CHAPTER 3

POWER-AWARE VIRTUAL MACHINE ALLOCATION IN CLOUD COMPUTING

3.1 INTRODUCTION

Cloud computing is an incrementally maturing technology that has been researched upon widely from all front. Though many services are provided by the cloud environment, the foundation of all the services is Infrastructure as a Service (IaaS). Apart from the benefits gained by the cloud computing, the pitfall is a huge amount of energy consumed by the datacenters which contribute to its operational cost. An important and daunting problem in a cloud environment is the power consumption in datacenters and its associated thermal heating resulting in reliability concerns. Lowering the power usage of datacenters becomes a challenging issue because computing applications and data are growing so rapidly, larger servers and disks are needed to process them fast enough within the required time period (Buyya et al. 2010). Datacenters consume 10-100 times more energy per square foot than typical office buildings (Scheihing et al. 2007). They can even consume as much electricity as a city (Markoff & Lohr 2002). Thus, datacenter resources need to be managed in a power-efficient manner to drive green cloud computing.
This problem needs to be concentrated on reducing the power consumption of the datacenters by power-aware VM allocation. There are several researchers addressing the problem of VM allocation in cloud computing environment, having the objective of minimizing the power consumption of datacenters. Although traditional heuristic algorithms find solutions for virtual machine mapping for power efficiency, they result in local optima. This multidimensional bin-packing problem can be solved by the meta-heuristic algorithms such as Genetic Algorithm (GA), Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO). They are well suited for virtual machine placement problem because of their ability to find out the near-optimal solution in a short period of time even for the large search space, provided the parameters are properly chosen. However, many studies illustrated that PSO algorithm is able to get the better solution than GA in distributed system and grid computing (Salman A et al. 2002) (Zhang L et al. 2008) and also observe that converge rate of PSO is faster than GA (Salman A et al. 2002).

Xiong et al. (2014) proposed Multi-Resource Energy Efficiency based on PSO (MREE-PSO) for placement of virtual machines in the cloud environment. The results are compared with two important traditional heuristic algorithms namely Modified Best Fit Decreasing (MBFD) and Modified Best Fit Heuristics (MBFH), it is observed that MREE-PSO has a better energy efficiency than MBFD and MBFH. Jeyarani et al. (2012) proposed Self-Adaptive PSO (SAPSO) for power-aware VM placement by exploiting the different sleep modes of the servers such as nap, sleep, winkle, and idle. The algorithm proposed by them chooses the host by means of the increase in the power consumption for VM placement. It is observed that SAPSO outperforms the Multi-Strategy Ensemble PSO (MEPSO) and the standard PSO in terms of reducing power consumption. A substantial amount of work has been carried out in the area of power-efficient cloud computing. From the
literature survey, it is observed that the existing algorithms have certain limitations such as the dynamic power model is not considered and real time benchmark data is not taken for result analysis.

To overcome the limitations of existing algorithms, Power-Aware Enhanced PSO (PA-EPSO) is proposed. The power model exhibited in this work considers the dynamic nature of resource utilization of the cloud environment. The goal of the proposed research is to multiplex the physical resources to the cloud users to reduce the power consumption of the datacenters. Firstly, the dynamic power consumption model of the cloud environment is discussed. As virtual machine mapping is a combinatorial optimization problem, it can be solved using swarm intelligence methods. Particle Swarm Optimization with linear decreasing inertia weight named Enhanced Particle Swarm Optimization is applied for the placement of virtual machines in the cloud environment to minimize the power consumption of the datacenters.

3.2 SYSTEM MODEL AND PROBLEM STATEMENT

The target system is an IaaS environment, represented by a large-scale datacentre consisting of N heterogeneous physical nodes. The type of the environment implies no knowledge of application workloads and time for which VMs are provisioned. In other words, the resource management system must be application-agnostic.

Virtual machine requests are considered as “N”. Out of “M” physical machines, “N” mappings should be found on the available physical machines that satisfied the virtual machine’s resource requirements as well as given optimum power consumption.
Since the cloud has an invariant number of resources, it is assumed that $N \ll M$. In the context of power-aware mapping, the hosts are modeled in terms of cores, memory, bandwidth, processing speed in Million Instructions Per Second (MIPS) and the power consumption. The servers are not having direct-attached storage devices; the storage devices used are Network Attached Storage (NAS) and Storage Area Network (SAN) to enable the virtual machine migration. The hosts are equipped with multi-core CPUs. Each core is having $m$ MIPS, so the total capacity of the processing power of the host with $n$ number of cores is $nm$ MIPS. Normally users submit virtual machine requests to popular virtual machine management systems with QoS requirements, such as required virtual processor speed, memory size, operating system and other hardware/software environments. The users establish SLAs with the cloud provider in terms of QoS requirements. The objective of this research work is to propose an algorithm for power aware virtual machine placement in the cloud environment. Datacenter, Host, and VM are represented as follows.

- **Datacenter** $DC : \{\text{Host, VM, PowerModel}\}$
- **Host** : $\{\text{Core, Memory, Bandwidth, MIPS, PowerConsumption}\}$
- **VM** : $\{\text{Core, Memory, Bandwidth, MIPS}\}$

### 3.2.1 Power Model

Power consumption model is formulated in order to effectively control the power consumption of cloud computing clusters. The power model helps to figure out how cloud computing swallows up so much energy, so the establishment of the power consumption model for cloud computing is a priority. Recent studies show that the power consumption of servers can be accurately described by a linear relationship between the power consumption
and CPU utilization (Fan et al. 2007). Hence, in this work, it is taken as power consumption of a physical machine is linearly related to resource utilization (e.g. CPU utilization) as mentioned in (Fan et al. 2007) (Beloglazov et al. 2012) (Beloglazov & Buyya 2010) (Beloglazov & Buyya 2012). The VM’s CPU utilization ($CPU_{vm,i}$) is calculated by the following Equation (3.1).

$$CPU_{vm,i} = VM_{mips,i}/HOST_{mips,j}$$  \hspace{1cm} (3.1)

The host CPU utilization ($CPU_{host,j}$) is calculated by the following Equation (3.2).

$$CPU_{host,j} = \sum_{i \in r_j} CPU_{vm,i}$$ \hspace{1cm} (3.2)

Finally, the power consumption of the $j$th host running multiple virtual machine instances is calculated by Equation (3.3).

$$p_j = \begin{cases} \left( p_j^{busy} - p_j^{idle} \right) * CPU_{host,j} + p_j^{idle}, & CPU_{host,j} > 0 \\ p_j^{idle}, & CPU_{host,j} = 0 \end{cases}$$ \hspace{1cm} (3.3)

Where,
- $VM_{mips,i}$: total required MIPS of $VM_i$
- $HOST_{mips,j}$: total capacity MIPS of $HOST_j$
- $r_j$: set of indexes of the VMs those were allocated on the $HOST_j$
- $p_j^{idle}$: the power consumption (Watts) of the host in idle (0% CPU utilization)
- $p_j^{busy}$: the maximum power consumption (Watts) of the host in 100% CPU utilization
- $p_j$: current power consumption of $j$-th host
3.2.2 Construction Graph and Constraints

Table 3.1 Construction Graph of M Physical Machine and N Virtual Machine

<table>
<thead>
<tr>
<th></th>
<th>VM_1</th>
<th>VM_2</th>
<th>...</th>
<th>VM_i</th>
<th>...</th>
<th>VM_N</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_1</td>
<td>x_{11}</td>
<td>x_{12}</td>
<td>...</td>
<td>x_{1i}</td>
<td>...</td>
<td>x_{1N}</td>
</tr>
<tr>
<td>PM_2</td>
<td>x_{21}</td>
<td>x_{22}</td>
<td>...</td>
<td>x_{2i}</td>
<td>...</td>
<td>x_{2N}</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>PM_j</td>
<td>x_{j1}</td>
<td>x_{j2}</td>
<td>...</td>
<td>x_{ji}</td>
<td>...</td>
<td>x_{jN}</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>PM_M</td>
<td>x_{M1}</td>
<td>x_{M2}</td>
<td>...</td>
<td>x_{Mi}</td>
<td>...</td>
<td>x_{MN}</td>
</tr>
</tbody>
</table>

The construction graph is shown in Table 3.1 (Xiong et al. 2014). The hosts are represented as M rows and the VMs are represented as N columns. \( x_{ji} \) is a binary variable that indicates whether \( VM_i \) can be assigned to \( HOST_j \) or not. If \( VM_i \) is allocated to \( HOST_j \), \( x_{ji} \) is set to 1, if not \( x_{ji} \) is set to 0.

The power consumption of datacenter is calculated using Equation (3.4). The appropriate mathematical model and constraints are applied during VM allocation. Equation (3.4) is the objective function to be minimized which measures the quality of the solution.

\[
P_{DC} = \sum_{j=1}^{M} P_j = \sum_{j=1}^{M} \left[ \left( P_{j}^{busy} - P_{j}^{idle} \right) \times CPU_{host,j} + P_{j}^{idle} \right]
\]

Subject to the constraints:
Equation (3.5) represents the constraint that a VM needs to be allocated to only one physical machine. The binary variable $x_{ji}$ indicates whether $VM_i$ can be assigned to $HOST_j$ or not. In Equation (3.6), $reqVM_i$ and $availHost_j$ denote the minimum resource requirements of the i-th virtual machine and the available resource of j-th host, respectively. This equation describes that the total resource requirement of the virtual machine does not exceed the resource availability of the physical machine.

The objective of this research work is to minimize the power consumption of the datacenters. Linearly Decreasing Inertia Weight PSO algorithm is used for the allocation of VM to the PM. The PA-EPSO algorithm is compared with existing Standard PSO (StdPSO) and Self-Adaptive PSO (SAPSO) algorithms for different problem instances in terms of power consumption of datacenters.

### 3.3 OVERVIEW OF PARTICLE SWARM OPTIMIZATION ALGORITHM

Generally, finding an allocation that meets a specific management objective can be formulated as an optimization problem. Particle Swarm Optimization (PSO) is a population-based optimization algorithm which may well be carried out and used simply to resolve numerous function optimization problems. Kennady & Eberhart (1995) introduced the Particle Swarm Optimization algorithm, which has been derived from the concept of swarming habits of animals such as birds and fishes. PSO uses the concept of...
social interaction for resolving a problem. Every single solution is a bird, called as a particle in the solution space. PSO maintains multiple potential solutions at a time. Swarm is a population that signifies the set of possible solutions. The solutions are evaluated by the fitness function to determine its value. The particles are flown and swarmed through the search space to find the optimum and the particle's best position obtained so far. Swarm maintains the global best position obtained so far. As time passes, the particles adjust their position with the help of their experience as well as with the experience of their neighbor particles. The particle’s velocity is used to define the direction and how fast the particle should move to find the optimal solution. The velocity is calculated using the experience of the particle (the best position the particle gained so far) and the experience of the swarm (the most successful particle in the swarm). The velocity is used to direct random particles to take off in the search space (Ercan & Li 2003). The new position of the particle is calculated by adding its current position to the new velocity. This process continues iteratively, until either the stopping criterion or a maximum number of iterations is met or no further improving solution is found (Visalakshi & Sivanandam 2009, Ercan & Li 2013). The particle's position and velocity are updated by the following Equations (3.7) and (3.8), correspondingly.

\[ v_{id}(t + 1) = w v_{id}(t) + c_1 r_1 [p_{best_{id}}(t) - x_{id}(t)] + c_2 r_2 [g_{best}(t) - x_{id}(t)] \]  

(3.7)

\[ x_{id}(t + 1) = x_{id}(t) + v_{id}(t + 1) \]  

(3.8)
Where,

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>Denotes the particles</td>
</tr>
<tr>
<td>$d$</td>
<td>Denotes the dimension</td>
</tr>
<tr>
<td>$v_{id}(t)$</td>
<td>Velocity of particle $i$ at $t^{th}$ iteration</td>
</tr>
<tr>
<td>$x_{id}(t)$</td>
<td>Position of particle $i$ at $t^{th}$ iteration</td>
</tr>
<tr>
<td>$w$</td>
<td>Inertia Weight</td>
</tr>
<tr>
<td>$r_1, r_2$</td>
<td>Uniformly distributed random numbers between 0 to 1</td>
</tr>
<tr>
<td>$c_1, c_2$</td>
<td>Acceleration coefficient between 1 to 4</td>
</tr>
<tr>
<td>$pbest_{id}(t)$</td>
<td>Personal best denotes the best position the particle has gained so far</td>
</tr>
<tr>
<td>$gbest(t)$</td>
<td>Global best denotes the swarm has gained so far</td>
</tr>
</tbody>
</table>

$w$ is an inertia weight, used as mechanisms for the control of the velocity’s magnitude. A large inertia weight facilitates global exploration by searching new spaces while a small inertia weight facilitates the local search. Suitable selection of the parameter $w$ provides a balance between the local and global exploration search abilities to reach the optimum in a minimum of iterations (Schutte et al. 2004). The inertia weight is constant or varying. The performance of the $pbest_{id} = (pbest_{i1}, pbest_{i2},..., pbest_{iN})$ is the personal best of particle $i$. $gbest_i = (gbest_{i1}, gbest_{i2},..., gbest_{iN})$ is the best position obtained by any particle so far. Each particle is moved by considering the initial random velocity and two weighted influences, individuality and sociality. Individuality means the readiness to go back to the particle’s best the previous state. Sociality means the readiness to run towards the neighborhood’s best previous
state. The initial population is generated randomly. The dimension of the particle is decided by the problem that needs to be optimized. The PSO looks for optima by updating generations. The pseudo-code for the PSO is as follows (Sudha & Viji 2008) (Jeyarani et al. 2012).

```
INITIALIZE

Assign random values to each particle’s position and velocity in all dimension of the solution space

Set each particle’s personalBest to particle position

Set globalBest to randomized particle’s position

REPEAT

For each particle

Calculate particle’s updated velocity using Eq.(3.7)

Calculate particle’s position using Eq. (3.8)

Evaluate the fitness function to all particles and find the fitness value

If(fitness value corresponding to new position is better than the personalBest in the history)

Set personalBest as the new position

End For

Choose the particle’s position with the best fitness value in the population as the gBest

UNTIL(Stopping Criteria is met)
```
3.4 PROPOSED ENHANCED PARTICLE SWARM OPTIMIZATION ALGORITHM (EPSO)

Linearly Decreasing Weight Particle Swarm Optimization is introduced by Shi & Eberhart (1999). In the proposed EPSO algorithm, Linearly Decreasing Weight Inertia Factor $w_{LDW}$ is included in the velocity updation Equation (3.7) instead of $w$ (Ercan & Li 2013). The performance of Linearly Decreasing Weight Particle Swarm Optimization is effectively improved when compared to the standard PSO, because the global and local search capabilities of the particle are balanced. In the proposed algorithm, $w_{LDW}$ is the inertia weight that linearly decreases from 0.9 to 0.1 throughout the search process (Yin et al. 2006). The updated velocity for the proposed EPSO algorithm is calculated by using Equation (3.9).

$$v_{i_d}(t + 1) = w_{LDW}v_{i_d}(t) + c_1 r_1[pbest_{i_d}(t) - x_{i_d}(t)]$$
$$+ c_2 r_2[gbest(t) - x_{i_d}(t)]$$

(3.9)

Linearly Decreasing Weight Inertia Equation is written as,

$$w_{LDW} = w_{end} + (w_{start} - w_{end}) \beta$$

(3.10)

Where,

$$\beta = \frac{1}{1 + \left(\frac{itr \alpha}{Total\itr}\right)}$$

(3.11)

$\alpha = $ constant; $itr =$ current iteration; $Total\itr = $ maximum number of iterations

3.4.1 Fitness Evaluation

A fitness function is evaluated after the solution construction. The proposed algorithm allocates each VM to a host that provides the least increase of power consumption due to its allocation. This allows the
heterogeneity of resources by choosing the most power-efficient nodes first. The objective of this research work is to minimize the power consumption of the datacenter by efficient mapping of VMs. In order to achieve maximum power saving, the fitness function is used which is given in Equation (3.12).

\[ P_{DC} = \text{Min} \sum_{j=1}^{M} P_j = \text{Min} \sum_{j=1}^{M} \left[ \left( p_j^{\text{busy}} - p_j^{\text{idle}} \right) \ast \text{CPU}_{\text{host,j}} + p_j^{\text{idle}} \right] \] (3.12)

3.5 POWER AWARE ENHANCED PSO (PA-EPSO) METHODOLOGY FOR VM PROVISIONING

This section deliberates the design of the Power-Aware Enhanced PSO methodology for VM provisioning in a cloud environment. The key issue in designing a successful PSO algorithm is the solution representation as it describes a direct relationship between the problem domain and the particles in PSO (Tasgetiren et al. 2004).

3.5.1 Solution Representation

Here, \( N \) number of virtual machines is to be allocated to the \( M \) number of physical servers. In the cloud environment, the number of resources available is enormous, So \( N \ll M \). The particles are represented as “\( N \)” dimensional vector. Each dimension could be mapped to the discrete set of possible assignments limited to “\( M \)”.

Equation (3.12) is the fitness function of the optimization algorithm. Suppose the number of virtual machines ‘\( N \)” and those should be allocated to ‘\( M \)” physical hosts in the datacenter, the position vector of particle is represented as \( x_{ij}(t) = (x_{1i}, x_{2i}, \ldots, x_{Ni}) \). Here, \( t \) is the iteration, \( i \) is the \( i^{th} \) possible solution, and \( j \) depicts the serial number of virtual machine. If there occurs three virtual machine requests, then the vector \( x_{ij}(t) = (x_{i1}, x_{i2}, x_{i3}) = (3, 2, 1) \), it represents that VM1 is mapped with host 3, VM2 is to host 2 and VM3 to host 1.
3.5.2 Pseudo-code of proposed Power-Aware Enhanced PSO (PA-EPSO)

The proposed algorithm utilizes the resources in a power-aware manner using the EPSO. Algorithm 3.1, VM-Allocation gets the VM requests \( N \). For allocating VMs on the hosts in a power-aware manner, VM-Allocation algorithm invokes the PA-EPSO algorithm with the parameters such as dimension, resource dynamics, and power model. The PA-EPSO algorithm in each of its iteration moves through its neighbor searching for a power-efficient host. PA-EPSO initializes the position vector, velocity vector and all the simulation parameters as mentioned in Table (3.3). Each particle corresponds to a VM mapping that allocates given VMs to PMs. The particles are generated with respect to the number of VMs and PMs. Each particle is evaluated using the fitness function as given in the Equation (3.12) which minimizes the power consumption of the datacenter. Further, it finds the gbest and pbest. This iteration is repeated as per the constraints. Finally, it returns the position vector of the gbest to the VM-Allocation Algorithm. VM-Allocation Algorithm dispatches the workloads according to the gbest position vector sent by the PA-EPSO and it updates the information in the resource pool. The following pseudo-code depicts the PA-EPSO inspired virtual machine allocation algorithm.
Algorithm 3.1 VM-Allocation()

Input: VMlist, Hostlist, PowerModel
Output: Power-aware VM allocation

1. Get the VM requests from the cloud users
2. Get the resource information from the cloud information registry
3. repeat
4. {
5. Particledimension = N (No. of VM Requests)
6. Get the current resource dynamics from resource pool
7. Power-aware VM mapping = PA-EPSO(Particledimension, resource dynamics, power model)
8. Dispatch the workloads (Power-aware VMmapping)
9. Update the information in the resource pool
10. }until all VMs placed

Algorithm 3.2 PA-EPSO

Input: Particledimension, resource dynamics, power model
Output: Power-aware VM Mapping

1. Initialize the position vector $x_{ij}$
2. Initialize the velocity vector $v_{ij}$
3. Initialize all the simulation parameters as Table 3.3.
4. Evaluate the fitness using the Equation (3.12)
5. Find the pbest and gbest

6. repeat

7. {

8.  Update the $x_{ij}$ and $v_{ij}$ using the equation (3.8)&(3.9)

9. Evaluate the fitness using the Equation (3.12)

10. Update the pbest and gbest

11. } until the termination criterion is met

12. return $x_{ij}$ of gbest

3.6 SYSTEM ARCHITECTURE OF THE POWER-AWARE VM PROVISIONER

Figure 3.1 Architecture of the Power-Aware VM Provisioner

Figure 3.1 depicts the system architecture of the Power-Aware VM provisioner. Since cloud environment has a non-homogeneous hardware infrastructure, assigning VMs to PMs becomes an explicitly demanding task.
The central component that manages the allocation of virtual resources to the physical resources is known as the cloud scheduler. The cloud user submits the list of required VM specification through the cloud interface software. The proposed VM scheduler has two components such as VM provisioner and Dispatcher. Power-Aware Enhanced Particle Swarm Optimization algorithm (PA-EPSO) is used in VM provisioner for initial placement of the virtual machine in a power-aware manner. The dispatcher unit executes the VM to PM mapping given by the VM provisioner. The power monitor is responsible for estimating the power consumption of the cloud datacenter based on the resource utilization of the available hosts. The cloud resource information registry gets updated periodically with the resource utilization, power consumption and the performance of the hosts in the datacenter. This information is taken by the VM provisioner for initial placement of the virtual machines as well as for doing the consolidation.

3.7 RESULTS AND DISCUSSION

In this Section, the experimental setup and the performance of the proposed algorithm are analyzed.

3.7.1 Experimental Setup

To test the proposed algorithms, experiments are performed on an Intel core i3 CPU processors running at 2.27 GHZ speed with 1.87 GB RAM and MS Windows 7, 64-bit operating system. The cloud datacenter architecture is implemented using Matlab R 2010a environment. The cloud datacenter is simulated in the hierarchy level of physical machines and virtual machines. The datacenter consists of 200 physical machines. Each physical machine is configured with its own resource specification, with varying number of cores, memory, storage, speed, and power consumption. The real
data on power consumption for the hosts are taken from the results of the SPEC benchmark (Fourth Quarter SPEC Results 2009). SPEC presented the SPECpower ssj2008 benchmark, which examined the relationship of power, performance and power consumption for servers at different performance levels, spanning from 100% segments over a set period of time. Table 3.2 gives the benchmark data set for hosts. Ten batches of the virtual machine each containing 20 requests are generated and given as the input to the algorithm and the data for virtual machines is taken from the Amazon EC2 instances. From the hosts, a subset of hosts may have the capacity for fulfilling the requirements of the virtual machine. However, apart from that, this algorithm chooses the power-aware host for launching the virtual machine from the subset, because the power-efficiency of a host changes according to CPU’s power-efficiency and speed.

### Table 3.2  Benchmark Data set for Hosts

<table>
<thead>
<tr>
<th>Hardware Vendor</th>
<th>Test Sponsor</th>
<th>CPU description</th>
<th>Processor MHz</th>
<th>Chips</th>
<th>Cores</th>
<th>Total memory (GB)</th>
<th>Sub measurements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acer incorporated</td>
<td>Intel Xeon 2266</td>
<td>2 8 8</td>
<td>196</td>
<td>81.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acer incorporated</td>
<td>Intel Xeon 2266</td>
<td>8 32 32</td>
<td>748</td>
<td>260</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dell Inc</td>
<td>Intel Xeon 2400</td>
<td>2 8 8</td>
<td>172</td>
<td>62.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hewlett-Packard</td>
<td>Intel Xeon 2400</td>
<td>8 32 32</td>
<td>664</td>
<td>191</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IBM Corporation</td>
<td>Intel Xeon 2933</td>
<td>1 4 8</td>
<td>115</td>
<td>43</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC corporation</td>
<td>Intel Xeon 2267</td>
<td>2 8 8</td>
<td>176</td>
<td>64.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plat ‘home</td>
<td>Intel Xeon 2400</td>
<td>2 8 8</td>
<td>172</td>
<td>56.2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.7.2 Sensitivity Analysis

The parameter setting of the PA-EPSO is chosen based on sensitivity analysis. Sensitivity analysis is used to find variation in the performance of proposed algorithm. The control parameters of the proposed algorithm are swarm size, maximum number of iterations, and the acceleration coefficients. These parameters have been optimally tuned to find the best combination. Table 3.3 shows the parameters used for the algorithm.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swarm size</td>
<td>50</td>
</tr>
<tr>
<td>Dimension</td>
<td>50-250</td>
</tr>
<tr>
<td>Maximum Iterations</td>
<td>200</td>
</tr>
<tr>
<td>Run</td>
<td>10</td>
</tr>
<tr>
<td>Self-cognition coefficient c1</td>
<td>1.9</td>
</tr>
<tr>
<td>Social coefficient c2</td>
<td>1.6</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>4</td>
</tr>
<tr>
<td>r1, r2</td>
<td>U(0,1)</td>
</tr>
<tr>
<td>$w_{start}$</td>
<td>0.9</td>
</tr>
<tr>
<td>$w_{end}$</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Figure 3.2  Sensitivity Analysis for Variation in Swarm Size

Figure 3.2 shows the sensitivity analysis for variation in swarm size in terms of power consumption. From Figure 3.2, it is observed that the better performance achieved by the PA-EPSO for swarm size = 50.

Figure 3.3  Sensitivity Analysis for Variation in Self-Cognition Coefficient c1
Figure 3.3 shows the sensitivity analysis for variation in self-cognition coefficient $c_1$ in terms of power consumption. From Figure 3.3, it is observed that the better performance achieved by the PA-EPSO for $c_1=1.9$.

Figure 3.4  Sensitivity Analysis for Variation in Social Coefficient $c_2$

Figure 3.4 shows the sensitivity analysis for variation in social coefficient $c_2$ in terms of power consumption. From Figure 3.4, it is observed that the better performance achieved by the PA-EPSO for $c_2=1.6$. 
3.7.3 Performance Comparison of the Proposed PA-EPSO Algorithm with the Existing Algorithms

![Graph](image.png)

Figure 3.5  Convergence of the Proposed PA-EPSO Algorithm for VM = 50

Figure 3.5 shows the convergence of the proposed PA-EPSO algorithm and the existing SAPSO and StdPSO algorithms by varying the number of iterations. For this experiment, 200 hosts and 50 VMs are taken. From the Figure 3.5, it is observed that the proposed PA-EPSO algorithm converges at 50 existing SAPSO converges at 108 and StdPSO converges at 125.
Figure 3.6 shows the convergence of the proposed PA-EPSO algorithm and the existing SAPSO and StdPSO algorithms by varying the number of iterations for 200 hosts and 100 VMs. From the Figure 3.6, it is observed that the proposed PA-EPSO algorithm converges at 62, existing SAPSO converges at 112 and StdPSO converges at 110.
Figure 3.7  Convergence of the Proposed PA-EPSO Algorithm for VM =150

Figure 3.7 shows the convergence of the proposed PA-EPSO algorithm and the existing SAPSO and StdPSO algorithms by varying the number of iterations for 200 hosts and 150 VMs. From the Figure 3.7, it is observed that the proposed PA-EPSO algorithm converges at 67, existing SAPSO converges at 89, and StdPSO converges at 91.
Figure 3.8  Convergence of the Proposed PA-EPSO Algorithm for VM = 200

Figure 3.8 shows the convergence of the proposed PA-EPSO algorithm and the existing SAPSO and StdPSO algorithms by varying the number of iterations for 200 hosts and 200 VMs. From the Figure 3.8, it is observed that the proposed PA-EPSO algorithm converges at 67, existing SAPSO converges at 84, and StdPSO converges at 92.
Figure 3.9 shows the convergence of the proposed PA-EPSO algorithm and the existing SAPSO and StdPSO algorithms by varying the number of iterations for 200 hosts and 250 VMs. From the Figure 3.9, it is observed that the proposed PA-EPSO algorithm converges at 46, existing SAPSO converges at 106, and StdPSO converges at 86.

The proposed algorithm is run for 200 iterations. The number of virtual machines is varying from 50 to 250 while the number of servers is 200. As can be seen, all the three algorithms minimize the power consumption over iterations. From the results, it is observed that the convergence of the proposed algorithm PA-EPSO is faster than the existing SAPSO and StdPSO. Similar to the case reported in Figure 3.5, in other figures also the same pattern is observed with the PA-EPSO converging to a good solution compared with the existing algorithms SAPSO and StdPSO.
Figure 3.10 Performance Comparison of the Proposed and Existing Algorithms by Varying Number of Iterations in terms of Power Consumption

Figure 3.10 gives a comparison of three algorithms in terms of power consumption by varying the number of iterations and the population size is set to 50. In the PA-EPSO algorithm, the power consumption obtained initially is 10730.94 Watts. When the number of iterations increases, the power consumption converges to the minimum value of 9639.816 watts at the 70th iteration and remains the same until last iteration. In the SAPSO, the power consumption initially obtained is 11166.48 and when the number of iterations increases the minimum power consumption reaches to 10584.18 at 90th iteration. In the StdPSO, the power consumption initially obtained is 11109.85 and when the number of iterations increases the minimum power consumption reaches to 10730.94 at 90th iteration. The results show that the proposed PA-EPSO algorithm achieves minimum power consumption while compared with SAPSO and StdPSO.
Figure 3.11 Performance Comparison of the Proposed PA-EPSO Algorithm with the Existing Algorithms SAPSO and StdPSO in terms of Power Consumption

Figure 3.11 shows the comparison of power consumption in the proposed PA-EPSO with standard PSO and existing SAPSO. The proposed PA-EPSO allocation algorithm is compared against the Self-Adaptive Particle Swarm Optimization (SAPSO) and the Standard PSO (StdPSO) in terms of power consumption. The total number of iterations is set to 200 and initial population size is 50. From the results obtained, it can be concluded that the proposed PA-EPSO performs better than the existing SAPSO and the conventional PSO in terms of power consumption. This is because the global and local search capabilities of the particle are balanced by inertia weight wLDW. The tuning of control parameters has been carried out proficiently which support to provide the optimal solution.
Table 3.4 Performance Comparison of the Proposed PA-EPSO Algorithm with the Existing Algorithms in terms of Power Consumption

<table>
<thead>
<tr>
<th>No. of Hosts</th>
<th>No. of VMs</th>
<th>Proposed PA-EPSO (Watts)</th>
<th>SAPSO (Watts)</th>
<th>StdPSO (Watts)</th>
<th>Rank of the Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Proposed PA-EPSO</td>
<td>SAPSO</td>
<td>StdPSO</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Proposed PA-EPSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SAPSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>StdPSO</td>
</tr>
<tr>
<td>200</td>
<td>50</td>
<td>2037.732</td>
<td>2550.976</td>
<td>2644.323</td>
<td>1</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>4293.716</td>
<td>5304.12</td>
<td>5387.618</td>
<td>2</td>
</tr>
<tr>
<td>200</td>
<td>150</td>
<td>7381.057</td>
<td>8429.372</td>
<td>8440.062</td>
<td>3</td>
</tr>
<tr>
<td>200</td>
<td>200</td>
<td>9639.816</td>
<td>10584.18</td>
<td>10730.94</td>
<td>1</td>
</tr>
<tr>
<td>200</td>
<td>250</td>
<td>13308.8</td>
<td>14247.82</td>
<td>14438.96</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Proposed PA-EPSO / SAPSO</strong></td>
<td><strong>13.42%</strong></td>
<td><strong>Proposed PA-EPSO / StdPSO</strong></td>
<td><strong>14.75%</strong></td>
</tr>
</tbody>
</table>

Table 3.4 shows the comparison of power consumption of proposed PA-EPSO algorithm with the existing SAPSO and StdPSO algorithms. The results show that the proposed PA-EPSO algorithm outperforms the SAPSO and StdPSO algorithms in terms of power consumption. From Table 3.4, it is observed that the proposed PA-EPSO algorithm achieves an average of 13.42% of power savings over SAPSO algorithm and 14.75% of power saving over StdPSO algorithm.

3.8 SUMMARY

An evolutionary technique, Enhance Particle Swarm Optimization (EPSO), is proposed for solving the VM allocation problem in the cloud computing environment. The performance of the proposed PA-EPSO, existing SAPSO and StdPSO has been validated for varying the
number of VMs. The PA-EPSO algorithm generated optimal VM mapping with the objective of minimizing the power consumption. It has been shown through different trials that the PA-EPSO algorithm is found to be better than the existing SAPSO and standard PSO in terms of minimizing the power consumption. The proposed PA-EPSO algorithm minimizes the power consumption of the datacenter on an average of 13.42% over existing SAPSO and 14.75% over standard PSO algorithm. In order to get further improvement in power efficiency, power-aware dynamic consolidation is proposed for eliminating idle power wastage which is dealt in the next chapter.