Effect of Acceleration Coefficient on Particle Swarm optimization for Task Scheduling in Cloud Computing

Nagresh Kumar¹,*

¹Meerut Institute of Engineering and Technology, Meerut, U.P, India

Abstract

Cloud computing emerges as a powerful platform to deliver IT services online. Due to the rapid development of cloud computing the user's dependence on the cloud has increased and hence user request per unit time is increases. Now scheduling and serving the user requests is a major challenge. Particle swarm optimization as a heuristic algorithm is the most suitable algorithm in such scenario to serve user requests for the most appropriate resources. Author written this research paper in continuation with previous research paper called Modified particle swarm optimization (MPSO) in which author controlled the inertia weight in PSO to find the best cost. This research paper investigates the effect of acceleration coefficient to achieve the best cost. The implementation results of PSO with different acceleration coefficient are produced and compared. Author has use MATLab to test the effect of acceleration coefficient on fitness value and also implemented in CloudSim simulator to test variation in execution time in various scenario. The purpose of author is also to test correctness of Reyes-Sierra and Coello [19] suggested acceleration coefficient.

Keywords: Particle Swarm Optimization (PSO), Modified Particle Swarm Optimization (MPSO), Inertia Weight, Acceleration coefficient.

Received on 19 October 2020, accepted on 21 December 2020, published on 15 January 2021

Copyright © 2021 Nagresh Kumar et al., licensed to EAI. This is an open access article distributed under the terms of the Creative Commons Attribution license, which permits unlimited use, distribution and reproduction in any medium so long as the original work is properly cited.

doi: 10.4108/eai.15-1-2021.168140

*Corresponding author. Email: nagresh@gmail.com

1. Introduction

Cloud computing is a model that enables convenient, on demand network access which is required for a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be quickly provisioned and deployed with minimal management effort or service provider interaction [2]. It is a new computing system that uses geographically shared computer resources, and is available to users with a variety of online services with minimal management effort [1]. Task scheduling is a major challenges related to accessing appropriate resources in the cloud to increase costs under a variety of factors [4].

Task scheduling refers to scheduling of task to appropriate resources to optimize computing costs. The computing performance depends on proper scheduling of task to the appropriate resources. Existing algorithms are using parameters such as time, makepan, speed, cost, setting the success rate, resource usage and so on. [1,5, 6, 7, 9, 10, 11]. Other parameters such as availability and reliability can play an important role and can be considered [4]. Since task scheduling is a complex problem, so a heuristic algorithm will be more suitable. The PSO heuristic algorithm has already been used to solve the scheduling problem [2,13,14,16,17]. The main objective of this research paper is to test correctness of acceleration coefficient suggested by Reyes-Sierra and Coello [19] and to find suitable parameter of PSO to improve the performance of the task scheduling to achieve better costs. This paper is written in continuation with research paper [16,25].

This research paper is structured as follows: a review of previous work is described in section 2. Section 3 contains details about the proposed algorithm. Section 4 describes the test setup, the implementation of the MPSO
algorithm and the results. Section 5 describes the future work and conclusion.

2. Related Work

Workflow scheduling in cloud computing refers to sending user tasks to available resources. It is an NP-Complete problem [12]. Various heuristic based algorithmic methods have already proposed and raised the issue. There are many heuristic algorithms suggested such as PSO, Ant colony algorithm, Genetic algorithm etc. [13]. Genetic algorithm is a great way to search the solution but sometimes it sticks to the local solution. It also deals with a large number of different parameters. The PSO convergence rate is higher than the genetic algorithm and works with a less number of parameter.

Jun Xue, S et al. [1] proposed a PSO algorithm for job scheduling that takes a small update in the position to reduce computing costs. The PSO embedded in crossover and mutation performs better compared to the original PSO in terms of convergence rate, accuracy, efficiency and success rate [15]. A PSO based heuristic proposed to schedule the user tasks considering parameters computation cost and transmission cost [14]. The test result shows better load distribution and reduced overall costs.

Figure-1 completely represents that how velocity and position of a particle updated its local best in the environment to get global best. There are four equations suggested by researcher [17, 18] by which a particle can update its position and velocity. These four equations are represented in equation-1, 2, 3 and 4.

\[ v_i(t+1) = w_{vi}(t) + c_1(p_i(t) - x_i(t)) + c_2(g(t) - x_i(t)) \]  \hspace{1cm} (1)

\[ x_i(t+1) = x_i(t) + v_i(t+1) \]  \hspace{1cm} (2)

\[ v_j(t+1) = w_{vij}(t) + r_1c_1(p_{ij}(t) - x_{ij}(t)) + r_2c_2(g(t) - x_{ij}(t)) \]  \hspace{1cm} (3)

\[ x_{ij}(t+1)) = x_{ij}(t) + v_{ij}(t+1) \]  \hspace{1cm} (4)

where:

- \( p_i(t) \): The ith particle best position at given time t
- \( g(t) \): Global best solution
- \( x_i(t+1) \): Particle new updated position
- \( x_{ij}(t+1) \): Particle position updated at time t+1
- \( v_{ij}(t+1) \): Particle velocity of \( i^{th} \) particle in time stamp t+1
- \( r_1, r_2 \): Random numbers between 0 and 1
- \( c_1, c_2 \): Acceleration coefficients
- \( w_{vi} \): The inertia component
- \( r_1c_1(p_{ij}(t) - x_{ij}(t)) \): The cognitive component
- \( r_2c_2(g(t) - x_{ij}(t)) \): The social component.

The inertia weight (w) in equation-3 controls the flow of particles resulting in an improved PSO convergence rate. Improved convergence rate is able to find the best global solution. The \( r_1 \) and \( r_2 \) used for random particle movements in the search space. Acceleration coefficient \( c_1 \) and \( c_2 \) is used to control the position of the global and personal best of particles position. The values of \( c_1 \) and \( c_2 \) are maintained constant to avoid influence by its positive position and that of the neighbours [19]. In this research paper variation of acceleration coefficient is the main goal. But acceleration coefficient will be constant in each execution. The w value is taken between 0.5 and 0.9 [16]. Reyes-Sierra and Coello [19] suggested acceleration coefficient \( c_1 \) and \( c_2 \) to be 1.5 for the higher convergence value.

In [16] the author proposed a new inertia weight controlled PSO that maintains a weight of inertia between 0.5 and 0.9 which is able to obtain better result than the original PSO. In implementation of previous research paper [25], the acceleration coefficient was constant, but in this research paper inertia weight varies between 1.5 and 2 on the basic of different scenario and kept constant during each execution. At the same time the purpose of this research paper is to prove the effectiveness of MPSO

A. Modified Particle swarm optimization-1 (MPSO-1)

This MPSO algorithm is implemented which keeps inertia weight between 0.5 and 0.9 according to Modified Simple Random Inertia Weight (MSRIW) as given below [16].

\[ w(t) = 0.9 - 0.5 \times \text{rand}() \]  \hspace{1cm} (5)

In above equation the value of w is between 0.4 and 0.9.

B. Modified Particle swarm optimization-2 (MPSO-2)

This algorithm implementation is based on Modified Oscillating Inertia Weight which keeps inertia weight between 0.5 and 0.9 [16].

\[ w(t) = \frac{w_{\text{min}} + w_{\text{max}}}{2} + \frac{w_{\text{max}} + w_{\text{min}}}{4} \cos \left( \frac{2 \pi t}{T} \right) \]  \hspace{1cm} (6)

where:

- \( w_{\text{min}} \) and \( w_{\text{max}} \): minimum and maximum inertia weight
- \( T \): Oscillation period.

In this research paper, the oscillation period is set to 100 iterations.
Where, $T$ is the number of iterations in which $w(t)$ completes a cycle. The equation of $T$ is:

$$T = \frac{2S_1}{3 + 2k}, \quad \text{-------------------------- (7)}$$

Where $S_1 = \frac{3S}{4}$ is total number of iterations required to oscillates, $S$ is the total number of iterations and $k$ is the frequency of oscillation.

### 3. Algorithm Design

First of all task resource mapping is done based on the task buffer size and the available computing resources. Then calculate the average computation cost corresponding to all task resources operation and the cost of communication between resources. The MPSO algorithm can be used to find the best task-resources mapping to optimize the cost by calculating the value of the fitness. Tasks are selected by a selection mechanism and are stored in the "ready task" repository before execution. The "ready-to-use" also used to verify task dependence to ensure that it should contain those tasks whose parent's task has already executed. The repetition of MPSO algorithm recomputed the task-resource mapping which make MPSO robust heuristic and dynamically balanced.

**Algorithm 1: MPSO Task Scheduling**

1. Compute task resource mapping
2. Find average computation cost of all task $(t_i)$ on all computing resources $(p_i)$
3. Execute PSO algorithm for all task $t_i$
4. For all ‘ready’ task $t_i \in T$ do
5. Assign task $t_i$ to resources $p_i$
6. End for loop
7. Dispatched all the mapped task in ready task buffer
8. Update ready task buffer
9. Update communication cost between resources
10. Repeat step 3 till unscheduled task remains

**MPSO Algorithm:**

- The PSO algorithm is used to find the best fitness value for a particle. Steps of algorithm are described below. PSO algorithm is started with initialization of position and velocity of the particle. Particles mean the number of user tasks that has to be assign to the resources and dimension means the task resource mapping. The performance of each particle is evaluated by their fitness value. The particles are continued to update their velocity and position corresponding to equation 3 and 4 until maximum number of iteration reached.

**Algorithm-2: MPSO algorithm.**

1. Initialize particle size. Particle size must be according to size of “ready task”.
2. Set particle position and velocity randomly.
3. Varies acceleration coefficient in each scenario.
4. Find fitness value of each particle.
5. If present fitness value is better than previous fitness value then updates ‘pbest’ with present fitness value.
6. Repeat step 3 and 4 for all particle and set best particle as ‘gbest’.
8. Repeat step 3 till maximum iteration reached.

### 4. Parameters used in CloudSim simulator

CloudSim is a framework for modeling and simulating cloud computing infrastructure and services. In CloudSim scheduling perform at two levels. The first is implemented between Hosts and Virtual Machines (VM) and then implemented between Virtual Machines and cloudlets [21, 22]. In Space-Shared editing object processing can be element can be allotted to a new virtual machine. While time-sharing scheduling, task can be shared to the processing elements for execution [23, 24].

Here we analyzed various scenarios with different parameters in the cloud computing system corresponding to MPSO algorithm as described above [16]. In each case, we change one parameter and keep the other parameters constant. On the basis of system behaviour, we analyze its performance and effectiveness. Virtual machine, task and host configuration are initially considered as given below:

**VM Parameters**

- long size = 10000;  //image size (MB)
- int ram = 512;  //vm memory (MB)
- int mips = 1000;
- long bw = 1000;
- int pesNumber = 1;  //number of cpus

**Task (cloudlet) parameters:**

- long length = 1000;
- long file Size = 300;
- long output Size = 300;
- int pes Number = 1;

**Host Parameters:**

- int ram = 4096;  //host memory (MB)
- long storage = 1000000;  //host storage
- int bw = 10000;
- CPU/Cores=Quad core and dual core

**Datacenter-2:** Contain 2-host with one quad core and one dual core (Total P.E. = 6)

**Virtual Machine (VM) =20**

**Cloudlet (Task) = 40(not divisible)**

**Scheduling:** Space Shared and Time Shared

VM allocation policy: this policy chooses a host for a VM with fewer processing elements (PEs) in use. This allocation policy does not perform any optimization of the VM allocation.
Scenario
In this scenario the parameters used are Datacenter= 2 and each Datacenter contain two hosts in which one host is quad-core and other is dual-core. Number of tasks is 40, population size=1000, c1=1.5, c2=2. VM’s Allocation Policy is space shared and changing the number of virtual machines (VM) metric of result is Execution Time (ET) value of tasks ( in milliseconds) taken by three different methods is shown in table-1.

<table>
<thead>
<tr>
<th>S.N</th>
<th>C1</th>
<th>C2</th>
<th>PSO</th>
<th>MPSO1</th>
<th>MPSO2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.5</td>
<td>1.5</td>
<td>8.04E-167</td>
<td>7.78E-215</td>
<td>6.73E-204</td>
</tr>
<tr>
<td>2</td>
<td>1.5</td>
<td>1.6</td>
<td>2.56E-189</td>
<td>3.92E-201</td>
<td>9.54E-229</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>1.7</td>
<td>7.15E-225</td>
<td>8.03E-213</td>
<td>6.38E-198</td>
</tr>
<tr>
<td>4</td>
<td>1.5</td>
<td>1.8</td>
<td>1.55E-170</td>
<td>9.76E-187</td>
<td>2.53E-168</td>
</tr>
<tr>
<td>5</td>
<td>1.5</td>
<td>1.9</td>
<td>9.20E-176</td>
<td>5.45E-189</td>
<td>7.15E-172</td>
</tr>
<tr>
<td>6</td>
<td>1.5</td>
<td>2.0</td>
<td>2.46E-173</td>
<td>1.28E-163</td>
<td>5.87E-174</td>
</tr>
<tr>
<td>7</td>
<td>2.0</td>
<td>1.5</td>
<td>6.10E-112</td>
<td>2.17E-170</td>
<td>7.89E-138</td>
</tr>
<tr>
<td>8</td>
<td>2.0</td>
<td>1.6</td>
<td>9.30E-134</td>
<td>4.84E-147</td>
<td>3.56E-140</td>
</tr>
<tr>
<td>9</td>
<td>2.0</td>
<td>1.7</td>
<td>7.51E-82</td>
<td>8.39E-156</td>
<td>1.34E-134</td>
</tr>
<tr>
<td>10</td>
<td>2.0</td>
<td>1.8</td>
<td>6.46E-72</td>
<td>3.36E-123</td>
<td>3.46E-101</td>
</tr>
<tr>
<td>11</td>
<td>2.0</td>
<td>1.9</td>
<td>1.58E-31</td>
<td>9.67E-104</td>
<td>7.19E-87</td>
</tr>
<tr>
<td>12</td>
<td>2.0</td>
<td>2.0</td>
<td>8.48E-21</td>
<td>5.53E-89</td>
<td>3.24E-51</td>
</tr>
</tbody>
</table>

Table 1.

![Execution time vs acceleration coefficient](image)

Fig. 2. Execution time vs acceleration coefficient

In the above graph serial numbers represents the corresponding to acceleration coefficients as shown in table-1.

5. Result Analysis
The implementation results of MPSO-1 and MPSO-2 with varying acceleration coefficients are shown in table-1 and plotted in figure-2. As Reyes-Sierra and Coello [19] suggested acceleration coefficient with value 1.5 is most suitable value in PSO. Our implementation also proved that acceleration coefficient close to 1.5 is most suitable value as shown in table. Our proposed modification in PSO which keep inertia weight value between 0.5 and 0.9 is also the performing much better than random inertia weight algorithm ,as shown in fig-2 that MPSO-1 and MPSO-2 gives better result than original PSO by keeping inertia weight between 0.5 and 0.9 [16].

6. Conclusion
It is clear from the implementation results of PSO, MPSO-1 and MPSO-2 that acceleration coefficient and inertia weight (w) plays an important role to find the best cost. MPSO-1 and MPSO-2 both keep the value of w between 0.5 and 0.9, and enhance the searching capabilities for best cost. The implementation results of MPSO on CloudSim show that on average it is capable to find best cost as compared to original PSO. Our experimental results conclude that acceleration coefficient should be close to 1.5. In future to accept MPSO as a scheduling algorithm some other parameters like population size, number of iterations, virtualization and cognitive component must be taken into account.

References
Effect of Acceleration Coefficient on Particle Swarm optimization for Task Scheduling in Cloud Computing


