. Calheiros, and R. Buyya, "Interconnected Cloud Computing Environments: Challenges, Taxonomy, and Survey", ACM Comput. Surv., vol. 47, no. 1, pp. 1-7, may 2014

[3] A. O. Akande, N. A. April, and J. V. Belle, "Management Issues with Cloud Computing", in Proceedings of the Second International Conference on Innovative Computing and Cloud Computing, New York, USA, pp. 119-124, 2013.

[4] E. Pacini, C. Mateos, and C. G. Garino, "Distributed job scheduling based on Swarm Intelligence: A survey", Computers \& Electrical Engineering, vol. 40, no. 1, pp. 252-269, 2014.

[5] P. Singhal, S. K. Agarwal, N. Kumar "Advanced Adaptive Particle Swarm Optimization based SVC Controller for Power System Stability" IJISA Vol. 7, No. 1, PP.101-110,

December 2014.
[6] G. B, F. Zohra, T, and Wieme, Z. "Approaches to Improve the Resources Management in the Simulator CloudSim" in ICICA 2010, LNCS 6377, pp. 189–196, 2010.
[7] M. A. Tawfeek, A. El-Sisi, A. E. keshk and F. A. Torkey, "Cloud Task Scheduling Based on Ant Colony Optimization", International Arab Journal of Information Technology (IAJIT), vol. 12, no. 2, pp. 129-137,2015.
[8] A. El-Sisi, M. A. Tawfeek, A. E. keshk and F. A. Torkey, "Intelligent Method for Cloud Task Scheduling Based on Particle Swarm Optimization Algorithm", The International Arab Conference on Information Technology(ACIT), Oman, 2014.
[9] M. A. Tawfeek, A. El-Sisi, A. E. keshk and F. A. Torkey, " Artificial Bee Colony Algorithm for Cloud Task Scheduling", International Journal of Computer and Information (IJCI), vol. 4, no. 1, pp. 1-9, 2015.
[10] R. F. T. Neto and M. G. Filho, "Literature Review regarding Ant Colony Optimization Applied to Scheduling Problems: Guidelines for Implementation and Directions for Future Research", Engineering Applications of Artificial Intelligence, vol. 26, no. 1, pp. 150-161, 2013.
[11] E. Talbi, "Metaheuristics from Design to Implementation", Hoboken, New Jersey: John Wiley & Sons, Inc., 2009.
[12] S. V. Kamble, S. U. Mane, A. J. Umbarkar "Hybrid Multi-Objective Particle Swarm Optimization for Flexible Job Shop Scheduling Problem" IJISA Vol. 7, No. 4, PP.54-61, March 2015
[13] B. Soumya, M. Indrajit, and P. Mahanti, "Cloud Computing Initiative using Modified Ant Colony Framework," In the World Academy of Science, Engineering and Technology, vol. 56, pp. 221-224, 2009.
[14] M. A. Tawfeek, A. El-Sisi, A. E. keshk and F. A. Torkey, "An Ant Algorithm for Cloud Task Scheduling", International Workshop on Cloud Computing and Information Security CCIS, pp. 169-172, China, 2013.
[15] S. Thomas and H. H. Holger, "MAX-MIN Ant System", Future Generation Computer Systems, vol. 16, no. 8, pp. 889-914, 2000.
[16] K. and Sharma, P. and Krishna, V. and Gupta, C. and Singh, K.P. and Nitin, N. and Rastogi, R. Nishant, "Load Balancing of Nodes in Cloud Using Ant Colony Optimization", in Computer Modelling and Simulation (UKSim), pp. 3-8, 2012.
[17] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud Task Scheduling Based on Load Balancing Ant Colony Optimization", in Chinagrid Conference (ChinaGrid), pp. 3-9, Aug 2011.
[18] A. E. keshk, A. El-Sisi, M. A. Tawfeek, F. A. Torkey, "Intelligent Strategy of Task Scheduling in Cloud Computing for Load Balancing", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), vol. 2, no. 6, pp.12-22,2013.
[19] S. Pandey, Linlin Wu, S. M. Guru, and R. Buyya, "A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments", in 24th IEEE International Conference on Advanced Information Networking and Applications (AINA) , pp. 400-407, April 2010.
[20] D. Babu L.D. and P. Venkata Krishna, "Honey Bee Behavior Inspired Load Balancing of Tasks in Cloud Computing Environments," Applied Soft Computing, vol. 13, no. 5, pp. 2292-2303, 2013
[21] Y. Lua et al., "Join-Idle-Queue: A Novel Load Balancing Algorithm for Dynamically Scalable Web Services," International Journal of Performance Evaluation, August 2011.
[22] V. Gupta, M. Harchol-Balter, K. Sigman, and W. Whitt, "Analysis of Join-the-shortest-queue Routing for Web Server Farms," Perforance Evaluation, vol. 64, pp. 1062–1081, 2011.
[23] B. Mondal, K. Dasgupta, and P. Dutta, "Load Balancing in Cloud Computing using Stochastic Hill Climbing A Soft Computing Approach," Procedia Technology , vol. 4, pp. 783–789, 2012.

Authors' Profiles

Medhat A. Tawfeek received the B.Sc., M.Sc. and PhD in in Computer Science from Menofia University, Faculty of computers and information in 2005, 2010 and 2015, respectively. His research interest includes cloud computing, smart card security, distributed system, fault tolerance and internet of things.

Gamal. F. Elhady: received the B.S, M.S and Ph.D degree in Computer Science at Faculty of Science, in 1998 and 2006, Mansoura University, Egypt. During 1998 and 2006, he works as the researcher student and Lecturer Assistance in Faculty of science computer science Dept. He is member of IAENG in USA (# 108463). His research interest includes software programing, software testing, distributed system, data mining, database, Artificial intelligent, image processing and bioinformatics.

How to cite this paper: Medhat A. Tawfeek, Gamal F. Elhady, "Hybrid Algorithm Based on Swarm Intelligence Techniques for Dynamic Tasks Scheduling in Cloud Computing", International Journal of Intelligent Systems and Applications (IJISA), Vol.8, No.11, pp.61-69, 2016. DOI: 10.5815/ijisa.2016.11.07