Cloudlet Scheduling Based Load Balancing on Virtual Machines in Cloud Computing Environment

Aida A. Nasr1, Nirmeen A. El-Bahnasawy2, Gamal Attiya2, Ayman El-Sayed2

1 Faculty of Artificial Intelligence, Kafrelsheikh University, Egypt.
2 Computer Science and Engineering Dept., Faculty of Electronic Engineering, Menoufia University, Egypt.
Aida.nasr2009@gmail.com, nermeeen.abd@el-eng.menofia.edu.eg, gamal.attiya@yahoo.com, ayman.elsayed@el-eng.menofia.edu.eg

Abstract

Management of cloud computing resources is critical, especially when several cloudlets are submitted simultaneously to cloud computing. Therefore, it is very important to use high efficient cloudlet scheduling techniques to guarantee efficient utilization of computing resources. This paper presents a two-phase approach, called SAAC, for scheduling cloudlets onto Virtual Machines (VMs) of cloud computing environment to balance workload on the available VMs and minimize makespan (i.e., the completion time at the maximum loaded VM). In the first phase, the SAAC approach applies the Simulated Annealing (SA) to find a near optimal scheduling of the cloudlets. While, in the second phase, the SAAC approach improves the cloudlets distribution by applying the Ant Colony Optimization (ACO) considering the solution obtained by the SA as the initial solution. The SAAC approach overcomes the computational time complexity of the ACO algorithm and low solutions quality that obtained by the SA. The proposed approach is evaluated by using the CloudSim, and the results are compared with that obtained by the most recent algorithms in terms of schedule length, load balancing, and time complexity.

Keywords: Cloud computing, Cloudlet scheduling, Load balancing, Ant colony, Simulated annealing

1 Introduction

Cloud computing is a new type of shared infrastructure. It has huge pools of computing resources and allows end users to use these resources over the Internet on-demand by pay-per-use strategy [1]. Users and enterprises can use cloud-computing to obtain high processing power without bearing the elevated price of building huge data centers [2]. Providers, like amazon EC2, let their clients to access, assign, and manage a group of Virtual Machines (VMs) that run inside the data centers and only charge them for the period of using the machines [3].

These days, cloud computing is becoming an efficient computing model. It provides high-performance computing over the Internet. However, one of the key challenges that degrade cloud-computing performance is cloudlets scheduling. This problem characterizes by the presence of limited number of VMs on which several cloudlets need to be executed. Therefore, an efficient cloudlet scheduling technique is required to guarantee good schedule of the cloudlets onto the VMs in order to minimize makespan and enhance cloud performance.

Recently, many researchers have developed meta-heuristic algorithms to solve the scheduling problem. However, most of the existing algorithms ignore several constraints that may affect the scheduling process [4]. In other words, they ignore the requirements of different cloudlets (i.e., memory and processing load requirements) and the availability of cloud computing resources (i.e., the available memory and processing power of VMs) [5].

This paper presents a new two-phase approach for cloudlets scheduling in cloud computing environment to balance workload on the VMs and minimize makespan (i.e., schedule length). The proposed approach, called SAAC, combines Simulated Annealing (SA) and Ant Colony Optimization (ACO). It takes into consideration the requirements of different cloudlets and the availability of computing resources. The proposed approach first finds a near optimal scheduling of cloudlets by applying the SA and then improves the cloudlets distribution by applying the ACO considering the solution obtained by the SA as the initial solution. The proposed SAAC approach overcomes the computational time complexity of the ACO algorithm and low solutions quality that obtained by the SA.

The remainder of this paper is organized as follows. Section 2 introduces cloud computing and cloudlet scheduling problem. Section 3 illustrates the formulation of scheduling problem. Section 4 presents the existing scheduling techniques, while the proposed approach is developed in Section 5. Section 6 presents...
the simulation results while Section 7 presents the conclusion of this research work.

2 Cloud Computing and Cloudlet Scheduling Problem

Cloud computing environment contains several components namely, client, cloud information system, datacenter broker, and Virtual Machines (VMs) [6], as shown in Figure 1. Client is responsible for submitting cloudlet(s) into cloud environment to be processed. All information of the submitted cloudlets is stored in cloud information system. This information includes cloudlet length, arrival time and resources requirements. The information system also monitors the availability of cloud-computing resources. Datacenter broker contains a scheduler, the backbone of scheduling process, which is responsible for assigning the cloudlets onto the VMs. It determines the execution order of each cloudlet. The VMs are the main components in cloud computing environment that responsible for execution of cloudlets.

Figure 1. Components of cloud computing environment

In cloud computing environment, several cloudlets arrive to the system at the same time. Each cloudlet needs to be assigned into a suitable VM to be executed in a shortest time. However, because the number of available VMs is less than the number of submitted cloudlets, the problem is how to find a good schedule of the cloudlets onto the VMs to minimize the execution cost or makespan. This problem is called cloudlet scheduling problem. Briefly, a set of n cloudlets needs to be processed on m VMs in the cloud-computing environment. The cloudlets require certain resources while the VMs have limited resources as memory and processing. Thus, the purpose is to schedule the cloudlets onto the VMs such that the completion time is minimized, the requirements of cloudlets are met, and the availability of the VMs resources is not violated.

3 Problem Formulation

The cloudlet-scheduling problem may be formulated as an optimization problem to be solved by optimization approaches. Designing a mathematical model to the cloudlet scheduling problem involves two steps, (i) formulate a cost function to represent the objective of the cloudlet scheduling, (ii) formulate set of constraints in terms of the cloudlets requirements and the availability of the VMs resources.

To formulate the scheduling problem, let n be the number of cloudlets, m be the number of VMs, \( e_{iv} \) be the processing time of cloudlet i on machine v, and \( X_{iv} \) is a binary variable such that \( X_{iv} = 1 \) if the cloudlet i is assigned to VM v and 0 otherwise, as shown in Eq. (1).

\[
x_{iv} = \begin{cases} 
1, & \text{if cloudlet i assigned to machine v} \\
0, & \text{Otherwise} 
\end{cases} 
\]  

(1)

Therefore, the scheduling problem may be formulated mathematically as:

\[
\min Z = \sum_{i=1}^{n} \sum_{v=1}^{m} e_{iv} x_{iv} 
\]  

(2)

Subject to

\[
\sum_{v=1}^{m} x_{iv} = 1, \quad \forall \text{cloudlet } i 
\]  

(3)

\[
\sum_{i=1}^{n} MI_{i} x_{iv} \leq L_{v} \quad \forall \text{VM} 
\]  

(4)

\[
\sum_{i=1}^{n} \text{mem}_{i} x_{iv} \leq \text{Mem}_{v} \quad \forall \text{VM} 
\]  

(5)

In this model, the objective of scheduling is formulated, Eq. (2), to minimize the execution time. Indeed, several constraints are modeled to meet the requirements of cloudlets and not violate the availability of VM resources. The first constrain, Eq. (3), guarantees that each cloudlet i is assigned to exactly one virtual machine v. The second constraint, Eq. (4), guarantees that the total processing load of all cloudlets, in Million Instructions (MI), assigned to a virtual machine v doesn’t exceed the available processing \( (L_{v}) \) of that VM. The third constraint, Eq. (5), guarantees that the total memory requirements of all cloudlets scheduled into a virtual machine v should not
exceed the maximum available memory of that VM. Where, mem, is the memory requirements of cloudlet i, and Mem, is the available memory of virtual machine v.

After formulating the scheduling problem, the solution may be obtained by solving the formulated problem. The optimal solution to the formulated problem may be achieved by applying an optimization method such as exhaustive search or branch-and-bound. However, the problems of applying such methods are that they provide very high time complexity and it is very difficult to be used in case of large number of cloudlets. Therefore, most of the research works trend to use heuristic methods which achieve near optimal solution.

The optimal solution (Sopt) may be defined as the best solution that achieves the lowest makespan. According to [7], the system can achieve the lowest makespan (i.e., optimal solution) if and only if the next conditions are met:
- Each cloudlet is assigned to distinct VM.
- Each cloudlet starts execution as soon as possible.

4 Related Work

Recently, several cloudlet-scheduling techniques are developed to solve the scheduling problem. Two types of algorithms are developed: heuristic algorithms and meta-heuristic algorithms. Heuristic algorithms use the predictions to achieve a near optimal solution [8]. On the other hand, the meta-heuristic methods search the solution space in a direct manner and produce efficient results on the broad domain problems, but these methods have high time complexity [9]. Genetic Algorithm (GA) and Ant Colony Optimization (ACO) are the famous algorithms. In [10], the authors use GA with ACO to develop a two-phase algorithm for the scheduling process. In the first phase, the authors apply one GA generation and select the best solution. In the second phase, they apply the ACO algorithm, using the GA solution of as initial solution, to obtain near optimal solution. Another metaheuristic cloudlet-scheduling algorithm is Simulated Annealing (SA) [11]. 

In [12], the authors developed a new cloudlet-scheduling algorithm for cloud computing based on SA and greedy. The algorithm uses greedy strategy as initial stage to get a near optimal solution and then improve the solution by applying the SA. Other meta-heuristic algorithm developed to solve the scheduling issue is the Particle Swarm Optimization (PSO) [13].

5 Proposed Approach

This section presents a two-phase approach for cloudlet scheduling in cloud computing to balance workload on the available VMs and achieve minimum schedule length. The first phase of the proposed algorithm finds a near optimal solution rapidly by applying the well-known Simulated Annealing (SA) while the second phase improves cloudlets allocation by applying the Ant Colony Optimization (ACO). The developed approach exploits the low time complexity of SA and the quality of solution of the ACO algorithm. In addition, the approach overcomes the drawback of the ACO that occurs in the first stage due to absence of pheromone by generating an initial solution using the SA algorithm.

5.1 Simulated Annealing

Simulated Annealing (SA) is a global optimization technique that attempts to find lowest point in an energy landscape [14]. The main idea of this method is derived from cooling molten metal slowly to generate a regular crystalline structure. The distinctive feature is that it starts with an initial solution, at high temperature, and incorporates random jumps to potential new solutions. At each temperature level, a new solution is generated and the quality of this solution is tested. The algorithm accepts the new solution as the current solution if it optimizes the objective function. Figure 2 shows sample of SA solution space of 500 cloudlets at 4 VMs. From the figure, solution B is generated after current solution A and the algorithm accepts solution B as it improves quality of solution. In SA, to achieve good solution, initial temperature should be high. In this paper, the SA is developed to generate a good solution at low running time. The algorithm starts at temperature Temp = 1000 and then searches for good solution. The algorithm repeats the searching process until the quality of a solution does not change 10 consecutive solutions. Algorithm 1 presents the SA algorithm steps.

Algorithm 1. Simulated Annealing (SA) Steps
1. Generate Current solution randomly
2. Set No_rejects = 0
3. Set good solution = Current solution
4. Calculate Egood and Ebetter
5. Initialize SA parameters; Temp and cooling rate \( \alpha \in [0-1] \)
6. Generate new solution

![Figure 2. Sample of SA solution space of 500 cloudlets and 4 VMs](image-url)
5.2 Ant Colony Optimization

The Ant Colony Optimization (ACO) algorithm takes its characteristics from the ability of ants to find the shortest path between the food and their nest. The ants connect by laying trails of pheromone. Ants choose their path by computing probability $P$. Where, $P$ depends on pheromone trails on the ground. When the pheromone of any path increases, the number of ants that choose this path will increase. Thus, each new solution is better than the previous [15]. Algorithm 2 illustrates the steps of ACO algorithm.

**Algorithm 2. An Iteration of the ACO Algorithm**

1. Assign $y$ ants on $m$ VMs according to the initial SA solution
2. Initialize $\tau_{iv}$ for each path between cloudlet $i$ and VM $v$
3. While cloudlet-list is not empty repeat
4. For $k=0$ to $y$
5. Ant$_k$ selects a suitable VM$_v$ for the selected cloudlet according to $P_{iv}^{\alpha}$ (T)
6. Insert VM$_v$ in Tabu and remove it from available
7. Remove the selected cloudlet from cloudlet_list
8. Update local pheromone
9. End For
10. End while
11. Update global pheromone

5.2.1 Initialization of Pheromone

When a cloudlet $i$ assigned to a virtual machine $v$, a new path is created with pheromone trial $\tau_{iv}$. By using the SA solution, the algorithm assigns each ant on a specific VM according to the order of initial solution [18]. Then, it initializes the pheromone trial for each edge according to Eq. (6):

$$
\tau_{iv}^{0} = \tau_{iv} + \frac{\text{MIPS(VM$_v$)}}{\text{task$_i$}} \text{length}
$$

Where, $\tau_{iv}^{0}$ is the pheromone trial value at initial iteration $t=0$ for ant $k$, and $\tau_{iv}$ is constant.

5.2.2 Virtual Machine Selection for Next Cloudlet

Each ant applies the next probability $P$ in Eq. 7 to choose VM$_v$ for next cloudlet $i$.

$$
P_{iv} = \frac{\tau_{iv} \times \eta_{iv}}{\sum_{v \in \text{allowed}} \tau_{iv} \times \eta_{iv}} \quad \text{if } v \in \text{allowed}
$$

Where, $\tau_{iv}$ is the pheromone trial of cloudlet $i$ in VM$_v$, allowed$_v$ is the available VMs of ant $k$ that are not chosen yet for any cloudlet by the ant. The VMs that are chosen are stored in tabuce. $\eta_{iv} = 1/d_{iv}$ is heuristic information representing the visibility of ant $k$ at iteration $t$, where $d_{iv}$ is the expected execution time of cloudlet $i$ at VM$_v$.

5.2.3 Pheromone Updating

After each ant creates a path, it updates the local pheromone of this path by Eq. 8.

$$
\Delta \tau_{iv}^{k} = \frac{Q}{L^{k}(t)} \quad \text{if } (i,v) \in T^{k}(t)
$$

Where, $T^{k}(t)$ represents $\text{Tabu}_k$ (the collection of VMs that ant $k$ visited) at iteration $t$, $L^{k}(t)$ is the expected makespan of ant $k$ and $Q$ is a control parameter. After generating a new solution, the global pheromone updates by the Eq. 9.

$$
\tau_{iv}(t+1) = (1-\rho)\tau_{iv}(t) + \Delta \tau_{iv}^{k}(t)
$$

Where, $\rho \in [0-1]$ is the trial volatility coefficient, and $\Delta \tau_{iv}^{k}(t)$ is computed by $\Delta \tau_{iv}^{k}(t) = \sum_{k=0}^{t} \Delta \tau_{iv}^{k}$. 

5.3 Two-Phase SAAC Approach

Algorithm 3 illustrates the developed SAAC approach. The approach combines both the well-known Simulated Annealing (SA) and the Ant Colony Optimization (ACO). In the first phase, the SA is applied to generate an initial solution, while, in the second phase, the ACO is applied to find the best schedule of cloudlets on the available VMs. The developed SAAC approach solves the tradeoff between
Algorithm 3. Two-Phase SAAC Approach
1. Set No_rejects=0
2. Set Current solution=SA solution and Best solution=SA solution
3. For t=0 to t_{\text{max}} do
4. Generate new solution using ACO algorithm
5. If new Solution quality is less than the current solution quality
   6. No_rejects++
   7. If (No_rejects>4)
      8. End the algorithm and Return Best Solution
   9. End If
10. Else
11. Best solution=new solution
12. End If
13. Current solution = new solution
14. End For

From Algorithm 3, the SAAC starts with setting No_rejects equal to zero. This value is used to prevent repeating the same solution and monitors the quality of searching. If No_rejects reaches to 4 this means that the quality is decreased and the solution is bad. At this moment, the SAAC approach will stop and gives the best solution.

5.4 Time Complexity

The time complexity of SA depends on two factors; the number of iterations (itr) and the product (n×m). It has time complexity \( O(\text{itr} \times n \times m) \), where itr is often >1000 for the initialize temperature=2500 and cooling rate=0.01. The time complexity of the ACO is the summation of steps for generate number of solutions \( O(t_{\text{max}} \times n \times m) \), and time complexity for evaluating the solutions \( O(t_{\text{max}} \times n) \). So, the overall time complexity of the ACO is \( O(t_{\text{max}} \times n \times m + t_{\text{max}} \times n) \). Where, \( t_{\text{max}} \) is the maximum number of generations. The time complexity of SAAC is the summation of the time complexity of both the SA and the ACO. The SA has time complexity \( O(\text{itr} \times n \times m) \), while the time complexity of the ACO is \( O(t_{\text{max}} \times n \times m) \). Thus, the overall time complexity of the new SAAC algorithm is \( O(\text{itr} \times n \times m + t_{\text{max}} \times n \times m) \). The SAAC approach uses No_rejects variable to decrease \( t_{\text{max}} \) (i.e. from the experiments \( t_{\text{max}} < 30 \ll n \)). So, the time complexity for SAAC approach may be recomputed as \( O(\text{itr} \times n \times m) \). Since, itr for SAAC algorithm is less than itr for SA algorithm (i.e. itr<200). Therefore, time complexity of SAAC approach is less than time complexity of SA algorithm.

6 Simulation Results

This section presents performance evaluation of the proposed two-phase SAAC scheduling approach. In this evaluation, the well-known CloudSim is used to simulate the cloud computing [16]. The simulation environment is a 64-bit windows 7 operating system installed in a laptop core i5 with 8 GB RAM. In addition, list of cloudlets are generated with lengths from 1000 to 10,000 MI, and a list of VMs is generated with \( MIPS \in [100-1000] \). The initial values of the SAAC approach parameters are \( y=50, \rho=0.5, \tau_c=0.3, \text{Temp}=1000, \alpha=0.01 \) and \( t_{\text{max}}=150 \). The results of the proposed SAAC approach are compared with both the SA and the ACO in terms of schedule length, load balancing and time complexity.

6.1 Schedule Length

Schedule length (SL) or makespan is the elapsed execution time at the maximum loaded virtual machine. Table 1 shows a simple comparison between the proposed SAAC approach and both the ACO and the SA algorithms in terms of makespan. From the table, the proposed SAAC approach is more efficient than the ACO and SA algorithms for all test cases. The new SAAC approach has lower schedule length at smaller number of VMs and larger number of cloudlets.

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Makespan in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Cloudlets</td>
<td>No. of VMs</td>
</tr>
<tr>
<td>500</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1500</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>2500</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>3000</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>3500</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>4000</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
From Table 1, both the SA and the ACO algorithms provide high schedule length with large number of cloudlets. In some cases (at 4000 cloudlets and 2 VMs), the new SAAC approach can save more than 2 hours than both the SA and the ACO algorithms. The SAAC approach minimizes the schedule length by 47.24% with SA and 14.5% with ACO. This is because the SAAC approach uses the advantages of SA and ACO algorithms with additional modifications to minimize the schedule length.

6.2 Balancing Degree

Balancing Degree (BD) is the degree of balancing the workload on the available VMs after scheduling. The balancing degree may be calculated as follows.

\[
BD = \frac{SL(S_{opt})}{SL(S_{fin})}
\]  

Where, \(S_{opt}\) is the optimal solution and \(S_{fin}\) is the final solution obtained from the applied algorithm. In this study, \(S_{opt}\) is assumed as the ideal solution, where \(S_{opt}\) is computed as the summation of total MI of all the cloudlets over the total MIPS of the available VMs. That is, \(S_{opt} = \text{Total MI}/\text{Total MIPS}\). From Eq. (10), the algorithm with high balancing degree achieves near optimal solution.

Figure 3, Figure 4 and Figure 5 show the balancing degree of scheduling different cloudlets by three different algorithms at 2, 4, and 8 VMs respectively. From the figures, the SAAC achieves higher BD ratio than SA and ACO algorithms, because it achieves the lowest makespan. Let’s take Figure 3 as an example. From the figure we note that the new algorithm always has higher BD ratio than the others. Although, the new algorithm gives BD ratios near of ACO in Figure 4, and Figure 5, it achieves the near optimal solution at very little time than the ACO. This is shown in Figure 6, Figure 7, and Figure 8. The developed SAAC improves BD ratio with 45.9% than SA and 4.8% than ACO algorithms.

6.3 Computation Time Complexity

Figure 6, Figure 7 and Figure 8 show the computation time complexity of the SAAC, the SA and the ACO at different values of VMs; 2, 4, and 8 respectively. The figures show that the SAAC approach has lower time complexity than both the SA and the ACO. The SAAC approach combines both the SA and the ACO algorithms to improve the overall performance of cloud computing. The new approach avoids the high time complexity by using a new variable called No_rejects, which counts the number of rejected solutions, and stops the repetition when reaching to the specific value. This prevents generating new bad solutions more times for minimizing time complexity.
Cloudlet Scheduling Based Load Balancing on Virtual Machines in Cloud Computing Environment

Figure 7. Running time of different algorithms at 4 VM

Figure 8. Running time of different algorithms at 8 VM

From Figure 6, Figure 7 and Figure 8, the SAAC has running time less than the SA and the ACO algorithms. For example, in Figure 6, the SAAC approach takes 40 second to find the near optimal solution, whereas the ACO takes 100 second, and SA takes 45 second. In addition, from Figure 8, the SAAC approach takes less than 100 second to find the near optimal solution of 4000 cloudlets, whereas the ACO algorithm gets a solution at 600 second. On other words, the SAAC approach is better than the SA and ACO in terms of running time. It decreases the running time 20.5% than SA and 87.12% than ACO algorithms.

In summary, from the figures, the new SAAC is better than both the SA and the ACO algorithms in terms of schedule length, balancing degree, and running time.

6.4 Real Experiments

To insure that the new approach is more efficient than the other algorithms, the proposed SAAC is applied to schedule a list of real cloudlets on different numbers of VMs. The cloudlets are generated by using a standard formatted workload of a high-performance computing center, called HPC2N, in Sweden as a benchmark [17]. Table 2 shows the makespan obtained by 4 different algorithms.

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Makespan in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Cloudlets</td>
<td>No. VMs</td>
</tr>
<tr>
<td>1000</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
<tr>
<td>6000</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
<tr>
<td>8000</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
</tbody>
</table>

From Table 2, the new SAAC algorithm achieves the lowest schedule length in all the test cases. It can distribute a large number of cloudlets into different numbers of VMs and achieves better schedule length than the current SA, ACO, PSO, and the Hybrid algorithm [18].

7 Conclusion

In this paper, a new two-phase SAAC approach is proposed for cloudlet scheduling in cloud computing environment. The proposed SAAC approach can schedule a large number of cloudlets into the available VMs considering several resources constraints, at low time complexity. It keeps the VMs in high balancing degree, and enhances the overall system performance. By comparing the new SAAC approach with the SA and the ACO, the new approach is more efficient than those algorithms. The experimental results show that the SAAC approach decreases running time by about 32.5% than SA and 87.12% than ACO algorithm. In addition, it achieves lower schedule length than SA and ACO algorithms. It decreases the schedule length by about 47.24% with SA and 14.5% with ACO. The SAAC also provides high load balancing degree. It improves balancing degree ratio by 45.9% than SA and 4.8% than ACO algorithms. Moreover, the new SAAC algorithm is compared with the PSO and the Hybrid algorithms.

References


Biographies

Aida A. Nasr received a Ph.D. in 2019, in the Dept. of Computer Science and Engineering, Faculty of Electronic Engineering, Menoufia University, Egypt. She is a Lecturer in the faculty of Artificial Intelligence, Kafrelsheikh University, Egypt. She received M.Sc. and B.S. from the Faculty of electronic Engineering, Menoufia University, Egypt in 2015 and 2010 respectively.

Nirmeen A. El-Bahnasawy received her B.S. in Electronic Engineering in 1998 and M.Sc. and Ph.D. degrees in Computer Science and Engineering from Menofia University in 2003 and 2013, respectively. Currently, she has been appointed as an Associate Professor at Menofia University in 2019. Her research interests include distributed computing, grid computing, and Cloud computing.

Gamal Attiya graduated in 1993 and obtained his M.Sc. degree in computer science and engineering from the Menoufia University, Egypt, in 1999. He received Ph.D. degree in computer engineering from the University of Marne-La-Vallée, Paris-France, in 2004. He is currently full Professor at the Department of Computer Science and Engineering, Faculty of Electronic Engineering, Menoufia University, Egypt.

Ayman El-Sayed received the B.Sc. (1994) and M.Sc. (2000) in computer science and engineering from the Menoufia University, Egypt. Ph.D. (2004) in computer network from Institute National De Polytechnique De Grenoble (INPG), France. He is full Professor/Dean of Faculty of Electronic Engineering, Menoufia University, Egypt. He is a IEEE senior member.