Computer Science and Information Systems 19(2):679–707 https://doi.org/10.2298/CSIS210512059P

PE-DCA: Penalty Elimination Based Data Center Allocation Technique Using Guided Local Search for IaaS Cloud

Sasmita Parida[1,2], Bibudhendu Pati[1], Suvendu Chandan Nayak[2], Chhabi Rani Panigrahi[1], and Tien-Hsiung Weng[3]

[1] Department of Computer Science,
Rama Devi Women's University, India
sasmitamohanty5@gmail.com
patibibudhendu@gmail.com
panigrahichhabi@gmail.com
[2] Department of Computer Science and Engineering,
Gandhi Institute for Technological Advancement,
BPUT, Bhubaneswar, India
suvendu2006@gmail.com
[3] Science and Information Engr. (CSIE),
Providence University, Taiwan
thweng@gm.pu.edu

Abstract. In Cloud computing the user requests are passaged to data centers (DCs) to accommodate resources. It is essential to select the suitable DCs as per the user requests so that other requests should not be penalized in terms of time and cost. The searching strategies consider the execution time rather than the related penalties while searching DCs. In this work, we discuss Penalty Elimination-based DC Allocation (PE-DCA) using Guided Local Search (GLS) mechanism to locate suitable DCs with reduced cost, response time, and processing time. The PE-DCA addresses, computes, and eliminates the penalties involved in the cost and time through iterative technique using the defined objective and guide functions. The PE-DCA is implemented using CloudAnalyst with various configurations of user requests and DCs. We examine the PE-DCA and the execution after-effects of various costs and time parameters to eliminate the penalties and observe that the proposed mechanism performs best.

Keywords: Cloud Computing, Data Center, Allocation, Penalty, Meta-heuristic, Guided Local Search.

1. Introduction

Data centers are located through an assortment of topographical regions in cloud computing. As per the user request, the cloud offers types of assistance utilizing the pay-per-use model. Cloud computing offers services through inter-process correspondence between various server farms[1]. For proficient inter-process correspondence, a center level is essential among the clients and cloud providers. The middle person is answerable for simple organization, allotment, and cloud administrations' executives to

the cloud clients. Presently, we essentially center around end clients' vicinity, so service providers guarantee clients' most extreme fulfillment with all availability of assets [2], [3]. Because of the rapid increase in the number of users in the cloud, administrations' requests are rapidly expanded. Accordingly, the cloud upgrades data centers' dependability and accessibility to offer independent services, just as heterogeneous clients [4]. The best DC selection is to oblige solicitations of a specific client. Furthermore, adjusting the heaps among the DCs ignoring the expense is also a monotonous issue [5].

The allocation of on-demand resource allocation initiates a few pre-requisite steps. It must be followed for better resource allocation, such as request processing, searching for data centers, allocating on-demand resources, computing, monitoring, and releasing resources [6]. The request processing phase analyzes the on-demand parameters such as the number of users, number of cluster nodes, request size, the demand of storage, CPU, memory, number of processors, type of operating systems, and several others on the cloud service provider [7], [8] then locating suitable data centers to fulfill the user requirements with minimum resources where the requests can be executed [9]. The network latency has much more impact in allocation which can be reduced if multiple instances will be deployed near by the users [10]. However, the searching mechanisms consider the basic parameters related to cost, time, Service Level Agreements (SLAs), and Quality of Service (QoS) [11], [12]. The resource allocation process occurs, keeping task scheduling and load balancing and initiating the computing process. The monitoring process keeps track of the computing process and resources; whenever the computing process is over, the allocated resources get released and ready to further allocation [13]. Among all these processes, the searching of data center processes favourably impact allocation. Eventually, it is a tedious task during peak hours to refer to the suitable DCs. Thus, the selection of a suitable DC for the on-demand request is a challenging aspect. Service broker policy routes the user requests after finding suitable DCs for resource allocation and keeps load balancing [14], [15].

In the last decades, various meta-heuristic and optimization-based DC allocation mechanisms have been proposed. These mechanisms target in optimizing cost and time, managing SLA and QoS by computing optimal DC lists. While computing the optimal DC list, researchers have focused less on the importance of searching techniques. Assume is the set of solution (DC list) computed through some techniques for a set of variables (user request) . Let f(x) computes the optimal solution si for user request RJ. While considering multiple parameters may not be an optimal solution for RJ. The optimal solution for RJ could be possible if we compute the penalty associated with the parameter set P. In cloud computing, if the data center DCi is selected for Rj, then the penalty for users' set for the parameters needs to be computed.

Motivation and Contribution. The DC allocation techniques need the discussion of search techniques. Local Search (LS) can be considered as the right solution with less time. However, out of all neighbors present in the search space, LS can be trapped as local optima- position. Over the year, different approaches are suggested for the improvement of the LS effectiveness. Simulated Annealing (SA), Tabu Search (TS), and Guided Local Search (GLS) are providing the supports to improve LS rather than the local optimum. The GLS technique is a meta-heuristic and a global optimization algorithm that utilizes an embedded LS algorithm. It is an expansion to the LS

algorithm, and for example, Hill Climbing is comparable in the system to the Tabu Search calculation and the Iterated Local Search calculation.

GLS is a penalty based meta-heuristic searching mechanism to solve issues related to local minima due to augmented objective function. GLS is used to deal with combinatorial optimization problems by improving efficacy of local search process. It calculates utility for each penalized feature. The basic aim is to assign penalties to all those features in the search space having high cost function values with maximum utility. While searching the suitable DC for allocation, the local search does not consider the penalty and utility. The local search techniques compute the searching with the constraints. The DC selection requires multiple parameters to select the suitable one. In case of random DC allocation, the penalty may be more with less utility. So the work suggests implementing GLS technique in the DC allocation mechanism.

In this work, we propose a new DC allocation technique named as PE-DCA, using the meta-heuristic search technique GLS. The PE-DCA mechanism allocates suitable DCs for the on-demand user requests by evaluating the penalty associated with time and cost metrics. The total cost of the VM is minimized, and the response time and processing time are reduced. The contributions of this work are mentioned as follows.

- Propose a novel meta-heuristic DC allocation mechanism for on-demand resources.
- Address the penalty associated with searching for a suitable DC.
- Formulate and discuss the proposed GLS based DC allocation technique (PE-DCA)
- Study the performance of the proposed mechanism through the java-based simulation tool called CloudAnalyst.
- Compare the performance parameters such as total cost, response, and processing time with the existing techniques.

The remainder of this work is organized as follows. Section 2 discusses the formulation of the problem statement and discusses the importance of the searching technique in allocation, while the background of the GLS technique is highlighted in Section 3. The detailed presentation of the proposed PE-DCA mechanism model and algorithm is discussed in Section 4. The simulation and the performance results are discussed in Section 5 to present the importance of PE-DCA in allocation. Section 6 presents the performance comparison results with the existing mechanisms for allocation in clouds. The related work for resource allocations are presented in Section 7, and finally Section 8 summarizes the contribution of the proposed work and the future directions.

2. Problem Statement Formulation

This section asserts DC allocation for on-demand resources for which the work is proposed. The importance of searching techniques and their improvement is addressed to select suitable DC. It also presents how the searching approaches can be improved to find globally optimal solutions from the local optimal solutions. The Table.1 presents the symbols and notations with descriptions used in this proposed work.

Table 1. Symbols and Notations

Symbols	Explanation
DC	Data center
PU	Physical machine Unit
VM	Virtual machine
R	Region
Ti	Time
C	Network constant
DTcost	Data transfer cost
VM_{total}	Total VM cost
VM_{cost}	VM cost
ST_{cost}	Storage cost
M_{cost}	Memory cost
α	Regularization parameter
ND_{Time}	Network delay
Res_{Time}	Response time
Req_{in_time}	Request initiated time
D_R	Data size per request in bytes
$BW_{avaliable}$	Available bandwidth
N_{dl}	Network latency due to delay
η	Data center request size
d	Delay parameter
Pro_{time}	Processing time
$RVM_{capacity}$	Request VM capacity
DC_{load}	Load of DC
$DC_{capacity}$	Capacity of DC

Cloud infrastructure is made up of various DCs, and all DCs are integrated with varying configurations and geographical locations are shown in Figure 1. These configuration parameters need to be considered in DC allocation. Let $R = \{R_1, R_2, \ldots R_n\}$ be the set of regions, $DC = \{DC_1, DC_2, \ldots DC_m\}$ be the set of DCs and $\forall DC_i \in R_j$, where $i, j \in n$. Each DC_i consists of a set of physical machine units (PU): $DC_i = \{PU_1, PU_2, \ldots, PU_n\}$ and PU_i represents with a set of VMs: $PU_i = \{VM_1, VM_2, \ldots, VM_n\}$. A set of objective function $F = \{F_1, F_2, \ldots, F_z\}$ can be enhanced. In general, time and cost parameters have an essential role in the resource allocation mechanism. A data center matrix DC_i can be computed with time and cost given in Eqn. (1).

$$DC_i(T_i, C_i) = \begin{bmatrix} T_1C_1 & \cdots & T_1C_n \\ \vdots & \ddots & \vdots \\ T_nC_1 & \cdots & T_nC_n \end{bmatrix} \quad (1)$$

The regions are separated geographically, so the $DC_i(T_i, C_i)$ can vary due to the parameter α, where α stands for data transfer cost and network constant C and is given as in Eqn. (2).

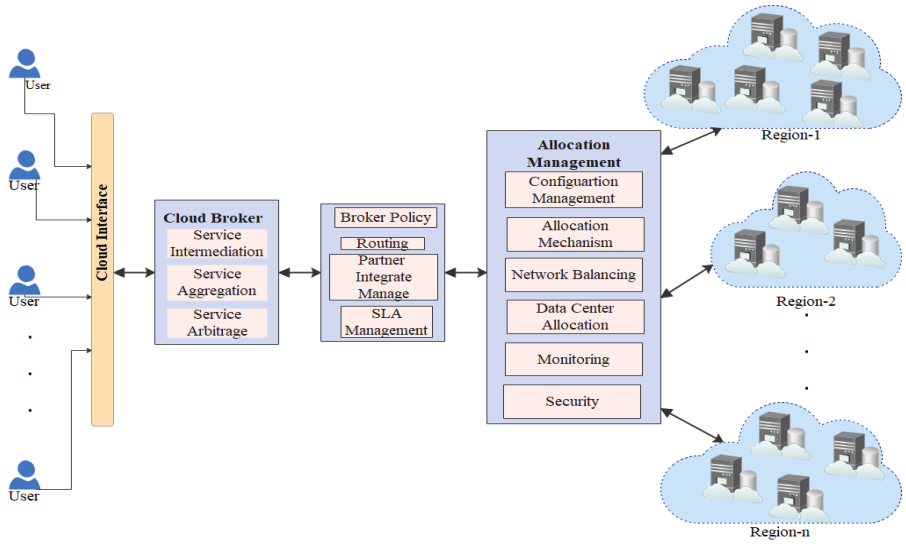

Fig. 1. Cross-cutting of Data Center Allocation

$$R_i(DC_i) = \alpha * DC_i(T_i, C_i) + C \qquad (2)$$

Consider the set of user requests $U = \{U_1, U_2, \ldots, U_m\}$. The on-demand resource of user Ui can be available in multiple data centers in various locations. The challenge is to select the best data center for Ui. All the optimal data centers are not the global optimal data center for Ui due to penalty and utility. The penalty and utility are reciprocal to each other and are influenced by α and C for a data center. It is essential to search the global optimal data center from the local optimal data center list and is shown as in Figure 2.

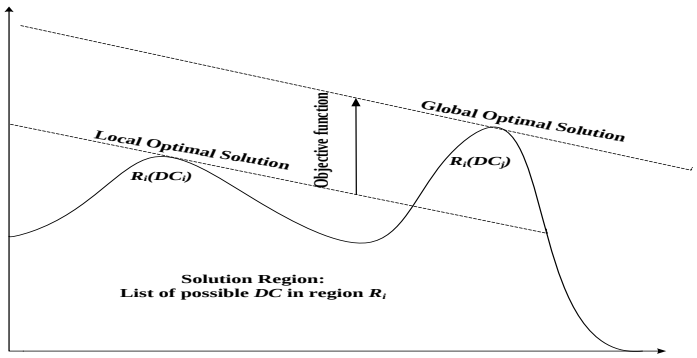

Fig. 2. Optimal Solution Approach

At this point, we have to follow combinatorial optimization as defined by a pair (D,c), where

D = the set of feasible solutions.

c = objective function, which maps each member with D.

The objective is to find a solution d in D that minimizes the objective function c and is formulated as:

$$Min(d)\ \forall\ d \in D \qquad (3)$$

Now, we define a neighborhood N for the pair (D,c) that is covers D and is given by

$$N \mid D \rightarrow 2^d$$

Consider *N(d)* as the neighborhood of *d*, and it has all possible solutions which are reachable from *d* with a single move. Here, the movie is an operator that transforms one form of solution to another with some modification.

Let *m* be the solution called as a local minimum with respect to *N(d)* if and only if

$$c(m) \leq c(d) \mid \forall d \in N(d) \qquad (4)$$

In the process of local searching primarily, we are minimizing the cost function *f* iteratively in various steps, and the current solution *m* being replaced by a solution *y* such that

$$y=c(d) < c(m) \mid d \in N(d) \qquad (5)$$

3. Guided Local Search (GLS)

This section presents the meta-heuristic-based GLS technique. It discusses the derived cost function for GLS in DC allocation, and the penalty function is formulated to find the different penalties associated with various allocation parameters.

To apply GLS, solution features must be characterized for the given problem. The solution features are characterized to recognize arrangements with various qualities, so areas of comparability around local optima can be perceived. The arrangement highlights' decision relies upon objective function and a limited degree on the local search algorithm[16]. The main idea to design the GLS algorithm system is to utilize punishments to empower a local search method to get away from local optima and find the global optima [17]. A local search algorithm is executed until it stalls out in local optima. The local optima highlights are assessed and punished, the consequences of which are utilized in an enlarged cost function utilized by the local search methodology. The local search is rehashed on various occasions utilizing the last local optima found and the enlarged cost function that guides investigation away from arrangements with highlights present in found local optima [18].

GLS is a meta-heuristic search technique[16] that sits on top of the local search technique to change its computation. It is an intelligent searching technique for a combinatorial optimization problem. The main idea states that the technique does iterative use of local search, gathers information from various sources and guides the local search towards a promising and suitable search space. It provides reasonable approximated solutions from local solution space due to its iterative searching mechanism. GLS follows a penalty-based mechanism that leads the local search technique through the guide function, which considers a feature where the cost and penalty are associated. The objective function *f(.)* defines the cost over the feature *i*. GLS develops punishments during a search. It utilizes penalties to assist the local search algorithm by looking through local minimal and plateaus. When the given local search

algorithm settles in a local optimum, GLS changes the objective function by utilizing a particular plan [19].

At that point, the local search utilizes an expanded objective function intended to bring the search out of the local optimum. The key idea is to change the objective function. GLS has been applied to a non-insignificant number of issues and found to be productive and viable. It is generally easy to execute and apply, with scarcely any parameters to tune. Assume the objective function *f(.)* and the guide function *g(.)* is treated as augmented objective function with the feature *i* and the corresponding penalty p_i in the solution space *s*. In each iteration *g(.)* contributes as a guide function and adjusts the increasing the penalties for the feature *i* from *s* and its utility can be evaluated. The feature *i* with the highest utility value is computed from *s*.

The greedy approach is followed in local searching techniques where we start with random solution space and stop with local minima. Here, instead of searching the whole solution space, we consider approximated solutions space using the iterative approach to compute the optimal solution.

In GLS, the general local search is given by the form as in Eqn. (6).

$$d_2 \leftarrow DCLocalSearch\ (d_1, c) \qquad (6)$$

where, d_2 is the local minimum and d_1 is the initial solution, and *c* is the objective cost function.

GLS defines solution feature as an augmented function, here we consider cost function as it and put the non-trivial solution element in the solution feature. Due to this property, each feature solution depends on the problem and interfaces within a particular application. The cost may affect directly or indirectly the solution feature.

A feature f_i *is* defined in Eqn. (7) as

$$I_i(d) = \begin{cases} 1, & if\ the\ solution\ has\ property\ i \\ 0, & Otherwise \end{cases} \quad and \quad d \in D \qquad (7)$$

The constraints on features are given by augmenting the cost function *c* to set a penalty.

Now, we have a new cost function called augmented cost function *g(d)* and is given in Eqn. (8).

$$g(d) = c(d) + \delta \sum_{i=1}^{n} P_i \,.\, I_i(d) \qquad (8)$$

where *n* = Number of features defined over solutions.

P_i= Penalty parameter for f_i

δ = Regularization parameter

Here, δ relates to the solution cost and has an impact on the search process and defined as in Eqn. (9).

$$\delta = {DC_{load}}/{DC_{capacity}} * VM_{load_i} \qquad (9)$$

A penalty vector is given over the defined solutions throughout the search process. During each iteration, the local search finds a local minimum over the possible set of solutions and let the penalty vector is given by $P = (P_1, P_2 \ldots P_n)$.

4. Proposed DC Allocation Mechanism (PE-DCA)

The detailed design and working mechanism of the proposed PE-DCA are described in this section. Various formulated functions followed by the diagrammatical representation of the proposed PE-DCA is also presented.

4.1. System Model

In this work, we propose a data center allocation mechanism named as PE-DCA using a meta-heuristic GLS technique. The local search does not consider the penalty for on-demand resources and multiple feasible data centers are also possible in different regions. The PE-DCA considers the penalty associated with feasible data centers. We derive the cost function *C( )* and time function *T( )* to consider different associated parameters. The total VM cost (VM_{total}) can be computed by VM cost (VM_{cost}) and data transfer cost (DT_{cost}) and is given as in Eqn. (10).

$$C(VM_{total}) = C(VM_{cost}) + C(DT_{cost}) \quad (10)$$

where, VM_{cost} is calculated as the sum of the cost per VM (VMP_{cost}), memory cost (M_{cost}), and the storage cost (ST_{cost}) and is given as in Eqn. (11).

$$C(VM_{cost}) = C(VMP_{cost}) + \sum_{i=1}^{x} C(M_{cost}) + \sum_{i=1}^{y} C(ST_{cost}) \quad (11)$$

where *x* defines the number of required memory units for MB main memory and *y* signifies the required storage units to X-MB [20][21]. These values are defined by the service provider during DC configuration using different pricing models [22]. DT_{cost} is derived by utilizing Eqn. (12) from available bandwidth ($BW_{avaliable}$), Data Size Request in bytes (D_R), and the number of PU as:

$$C(DT_{cost}) = \sum_{i=1}^{PU} \left(\beta * \frac{D_R}{BW_{avaliable}}\right) \quad (12)$$

The VM_{cost} is fixed in the feasible solution space, though the DT_{cost} is varied from the data center to the data center. So, VM_{total} also has impact in searching for the optimal global solution. We also compute the response time utilizing the Eqn. (13) and consider its importance in searching for the optimal solution.

$$T(Res_{Time}) = \left(c_1 + T\left(t_i - Req_{in\_time}\right)\right) + \left(\frac{D_R}{BW_{avalaible}} + ND_{Time}\right) + c_2 \quad (13)$$

where Res_{Time}= Response time, c_1= Time required to take decision for allocation, t_i=tentative start time of allocation, Req_{in_time}=Request initiated time, ND_{Time} = network delay and c_2 = Time required for configuration checking. The proposed architecture is shown as in Figure 3.

The PE-DCA finds the optimal data center for the on-demand request. If the searched data center is overloaded, then PE-DCA refers to the next optimum data center within the same region. In a region when all the data centers are allocated, then PE-DCA searches the next closest region.

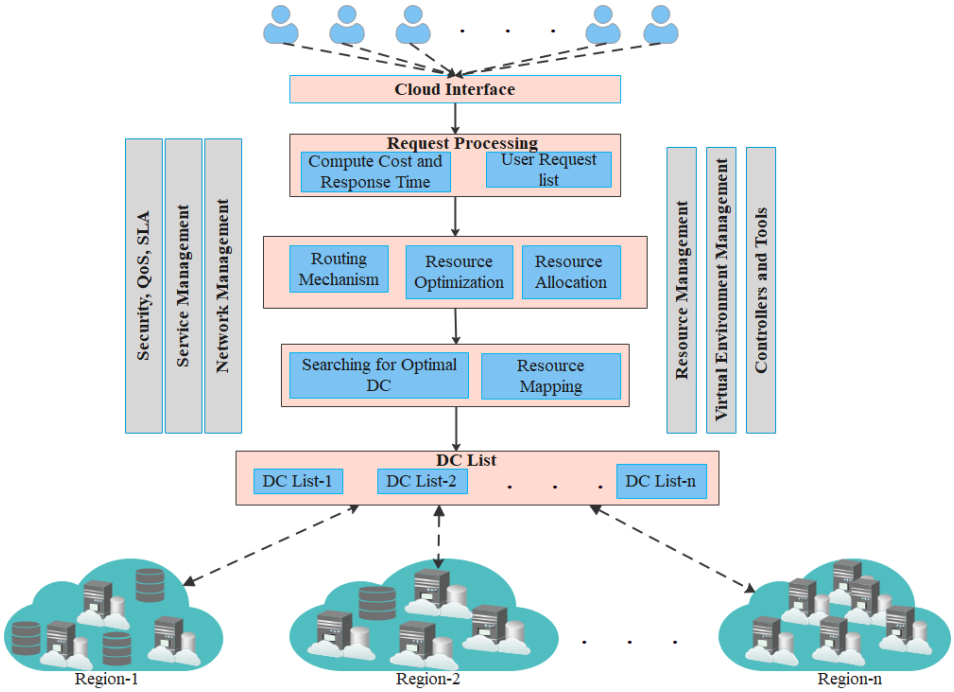

Fig. 3. Proposed PE-DCA Architecture

The proposed mechanism considers the network latency due to delay (N_{dl}) along with cost and time to search a suitable data center. We can compute N_{dl} as the time delay for a request from its initiation to retrieve a processing response. The latency directly depends on the $BW_{avaliable}$ and the delay parameter (d) and cooperatively impacts the response time, though discursively increase the overall cost for the VM.

$$N_{dl} = d + \eta\,[request/job]/BW_{avaliable} \quad (14)$$

$$\min Res_{Time} \propto \min N_{dl} \geq \min VM_{total}$$

$$\min Res_{Time} \propto \max BW_{avaliable}$$

$$\min VM_{total} \propto \min Pro_{Time}$$

where η = Data center request size and Pro_{Time} = Processing time.

The bandwidth alludes to the amount of data that can be conveyed inside the network from users to various DCs farms at a time instance. Likewise, it fundamentally influences the required response time of the request. Besides, the response time and all-out expense of the cloud environment proportionately influence the response time. Appropriately, the data centers farm can deal with more demands within a time unit when the processing time reduces, *i.e.*, roughly with an improved response time. Thus, the cloud framework's final cost decreases with processing time, resulting in refined overall performance.

4.2. Penalty Elimination and Allocation Technique

The proposed PE-DCA allocates the on-demand resources on the computed values of cost and time and is defined in Eqn. (10) and (13). The network latency as defined in Eqn. (14) is countered with cost and time in the suitable data center's searching mechanism. The various computational steps of PE-DCA are highlighted in Figure 4. The basic principle of GLS is analyzing constraints on solution features. For each iteration, the local search tends to a local minimum, and the penalty parameter defining one or more features over different solutions can be incremented by GLS. As the penalty parameter gradually increases for one or more features over a possible solution set the increment of penalty signifies that the penalized feature must be avoided by local search. Thus, high-cost features have more penalties in comparison to low-cost features.

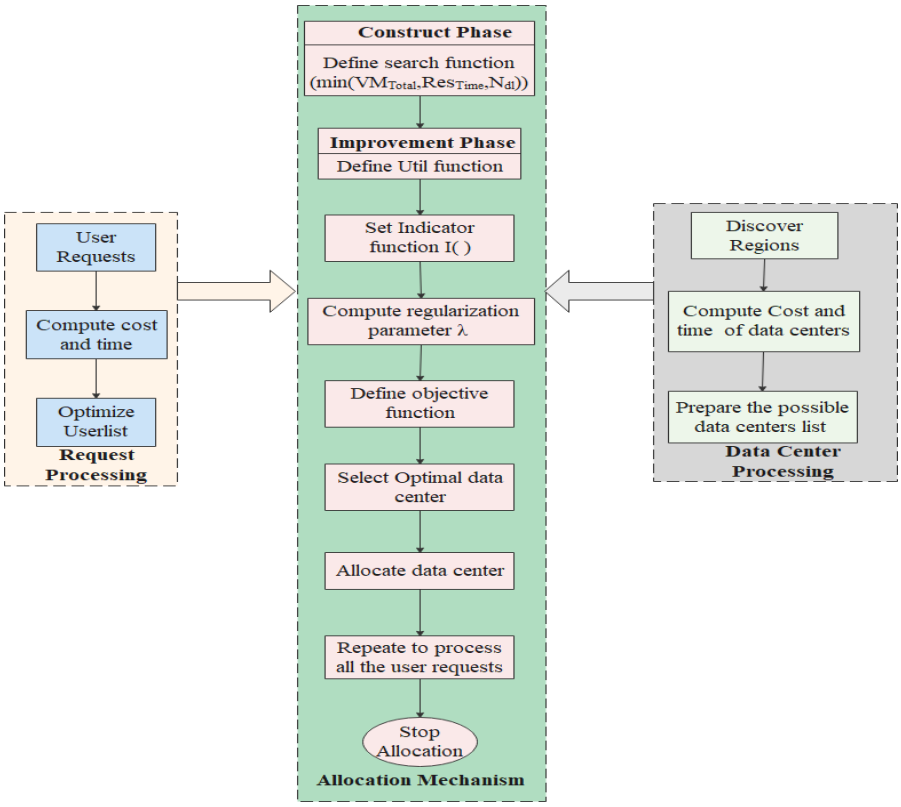

Fig. 4. Proposed PE-DCA Model and Design Steps

We assume that all the penalty values for all feature is set to 0 initially. Let each feature f_i is assigned to a total cost C_i and is represented by a vector as given in Eqn. (15).

$$C(d) = (C_1, C_2 \ldots C_n) \mid \forall C_i \geq 0 \qquad (15)$$

If the feature f_i is found as a local minimum solution D_* then

$I_i(D_*) = 1$ as per Eqn. (7).

Let a vector *L* considered as indicator function values which keep local minimum D_* and is represented as in Eqn. (16).

$$L(D_*) = \left(I_1(D_*), I_2(D_*) \dots I_n(D_*)\right) \quad (16)$$

There is a utility function related to each data center's possible solutions. In local minimum (D_*), the penalty values are increased by one if the utility function is maximized for each feature f_i.

$$DC_{Util}(D_*, f_i) = I_i(D_*) \cdot \frac{C_i}{1 + P_i} \quad (17)$$

The DC with minimum utility is selected before being assigned in a local minimum. Thus the utility function uses Vector *L(D∗)* and cost vector *C*. We introduced DC utility associated with each instance specified in Eqn.(17) to avoid the penalty incorporated to be of high cost. The PE-DCA evaluates the during allocation. It is calculated using Eqn. (18).

Request VM capacity ($RVM_{capacity}$): $RVM_{capacity} = P_s * N \quad (18)$

where P_s= processing speed of the processor in MIPS and *N*=Number of processors.

VM Taskload (VM_{load}): The VM_{load} states the workload of each VM with service rate at a time *t*. So the VM_{load} for i^{th} VM at time *t* is calculated by using Eqn. (19).

$$VM_{load_{i,t}} = k * {TL_i}/{X\left(VM_{i,t}\right)} \quad (19)$$

where k=1,2, …, n, TL_i = Task Length in Million Instructions (MI) and $X(VM_{i,t})$ = service rate of VM_i at time *t*.

Datacenter load (DC_{load}): It represents the number of the task of i^{th} VM in a request for a data center and is computed by using Eqn. (20).

$$DC_{load} = \sum_{j=1}^{m} pVM_{i,t} \quad (20)$$

where *p* = number of tasks associated with the request.

Datacenter capacity ($DC_{capacity}$): The DC capacity represents the sum of VM capacity and is calculated by using Eqn. (21).

$$DC_{capacity} = \sum_{j=1}^{m} RVM_{capacity} \quad (21)$$

Expected processing time (EPT): It is the time counted before allocation to complete the task with the on-demand request in the data center. EPT is defined as a total load of a data center and the total capacity and is given as in Eqn. (22).

$$EPT = {DC_{load}}/{DC_{capacity}} \quad (22)$$

Algorithm1:DC_Guided_Local_Search(P,g(d),α,[$I_1(D_*), I_2(D_*) \dots I_n(D_*)$],

[$C_1, C_2 \ldots C_n$],n)
Inputs: dcList=[*1,2,…,n*]
Output: **D**$_*$

1 Begin
2 k←0
3 D_0←InitialAllocation(P)
4 Set all penalties cost to 0
5 **for** *i*←*1* to *n*
6 P_i←0
7 $$g(d) = c(d) + \alpha \sum_{i=1}^{n} P_i \cdot I_i(d)$$
8 **While** stoppingCriteria==false
9 do
10 D_{k+1}←DClocalSearch(D_k,g)
11 **for**←*1* to *n* do
12 $$DC_{Util}(D_*, f_i) = I_i(D_*) \cdot \frac{C_i}{1 + P_i}$$
13 **for** each *i* such that DC_{Util_i} is maximum **do**
14 P_i←P_i+1
15 End for
16 k← k+1
17 End for
18 End while
19 *dcName*=D_*← best solution with the minimum of DC_{total} and Res_{Time}
20 Return D_*
21 End for

The proposed mechanism PE-DCA implements two procedures *DC_Guided_Local_Search() and Select_DC()*. The *DC_Guided_Local_Search()* empowered with the meta-heuristic search mechanism GLS. The design approach follows the *InitialAllocation()* to construct the initial solution for *D* over problem *P*. The *DClocalSearch(Dk,g)* searches local minimum and improves the solution with D_k and compute the utility till the *stoppingCriteria* fails. By eliminating the penalty, the best solution is selected based on the objective function for DC_{total} and Res_{Time} as defined in Eqn. (10) and (13). The procedure *Select_DC()* emphasizes in selecting the data center with minimum penalty. It computes the cost and time using the defined objective functions for each user request to penalty while allocating the data center. The minimum penalty of cost and time defined in step 7 is the selected data center's parameters. For allocating the computing data center, we compute DC_{load} and $DC_{capacity}$ as defined in Eqn. (20) and (21) to know whether the computed data center is a suitable one or not; if suitable, the *dcName* is tracked with the index for allocating the user's resources.

Algorithm 2:**Select_DC( )**
Inputs: dcList=[*1,2,…,n*], regionList=[*1,2,…,m*], where *m*<*n*
Output: **dcName[]**

Begin
For the selected region get data center index *regionList*
Get regionalDatacenterIndex.get(region)
Keep region list for selected data center
if *regionList* is not *NULL* then

listSize_size(regionList)

if *listSize_size* is 1 then
dcName_regionList.get(0)
else

for i=1 to n

Compute $C(VM_{total})$ and $T(Res_{Time})$for each DC

Create the list p for DC with $min(VM_{total}, Res_{Time})$

End for

for all *p*

Compute $DC_{capacity}$and DC_{load}

if (DC_{load}< $DC_{capacity}$)

get DC index and prepare *dcName[ ]*

dcName[index]=dcName_regionList.get(p)

End if

End for
End if else
End if
Return dcName[ ]
End

5. Simulation and Results

This section presents the proposed PE-DCA outcomes, briefing the required simulation tool, setup, and configuration, and evaluating different parameters such as cost, response time, and processing time obtained through simulation and analyzes the performance. CloudAnalyst [20] is an open-source Java-based cloud tool issued to study the proposed PE-DCA behavior to support a built-in environment.

5.1. Simulation Environment

The standard parameters of CloudAnalyst were customized to examine the results of PE-DCA. CloudAnalyst defines six geographical regions indexed as ($R_0, R_1, \ldots, R_5$) to locate DCs in its simulation environment. We consider various network delays in milliseconds among the simulation regions and is given as in Table.2. We assume a minimum network delay of 25 milliseconds in a similar region and vary to a maximum of 500 milliseconds for other regions during our simulation. We consider various bandwidths ranging from 800 to 2500 Mbps for data transmission among the regions and are given in Table.3 in our simulation study. PE-DCA is examined for a different set of user requests and DCs and is presented in Table.4.

Table 2. Network delay in regions (ms)

Region	R_0	R_1	R_2	R_3	R_4	R_5
R_0	25	100	150	250	250	100
R_1	100	25	250	500	350	200
R_2	150	250	25	150	150	200
R_3	250	500	150	25	500	500
R_4	250	350	150	500	25	500
R_5	100	200	200	500	500	25

Table 3. Bandwidth in regions (Mbps)

Region	R_0	R_1	R_2	R_3	R_4	R_5
R_0	2000	1000	1000	1000	1000	1000
R_1	1000	800	1000	1000	1000	1000
R_2	1000	1000	2500	1000	1000	1000
R_3	1000	1000	1000	1500	1000	1000
R_4	1000	1000	1000	1000	500	1000
R_5	1000	1000	1000	1000	1000	2000

Table 4. Experimental User and Data Center Set

User Set	No of Users	No of DCs
1	100	28
2	150	43
3	200	55
4	250	68
5	300	80

The network bandwidth and transmission delay for the considered experiments and scenarios were kept constant during the simulation. The proposed simulation model is shown in Figure 5 and the user requests are generated using the user base interface of CloudAnalyst. Similarly, the data center configuration interface of CloudAnalyst is used to configure the data centers. Here, we created two new classes under the package of *cloudsim.ext* for implementing PE-DCA, and those classes along with the package are imported to the package *cloudsim.ext.gui*. Finally, all the classes are executed from the caller's primary method *GuiMain.java* class. The Java-based application is developed to implement the computed functions and is shown in Figure 5.

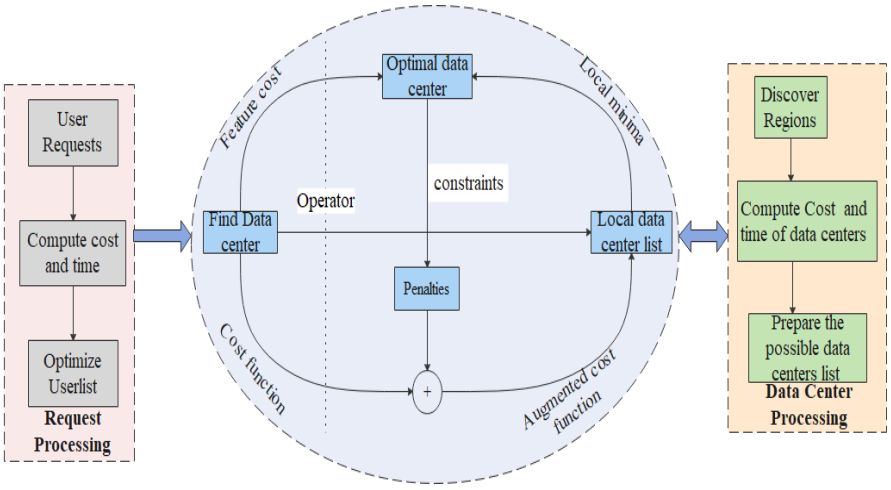

Fig 5. Implementation Model of PE-DCA for Simulation

Here, we consider cost and time to prepare the user list and derive total VM cost in Eqn. 10 and response time in Eqn. 13. The user list is computed with minimum total VM cost and response time. Similarly, we prepare the DC list considering the compute cost and response time. As we earlier discussed GLS work on the top of the local search technique in incremental way. We implement the cost and the response time function as the augmented function for incremental growth of the searching technique. The java-based function is developed which consider the 1st user request from the user list and estimate the cost to the corresponding DC list. This process goes incrementally and computes the penalty and utility corresponding to the DCs. The similar process is repeated for the response time augmented function. The application keeps all the records of the penalties and utilities of each user request with respect to the DCs. Here, we consider DC_{load} and $DC_{capacity}$ as the searching constraints. If DC_{load} is less than $DC_{capacity}$ then the proposed system selects the minimum penalty and returns the DC index for allocation.

6. Performance Evaluation

The study of PE-DCA is performed by considering diverse userbase (user request) and data center configuration specifications. The performance was obtained for a variable number of user requests, with a maximum of up to 300 in different scenarios. The number of user requests per hour varies from 60-90, the request size is set to 1-500 KB, and the average off-peak user is 50 and 100. The data center is configured as x86 architecture, Linux OS, Xen VMM with variable physical H/W units (1-4) with four numbers of processors, each having 10,000 MIPS. We consider different standard costs for the data center: the cost per VM per Hr as 0.1$, memory cost as 0.05$, and data transfer cost per GB as 0.1$. The performance is studied by considering a number of data centers vary between 28-80, which is less than 30% of the number of user requests in different scenarios.

The performance of PE-DCA is also studied under different scenarios to determine suitable data centers for a user request. We implement the derived function (Eqn.(10)) for VM_{total} and Res_{ume} for each user request (Eqn. (13)) in the set. The DC_{util} function as in Eqn.(17) is computed to examine the utilization for each data center before allocation. Implementing DC_{util} is to maintain load balancing among the DCs and is used as the *stoppingCriteria* in PE-DCA. We compute the response time penalty in millisecond and cost penalty in dollar ($). However, from the obtained results it was found that the penalty in allocation increases the response time, and the penalty in cost improves the total VM cost. So, as the penalty is reduced for time and cost, the response time and total VM cost are reduced, and better data center allocation is achieved.

PE-DCA evaluates the deviation in the cost for each user request for the DCs. On the other hand, we had also evaluated the response time deviation for each user request and the selected DCs. The penalties may vary for each set of the user request and the DC. So, we evaluated the penalty of cost and response time in average and is noted in Table.5 for users set. Figure 6 depicts the response time penalty associated with a different set of user requests. It signifies that the deviations in cost and time for each user request need to be eliminated. Note that the increase in deviation may increase the total cost and response time. The minimum and maximum response time penalty of 0.02 ms and 0.8 ms were observed during the simulation. It was noticed that the average penalties varies between 0.6025 ms to 1.0575 ms during the experimentation. The cost ($) penalty associated to user requests in different user sets is shown in Figure 7, where the minimum and maximum cost penalties are 0.02$ and 0.28$ and the simulation studies consider the average cost penalties within the range 0.1231$ to 0.1394$ which are responsible for increasing the total VM cost and need to be eliminated.

Table 5. The Cost and Response Time Penalty

User Set	Average *P(c)* in $	Average *P(t)* in ms
1	0.1231	0.6025
2	0.1338	0.8583
3	0.1352	0.8709
4	0.1371	0.9732
5	0.1394	1.0575

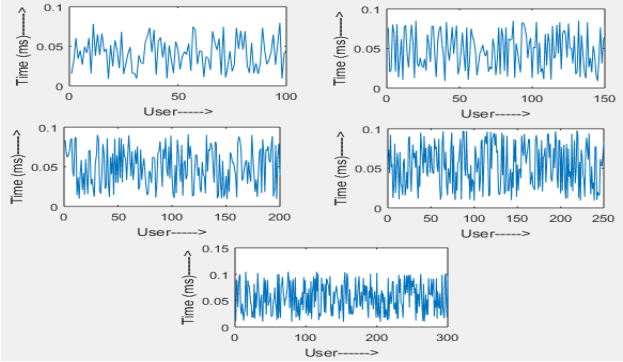

Fig. 6. Computed Response Time Penalty (ms) to User Request in Different User Set

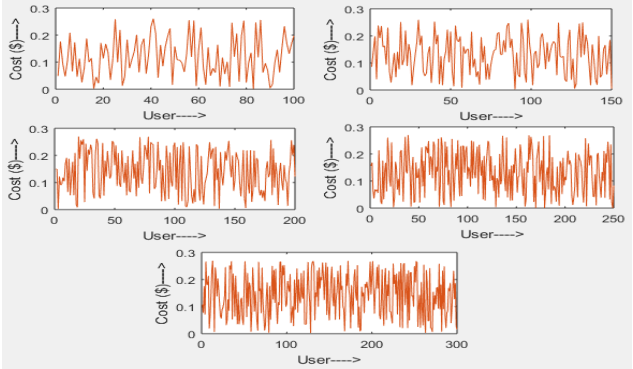

Fig. 7. Cost Penalty ($) associated to User Request in Different User Set

7. Results and Discussion

This section presents simulation results along with the analysis of results. The simulation results of the total cost, response time, and data center processing are discussed for various users. The advantages of the PE-DCA for data center allocation are also highlighted.

7.1. Overall Response Time

The response time computed during the simulation addresses the time required to allocate the data center. It can vary from data center to data center for various bandwidth and network delays. The computation of response time was computed with regard to bandwidth and network delays are given in Table.2 and 3. The response time penalties

occur due to improper allocation, which may be considered as the deviation during allocation. We derived the function for response time (Eqn.(13)) and computed it by considering the penalty associated with each user request. The suitable data center was selected by evaluating the δ defined in Eqn.(9) which examines whether the data center can accommodate the request or not. To evaluate δ, we implemented Eqn. (19), (20), and (21) and computed the minimum, maximum, and average response time for all the user sets are noted in Table.6. It was observed from obtained simulation results that the proposed PE-DCA mechanism requires a minimum of 46.0875677158 ms and a maximum of 173.990628155 ms to allocate the data center. The overall response time for various user request set is monitored and shown in Figure 8.

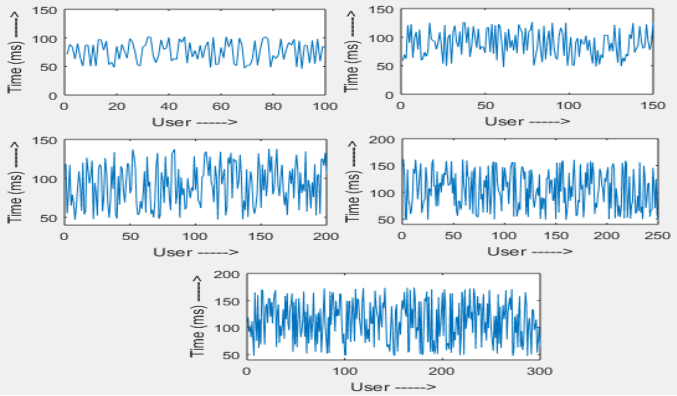

Fig. 8. Overall Response Time (x-axis: Number of User Request, Y-axis: Response Time (ms))

Table 6. Overall Response Time for Various User Sets

User Set	Over All Response Time		
	Minimum	Maximum	Average
1	46.09	102.25	76.28
2	47.74	126.41	88.85
3	47.08	138.05	92.09
4	47.32	161.83	104.67
5	47.34	173.99	111.52

7.2. Overall Total Cost

The CloudAnalyst simulator supports for evaluating different costs such as VM cost, data transfer cost, and total cost. The total cost is the sum of VM cost and data transfer cost. We implemented Eqn. (11) and (12) to evaluate the total VM cost. It can be varied from request to request due to the Request VM capacity defined in Eqn. (18). We computed each request VM capacity and the corresponding cost. We computed the penalty cost associated to user request is shown in Figure 7. In PE-DCA, we observed that the penalty cost was gradually decreasing to a minimum level and was not changed

further as shown in Figure 9. We considered the minimum penalty cost for computing the overall cost metric: VM cost (VM_{cost}), data transfer cost (DT_{cost}), and total cost (VM_{total}) for all user sets and were recorded in Table.7.

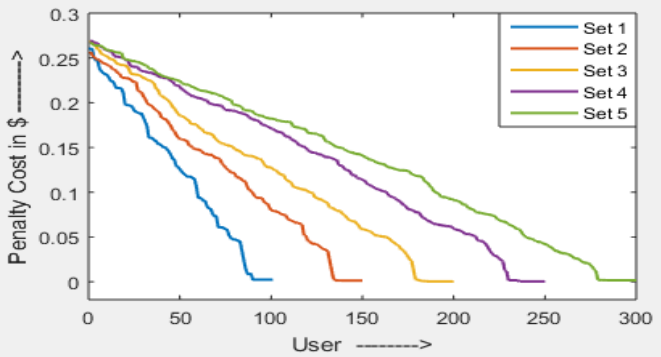

Fig. 9. Penalty Cost Reaching to Minimum Level

Table 7. Overall Cost Metric in $

User Set	VM_{cost}	DT_{cost}	VM_{total}
1	71.73	53.57	125.30
2	84.26	71.48	155.74
3	114.44	83.67	198.11
4	141.76	94.67	236.43
5	176.78	108.54	285.32

7.3. Overall Processing Time

During simulation, we computed the overall processing time for all the user sets. The processing time varies for different types of users due to the on-demand of resources. For each user request, we computed the EPT as defined in Eqn. (22) and considered min(EPT) in the implementation to evaluate the overall processing time. As the proposed work was simulated using the CloudAnalyst tool, we used the in-built mechanism and examined the overall processing time for our configured data centers and user requests rather than defining any procedure. The overall processing time for various user sets is shown in Figure 10, and noted the minimum and maximum time as 3.98 ms and 1371.28 ms in our simulation. The minimum, maximum, and average computational overall processing time was recorded as given in Table.8.

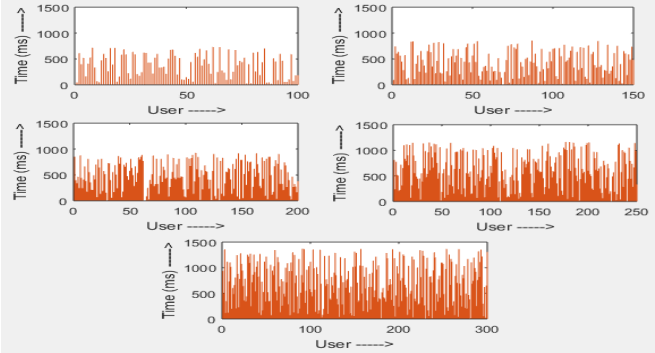

Fig. 10. Overall Processing Time (X-axis: Number of user requests, Y-axis: Processing time (ms))

Table 8. Overall Processing Time (ms)

User Set	Minimum	Maximum	Average
1	19.98	730.47	371.5
2	3.98	855.17	427.99
3	5.05	926.09	475.38
4	4.48	1162.04	607.31
5	6.03	1371.28	692.79

8. Comparison

In this section, we represent the performance comparison of the proposed PE-DCA with different prevailing techniques. We compare the in-built broker policies of CloudAnalyst first, followed by comparing various techniques as proposed researchers for data center allocation.

8.1. PE-DCA vs Benchmark Broker Policy

We compare the performance parameters such as total VM cost, overall response time, and overall processing time for various user sets with the benchmark broker policies of CloudAnalyst. The *Closest Data Center* [20] broker policy focuses on routing user requests based upon the nearest data center. It does not consider other computing parameters for allocation. It allocates the data center by considering the distance among the data center. CloudAnalyst defines *Optimize Response Time* [20] broker policy to optimize the response time of the user request. This policy allocates data centers based upon minimizing the response time of the request. The comparison of average total cost in $ is noted in Table.9.

Table 9. Comparison of Average Total Cost in $

User Set	Closest DC Policy	Optimize Response Time	PE-DCA
1	1.3593	1.2984	1.2530
2	1.1320	1.1471	1.0382
3	1.0332	1.0957	0.9905
4	1.0629	1.1051	0.9457
5	1.0811	1.0444	0.9510

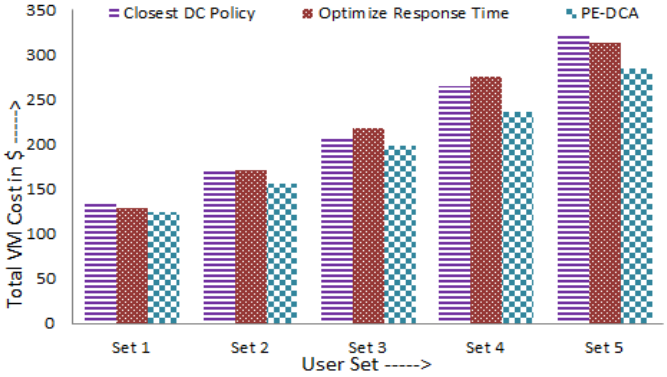

Fig. 11. Comparison of Total VM Cost for User Sets

Fig. 12. Comparison of Average Overall Response Time for User Sets

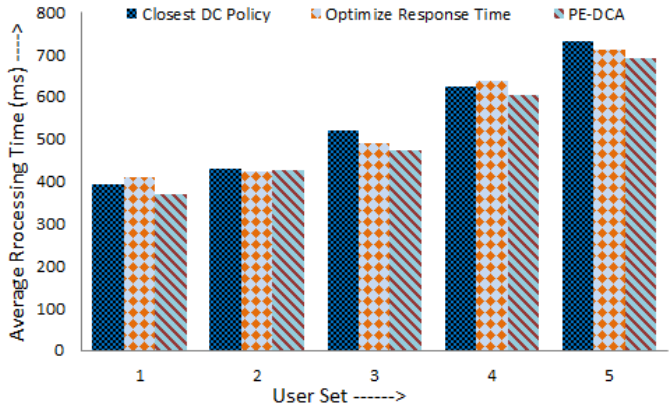

Fig. 13. Comparison of Average Overall Processing Time for User Sets

8.2. PE-DCA vs Existing Approaches

Many researchers have recently used optimization techniques for resource management, mainly the Particle Swarm Optimization (PSO) impacted more. We compare our proposed work with the existing PSO-based mechanisms for cost and response time optimization in resource management for cloud computing. To establish a comparison with existing mechanisms, we develop new java classes under the package of *cloudsim.ext* and import them to the package *cloudsim.ext.gui*. To compare costs, cost-aware PSO (CA-PSO) [23], novel PSO (NPSO) [24], and Modified PSO (MPSO)[25] are considered, where such techniques are executed, and the number of iterations is similar to the number of user requests in the set. The comparison of total cost between the above approaches and proposed PE-DCA is shown in Figure 14, and the average overall response time is depicted in Figure 15. Note that the total VM cost of PE-DCA and existing NPSO and MPSO are found to be approximately equal as shown in Figure 14, but the average overall response time is found to be more than PE-DCA as noticed in Figure 15.

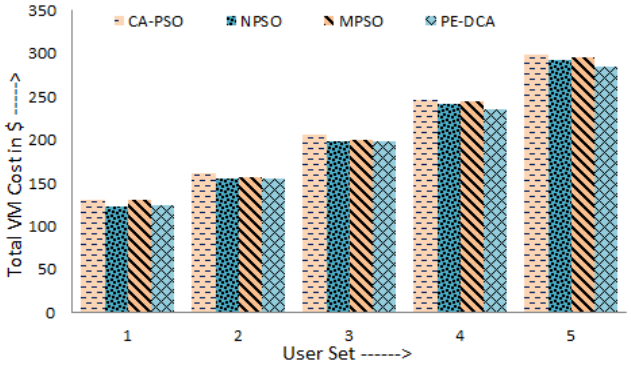

Fig. 14. Comparison of Total VM Cost with PSO Based Techniques

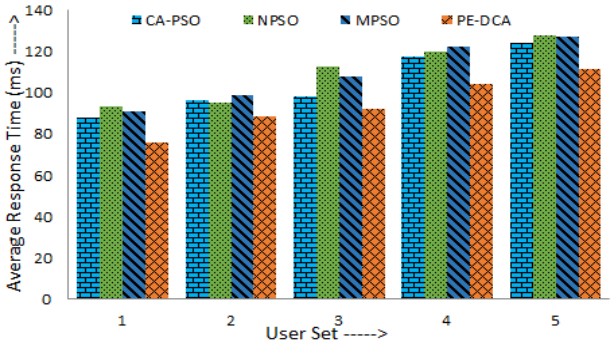

Fig. 15. Comparison of Average Response Time with PSO Based Techniques

9. Related Work

This section elaborates various related work on DC allocation and meta-heuristic search techniques for selecting suitable DC in cloud computing. Firstly, we discuss the recently proposed optimization-based approaches for allocation and then the searching-based techniques for cloud computing are presented.

For assigning the cloud services, the selection of suitable data center is much required. For which different allocation mechanisms have been proposed in the literature. Here, we present only the optimization-based techniques that target to optimize the allocation parameters such as cost and time. Manasrah *et al.* [21] designed a meta-heuristic-based Variable Service Routing Policy (VSBRP) which reduces the data centers overload and significantly minimized the response time and processing time. Jessica *et al.* [26] proposed a Multi-Objective Genetic Algorithm based cloud brokering policy which reduces the response time and cost. The authors simulated the real data over Amazon EC2. Manasrah *et al.* [27] discussed an optimized service broker routing policy based on a differential evolution algorithm. It aimed to minimize response

time, processing time, and overall cost in fog and cloud environments. The authors implemented the new broker policy using six different scenarios.

Pengcheng *et al.*[28] considered the scheduling mechanism as a multi-objective optimization problem and presented the mechanism to optimize the cost and makespan. It eliminates useless resources to narrow down the search space. A fuzzy-based approach to minimize the end-user cost is discussed using the Ant Colony Optimization (ACO) mechanism to allocate the computing and network resources in [29]. Recently, Zhenxin [30] modified Artificial Bee Colony (ABC) to elite-guided ABC for large number of particle problems. Mapetu *et al.*[31] discussed load balancing mechanism using binary PSO to schedule the task with low-cost and low-time complexity. A cost optimization-based mechanism is presented for the enterprise cloud to optimize computing cost, bandwidth cost, and I/O cost for resource allocation [32].

Table 10. Summary of Literature Study

Reference	Optimization Based					Without Penalty Based Sear.	Penalty Based Sear.
	Cost	Response Time	Processing Time	Makespan	Other		
Manasrah *et al.*[21]	×	√	√	×	×	×	×
Jessica *et al.*[26]	√	√	×	×	×	×	×
Manasrah *et al.*[27]	√	√	√	×	×	×	×
Pengcheng *et al.*[28]	√	×	×	√	×	×	×
Mishra *et al.* [29]	√	×	×	√	×	×	×
Mapetu*et al.* [31]	√	×	√	×	×	×	×
Suchintan *et al.*[32]	√	×	×	×	×	×	×
Larumbe *et al.*[34]	×	×	×	×	√	√	×
Tellez *et al.*[35]	×	×	×	×	√	√	×
Pan *et al.*[36]	×	×	×	×	√	√	×
Hamza *et al.*[37]	×	×	×	×	√	√	×
Parida *et al.*[38]	√	√	√	×	×	×	×
Divya *et al.*[39]	√	×	√	×	×	×	×
Ali *et al.*[40]	×	×	×	√	×	√	×
Alkhashai *et al.*[41]	√	×	√	×	×	√	×
Parida *et al.* [42]	√	×	√	×	×	×	×
Proposed PE-DCA	√	√	√	×	×	√	√

Researchers also utilized various searching mechanisms for resource management, task scheduling, VM allocation, and DC allocation in cloud computing. Leonard *et al.*[33] integrated the brokering concept with multi-criteria location-based selection using neighborhood search approaches for virtual machines residing in the multi-cloud era. The author implemented in CloudSim with the Greedy approach and meta-heuristic search. This work also proved to give improved latency and optimized cost. Larumbe *et al.*[34] determined the DC location using the Tabu search optimization technique. The work aims to optimize network performance and CO_2 emission. Tellez *et al.*[35] presented the Tabu search technique for optimal load balancing between cloud and fog nodes. To solve the joint resource allocation task scheduling problem, Pan *et al.*[36]

proposed a Tabu search-based heuristic approach for cloud computing. Hamza *et al.*[37] discussed a hybrid approach using Tabu search and simulated annealing for load balancing in cloud computing. In [38] authors proposed a new meta-heuristic approach for data center allocation with minimum cost, response time, and processing time. The summary of the literature review is presented in Table.10. From the literature study it is found that the penalty estimation and elimination are the major issues while searching for a suitable data center for allocation and is not found from the literature review.

10. Conclusions and Future Work

The datacenter allocation to the on-demand resources in cloud computing is an essential yet open issue for fair allocation with minimized response time, processing, and cost. Due to the dynamic nature of cloud computing, DC allocation is an NP-hard problem. In this work, we study and examine the impact of a penalty during the allocation of resources. While allocating the resource R_i to on-demand request U_i, the penalty may be associated with another user request U_x, which means R_i might be the best suitable resource for U_x. To search the suitable data centers for the on-demand user, we suggest PE-DCA, a new search-based allocation technique addressing the penalty associated with response time, processing time, and cost using the GLS meta-heuristic technique. The selection of data centers for the on-demand resources is established through finding and eliminating the penalties for each user request in allocation to achieve fair allocation. During the implementation, the number of user requests and data centers were configured using the CloudAnalyst tool and the time and cost parameters were computed. The performance was compared and studied with benchmark allocation techniques of CloudAnalyst and other PSO-based techniques (CA-PSO, NPSO, and MPSO). From the simulation results it was found that as compared with the CloudAnalyst benchmark mechanisms, the proposed PE-DCA performs better in minimizing the time and cost parameters as depicted in Figures 11, 12, and 13. The total VM cost of PE-DCA is approximately equal to the evaluated cost of existing techniques of NPSO and MPSO. In contrast, the average response time of PE-DCA is found to be less as compared to NPSO and MPSO. The PE-DCA provides better data center allocation with minimum response time, cost, and processing time.

As future directions, we will further consider additional constraints for data center allocation in cloud computing such as power consumption, deadline constraint workflow, SLA, and QoS. The penalties related to power consumption, task penalty for deadline-based workflow scheduling, evaluation of SLA, and QoS penalty violation will also be investigated in the context of cloud computing. The exploration of other penalty-related searching techniques is also needed to enhance the cloud environment performance. The data center allocation task can be related to energy consumption to formulate an optimized allocation algorithm for cloud systems. The dynamic nature of cloud computing can be further added to explore more on resource allocation mechanism along with multiple objectives. Such approaches can also be cascaded with penalties by eliminating the searching approach to enhance the efficiency of cloud systems.

References

1. Z. Zhang, C. Wu, and D. W. L. Cheung, “A Survey on Cloud Interoperability: Taxonomies, Standards, and Practice,” SIGMETRICS Perform. Eval. Rev., vol. 40, no. 4, pp. 13–22, 2013.
2. Y. Cao, L. Lu, J. Yu, S. Qian, Y. Zhu, and M. Li, “Online cost-rejection rate scheduling for resource requests in hybrid clouds,” Parallel Comput., vol. 81, no. 800, pp. 85–103, 2019.
3. W. Liang, D. Zhang, X. Lei, M. Tang, K. C. Li, and A. Zomaya, “Circuit Copyright Blockchain: Blockchain-based Homomorphic Encryption for IP Circuit Protection,” IEEE Trans. Emerg. Top. Comput., vol. 6750, no. c, pp. 1–11, 2020.
4. A. Shawish and M. Salama, “Cloud Computing: Paradigms and Technologies,” Inter-cooperative Collect. Intell. Tech. Appl., vol. 495, pp. 39–68, 2014.
5. J. L. Sarkar, C. R. Panigrahi, B. Pati, A. K. Saha, and A. Majumder, “MAAS: A mobile cloud assisted architecture for handling emergency situations,” Int. J. Commun. Syst., vol. 33, no. 13, pp. 1–15, 2020.
6. S. C. Nayak, “Multicriteria decision - making techniques for avoiding similar task scheduling conflict in cloud computing,” Int. J. Commun. Syst., no. July 2018, pp. 1–31, 2019.
7. Sasmita Parida, Suvendu Chandan Nayak, et al. “Truthful Resource Allocation Detection Mechanism for Cloud Computing,” in Third International Symposium on Women in Computing and Informatics (WCI ’15), Indu Nair (Ed.). ACM, 2015, pp. 487–491.
8. S. Mohapatra, C. R. Panigrahi, B. Pati, and M. Mishra, “MSA: A task scheduling algorithm for cloud computing,” Int. J. Cloud Comput., vol. 8, no. 3, pp. 283–297, 2019.
9. J. Proaño, C. Carrión, and B. Caminero, “Empirical modeling and simulation of an heterogeneous Cloud computing environment,” Parallel Comput., vol. 83, pp. 118–134, 2019.
10. G. Zou, Z. Qin, S. Deng, K. C. Li, Y. Gan, and B. Zhang, “Towards the optimality of service instance selection in mobile edge computing,” Knowledge-Based Syst., vol. 217, p. 106831, 2021.
11. R. Buyya, S. K. Garg, and R. N. Calheiros, “SLA-Oriented Resource Provisioning for Cloud Computing : Challenges , Architecture , and Solutions,” in International Conference on Cloud and Service Computing, 2011, no. Figure 1, pp. 1–10.
12. J. Li, S. Su, X. Cheng, M. Song, L. Ma, and J. Wang, “Cost-efficient coordinated scheduling for leasing cloud resources on hybrid workloads,” Parallel Comput., vol. 44, pp. 1–17, 2015.
13. S. C. Nayak and C. Tripathy, “Deadline based task scheduling using multi-criteria decision-making in cloud environment,” Ain Shams Eng. J., vol. 9, no. 4, pp. 3315–3324, 2018.
14. S. Nanda, C. R. Panigrahi, and B. Pati, “Emergency management systems using mobile cloud computing: A survey,” Int. J. Commun. Syst., no. May 2019, pp. 1–20, 2020.
15. Parida S., Pati B., Nayak S.C., Panigrahi C.R. (2020) Offer Based Auction Mechanism for Virtual Machine Allocation in Cloud Environment. Proceedings of ICACIE 2018, Volume 2, vol. 2. pp. 339-352, 2020.
16. C. Voudouris, “Chapter 7 Guided Local Search,” Handb. Metaheuristics, pp. 185–218, 2003.
17. A. Alsheddy and E. P. K. Tsang, “Empowerment scheduling for a field workforce,” J. Sched., vol. 14, no. 6, pp. 639–654, 2011.
18. P.H. Mills “Extensions To Guided Local Search: A thesis submitted for the degree of Ph . D . Department of Computer Science University of Essex,” 2002.
19. M. Gendreau and J.-Y. Potvin, Variable Nneighborhood search (chapter), vol. 146. pp.211-238, 2010.
20. B. Wickremasinghe, R. N. Calheiros, and R. Buyya, “CloudAnalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications,” Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA, pp. 446–452, 2010.

21. A. M. Manasrah, T. Smadi, and A. ALmomani, “A Variable Service Broker Routing Policy for data center selection in cloud analyst,” J. King Saud Univ. - Comput. Inf. Sci., vol. 29, no. 3, pp. 365–377, 2017.
22. J. Huang, R. J. Kauffman, and D. Ma, “Pricing strategy for cloud computing: A damaged services perspective,” Decis. Support Syst., vol. 78, pp. 80–92, 2015.
23. G. Zhao, “Cost-Aware Scheduling Algorithm Based on PSO in Cloud Computing Environment,” Int. J. Grid Distrib. Comput., vol. 7, no. 1, pp. 33–42, 2014.
24. R. Pragaladan and R. Maheswari, “Improve Workflow Scheduling Technique for Novel Particle Swarm Optimization in Cloud Environment,” Int. J. Eng. Res. Gen. Sci., vol. 2, no. 5, pp. 675–680, 2014.
25. S. gaelle Mohamod, Kasim, “Council for Innovative Research,” J. Adv. Chem., vol. 10, no. 1, pp. 2146–2161, 2014.
26. Y. Kessaci, N. Melab, and E. G. Talbi, “A pareto-based genetic algorithm for optimized assignment of VM requests on a cloud brokering environment,” 2013 IEEE Congr. Evol. Comput. CEC 2013, pp. 2496–2503, 2013.
27. A. M. Manasrah and A. B. B. Gupta, “An optimized service broker routing policy based on differential evolution algorithm in fog / cloud environment,” Cluster Comput.,Vol.22, pp. 1639–1653, 2019..
28. P. Han, C. Du, J. Chen, F. Ling, and X. Du, “Cost and makespan scheduling of workflows in clouds using list multiobjective optimization technique,” J. Syst. Archit., Volume 112, pp. 809-837, 2021.
29. S. B. Suchintan Mishra, Arun Kumar Sangaiah,Manmath Narayan Sahoo, “Pareto-optimal cost optimization for large scale cloud systems using joint allocation of resources,” J Ambient Intell Hum. Comput, 2019.
30. Z. Du, D. Han, and K. C. Li, Improving the performance of feature selection and data clustering with novel global search and elite-guided artificial bee colony algorithm, vol. 75, no. 8. Springer US, 2019.
31. J. P. B. Mapetu, Z. Chen, and L. Kong, “Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing,” Appl. Intell., vol. 49, no. 9, pp. 3308–3330, 2019.
32. S. Mishra, M. N. Sahoo, A. Kumar Sangaiah, and S. Bakshi, “Nature-inspired cost optimisation for enterprise cloud systems using joint allocation of resources,” Enterp. Inf. Syst., no. 0123456789, 2019 (Inpress).
33. L. Heilig, R. Buyya, and S. Voß, “Location-aware brokering for consumers in multi-cloud computing environments,” J. Netw. Comput. Appl., vol. 95, pp. 79–93, 2017.
34. F. Larumbe and B. Sansò, “A tabu search algorithm for the location of data centers and software components in green cloud computing networks,” IEEE Trans. Cloud Comput., vol. 1, no. 1, pp. 22–35, 2013.
35. N. Téllez, M. Jimeno, A. Salazar, and E. D. Nino-Ruiz, “A Tabu search method for load balancing in fog computing,” Int. J. Artif. Intell., vol. 16, no. 2, pp. 106–135, 2018.
36. P. Yi, H. Ding, and B. Ramamurthy, “A Tabu search based heuristic for optimized joint resource allocation and task scheduling in Grid/Clouds,” 2013 IEEE Int. Conf. Adv. Networks Telecommun. Syst. ANTS 2013, 2013.
37. F. Youssef, B. L. El Habib, R. Hamza, Labriji El Houssine, E. Ahmed, and M. Hanoune, “A New Conception of Load Balancing in Cloud Computing Using Tasks Classification Levels,” Int. J. Cloud Appl. Comput., vol. 8, no. 4, pp. 118–133, 2018.
38. S. Parida and B. Pati, “A Cost Efficient Service Broker Policy for Data Center Allocation in IaaS Cloud Model,” Wirel. Pers. Commun., no. 0123456789, 2020 (Inpress).
39. D. Chaudhary and B. Kumar, “A New Balanced Particle Swarm Optimisation for Load Scheduling in Cloud Computing,” J. Inf. Knowl. Manag., vol. 17, no. 1, 2018.
40. A. Al-maamari and F. A. Omara, “Task Scheduling Using PSO Algorithm in Cloud Computing Environments,” Int. J. Grid Distrib. Comput., vol. 8, no. 5, pp. 245–256, 2015.

41. H. M. Alkhashai and F. A. Omara, “An enhanced task scheduling algorithm on cloud computing environment,” Int. J. Grid Distrib. Comput., vol. 9, no. 7, pp. 91–100, 2016.
42. Parida S., Pati B., Nayak S.C., Panigrahi C.R. (2021) Offer Based Auction Mechanism for Virtual Machine Allocation in Cloud Environment. Proceedings of ICACIE 2019, Volume 1, vol. 2. pp. 621-633, 2021.

Sasmita Parida completed B.Tech. and M.Tech. in Computer Science and Engineering from Biju Patnaik University of Technology, India in the year 2004 and 2010 respectively. She is currently working as Assistant Professor in the Department of Computer Science and Technology at GITA Autonomous College, Bhubaneswar, BPUT, India. She has around 15 years of experience in teaching and research. She is pursuing Ph.D. degree at Rama Devi Women′s University, India. Her area of research is cloud computing, Bigdata, IoT, Machine Learning and parallel computing. He has published more than 18 research papers in different journals and conference proceedings.

Bibudhendu Pati completed his Ph.D. degree from IIT Kharagpur, India. He is currently working as Associate Professor in the Department of Computer Science at Rama Devi Women's University, Bhubaneswar, India. He has around 23 years of experience in teaching and research. His areas of research interests include Wireless Sensor Networks, Cloud Computing, Big Data, Internet of Things, and Advanced Network Technologies. He has got several papers published in reputed journals, conference proceedings, and books of international repute. He has been involved in many professional and editorial activities. He is a Life Member of Indian Society for Technical Education, Computer Society of India and Senior Member of IEEE.

Suvendu Chandan Nayak received his Ph.D. degree from Veer Surendra Sai University of Technology (Formally University College of Engineering, Burla) India in Computer Science & Engineering. He is currently working as an Associate Professor in the Department of Computer Science and Information Technology at GITA Autonomous College Bhubaneswar, BPUT, India. He has 14 years of teaching and research experience in the field of computer science. His area of research is cloud computing, Bigdata, IoT and parallel computing. He has published more than 20 papers in different International / National journals and conference proceedings.

Chhabi Rani Panigrahi received her Ph.D. in Computer Science and Engineering from IIT Kharagpur, India. She is currently an Assistant Professor in the Department of Computer Science at Rama Devi Women's University, Bhubaneswar, India. Prior to this, she was working as Assistant Professor in Central University of Rajasthan, India. Her research interests include Software Testing, Mobile Cloud Computing, and Machine Learning. She holds 20 years of teaching and research experience. She has published several international journals, conference papers, and books. She served as chairs and technical program committee member in several conferences of international repute.

Tien-Hsiung Weng is currently working as a Professor in the Department of Computer Science and Information Engineering, Providence University, Taichung City, Taiwan. He received his PhD in Computer Science from the University of Houston, Texas. His research interests include parallel computing, high performance computing, scientific computing, and machine learning. He has has published several international journals, conference papers, and books.

Received: May 12, 2021; Accepted: September 22, 2021.