Abstract
Recent advancement in public cloud allow the providers to rapidly diversify their products and pricing plans. To increase cloud provider’s profit, tasks are outsourced from internal cloud to External Cloud (EC). The main objective is to maximize the internal cloud utilization and minimize the cost of outsourced task to external cloud. This problem is formulated as an integer programming model and is solved using cuckoo driven Particle Swarm Optimization (CS-PSO) approach. This perform the local search more efficiently and it avoids the local optima problem of PSO. The task utilization runtime and the cost for each task is calculated using cloudsim simulator. Experimental results shows that the average profit obtained by this approach is higher than the standard PSO and Self adaptive Learning PSO (SLPSO) for problems of non trivial size.

Keywords: Cuckoo Search, External Cloud (EC), Hybrid Cloud, Local Optima, Particle Swarm Optimization (PSO)

1. Introduction
Cloud computing provides large scale computational resources to the users in on-demand, pay-as-per use, reliable and low-cost services. Cloud provide a variety of services. Infrastructure as a service serves as a foundation of other high level services such as PaaS and SaaS. IaaS provider allows the users to rent the resources such as CPU, memory, virtual machines based on different pricing models. The major challenge for the IaaS provider is how to effectively schedule their resources to guarantee the promised QoS. When an IaaS provider receives a massive request, which it is not able to satisfy with its resources, then it can either over-purchase the resources in advance or integrate different clouds (cloud federation). But both these solutions are not efficient as the former is not cost efficient while the latter is hardly feasible in practice.

To tackle this problem, we propose a resource allocation framework in which the IaaS provider can utilize the External Cloud (EC). IaaS provider has its own cloud called private cloud and external cloud to which the tasks are outsourced, when the local resources are not sufficient. Since each task is associated with a strict deadline, this problem can be considered as a Deadline Constrained Task Scheduling (DCTS) problem.

2. Related Work
Many researches and schemes have been applied to effectively allocate the resources and schedule the tasks in a single cloud environment. Mohana R.S proposed Position Balanced Particle Swarm Optimization (PB-PSO) technique in which each task is considered as a particle. The tasks location and movement of allocated resources in the cloud virtual machine is considered as the position and velocity of the particle. The local best and global best values are computed based on the fitness value. Based on which the task best response time and price of allocation in cloud is calculated.
Dhingra and Paul\cite{2}, suggested Bacterial Foraging Optimization (BFO) technique in which the migration of virtual machines in done in two phases. Energy conservation in cloud resource allocation is done using Modified Best Fit Decreasing (MBFD) and Bacterial Foraging algorithm. Umarani\cite{3}, proposed an ACO based algorithm for task scheduling in cloud computing. ACO algorithm consists of two phases namely the training phase and the testing phase. In training phase a task set model is generated and in testing phase new tasks are attempted to fit into the task model.

Nathani et.al.\cite{4} suggested an algorithm to schedule the deadline constrained task. This algorithm uses swapping and backfilling method to accept more tasks, but it considers only a limited number of resources, thereby rejecting the users request which fails to satisfy user expectations. Zhao et al.\cite{5} suggested a genetic algorithm to schedule the independent and divisible tasks in a heterogeneous environment. Though the genetic algorithm attempts to optimize the resources, it fails to provide global optimization. Zhang et al.\cite{6} proposed a PSO based task scheduling algorithm on a grid environment, in order to reduce completion time. Results have shown that the PSO-based method outperforms the GA approach. All the above algorithms considers only allocation in a single cloud, they do not consider multiple cloud environment. These solutions will reject the users tasks when the resources are not sufficient.

To make the cloud more scalable and reliable, task scheduling among multiple clouds must be considered. Bossche et al.\cite{7} proposed mathematical model in which tasks are outsourced to public cloud so as to minimize the cost of the cloud provider. This model does not scale for large number of tasks. Xingquan\cite{8}, proposed a Self-adaptive Learning PSO- based scheduling method (SLPSO) for scheduling the tasks in a hybrid cloud using four velocities updating strategies. However, this increases the complexity of the PSO algorithm.

### 3. Solution Framework

Our solution framework consist of Private Cloud and Elastic Cloud (ECs) as shown in Figure 1. Private cloud is the primary cloud that contains its own resources and Elastic cloud refers to other clouds to which the tasks are outsourced. In the Private Cloud, the users request is received through the user interface. The Request manager collects and manages all the users request. The Resource monitor monitors the resource pool such as CPU pool, storage pool, memory pool, etc..<br>Figure 1. Hybrid Cloud Framework.

The cloud interface acts as an interface to both private cloud and Elastic cloud. It collects the pricing model of ECs and sends the tasks to ECs. The Scheduler schedules the task between the private cloud and ECs by collecting the scheduling information from the Request manager, Resource monitor and cloud interface. The scheduler then decides whether the task is to be allocated to private cloud or one of the ECs. If the task is allocated to an EC then its pricing model is sent to the scheduler through the Cloud interface.

### 4. Problem Description

The proposed solution usually considers batch workload, i.e. a bag of independent tasks which can be used in scientific computations, sensor data analysis, big data analysis, data processing, image or data processing etc. These tasks are usually independent and parallel. In our model, tightly coupled tasks, online transaction processing, complex computations is not taken into consideration.

Consider an application which has a large number of parallel and independent tasks. Each task has a strict deadline before which the task has to be completed. Each task is capable of executing in only one VM instance. Once the task is assigned to one VM, it cannot be interrupted.

Let \( CP = \{CP_1, CP_2, \ldots, CP_n\} \) and \( VM = \{VM_1, VM_2, \ldots, VM_I\} \) be the set of cloud providers and set of VM instances respectively. Assume \( CP_1 \) is the private or Internal Cloud and \( CP_2, \ldots, CP_n \) are the Elastic clouds. Consider \( A = \{a_1, a_2, \ldots, a_w\} \) is the set of applications where each application a consist of a task set \( T = \{t_{j1}, t_{j2}, \ldots, t_{jT}\} \) with a strict deadline \( d \) and runtime \( r \).

The main objective is to allocate the ‘w’ applications to \( CP_k \) where \( k \in \{1, 2, \ldots, n\} \) so as to maximize the profit of the CP1. This can be formulated as
\[ \text{Profit} = \sum_{j=1}^{W} \sum_{v=1}^{I} T_j b_{jv} p_v r_j - \sum_{j=1}^{W} \sum_{l=1}^{T_j} \sum_{v=1}^{I} Y_{jlk} b_{jvl} c_{kv} r_j \]  

(1)

Where,

\[ \sum_{k=1}^{n} y_{jlk} = 1, \quad \forall j \in \{1, 2, \ldots, W\}, \quad l \in \{1, 2, \ldots, T_j\} \]

\[ \sum_{s=1}^{d_{j}} z_{jls} = y_{j1} r_{j}, \quad \forall j \in \{1, 2, \ldots, W\}, \quad l \in \{1, 2, \ldots, T_j\} \]

\[ s_{jl} \geq 1, \quad \forall j \in \{1, 2, \ldots, W\}, \quad l \in \{1, 2, \ldots, T_j\} \]

\[ s_{jl} \geq d_{j} - r_{j} + 1 \quad \forall j \in \{1, 2, \ldots, W\}, \quad l \in \{1, 2, \ldots, T_j\} \]

\[ \sum_{j=1}^{W} \sum_{l=1}^{T_j} \sum_{v=1}^{I} Z_{jvl} b_{jvl}cpu_{v} \leq \text{Total} - \text{cpu}, \forall s \in \{1, 2, \ldots, S\} \]

\[ \sum_{j=1}^{W} \sum_{l=1}^{T_j} \sum_{v=1}^{I} Z_{jvl} b_{jvl}mem_{v} \leq \text{Total} - \text{mem}, \forall s \in \{1, 2, \ldots, S\} \]

Table 1. Parameter Description.

| Parameters | Description                  |
|------------|------------------------------|
| I          | Number of VM instances       |
| N          | Number of Cloud Providers    |
| W          | Number of Applications       |
| d_{j}      | Deadline of the jth Application |
| r_{j}      | Runtime of jth Application   |
| p_{v}      | Price of the Vth VM type in CP_{1} |
| c_{kv}     | Cost of the Vth VM type in CP_{k} |
| S          | Maximum Deadline of the Application |
| T_{j}      | Number of Tasks in jth Application |
| cpu_{v}    | Number of CPUs in each VM of CP_{v} |
| mem_{v}    | Amount of memory in each VM of CP_{v} |
| total_cpu  | Total Number of CPU in CP_{1} |
| total_mem  | Total Amount of memory in CP_{1} |

Table 2. Decision Variables.

| Decision Variables | Description |
|--------------------|-------------|
| b_{jv}             | Binary Decision Variable |
|                      | B_{jv} = 1, if jth application uses vth VM instance; otherwise 0 |
| y_{jlk}            | Binary Decision Variable |
|                      | Y_{jlk} = 1, if task t_{jl} is allocated to CP_{1}; otherwise 0 |
| z_{jvl}            | Binary Decision Variable |
|                      | Z_{jvl} = 1, if task t_{jl} is allocated to the time slot of v in CP_{l} |
| s_{jl}             | Integer Decision Variable |
|                      | Start time of task t_{jl} |

5. Standard PSO

PSO is a random optimization technique which is developed based on the flocking and schooling patterns of the swarm. PSO usually optimizes a problem by considering a population of candidate solutions.

Each individual in the swarm is represented as a particle in a D-dimensional space. Each particle is represented by its position \(X_{i}(t)\) and velocity \(V_{i}(t)\). The particle's personal best position is given by \(P_{i} = \{p_{i1}, p_{i2}, \ldots, p_{iD}\}\) and the Global best of all particles is given by \(G = \{g_{1}, g_{2}, \ldots, g_{D}\}\).

The algorithm starts with a set of particles whose position and velocity are initialized randomly. Each particle's fitness value is calculated. The fitness value of each particle is recorded as its personal best (pbest) value. The best fitness value, obtained so far by the entire particles is considered as the global best (gbest). Once these two values are obtained, the each particle will updates its position and velocity using the two equations (2) and (3).

\[ V_{id}(t) = w_{id}(t) + c_{1} r_{1} [X_{id}(t) - p_{id}(t)] + c_{2} r_{2} [X_{id}(t) - g_{id}(t)] \]  

(2)

\[ X_{id}(t) = X_{id}(t) + V_{id}(t) \]  

(3)

Where \(c_{1}\) and \(c_{2}\) are the Cognitive factors, \(r_{1}\) and \(r_{2}\) are the values randomly chosen between 0 and 1 and \(w\) is the cognitive weight factor.

6. Cuckoo Search

Cuckoo search is a search algorithm based on the natural behavior of blood parasitism of cuckoo birds. Cuckoo birds
have an aggressive breeding behavior. Cuckoo birds are not in the habit of building their own nest instead they lay their eggs in some other host birds. The host bird is unaware of the cuckoo eggs as it imitate the color and pattern of the host bird. Cuckoo eggs are hatched earlier than the host eggs. Once the host birds discover that the eggs are not their own, they will either simply throw these foreign eggs away or destroy these eggs and build the nest elsewhere.

These fascinating breeding behaviors of cuckoos will help in many optimization problems. Cuckoo search is based on three assumptions namely

- Each cuckoo is capable of laying only one at a time and these eggs are randomly placed into any nest.
- The nest containing eggs of better quality will be passed to the subsequent iteration.
- The quantity of host nest is static and cannot be changed.

Step (0): Initialization
Objective function \( f(x) \), \( x = \{x_1, x_2, \ldots, x_d\} \)
Generate an initial population of \( n \) host nests \( x_i, i = 1, 2, \ldots, n \)

Step (1): Updation loop
While (Stop Criterion)
Select a cuckoo bird (i) randomly using levy Flights
Find its Fitness function \( F_i \)
Select a nest (j) randomly among \( n \)
If (Fitness\(_i\) > Fitness\(_j\))
Replace \( j \) by \( i \)
End
A Probability (\( P_a \)) of worst nest is removed.
Build the new nest
Record the best solutions
Sort these solutions and find current best
End While
Pass the best solution to next iteration
End

Pseudocode for cuckoo search algorithm

7. Proposed Algorithm

Step 1: Particle Representation
Each application has many tasks and these tasks should be mapped to the particles in PSO algorithm. The initial particle with its position and velocity is produced randomly using these two equations

\[
 x^i_t = x_{\text{min}} + (x_{\text{max}} - x_{\text{min}}) \times r \\
 v^i_t = v_{\text{min}} + (v_{\text{max}} - v_{\text{min}}) \times r
\]  (4)

where \( x_{\text{min}} = -0.4, x_{\text{max}} = 4.0 \), \( r \) is a random value between 0 and 1.

All the above obtained values are continuous values. Convert these values into discrete values using Small Position Value (SPV) rule as shown in Table 3.

Step 2: Cost Evaluation Function
The permutation of tasks is optimized to increase the profit of \( CP_i \). An evaluation function serves as a fitness function of each particle in the swarm. The pseudocode is as follows

- **Initialization**

\[
 \text{Total}_\text{cost}=0; \text{Avail}_\text{cpu}=\text{Total}_\text{cpu}; \text{Avail}_\text{mem}=\text{Total}_\text{mem};
\]

- **Calculate the total income**

\[
 \text{Total}_\text{income} = \sum_{l=1}^{D} \sum_{v=1}^{B} b_{app(l)} v_{p_{app(l)}}
\]

- **Calculate the total cost**

Arrange the task in an ascending order according to the rank values of permutation tasks.
Let the \( l \)th task start time be \( s_t \)
For \( l = 1 \) to \( D \)
While \( (s_t \leq d_{app(l)} - r_{app(l)} - 1) \)
\( PC=true \)
If \( \left( \sum_{v=1}^{B} b_{app(l)} v_{p_{app(l)}} \geq \text{Avail}_\text{cpu} \text{ or } \sum_{v=1}^{B} b_{app(l)} v_{p_{app(l)}} \geq \text{Avail}_\text{mem} \right) \)
\( PC=false \)
End If
If (PC=true)
Calculate the cost for a task using
\[
 \text{Cost}_i = \sum_{v=1}^{B} C_{1v} b_{app(l)} v_{p_{app(l)}}
\]
Update the \( \text{Avail}_\text{cpu} \) and \( \text{Avail}_\text{mem} \)
End If
End While
If (PC =false)
Select the elastic cloud and Calculate the cost using
\[
 \text{Cost}_i = \sum_{v=1}^{B} C_{(EC)v} b_{app(l)} v_{p_{app(l)}}
\]
End If

Table 3. Particle Encoding.

| Dimension | 1   | 2   | 3   | 4   | 5   | 6   | 7   |
|-----------|-----|-----|-----|-----|-----|-----|-----|
| Position Value | 0.1587 | 3.6189 | 2.3824 | 0.0292 | 0.8254 | 2.0063 | 3.8130 |
| Permutation vector | 2   | 6   | 5   | 1   | 3   | 4   | 7   |
Total\_cost = Total\_cost + cost, 
End For 

(4): Profit = Total\_income - Total\_cost

Pseudocode For Evaluating Fitness Function

Step 3: Cuckoo Driven PSO based Approach

Step (0) Initialization
Initialize particle's position and velocity $X = (X_{i1}, X_{i2}, \ldots, X_{iD})$ and $(V_{i1}, V_{i2}, \ldots, V_{iD})$ respectively.
Set Iteration = 0 
Evaluate the fitness values for each particle $F = (f1, f2, \ldots, fp)$.
Set each particle's current position as its pbest.
Set the best one among all the particle's as its gbest

Step (1) Local search and updating loop
While (t<MaxGeneration) 
Select a cuckoo bird (i) by Levy flights randomly and maintain its pbest value 
Calculate its fitness function $F_i$ 
Select a nest (j) randomly among n nests 
If(Fitness$_i$>Fitness$_j$)
Replace j by i.
End
Update Cuckoo Position and velocity using equation 2 and 3.
Remove a of (p$_a$) worse nests and build new ones.
Record the best solution and sort those solutions 
Calculate the current Best
EndWhile

Step (2) Termination condition
If the termination condition, not reached, go to Step (1), otherwise end

Pseudocode For Cuckoo Driven PSO Approach

8. Experimental Results

The proposed Cuckoo Search driven PSO (CS-PSO) scheduling approach is compared with standard PSO, SLPSO. The effectiveness of our approach is verified using three problem instances. Instance 1 consists of 8 applications, instance 2 consist of 5 applications and instance 3 of 10 applications. The parameters are shown in the table 4, 5, and 6.

The graphical results depicted in Figure 2 shows that the average profit obtained by CS-PSO is higher than the SPSO and SLPSO.

9. Conclusion and Future Work

In this paper, a framework for resource allocation in a hybrid cloud is proposed. This resource allocation problem is solved using a cuckoo driven PSO approach. By combining cuckoo with PSO algorithm the search space gets
increased and it effectively obtains better solutions, thereby avoiding the local optima problem of PSO. The proposed solution guarantees the QoS requirements and increase the IaaS providers benefit. Experimental results show that it is better than standard PSO and SLPSO for larger problem size. In future some other optimization algorithms can be combined with PSO to improve further performance.

10. References

1. Bhardwaj S, Jain L, Jain S Cloud computing: A study of infrastructure as a service. International Journal of engineering and information Technology. 2010; 2(1):60–3.
2. Buyya R. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation computer systems. 2009; 25(6):599–616.
3. Breitgand D, Marashini A, Tordsson J. Policy-driven service placement optimization in federated clouds. IBM Research Division.2011:1–10.
4. Mohana RS. A position balanced parallel particle swarm optimization method for resource allocation in cloud. Indian Journal of Science and Technology. 2015 Feb; 8(S3):182–8.
5. Dhingra A, Paul S. Green Cloud: Heuristic based BFO Technique to Optimize Resource Allocation. Indian Journal of Science and Technology. 2014 May; 7(5):685-691.
6. Umarani Srikanth G, Uma Maheswari V, Shanthi AP, Sivamoney A Task Scheduling Model. Indian Journal of Science and Technology. 2015; 8(S7):33–42.
7. Nathani A, Chaudhary S, Somani G. Policy based resource allocation in IaaS cloud. Future Generation Computer Systems.2012; 28(1):94–103.
8. Zhao C. Independent tasks scheduling based on genetic algorithm in cloud computing, 5th International Conference on Wireless Communications, Networking and Mobile Computing (WiCom’09); 2009. p. 1-4.
9. Zhang L. A task scheduling algorithm based on PSO for grid computing. International Journal of Computational Intelligence Research. 2008; 4(1):37–43.
10. Van den Bossche R, Vanmechelen K, Broeckhove J. Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads. 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD); 2010. p. 228–35.
11. Zuo X, Zhang G, Tan W, Wei T. Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud. IEEE Transactions on Automation Science and Engineering. 2014; 11(2):564–73.