Received: 12 June 2019 | Revised: 18 December 2019 | Accepted: 20 January 2020

DOI: 10.4218/etrij.2019-0294

ORIGINAL ARTICLE

ETRI Journal WILEY

Resource-efficient load-balancing framework for cloud data center networks

Jitendra Kumar[1*] | **Ashutosh Kumar Singh**[1] | **Anand Mohan**[2]

[1]Department of Computer Applications, National Institute of Technology, Kurukshetra, India

[2]Department of Electronics Engineering, Indian Institute of Technology, Banaras Hindu University, Varanasi, India

Correspondence
Jitendra Kumar, Department of Computer Engineering and Applications, GLA University, Mathura, India.
Email: jitendrakumar@ieee.org

***Present address**
Jitendra Kumar, Department of Computer Engineering and Applications, GLA University, Mathura, India

Funding information
This research was supported by the Ministry of Electronics & Information Technology (MeitY), Government of India.

Cloud computing has drastically reduced the price of computing resources through the use of virtualized resources that are shared among users. However, the established large cloud data centers have a large carbon footprint owing to their excessive power consumption. Inefficiency in resource utilization and power consumption results in the low fiscal gain of service providers. Therefore, data centers should adopt an effective resource-management approach. In this paper, we present a novel load-balancing framework with the objective of minimizing the operational cost of data centers through improved resource utilization. The framework utilizes a modified genetic algorithm for realizing the optimal allocation of virtual machines (VMs) over physical machines. The experimental results demonstrate that the proposed framework improves the resource utilization by up to 45.21%, 84.49%, 119.93%, and 113.96% over a recent and three other standard heuristics-based VM placement approaches.

KEYWORDS
cloud data center, genetic algorithm, load balancing, multi-objective, resource utilization

1 | INTRODUCTION

Cloud computing has become a salient technology, and its drastic growth has resulted in the development of numerous large-scale data centers across the globe. These data centers consume large amounts of electricity for their smooth operation, which makes them one of the major sources of CO_2 emissions. It is expected that data centers will consume 95 TWh of electricity by 2040 as per a report published by the US Energy Information Administration [1]. A data center consumes more power during its cooling process rather than during its computing operations, as the power consumption during the latter process can be reduced via an effective and optimal usage of its resources [2].

As per a study conducted by IBM, the mean utilizations of the central processing unit (CPU) and memory were only 17.76% and 77.93%, respectively [3]. Similarly, a study conducted using Google cluster trace determined that the utilizations of the CPU and memory were less than 60% and 50%, respectively [4]. The underutilization of resources results in high operational costs owing to the excess usage of electricity. The resources must be used effectively because an active ideal machine consumes more than half the peak power [5]. Therefore, an efficient resource-management scheme that can reduce the operational costs by maximizing the resource utilization and minimizing the power consumption of the cloud system is required [6,7].

Efficient virtual-machine (VM) consolidation can be used to improve the cloud performance, and this includes the selection of an appropriate scheme for migrating VMs and their placement to the most favorable server (PM) [8]. Various approaches have been proposed for solving this

problem, but most of them involve the use of a weighted-sum approach [9–13], wherein each solution is assigned a weight. A multi-objective problem is converted into the form of $\sum_{i=1}^{k} \omega_i f_i$, where ω_i represents the weight corresponding to the i^{th} objective. The fundamental issue with weighted-sum approaches is the selection of optimal weights to be equilibrated for the differences in the objective function and setting weights to indicate the relative importance of an objective [14,15]. In this study, a true multi-objective approach that considers each objective individually to determine the best possible VM allocations is used.

1.1 | Our contributions

This paper aims to present a novel multi-objective resource-efficient load-balancing framework (RELB) for realizing effective VM placement. The key contributions of the paper are as follows: (a) a multi-objective framework is developed to improve the resource utilization and power consumption of a data center and (b) a selection operator guided by non-dominated sorting is introduced. It is capable of selecting the best-suited individuals from the population having multiple cost values associated with the respective objectives of the VM allocation problem.

The rest of this paper is organized as follows. Section 2 summarizes the related work on the load balancing of cloud servers. Section 3 introduces the framework design, which is followed by a detailed discussion on the proposed VM allocation policy in Section 4. Section 5 presents the experimental findings. Finally, the conclusions of this study are presented in Section 6.

2 | RELATED WORK

Modern cloud systems have numerous open research challenges including resource management, load balancing, security, privacy, and multi-clouds [16–22]. In this section, the recent works on load balancing are summarized as this study concentrates on multi-objective load balancing. Numerous approaches have been proposed for optimizing the placement of VMs while considering various aspects such as resource utilization, power consumption, and bandwidth or traffic management [18,20,23]. VM placement approaches can be classified based on the number of objectives being optimized and the optimization scheme being used. Based on the number of objectives, these algorithms can be classified into single-objective and multi-objective solutions [16,22,24]. Furthermore, VM placement schemes can be classified into mono-objective, multi-objective as mono-objective (MAM), and true multi-objective approaches based on the optimization scheme employed [25].

The load-balancing approaches that handle a single objective involve the use of various heuristic-based techniques. For instance, vector bin packing problem generation has been applied to the modeling of VM placement [26]. Similarly, first-fit decreasing and best-fit algorithms have been applied to the development of greedy heuristics to compute the feasible and near-optimal solutions for resource allocation in the distributed computing paradigm [27]. Simulated annealing has also been explored for improving the cost efficiency of data centers by reducing the number of active machines [28]. The mono-objective optimization approach optimizes only one parameter of the problem at a time, and it was observed that 64% of the studies on VM placement belong to this category [25]. Single-objective-based schemes cannot balance the different objectives involved in VM placement or allocation, and researchers have thus started developing multi-objective methods.

Multi-objective optimization algorithms have been of significant interest to researchers in different applications of engineering and optimization. The multi-objective approach has also been applied to the selection of the most suitable host or physical machine (PM) for placing the VMs. The second category of approaches, that is, MAM approaches, optimizes multiple parameters simultaneously; however, these parameters are converted into a single objective by using the weighted sum or other similar approaches. Such proposals account for 32% of the VM placement approaches. The true multi-objective schemes consider multiple objectives for optimization in cases wherein two or more objectives may be conflicting in nature. However, only 4% of existing studies belong to this category [25].

The other parameter used to classify the placement schemes is the implementation approach employed to obtain the optimal or near-optimal mapping. A placement scheme involves the use of an approach such as constraint programming, bin packing, or genetic algorithms. Constraint programming is a form of logic programming that defines the variable relations in a special form referred to as constraints. Usually, the constraints define the characteristics of a solution being searched. The constraints are embedded into a logic program. The VM placement problem is formulated as a mathematical problem with constraints, and a logic program can be developed using constraint-programming logic. Numerous approaches have been proposed using the constraint-programming paradigm. For instance, Yu and Gao formulated a placement problem as a multidimensional bin packing and used the constraint solver to optimize the mapping [29]. The objective of this approach is to minimize the number of active PMs to improve the resource utilization and power consumption. Similarly, Zhang and others developed a resource allocation approach using constraint programming with the objective of reducing the resource usage cost and improving the quality of service [30].

Bin packing is the process of assigning different items to a minimum number of containers referred to as bins. In a standard bin-packing problem, an attempt is made to minimize the number of occupied bins of capacity 1 to pack items of capacity (0, 1). Song and others presented a resource allocation framework that transforms the problem into an online bin-packing problem [31]. The approach is called variable-item-size bin packing, and it is capable of adjusting the available resources. The algorithm also handles run-time changes in the resource demands of the VMs. However, this approach results in increased violations of service-level agreements. VM consolidation is the process of fitting the VMs into a minimum number of servers in the datacenter and is often used for load balancing. Numerous heuristics have been introduced for VM consolidation, and they consider the heterogeneity in resource requirements [32]. The proposed heuristics have been extensively studied and they have achieved a performance similar to that of multidimensional algorithms. The energy-aware VM placement schemes called dynamic round robin and hybrid have been introduced in [33]. The hybrid algorithm fuses the dynamic round robin with the first fit to achieve better placements.

Evolutionary algorithms select the best suitable solution from all the possible solutions in the search space. These algorithms have been widely explored for developing VM placement schemes. For instance, the VM reallocation approach that anticipates the future workload information has been presented, and this information is passed to a genetic-algorithm-based model to determine the optimal reconfiguration of the VMs [34]. The ant-colony-optimization–based (ACO-based) VM placement scheme was introduced in [35], and its objective was to improve the resource utilization and power consumption. The authors model the VM consolidation as a multidimensional vector-packing problem and use vector algebra to model the different objectives. A VM consolidation approach that uses biogeography-based optimization to optimize multiple objectives has been proposed in [36]. This approach is hybridized with differential evolution along with the cosine migration model and Gaussian mutation to obtain candidate solutions of better quality. The approach considers various parameters such as workload, power consumption, and resource utilization for obtaining the best solution for the VM assignment. Tang and Pan presented a hybrid genetic algorithm for obtaining an approximation of the optimal solution for assigning VMs on servers [37]. Their approach considers the power consumption in a communication network of data centers in addition to the power consumption of the servers. Similarly, an ACO-based multi-objective approach has been developed for optimizing the VM assignment in a large data center [38]. The purpose of this approach was to find a set of solutions that minimize the resource wastage and power consumption. Another approach based on the ant-colony system was presented in [39] wherein the VM assignment problem was formulated as a constrained combinatorial optimization problem, and the information of the VMs and servers was used to minimize the power consumption. Several other intelligent approaches have been presented for optimizing the power consumption [40–48]. However, in most of these approaches, the VM assignment problem is converted into a single-objective problem by using the weighted-sum approach. As discussed earlier, the assignment of weights to each objective is crucial, and an inappropriate selection may result in sub-optimal solutions. Therefore, we present a VM allocation model that treats each objective individually and provides a set of solutions.

3 | FRAMEWORK DESIGN

The proposed RELB framework is a multi-objective approach and is presented in Figure 1. In a cloud computing environment, the resources ranging from infrastructure to software are shared among multiple users. A load-balancing scheme assigns each task to different computing servers based on criteria such as utilization, response time, and quality of service (QoS). The objective of a load-balancing

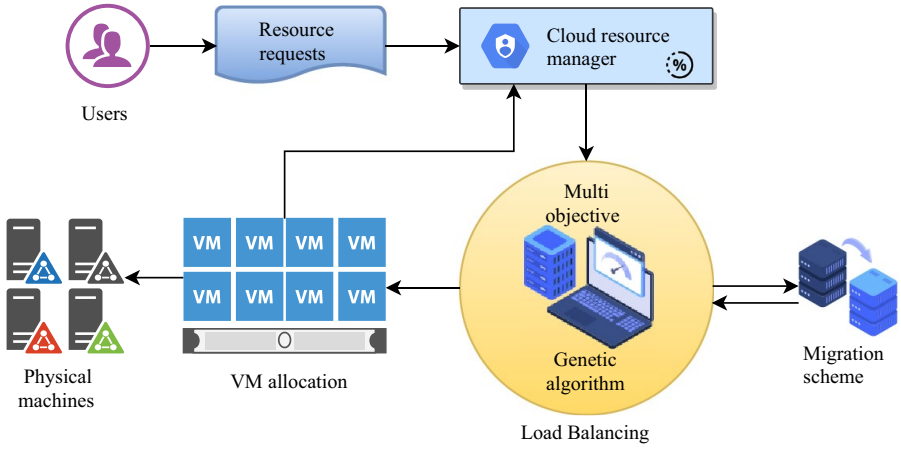

FIGURE 1 RELB framework schematic workflow

scheme could be to optimize a single objective or multiple objectives. However, an ideal load-balancing scheme should be capable of optimizing multiple objectives simultaneously as the assignment of cloud resources directly affects several other system parameters. An optimization problem is considered a multi-objective problem if multiple objectives are to be optimized simultaneously. Mathematically, this can be stated as follows:

$$\min\left(f_1(y), f_2(y), \ldots, f_k(y)\right) \quad \text{s.t.}\, y \in Y,$$

where k defines the number of objective functions, which should be greater than 1, that is, $k \geq 2$, and Y is the feasible set of decision vectors or constraint functions.

The RELB framework assigns the VMs to PMs in a manner such that the resource utilization is maximized. Its objective is to minimize the number of active machines to reduce the power consumption and carbon footprint. The users send the resource demands to the cloud-resource manager, which keeps track of the resource states and passes this information to the load-balancing unit. The framework executes the resource-efficient load-balancing algorithm to determine the near-optimal allocation. If an infeasible solution is observed during the optimization process, the infeasible allocations are migrated to feasible servers. The framework uses a genetic algorithm to approximate the VM allocation, which is an optimization approach inspired from genetic evolution. A set of randomly generated solutions in the search space is evolved toward better solutions. Each solution is encoded with a set of properties that can be altered over time using genetic operators such as mutation and crossover. The candidate solutions are evaluated using a fitness function, and new solutions are generated in each iteration. The better solutions among the old and new solutions are stochastically selected for the next iteration.

4 | MULTI-OBJECTIVE VM PLACEMENT

The objective of the VM placement process is to allocate the VM on the most suitable PM or server. The process is required to consider multiple factors such as resources and performance. The clients (users) and service providers have different objectives and priorities in a sufficiently large distributed and virtualized computing environment. For instance, the users may prioritize the response time, QoS, and other parameters, whereas the service providers may seek to minimize the operational cost of the data center for maximizing their revenue. The objective of the proposed framework is to reduce the operational cost of a data center by maximizing the resource utilization and minimizing the power consumption. The framework attempts to find a suitable PM to host a VM in such a manner that both objectives are reasonably achieved.

Let us consider that n VMs (VM_1, VM_2, …, VM_n) of p users (U_1, U_2, …, U_p) are hosted in a data center DC that consists of m PMs or servers (PM_1, PM_2, …, PM_m). Let us also consider that α is a vector of size m that shows the status of each PM. The PM_j is said to be active ($\alpha_j = 1$) if one or more VMs are hosted on the server j; else, PM_j is in the ideal state ($\alpha_j = 0$). Similarly, β is a matrix of size $n \times m$ that keeps track of the VM placement. If PM_j hosts VM_i, then $\beta_{ij} = 1$; else, $\beta_{ij} = 0$. Let PM_j have $\mathrm{PM}_j^{\mathrm{CPU}}$ and $\mathrm{PM}_j^{\mathrm{MEMORY}}$ capacities of the CPU and memory, respectively. The utilizations of the CPU and memory of VM_i are $\mathrm{VM}_i^{\mathrm{CPU}}$ and $\mathrm{VM}_i^{\mathrm{MEMORY}}$, respectively. The resource utilization of each resource is monitored independently, and the CPU utilization of each PM is computed using (1), which adds the CPU utilization of each allocated VM and divides it by the CPU capacity of the server. Similarly, the memory utilization of each server is observed using (2).

$$\mathrm{RU}_j^{\mathrm{CPU}} = \frac{\sum_{i=1}^{n} \beta_{ij} \times \mathrm{VM}_i^{\mathrm{CPU}}}{\mathrm{PM}_j^{\mathrm{CPU}}} \tag{1}$$

and

$$\mathrm{RU}_j^{\mathrm{MEMORY}} = \frac{\sum_{i=1}^{n} \beta_{ij} \times \mathrm{VM}_i^{\mathrm{MEMORY}}}{\mathrm{PM}_j^{\mathrm{MEMORY}}}. \tag{2}$$

The CPU and memory resources are used to compute the resource utilization. However, the framework is sufficiently general to accommodate and consider any number of additional resources such as a disk. The average resource utilization of the data center ($\mathrm{RU_{DC}}$) is computed using (3), and the objective of the proposed allocation scheme is to maximize it. Here, $|\mathrm{R}|$ represents the number of resources under consideration, that is, $|\mathrm{R}| = 2$.

$$\mathrm{RU_{DC}} = \frac{\sum_{j=1}^{m} \mathrm{RU}_j^{\mathrm{CPU}} + \sum_{j=1}^{m} \mathrm{RU}_j^{\mathrm{MEMORY}}}{|\mathrm{R}| \times \sum_{j=1}^{m} \alpha_j}. \tag{3}$$

A large-scale data center produces a large amount of heat during its operation. Therefore, a major part of its total consumed power is used to maintain the operational temperature. After the cooling infrastructure, the highest power consumption is that of the CPU [49], and the resource-utilization rate has a close relationship with the power consumption [50–52]. The energy-saving schemes applied in the case of a single CPU follow the CPU states, that is, busy or ideal. In a busy state, the energy consumed by a processor depends on multiple variables, such as the utilization rate. However, in the ideal state, some of the components are switched off by the resource manager, and thus, the operating frequency is also reduced to reduce the power consumption. The framework

uses power consumption modeling to measure the power consumption based on the resource utilization [53]. This model has been widely used in the design of energy-aware VM placement approaches [40,46–48].

Considering that PW_j^{min} and PW_j^{max} are the minimum and maximum powers required by PM_j, respectively, PM_j also requires PW_j^{idle} amount of power in the ideal state. Therefore, the power requirements or consumption of PM_j can be calculated using (4). Thus, the power consumed by the data center can be obtained by adding the power consumed by each server, as shown in (5). The objective of the proposed framework is to minimize PW_{DC}, that is, $\min_{\forall PM_j} \sum_{j=1}^{m} PW_j$.

$$PW_j = \left(PW_j^{max} - PW_j^{min}\right) \times RU_j + PW_j^{idle} \tag{4}$$

and

$$PW_{DC} = \sum PW_j; \quad \forall j = \{1, 2, \ldots, m\}. \tag{5}$$

4.1 | Proposed algorithm

As determining the optimal allocation of n VMs on m servers strictly belongs to the NP-hard class of problems [54,55], no existing algorithm can be used to find an optimal solution in polynomial time. Therefore, the proposed framework approximates the allocation using a genetic algorithm. The model iteratively improves the VM allocation while obtaining near-optimal solutions. Algorithm 1 shows the pseudocode of the approach, wherein a population (ϵ) of n random solutions (line 3) is first initialized. A solution places the i^{th} VM into a randomly selected j^{th} PM that is active and satisfies the constraints listed in (6) and (7). Both the constraints are used to ensure that no VM is allocated to a server if sufficient resources are unavailable.

$$\sum_{i=1}^{n} VM_i^{CPU} \times \beta_{ij} \leq PM_j^{CPU}; \quad \forall j \in \{1, 2, \ldots, m\} \tag{6}$$

$$\sum_{i=1}^{n} VM_i^{MEMORY} \times \beta_{ij} \leq PM_j^{MEMORY}; \quad \forall j \in \{1, 2, \ldots, m\}. \tag{7}$$

The solution or chromosome (χ) is a vector of length m that represents the list of VMs hosted on each PM. Let VP_j be the set of VMs hosted on PM_j, that is, for instance, $VP_j = \{VM_1, VM_3, VM_8\}$ indicates that PM_j is hosting VMs indexed as 1, 3, and 8. Thus, a chromosome solution may be represented as $\chi = (VP_1, VP_2, \ldots, VP_m)$. Let us assume that PM_1, PM_2, PM_3, and PM_4 have the CPU (mips) and memory (MB) capacities of (2500, 4000), (1000, 500), (500, 1000), and (1500, 3000), respectively. We consider that VM_1, VM_2, …, VM_{10} with the resource requests (250, 100), (200, 500), (500, 200), (100, 500), (1000, 1000), (500, 1000), (200, 500), (500, 1000), (200, 700), and (1000, 500) are hosted among these machines as per the configuration shown in (8). The chromosome of the aforementioned VM placement is presented in Figure 2.

$VM_4, VM_6, VM_8, VM_{10}$	VM_1, VM_3	VM_2	VM_5, VM_7, VM_9

FIGURE 2 Chromosome encoding (placement of VMs)

$$\beta = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{8}$$

Algorithm 1 RELB framework pseudocode

1: Randomly generate N feasible solutions $\epsilon^0 = \{\chi_1, \chi_2, \ldots, \chi_N\}$
2: **for** each $i = (1, 2, \ldots, N)$ **do**
3: $[f_{\chi_i}^{RU}, f_{\chi_i}^{PW}] = \Psi(\chi_i, DC^{\kappa})$
4: **end for**
5: **while** termination criterion is not met **do**
6: $O^c = [], O^m = []$
7: **for** each $i = (1, 2, \ldots, N)$ **do**
8: $idx = rand(1, N)$ s.t. $i \neq idx$
9: $[o_1^c, o_2^c] = \mathcal{Q}(\chi_i, \chi_{idx}, DC^{\kappa})$
10: $[o^m] = \mathcal{Z}(\chi_i, DC^{\kappa})$
11: $O^c = [O^c, o_1^c, o_2^c]$
12: $O^m = [O^m, o^m]$
13: **end for**
14: **for** each $i = (1, 2, \ldots, 2N)$ **do**
15: $[f_{o_i^c}^{RU}, f_{o_i^c}^{PW}] = \Psi(o_i^c, DC^{\kappa})$
16: **end for**
17: **for** each $i = (1, 2, \ldots, N)$ **do**
18: $[f_{o_i^m}^{RU}, f_{o_i^m}^{PW}] = \Psi(o_i^m, DC^{\kappa})$
19: **end for**
20: $[\epsilon^{g+1}] = \mathcal{S}(\epsilon^g, O^c, O^m)$
21: **end while**

Each solution (χ_i) is evaluated using a cost function (Ψ) and the fitness values ($f_{\chi_i}^{RU}$ and $f_{\chi_i}^{PW}$) associated with both the objectives that are computed. The Ψ evaluates any given allocation by computing $f_{\chi_i}^{RU}$ and $f_{\chi_i}^{PW}$ for resource utilization

Algorithm 2 Non-dominated-sorting-based selection operator

```
1:  function S(ε^g, O^c, O^m)
2:      for each χ_i^g do
3:          for each χ_j^g;   ∀j ∈ {1, 2, ..., N} and i ≠ j do
4:              S_i = ∅
5:              n_i = 0
6:              if χ_i^g ≺ χ_j^g then
7:                  S_i = S_i ∪ {χ_j^g}
8:              end if
9:              χ_i^g.r = 1
10:             if n_i == 0 then
11:                 F_1 = F_1 ∪ {χ_i^g}
12:             end if
13:         end for
14:     end for
15:     i = 1
16:     while F_i ≠ ∅ do
17:         Q = ∅
18:         for each χ_i^g ∈ F_i do
19:             for each χ_j^g ∈ S_i do
20:                 n_j = n_j − 1
21:                 χ_j^g.r = i + 1
22:                 if n_j == 0 then
23:                     Q = Q ∪ {χ_j^g}
24:                 end if
25:             end for
26:         end for
27:     end while
28:     i = i + 1
29:     F_i = Q
30:     ε^{g+1} = Select first N solutions from F_i   ∀i
31:     return ε^{g+1}
32: end function
```

and power consumption, respectively. To learn the best possible assignment of VMs, an iterative process that includes the mutation, crossover, and selection as key operators is applied.

The crossover ($\mathcal{Q}$) is a recombination operator that mimics the reproduction system of genetic evolution wherein two individuals (parents) meet to generate offspring solutions (O), where o^{c} and o^{m} are the newly generated solutions from crossover and mutation operator respectively. We applied the single-point crossover, wherein a random point in the range [1, D] is selected, and the tails of both the individuals are exchanged, as shown in Figure 3, where D is the length of the candidate solution. The mutation operator allows the candidate solutions to explore the search space. In the approach, a swap-operation-based mutation operator ($\mathcal{Z}$) is applied, as shown in Figure 4. Here two distinct VM allocations are selected randomly, and their positions are exchanged accordingly. The newly generated solutions are evaluated using a cost function. The N solutions are selected for the next iteration by applying the selection operator ($\mathcal{S}$), as shown in Algorithm 2, which uses non-dominated sorting to rank the solutions and selects the best N solutions for the next iteration.

4.2 | Complexity analysis

This section discusses the functions used in detail and presents an analysis of the time complexity of the approach. The line 1 of the RELB framework (Algorithm 1) randomly initializes n solutions. The framework first scans for a feasible PM and assigns the VM to an identified server that takes $\mathcal{O}(m)$. In the process, n VMs are assigned in each solution. Therefore, the time elapsed in initializing N solutions becomes $\mathcal{O}(nmN)$. Next, in lines 2–4, each individual is evaluated, and the fitness values corresponding to both objectives are computed by applying the Ψ function, which consumes $\mathcal{O}(mN)$. Lines 9 and 10 correspond to the recombination operators, crossover ($\mathcal{Q}$) and mutation ($\mathcal{Z}$), respectively. The single-point crossover is applied and randomly selects a distinct individual and exchanges its tail with the i^{th} individual. It consumes $\mathcal{O}(m)$ for performing the operation once. However, it can generate infeasible solutions, and therefore, the feasibility of the solutions must be verified. The feasibility of the solutions is verified by comparing the amount of resources requested by the assigned VMs with the capacity of servers that consume $\mathcal{O}(m)$. If any infeasible solution is detected, the appropriate VMs are migrated to other servers. As the approach first reallocates the VM that consumes the highest resources, it is very unlikely that a large number of VMs are required to be migrated. Assuming

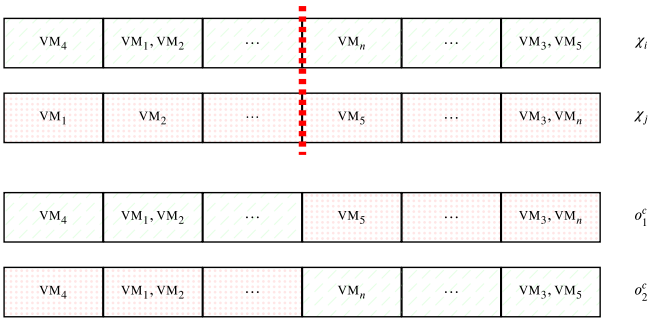

FIGURE 3 Crossover operator

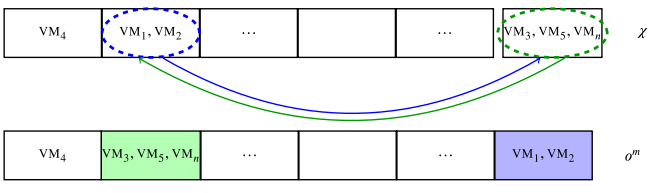

FIGURE 4 Mutation operator

TABLE 1 Server configuration

Name	PE	MIPS	Memory	PW^{max}	PW^{min}	PW^{idle}
PM_{T_1}	2	2660	4096	135	93.7	93.7
PM_{T_2}	4	3067	8172	113	42.3	42.3
PM_{T_3}	12	3067	16 384	222	58.4	58.4

TABLE 2 VM configuration

Name	PE	MIPS	Memory
VM_{T_1}	1	500	512
VM_{T_2}	2	1000	1024
VM_{T_3}	3	1500	2048
VM_{T_4}	4	2000	3072

that the migration process migrates k VMs, where $k \ll n$, it consumes $\mathcal{O}(m)$. Therefore, the crossover operator requires $\mathcal{O}(m)+\mathcal{O}(m)+\mathcal{O}(m)$, that is, $\mathcal{O}(m)$. Similarly, the mutation operation interchanges the assignments of two randomly selected VMs that may be performed in $\mathcal{O}(1)$. This operation also migrates the infeasible allocations that may be produced during mutation. Therefore, the total time elapsed in the mutation becomes $\mathcal{O}(m)$. As both operations are iterated N times, the total time elapsed in lines 7–13 becomes $\mathcal{O}(mN)$. In lines 14–19, the newly generated solutions and elapsed $\mathcal{O}(mN)$ are evaluated. In line 20, the population is selected for the next iteration using the selection (S) operator described in Algorithm 2, which takes $\mathcal{O}\left(kN^2\right)$, where k is the number of objectives to be optimized. The total time elapsed in lines 6–20 becomes $\mathcal{O}(mN)+\mathcal{O}(mN)+\mathcal{O}\left(kN^2\right)$. As k is a relatively small number, it can be written as $\mathcal{O}\left(mN+N^2\right)$. As the process is repeated g times (maximum iterations), the total time becomes $\mathcal{O}\left(g\left(mN+N^2\right)\right)$. After adding the initialization cost, the total cost obtained is $\mathcal{O}\left(g\left(mN+N^2\right)+nmN\right)$.

5 | RESULTS AND DISCUSSION

We conducted a series of simulations on a machine equipped with an Intel Xeon CPU E5-2630 having 128 GB of memory. The CPU and memory utilizations are measured in millions of instructions per second "mips" and megabytes "MB," respectively. The simulation environment consisted of three types of servers (HP and IBM) with the CPU and memory capacities of (2660, 4096), (3067, 8172), and (3067, 16 384), as shown in Table 1 [56–58], where "PE" defines the number of processing elements, and "MIPS" measures the capacity or speed of the machine in millions of instructions per second. Four types of VMs having the CPU and memory requests of (500, 512), (1000, 1024), (1500, 2048), and (2000, 3072) are used, as shown in Table 2. The efficacy of the approach is evaluated using 20, 60, 100, 300, and 500 VMs running on the data center, which is sufficiently capable of hosting all the VMs. The simulation environment ensures that sufficient resources are available to accommodate the VM requests, and the number of servers for each type is selected as $\left\lfloor \frac{\#VMs}{\text{type of servers}} \right\rfloor$, for example, $\left\lfloor \frac{100}{3} \right\rfloor$ to experiment with

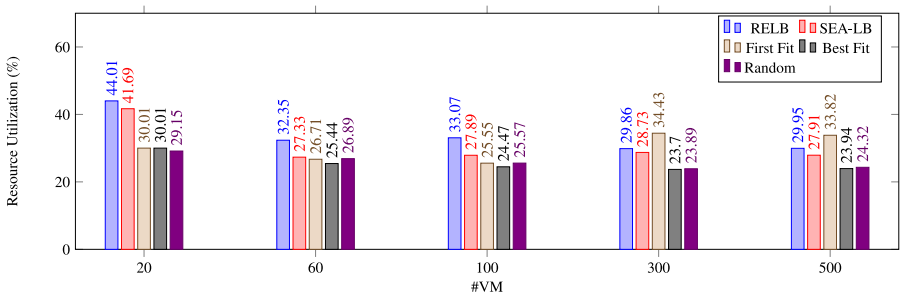

FIGURE 5 Resource utilization for homogeneous VM requests

TABLE 3 Power consumption (W) for homogeneous VM requests

VMs	RELB	SEALB	Best fit	First fit	Random
20	3.42E+04	4.34E+04	3.83E+04	3.99E+04	4.42E+04
60	1.16E+05	1.18E+05	1.09E+05	1.29E+05	1.41E+05
100	2.15E+05	2.14E+05	1.88E+05	2.06E+05	2.42E+05
300	6.27E+05	6.49E+05	5.62E+05	6.96E+05	7.20E+05
500	1.04E+06	1.06E+06	9.15E+05	1.18E+06	1.21E+06

100 VMs. We evaluated the performance of the algorithm in two different scenarios, that is, homogeneous and heterogeneous requests of VMs. In the homogeneous environment, each VM requests the same amount of resources, whereas in the heterogeneous environment, the VMs are not bound to request the same amount of resources. We also compared the performance of the approach with the performances of one of the most recent approaches, that is, the secure and energy-aware load-balancing approach (SEALB), and three heuristic-based standard VM allocation schemes [59].

5.1 | Homogeneous VM requests

This section presents an evaluation of the performance of the proposed scheme for homogeneous VM requests. The resource utilization realized using each approach is presented in Figure 5 for various numbers of VMs hosted in the data center. The resource utilization of the proposed approach is better than that of other approaches. The maximum relative improvement realized using the proposed scheme is up to 7.29%, 46.64%, 46.64%, and 50.99% over the SEALB, best-fit, first-fit, and random heuristics-based allocation approaches, respectively. However, the proposed scheme could not outperform the first-fit-based allocation approach for 300 and 500 VMs because the first-fit approach selects a PM based on its availability and index, whereas the proposed scheme follows the convention of sorting the PMs based on the resource availability and selects the one that can host the VM. The first-fit-based allocation approach improved the marginal resource utilization by a maximum factor of ≈13%. The amounts of power consumed by the data center for homogeneous VM requests with different approaches are presented in Table 3. The proposed framework also reduced the power requirements for the data-center operation.

5.2 | Heterogeneous VM requests

A typical data center hosts VMs having various requirements. Therefore, we also conducted an in-depth experimental analysis with heterogeneous VM requests. The resource utilizations of the approaches are presented and compared in Figure 6. The resource utilization of the proposed scheme is better than that of all other approaches in all the cases considered. The maximum relative improvement achieved with the proposed scheme is up to 45.21%, 84.49%, 119.93%, and 113.96% over the SEALB, best-fit, first-fit, and random-based allocation approaches, respectively. The proposed approach outperforms the secure and energy-aware load-balancing scheme in resource utilization, which reflects the cost involved in achieving secure load balancing. Each workload may not require security during the VM placement in real-world scenarios owing to various reasons, for example, overhead costs and insensitive data. Therefore, a load balancer should be capable of considering such requests and placing them appropriately. The power consumptions of all the approaches are listed in Table 4. The power consumption of the proposed approach is better than that of the best-fit and random placement approaches. However, the SEALB and first-fit placement schemes consume less power.

These results indicate that the workload of the cloud data center must be allocated carefully to reduce the overhead costs of service providers. Other parameters such as the

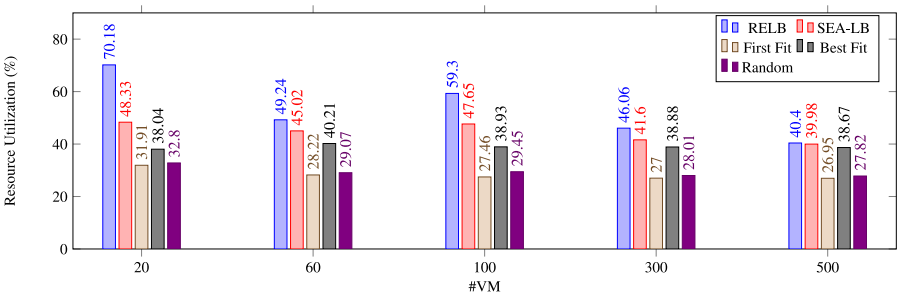

FIGURE 6 Resource utilization for heterogeneous VM requests

VM	RELB	SEALB	BF	FF	RND
20	4.31E+04	3.90E+04	4.80E+04	4.30E+04	5.00E+04
60	1.51E+05	1.40E+05	1.70E+05	1.40E+05	1.70E+05
100	2.69E+05	2.40E+05	2.90E+05	2.40E+05	2.90E+05
300	8.62E+05	7.50E+05	8.60E+05	7.30E+05	8.70E+05
500	1.35E+06	1.30E+06	1.40E+06	1.20E+06	1.40E+06

TABLE 4 Power consumption (W) for heterogeneous VM requests

response time and QoS can also be improved using an effective load-balancing scheme.

6 | CONCLUSION

The realization of load balancing in a dynamic cloud system has always been a challenging task. This paper presented a load-balancing approach for optimizing the resource utilization and power consumption of data centers. The approach involved the use of a genetic algorithm to find an approximated solution. A non-dominated sorting-based selection operator is used to rank the solutions that have multiple fitness values associated with each objective. The proposed scheme achieved a significant improvement over one of the most recent and three other standard heuristics-based load-balancing approaches. The experimental results validate the efficacy of the proposed approach. In the future, a load-balancing framework that considers more than two dimensions of load balancing can be introduced for improving the resource utilization and reducing the power consumption.

CONFLICT OF INTEREST

The authors declare that they have no known competing financial interests or personal relationships that influenced the work reported in this paper.

ORCID

Jitendra Kumar https://orcid.org/0000-0002-2938-6432

REFERENCES

1. Navigant Consulting Inc. *SAIC, Analysis and Representation of Miscellaneous Electric Loads in NEMS*, prepared for the U.S. Energy Information Administration, Navigant Reference: 160750, 2017, pp. 1–138.
2. J. Kumar and A. K. Singh, *Cloud datacenter workload estimation using error preventive time series forecasting models*, Cluster Comput (2019), in press.
3. R. Birke et al., *Data Centers in the Wild: A Large Performance Study*, Tech. report, IBM Research - Zurich, Switzerland, 2012.
4. C. Reiss et al., *Heterogeneity and dynamicity of clouds at scale: google trace analysis*, in Proc. ACM Symp. Cloud Comput. (San Jose, CA, USA), Oct., 2012, pp. 1–18.
5. L. Barroso, J. Clidaras, and U. Hölzle, *The Datacenter as a Computer An Introduction to the Design of Warehouse-Scale Machines*, 2 ed, Morgan & Claypool Publishers, 2013.
6. J. Kumar and A. K. Singh, *Workload prediction in cloud using artificial neural network and adaptive differential evolution*, Future Generation Comput. Syst. **81** (2018), 41–52.
7. J. Kumar, R. Goomer, and A. K. Singh, *Long short term memory recurrent neural network (LSTM-RNN) based workload forecasting model for cloud datacenters*, Procedia Comput. Sci. **125** (2018), 676–682.
8. P. D. Bharathi, P. Prakash, and M. V. K. Kiran, *Virtual machine placement strategies in cloud computing*, in Proc. Innovations Power Adv. Comput. Technol. (Vellore, India), Apr. 2017, pp. 1–7.
9. A. C. Adamuthe, R. M. Pandharpatte, and G. T. Thampi, *Multiobjective virtual machine placement in cloud environment*, in Proc. Int. Conf. Cloud Ubiquitous Comput. Emerg. Technol. (Pune, India), Nov. 2013, pp. 8–13.
10. T. Ferreto, C. A. F. De Rose, and H. Heiss, *Maximum migration time guarantees in dynamic server consolidation for virtualized data centers*, E Jeannot, R Namyst, and J Roman (eds), Euro-Par 2011 Parallel Processing, Springer Berlin Heidelberg: Berlin, Heidelberg, 2011, pp. 443–454.
11. S. Shigeta et al., *Design and implementation of a multi-objective optimization mechanism for virtual machine placement in cloud computing data center*, M. Yousif and L. Schubert (eds), Cloud Computing (Cham), Springer International Publishing, 2013, pp. 21–31.
12. J. Xu and J. A. B. Fortes, *Multi-objective virtual machine placement in virtualized data center environments*, in Proc. IEEE/ACM Int. Conf. Green Comput. Commun. Int. Conf. Cyber, Phys. Social Comput. (Hangzhou, China), Dec. 2010, pp. 179–188.
13. F. Tseng et al., *Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm*, IEEE Syst J. **12** (2018), no. 2, 1688–1699.
14. N. Gunantara, *A review of multi-objective optimization: methods and its applications*, Cogent Eng. **5** (2018), no. 1, 1502242:1–16.
15. R. T. Marler and J. S. Arora, *The weighted sum method for multi-objective optimization: new insights*, Structural Multidisciplinary Optimization **41** (2010), no. 6, 853–862.
16. F. Fang and B. B. Qu, *Multi-objective virtual machine placement for load balancing*, ITM Web Conf.: Int. Conf. Inf. Sci. Technol. **11** (2017), 01011:1–9.
17. J. Kumar and A. K. Singh, *Cloud resource demand prediction using differential evolution based learning*, in Proc. Int. Conf. Smart Comput. Commun. (Sarawak, Malaysia), June 2019, pp. 1–5.
18. J. Zhang et al., *Load balancing in data center networks: a survey*, IEEE Commun. Surveys Tutorials **20** (2018), no. 3, 2324–2352.
19. J. Kumar and A. K. Singh, *Dynamic resource scaling in cloud using neural network and black hole algorithm*, in Proc. Int. Conf. Eco-friendly Comput. Commun. Syst. (Bhopal, India), Dec 2016, pp. 63–67.
20. I. Cuadrado-Cordero, A. Orgerie, and C. Morin, *GRaNADA: A network-aware and energy-efficient PaaS cloud architecture*, in Proc. IEEE Int. Conf. Data Sci. Data Intensive Syst. (Sydney, Australia), Dec. 2015, pp. 412–419.
21. J. Kumar and A. K. Singh, *An efficient machine learning approach for virtual machine resource demand prediction*, Int. J. Adv. Sci. Technol. **123** (2019), 21–30.
22. I. De Falco et al., *Effective processor load balancing using multi-objective parallel extremal optimization*, in Proc. Genetic Evolutionary Comput. Conf. Companion (New York, NY, USA), July 2018, pp. 1292–1299.
23. Z. Á. Mann, *Multicore-aware virtual machine placement in cloud data centers*, IEEE Trans. Comput. **65** (2016), no. 11, 3357–3369.
24. F. Ramezani et al., *A multi-objective load balancing system for cloud environments*, Comput. J. **60** (2017), no. 9, 1316–1337.
25. F. L. Pires and B. Baràn, *A virtual machine placement taxonomy*, in Proc. IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (Shenzhen, China), May 2015, pp. 159–168.

26. M. Gabay and S. Zaourar, *Vector bin packing with heterogeneous bins: application to the machine reassignment problem*, Ann. Oper. Res. **242** (2016), no. 1, 161–194.
27. P. Silva, C. Perez, and F. Desprez, *Efficient heuristics for placing large-scale distributed applications on multiple clouds*, in Proc. IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (Cartagena, Colombia), May 2016, pp. 483–492.
28. A. Marotta and S. Avallone, *A simulated annealing based approach for power efficient virtual machines consolidation*, in Proc. IEEE Int. Conf. Cloud Comput. (New York, USA), June 2015, pp. 445–452.
29. Y. Yu and Y. Gao, *Constraint programming-based virtual machines placement algorithm in datacenter*, Z. Shi, D. Leake, and S. Vadera (eds), Intelligent Information Processing VI, Springer Berlin Heidelberg: Berlin, Heidelberg, 2012, pp. 295–304.
30. L. Zhang, Y. Zhuang, and W. Zhu, *Constraint programming based virtual cloud resources allocation model*, Int. J. Hybrid Inf. Technol. **6** (2013), no. 6, 333–344.
31. W. Song et al., *Adaptive resource provisioning for the cloud using online bin packing*, IEEE Trans. Comput. **63** (2014), no. 11, 2647–2660.
32. Y. Zhang and N. Ansari, *Heterogeneity aware dominant resource assistant heuristics for virtual machine consolidation*, in Proc. IEEE Global Commun. Conf. (Atlanta, GA, USA), Dec. 2013, pp. 1297–1302.
33. C. Lin, P. Liu, and J. Wu, *Energy-efficient virtual machine provision algorithms for cloud systems*, in Proc. IEEE Int. Conf. Utility Cloud Comput. (Victoria, Australia), Dec. 2011, pp. 81–88.
34. H. Mi et al., *Online self-configuration with performance guarantee for energy-efficient large-scale cloud computing data centers*, in Proc. IEEE Int. Conf. Services Comput. (Miami, FL, USA), July 2010, pp. 514–521.
35. Md H Ferdaus et al., *Virtual machine consolidation in cloud data centers using ACO metaheuristic*, F. Silva, I. Dutra, and V. S. Costa (eds), Euro-Par 2014 Parallel Processing, Springer International Publishing, Porto, 2014, pp. 306–317.
36. Q. Zheng et al., *Multi-objective optimization algorithm based on bbo for virtual machine consolidation problem*, in Proc. IEEE Int. Conf. Parallel Distr. Syst. (Melbourne, Australia), Dec. 2015, pp. 414–421.
37. M. Tang and S. Pan, *A hybrid genetic algorithm for the energy efficient virtual machine placement problem in data centers*, Neural Process. Lett. **41** (2015), no. 2, 211–221.
38. Y. Gao et al., *A multi-objective ant colony system algorithm for virtual machine placement in cloud computing*, J. Comput. Syst. Sci. **79** (2013), no. 8, 1230–1242.
39. F. Alharbi et al., *An ant colony system for energy-efficient dynamic virtual machine placement in data centers*, Expert Syst. Appl. **120** (2019), 228–238.
40. N. Sharma and R. M. Guddeti, *Multi-objective energy efficient virtual machines allocation at the cloud data center*, IEEE Trans. Serv. Comput. **12** (2018), no. 1, 158–171.
41. M. Dabbagh et al., *Energyefficient resource allocation and provisioning framework for cloud data centers*, IEEE Trans. Netw. Serv. Manage. **12** (2015), no. 3, 377–391.
42. X. Zhang et al., *Energy-aware virtual machine allocation for cloud with resource reservation*, J. Syst. Softw. **147** (2019), 147–161.
43. Z. Xiao, W. Song, and Q. Chen, *Dynamic resource allocation using virtual machines for cloud computing environment*, IEEE Trans. Parallel Distrib. Syst. **24** (2013), no. 6, 1107–1117.
44. S. Chhabra and A. K. Singh, *Dynamic hierarchical load balancing model for cloud data centre networks*, Electron. Lett. **55** (2019), 94–96.
45. Y. Zhang, J. Yao, and H. Guan, *Intelligent cloud resource management with deep reinforcement learning*, IEEE Cloud Comput. **4** (2017), no. 6, 60–69.
46. Z. Li et al., *An exploration of designing a hybrid scale-up/out hadoop architecture based on performance measurements*, IEEE Trans. Parallel Distrib. Syst. **28** (2017), no. 2, 386–400.
47. N. J. Kansal and I. Chana, *Energy-aware virtual machine migration for cloud computing - a firefly optimization approach*, J. Grid Comput. **14** (2016), no. 2, 327–345.
48. M. Bala and D. Padha, *An adaptive overload detection policy based on the estimator sn in cloud environment*, Int. J. Serv. Sci. Manag. Eng. Technol. **8** (2017), no. 3, 93–107.
49. D. M. Quan et al., *T-alloc: A practical energy efficient resource allocation algorithm for traditional data centers*, Future Generation Comput. Syst. **28** (2012), no. 5, 791–800.
50. W. Yan, J. Chen, and L. Li, *A power-aware aco algorithm for the cloud computing platform*, in Proc. Int. Conf. Commun. Inf. Process. (New York, NY, USA), Nov. 2018, pp. 1–6.
51. E. Arianyan, H. Taheri, and S. Sharifian, *Novel heuristics for consolidation of virtual machines in cloud data centers using multi-criteria resource management solutions*, J. Supercomput. **72** (2016), no. 2, 688–717.
52. N. Akhter and M. Othman, *Energy aware resource allocation of cloud data center: review and open issues*, Cluster Comput. **19** (2016), no. 3, 1163–1182.
53. L. Minas and B. Ellison, *Energy efficiency for information technology: How to reduce power consumption in servers and data centers*, Intel Press, 2009.
54. X. Li et al., *Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center*, Math. Comput. Modelling **58** (2013), no. 5, 1222–1235.
55. F. Farahnakian et al., *Using ant colony system to consolidate vms for green cloud computing*, IEEE Trans. Serv. Comput. **8** (2015), no. 2, 187–198.
56. SPECpower, Accessed: 2019-12-18 [http://www.spec.org/power_ssj2008/results/res2011q1/power_ssj2008-20110124-00339.html].
57. SPECpower, Accessed: 2019-12-18 [http://www.spec.org/power_ssj2008/results/res2011q2/power_ssj2008-20110406-00368.html].
58. SPECpower, Accessed: 2019-12-18 [http://www.spec.org/power_ssj2008/results/res2010q4/power_ssj2008-20101001-00297.html].
59. A. K. Singh and J. Kumar, *Secure and energy aware load balancing framework for cloud data centre networks*, Electron Lett. **55** (2019), 540–541.

AUTHOR BIOGRAPHIES

Jitendra Kumar received his PhD in machine learning from the National Institute of Technology, Kurukshetra, India (An Institution of National Importance) in 2019. Currently, he is an assistant professor with the Department of Computer Engineering & Applications, GLA University, Mathura, India. Dr. Kumar has published several papers in international journals and conferences of high repute. His current research interests include machine learning, cloud computing, data analytics, and parallel processing.

Ashutosh Kumar Singh is currently working as a professor and Head of the Department of Computer Applications, National Institute of Technology, Kurukshetra, India. He has more than 19 years of research and teaching experience in various universities of India, the UK, and Malaysia. He received his PhD in electronics engineering from Indian Institute of Technology, BHU, India, and Post Doc from the Department of Computer Science, University of Bristol, the UK. He is also a Chartered Engineer of the UK. His research areas include verification, synthesis, design, and testing of digital circuits; data science; cloud computing; machine learning; security; and big data. He has published more than 225 research papers in various journals, conferences, and news magazines.

Anand Mohan has nearly 44 years of experience in teaching and research and the administration and management of higher educational institutions. He is currently an institute professor in the Department of Electronics Engineering, Indian Institute of Technology (BHU), Varanasi, India. Besides his present academic assignment, Prof. Mohan is a Member of the Executive Council of Banaras Hindu University and Vice-Chairman of the Board of Governors of Indian Institute of Technology (BHU), Varanasi, India. Prof. Mohan served as Director (June 2011--June 2016) of the National Institute of Technology (NIT), Kurukshetra, Haryana, India and was also Mentor Director of the National Institute of Technology, Srinagar, Uttarakhand, India. For his outstanding contributions in the field of Electronics Engineering, Prof. Mohan was conferred the "Lifetime Achievement Award" (2016) by Kamla Nehru Institute of Technology, Sultanpur, India.