Gopu *et al. Journal of Cloud Computing* (2023) 12:124
https://doi.org/10.1186/s13677-023-00501-y

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Energy-efficient virtual machine placement in distributed cloud using NSGA-III algorithm

Arunkumar Gopu[1], Kalaipriyan Thirugnanasambandam[2], Rajakumar R[3], Ahmed Saeed AlGhamdi[4], Sultan S. Alshamrani[5], K. Maharajan[6] and Mamoon Rashid[7*]

Abstract

Cloud computing is the most widely adapted computing model to process scientific workloads in remote servers accessed through the internet. In the IaaS cloud, the virtual machine (VM) is the execution unit that processes the user workloads. Virtualization enables the execution of multiple virtual machines (VMs) on a single physical machine (PM). Virtual machine placement (VMP) strategically assigns VMs to suitable physical devices within a data center. From the cloud provider's perspective, the virtual machine must be placed optimally to reduce resource wastage to aid economic revenue and develop green data centres. Cloud providers need an efficient methodology to minimize resource wastage, power consumption, and network transmission delay. This paper uses NSGA-III, a multi-objective evolutionary algorithm, to simultaneously reduce the mentioned objectives to obtain a non-dominated solution. The performance metrics (Overall Nondominated Vector Generation and Spacing) of the proposed NSGA-III algorithm is compared with other multi-objective algorithms, namely VEGA, MOGA, SPEA, and NSGA-II. It is observed that the proposed algorithm performs 7% better that the existing algorithm in terms of ONVG and 12% better results in terms of spacing. ANOVA and DMRT statistical tests are used to cross-validate the results.

Keywords Cloud computing, Distributed system, Energy efficient, Virtualization, NSGA-III algorithm

*Correspondence:
Mamoon Rashid
mamoon873@gmail.com
[1] Department of Computer Science and Engineering, Dayananda Sagar University, Bangalore, Karnataka, India
[2] Centre for Smart Grid Technologies, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai Campus, Chennai, Tamilnadu, India
[3] Centre for Automation, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai Campus, Chennai, Tamilnadu, India
[4] Department of Computer Engineering, College of Computer and Information Technology, Taif University, PO Box. 11099, 21994 Taif, Saudi Arabia
[5] Department of Information Technology, College of Computers and Information Technology, Taif University, PO Box 11099, 21944 Taif, Saudi Arabia
[6] Department of Computer Science and Engineering, School of Computing, Kalasalingam Academy of Research and Education, Krishnankoil 626126, India
[7] Department of Computer Engineering, Faculty of Science and Technology, Vishwakarma University, Pune 411048, India

Introduction

Cloud computing is a model for outsourcing an organization's computing power to a rented infrastructure. Cloud computing is possible because of emerging service-oriented architecture, sophisticated servers, and software-defined networking technologies. The physical machine can host multiple operating systems with the help of a hypervisor software module installed in physical devices [1, 2]. Virtualization significantly reduces resource wastage instead of using an entire machine hosted with a single operating system. Resource wastage is the unused CPU and RAM after placing the virtual machine in the respective physical machine (Residual). In simple terms, we can express $R_w = R_a - R_u$, where R_w denotes the resource waste, R_a denotes the available resource in the physical machine, R_u denotes the resource consumed or utilized by the number of virtual machines hosted in the physical machine [3].

The networking infrastructure is isolated using SDN and assigned to individual virtual machines for communication. SOA is used to expose the virtualized data center to the end-users over the internet. Cloud supports elasticity, service on demand, and the pay-as-you-go model. Cloud provides three fundamental service models to the end-user: IaaS, PaaS, and SaaS. Many other prefabricated services like databases and Hadoop are also in existence. For creating a virtual machine, the user needs to specify the operating system, Memory, CPU cores, and Storage [4]. The preconfigured operating system is a machine image stored in the SAN network that can be executed directly on the virtualized hardware without installation. The machine image is an operating system deployment file compatible with the hypervisor software. The CPU and RAM are partitioned from the physical server and assigned to run the virtual machines. Virtual machine favors the data center with consolidation, migration, and load balancing. When two or more physical devices are underutilized, the virtual machine can be migrated to a single physical machine to save resources. The unused servers can be put to hibernate mode, to consume minimal energy.

The challenge for efficiently utilising a data center lies in using the underlying data center resources. As per the Gartner report [5], the physical machine consumes 60% of data center power, and the remaining 40% is consumed by networking, cooling, and storage infrastructure. It is crucial to efficiently utilize the data center resources by hosting an appropriate virtual machine to the server. Reducing resource utilization will significantly reduce the expense of a data center. Another vital aspect is placing a virtual machine in a data center with less latency [6]. The data centers are distributed in various geographical locations. When a VM is placed in an area having more latency, it suffers from a performance bottleneck. Consider a virtual machine configured to host a database server in a location with more significant latency. Even though the workload is hosted in sophisticated servers with more excellent configurations, it will only help retrieve the data. The delivery of the information solely depends on network bandwidth and latency. As the latency increases, the user will experience a delay in content delivery in both get and put requests.

The main objectives of this paper are summarized as follows:

- To optimize the placement of virtual machines (VMs) in a data center to minimize resource wastage, power consumption, and network transmission delay.
- To develop an efficient methodology for cloud providers to achieve economic revenue and contribute to developing green data centers.
- To propose using NSGA-III, a multi-objective evolutionary algorithm, to simultaneously reduce resource wastage, power consumption, and network transmission delay.
- To compare the performance metrics (Overall Non-dominated Vector Generation and Spacing) of the proposed NSGA-III algorithm with other existing multi-objective algorithms, namely VEGA, MOGA, SPEA, and NSGA-II.
- To validate the results of the proposed algorithm using ANOVA and DMRT statistical tests to ensure the reliability and accuracy of the findings.

The motivation behind designing an efficient algorithm to place virtual machines in appropriate servers is to address resource wastage and power consumption in data centers. Currently, data centers consume approximately 2% of the total electricity generated by nations. This significant energy consumption needs substantial efforts to generate electricity, leading to environmental impacts and resource depletion. With the rapid growth of businesses adopting cloud platforms for their operations, data center electricity consumption is projected to increase to 95% in the coming years. This surge in demand makes it primary to find solutions to reduce electricity consumption in data centers, given its crucial role in meeting the escalating digital needs. By developing practical VM placement algorithms, we can optimize resource utilization, distribute workloads efficiently, and minimize energy consumption in data centers. This proactive approach towards energy efficiency aligns with the urgent need to mitigate environmental impact and promote sustainable computing practices. As cloud computing becomes an integral part of modern business operations, the quest to reduce electricity consumption

becomes paramount, and an efficient VM placement algorithm emerges as the need of the hour.

The paper is structured as follows: "Literature survey" section provides a comprehensive literature review of existing algorithms; "Virtual machine placement objective formulation" section presents the formulation of the VMP objective; "Proposed methodology" section introduces the NSGA-III algorithm; "Experimental setup" section describes the experimental setup; "Performance evaluation and discussions" section offers the performance evaluations using ANOVA and DMRT; and finally, the paper concludes with a summary of findings.

Literature survey

Building an energy-efficient data center is a crucial concern for any cloud provider. Server virtualization technologies give the flexibility to host multiple operating systems with a partitioned resource called a virtual machine in the same physical machine [3]. It has improved the utilization of cloud servers to a great extent. The challenges are replaced, and the issues are now related to the placement of virtual machines in the cloud server to increase its utilization even further. Thus, objectives emerged to place VM to PM considering criteria like maximizing resource utilization of servers and networking devices, minimizing power consumption, maximizing economic revenue, etc. Consumption or Power Consumption means the amount of electricity the physical machine consumes [7]. A heuristic algorithm like bin packing [8] and linear programming-based formulation [9] is used to achieve better results in problems on a smaller scale. Many novel stochastic algorithms are proposed to achieve maximum benefits from the large-scale data centre. A bio-inspired and evolutionary algorithm is extensively applied out of many stochastic algorithms, and the literature is presented in this section.

Swarm intelligence

Swarm intelligence (SI) is a technique that mimics the natural behaviour of a species to find a food source or a mate. Many researchers used swarm intelligence algorithms to solve virtual placement problems [10, 11]. Ant exhibits their intelligence in finding the food source, whereas the firefly exhibits intelligence in finding a mate. In swarm intelligence, randomly, each agent works until it finds a solution then the information is communicated with the remaining individuals. The remaining individuals will tune themselves to achieve a better solution. The global solution is the individual that dominates all the remaining individuals. Every swarm intelligence algorithm works based on two factors called exploration and exploitation [12]. Exploration is searching for a solution in the overall solution space, and exploitation is searching within the best-known solution space. The solution space is defined using the objective function. For many of the problems, there might be more than a single objective function that either needs to be minimized or maximized. Minimizing an objective function may have a negative impact on other objective functions. When an algorithm is constructively optimized, two or more objective functions are called a multi-objective optimization algorithm [13].

In [14] proposed a multi-objective ant colony algorithm to minimize power consumption (η_1) and maximize the revenue of communication (η_2). The movement of an ant to a food source is mapped to the VM to be placed in PM. The favorability of placing VM_i to PM_j is based on the pheromone trails $\eta(i, j)$. The multi-objective problem solution is converted to scalar quantity using the weighted sum approach $\eta(i,j) = \eta_1(i,j) + \eta_2(i,j)$. In [15], proposed a modified ACA called Order Exchange and Migration ACS to minimize the number of active servers favours energy-efficient data centres. The proposed algorithm is compared with ACS and shows significant performance improvement with a single objective function. The algorithm also focuses on ordering and migrating overloaded and underloaded server loads. The congested server's VM configurations are sorted, and the VM utilizing higher resources is swapped with an underutilized server called load balancing. A load-balancing operation is a network-intensive task once the virtual machine is placed into a physical machine. In [16] proposed work, ant colony-based power-aware and performance-guaranteed methodology (PPVMP) is used to optimize the data centre power consumption and improve VM performance in a physical machine. In [4] proposed Energy Efficient Knee point driven Evolutionary Algorithm (EEKnEA) uses the evolutionary algorithm framework with a modified selection strategy called KnEA where the highest fit Pareto optimal solutions are considered along with knee points for the next generation. The algorithm uses a single-point crossover technique. The chromosomes are checked for feasibility during each population generation, and infeasible chromosomes are subjected to solution repair. In this work, the author addressed the objectives: the energy consumption of servers, the energy consumption of inter-VM communication, Resource Utilization, and Robustness.

Kuppusamy et al. [17] proposed a reinforced social spider optimization to handle job scheduling in a fog-cloud environment. The author performed extensive experimentation using the FogSim simulator to generate the

dataset. In addition, they achieved the minimized cost function by considering the CPU processing time and allocated memory resources. Huanlai Xing et al. [18] proposed an ACO algorithm to address the virtual machine placement problem by considering energy consumption and network bandwidth. The proposed algorithm enables the information exchange that inherits the indirect information exchange among the ants in ACO.

Genetic algorithm

Genetic algorithm is inspired by the evolutions of living beings based on the concepts of Darwin's theory of evolution [19]. The genetic algorithm works based on three techniques – selection, crossover, and mutation. Selection is the process of finding the best individual from the entire population. The selected individual's chromosomes are exchanged in varying proportions to form offspring. The mutation is used to achieve something newer from the population. Mutation is a process of voluntarily changing chromosomes to generate unique offspring. Then the offspring are subjected to the fitness function or objective function. If the offspring is a valid chromosome, it survives to the next generation of the population; else, they discard the chromosome. If the progeny survives with the most significant fitness value, then the offspring is likely to be selected in the next mating pool. A better solution can be achieved by iterating the process [20].

The author in [21] proposed a novel hybrid genetic and PSO algorithm (HGAPSO) to optimize power consumption, resource wastage, and SLA violation. Genetic algorithm concepts of crossover and mutation are used to find globally optimal solutions, whereas PSO is used to achieve faster convergence. Roulette wheel selection, single-point crossover with shuffling mutation operator is used in the GA phase. This work converts an ordered encoding chromosome into a binary encoding method to apply PSO. In [22], they considered the NSGA-II algorithm to optimize computing resources and network bandwidth. In [23], a modified genetic algorithm with the fuzzy model optimises the computing resource and thermal efficiency.

In [24], the authors propose a secure and self-adaptive resource allocation framework integrated with an enhanced spider monkey optimization algorithm. The proposed framework addresses workload imbalance and performance degradation issues while meeting deadline constraints. Experimental results demonstrate its superiority over state-of-the-art approaches like PSO, GSA, ABC, and IMMLB in terms of time, cost, load balancing, energy consumption, and task rejection ratio. In [25], the author addresses the challenges of cloud-fog computing. IoT systems generate vast amounts of data that need to be processed. Instant response tasks are sent to fog nodes for low delay but high end-user energy consumption, while complex tasks go to cloud data centres for extensive computation. To address these challenges, the author proposes the MGWO algorithm, which reduces fog brokers' QoS delay and energy consumption. The proposed algorithm is verified in simulations against state-of-the-art algorithms. In [26], the authors introduce the ARPS framework for efficient multi-objective scheduling of cloud services to meet end-user's QoS requirements. The framework optimizes execution time and costs simultaneously using the spider monkey optimization algorithm. Extensive simulation analysis with Cloudsim demonstrates its superiority over four existing mechanisms regarding processing time, cost, and energy consumption.

In [27], the authors focus on microservices and the challenges of meeting end-user demands in cloud computing while adhering to SLA constraints. Using the Fine-tuned Sunflower Whale Optimization Algorithm (FSWOA), the proposed QoS-aware resource allocation model optimizes microservice deployment for improved efficiency and resource utilization. Experimental results show that the proposed approach outperforms baseline methods (SFWOA, GA, PSO, and ACO) with reductions in time, memory consumption, CPU consumption, and service cost by up to 4.26%, 11.29%, 17.07%, and 24.22% respectively. In [28], the authors develop a task-processing framework for cloud computing that selects optimal resources at runtime using a modified PSO algorithm. The proposed algorithm addresses conflicting objectives, optimizing multiple parameters simultaneously, such as time, cost, throughput, and task acceptance ratio. Experimental results using Cloudsim demonstrate its significant superiority over baseline heuristic and meta-heuristic methods. In [29], the authors address the resource provisioning and scheduling challenges in cloud computing due to resource heterogeneity and dispersion. To mitigate environmental concerns caused by increased data centres for high computational demand, the authors propose an efficient meta-heuristic technique using a modified transfer function for binary particle swarm optimization (BPSO).

Comparison on state of art multi-objective optimization algorithms

This section aims to compare some of the leading multi-objective optimization algorithms comprehensively. The difference between the algorithm working and its performance in attaining optimal solutions varies between the algorithm and the problem. By analyzing each algorithm's strengths, weaknesses, and performance metrics, we aim

Table 1 Comparison of state of art multi-objective optimization algorithms

Algorithm	Exploration	Exploitation	Convergence Rank	Computational Complexity	Uniqueness	Cons
Genetic Algorithm (GA)	Crossover and mutation	Selection	4	Exponential time complexity	Simple to implement and understand. Can handle complex problems.	Exponential time complexity may not find the optimal solution for all objectives.
Particle Swarm Optimization (PSO)	Velocity update	Inertia weight and cognitive and social factors	5	Quadratic time complexity	Fast and efficient. Can find suitable solutions in a short amount of time.	Slow for small problems, may not find the optimal solution for all objectives.
Ant Colony Optimization (ACO)	Pheromone evaporation and updating	Ant recruitment and ant selection	6	Quadratic time complexity	Robust and scalable. Can handle large problems.	Sensitive to the initial conditions, may not find the optimal solution for all objectives.
Multi-objective Evolutionary Algorithm (MOEA)	Genetic operators	Selection	3	Polynomial time complexity	Can handle multiple objectives simultaneously. It can find the Pareto optimal set.	It can be complex to implement and understand and may not find the optimal solution for all objectives.
Non-dominated Sorting Genetic Algorithm II (NSGA-II)	Crossover and mutation	Selection	2	Polynomial time complexity	Based on the concept of non-dominated sorting. Ensures that the Pareto front is always represented in the population.	It can be slow for minor problems and may not find the optimal solution for all objectives.
Non-dominated Sorting Genetic Algorithm III (NSGA-III)	Crossover and mutation	Selection	1	Polynomial time complexity	It uses a niching mechanism to promote diversity. Ensures that there are enough solutions in each niche of the search space.	Can be slow for small problems, may not find the optimal solution for all objectives.

to identify their suitability for specific problem types and offer insights into their practical applications. The algorithms under review include NSGA-II, MOEA/D, NSGA-III, Genetic Algorithm, Particle Swarm Algorithm and Ant Colony Algorithm. The metrics considered for comparison are explained below, and Table 1 compares the mentioned algorithms.

Exploration

The degree to which the algorithm explores the unvisited search space. The mechanisms implemented in the algorithm, such as mutation, pheromone updates and reference points, favour exploration in the algorithms.

Exploitation

The degree to which an algorithm focuses on improving the existing solution is called exploration. As the iteration increases, the monotonic decay or reduction in crossover probability favours exploitation. Exploitation is searching for better solutions that are closer to the existing solution.

Convergence

Convergence measures how quickly an algorithm finds the optimal or Pareto optimal solution for a given problem, given a fixed number of iterations. To generalize the ranks of the algorithms listed in the Table 1, we used the Rosenbrock function and ranked them according to their convergence speed.

Computation complexity

Computational complexity defines the runtime of an algorithm. Optimization problems are generally NP-hard, meaning they are computationally intractable and require exponential time. Polynomial time complexity is the most desirable for optimization problems, as the algorithm will run in a reasonable amount of time. The higher the complexity, the longer the runtime will be.

Extract from the literature

The above literature shows that many leading researchers are applying bio-inspired swarm optimization or genetic algorithms to improve the various efficiency aspects of cloud resources. In specific, ACO is widely used in bio-inspired algorithms. The cloud servers are both time and space components, allowing the cloud provider to over-commit their cloud resources. Our research considers that the servers are only space-shared components, and over-committing server resources are not considered.

Virtual machine placement objective formulation

Minimize resource wastage

A cloud data centre may have any number of physical machines. A physical machine has resources in terms of CPU and RAM. In the cloud environment, the storage is given to a physical machine in terms of a Storage Area Network and can be dynamically increased to any volume. Our research considers only the CPU and memory in the optimization objective calculation. About the illustration above, assume we have two physical machines, namely $PM = \{PM_1, PM_2\}$ in a

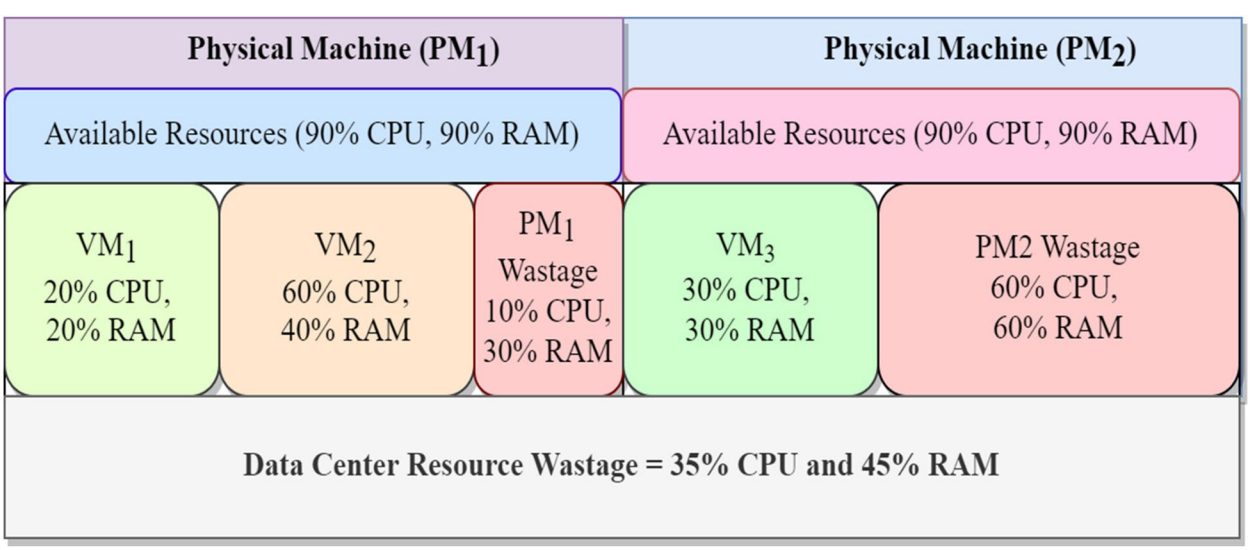

Fig. 1 Illustration on resource wastage objective calculation

data centre with an available resource capacity of 90% each. The remaining 10% of the CPU and Memory is reserved for running the operating system and the hypervisor software. Three VM requests are, namely $VM = \{VM_1, VM_2, VM_3\}$ and each VM need a different resource for execution. VM_1 requires 20% of CPU and 20% of RAM for execution, VM_2 requires 60% CPU and 40% of RAM, VM_3 requires 30% CPU and 30% of RAM.

Figure 1 depicts the possible way to schedule the virtual machine. The PM_1 holds $\{VM_1, VM_2\}$ and VM_3 is placed in PM_2 because the PM_1 don't have enough resources. The wastage is highlighted in red color. The total resource wastage is the sum of wastage in $\{PM_1, PM_2\}$. The above process can be mathematically represented using the below Eq. 1.

$$Min \sum_{i=1}^{M} W_i = \sum_{i=1}^{M} \left[y_i \times \frac{\left| (\theta_{P\,i} - \sum_{j=1}^{N} (x_{i,\,j.}\, R_{p,\,j})) - (\theta_{M\,i} - \sum_{j=1}^{N} (x_{i,\,j.}\, R_{M,\,j})) \right| + \varepsilon}{\sum_{j=1}^{N} (x_{i,\,j.}\, R_{p,\,j})) + \sum_{j=1}^{N} (x_{i,\,j.}\, R_{M,\,j}))} \right] \tag{1}$$

$$x_{i,j} = \begin{cases} 1 \; if \;\; VM_j \; is \; allocated \; to \; PM_i \\ 0 \; otherwise \end{cases} \tag{2}$$

$$y_i = \begin{cases} 1 \; if \;\; PM_i \; is \; used \\ 0 \; otherwise \end{cases} \tag{3}$$

where $R_{M,j}$, $R_{p,j}$ is CPU and memory demand of each Virtual Machine, θ_{Mi}, θ_{Pi} Each PM's upper limit value is usually set to 90%, where the remaining 10% is used to run hypervisors and server monitoring modules.

Minimize power consumption

Considering the $\{VM_1, VM_2\}$ placed in PM_1 and VM_3 is placed in PM_2 the calculation is carried out in this section. In the literature [16], correlation indicates a linear relationship between CPU utilization and power

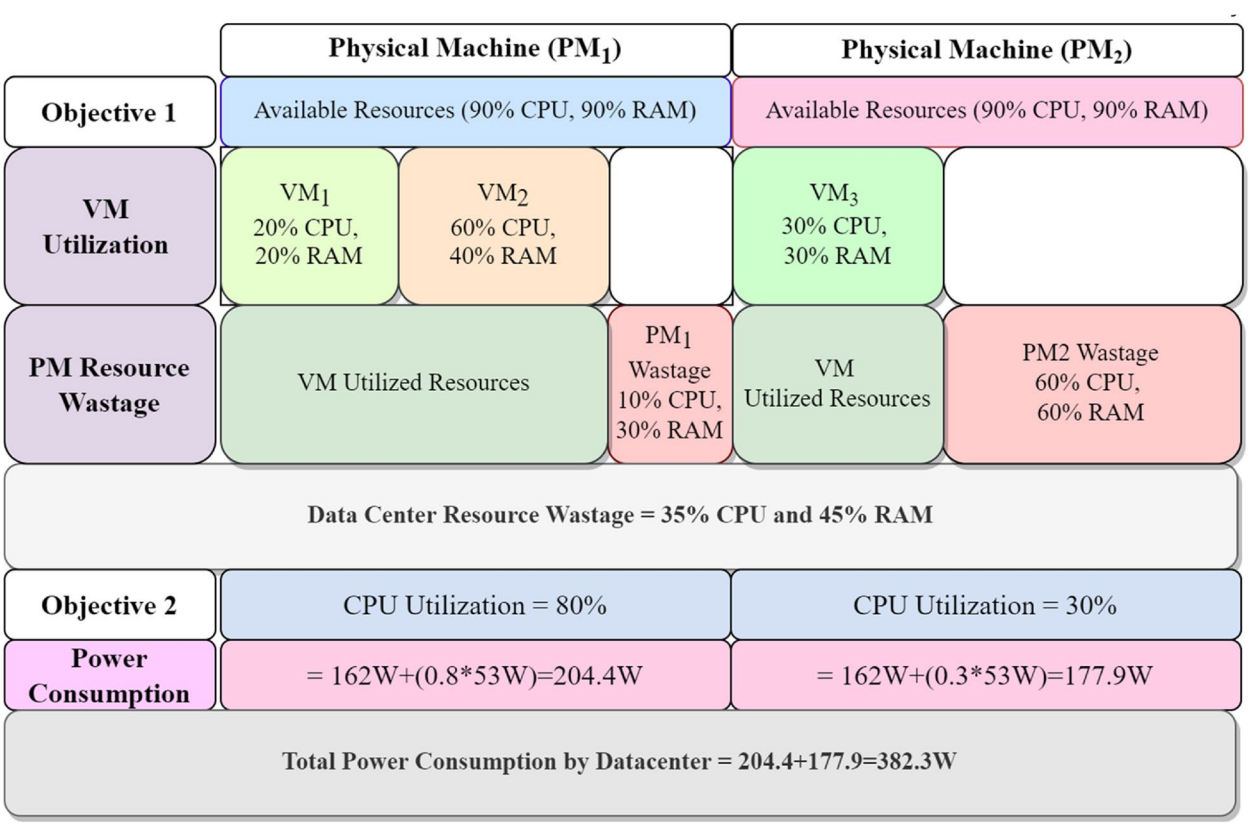

Fig. 2 Illustration on power consumption objective calculation

consumption, where an increase in CPU utilization corresponds to a proportional increase in power consumption. The reference value is also incorporated from the literature [16]. When the CPU is idle, not placed with any VM, the power consumption is observed to be 162W. When the PM is thoroughly utilized, the power consumption is 215W. Hence the power consumption of a physical machine ranges from a lower limit of 162W to a higher limit of 215W. In Fig. 2, PM_1 is hosted with $\{VM_1, VM_2\}$ has a total CPU utilization of 80%. 80% of 53W is consumed in addition to 162W. The power consumption of PM_1 is said to $162W + (80\% * 53W)$ equals 204.4W.

Likewise, the power consumption of all physical machines is calculated to find the total power consumption of the cloud data center. It is mathematically represented as in Eq. 4.

$$Min \sum_{i=1}^{M} P_i = \sum_{i=1}^{M} \left[y_i \times \left(\left(P_i^{active} - P_i^{idle} \right) \times \sum_{j=1}^{N} \left(x_{i,j.} R_{p,j} \right) + P_i^{idle} \right) \right] \tag{4}$$

P_i^{idle} denotes the power consumption of a physical server without any virtual machine hosted in it, P_i^{active} is the power consumption of a physical machine at its maximum hosted load.

Minimize propagation time

$$Min \sum_{n=1}^{N} PDelay_{i,j}^{n} \ \forall i \in M, \forall j \ \varepsilon \ N \tag{5}$$

where *PDelay* denotes the latency of a virtual machine VM_i to be placed in a physical machine PM_j.

Proposed methodology

NSGA-III [30] is much like the working model of the NSGA-II algorithm [22]. The NSGA II algorithm proposed by Deb follows the pattern of the Pareto-based approach extensively [31]. During the initialization phase, a problem-specific initial population is generated. Then the population is evaluated using the objective function. Now, the population will have their fitness values. This existing population is called a parent population. This parent population is chosen in random or probabilistic-based approaches to generate the children's population. These parent population pairs are subjected to crossover and mutation operators to produce several individuals in the children population. Until this step, this algorithm follows the identical framework of NSGA II. The NSGA III steps are majorly divided into two. They are non-dominated sorting and calculation of crowding distance.

The parent and children populations are merged to find the Pareto front in non-dominated sorting. For example, consider there are five parent

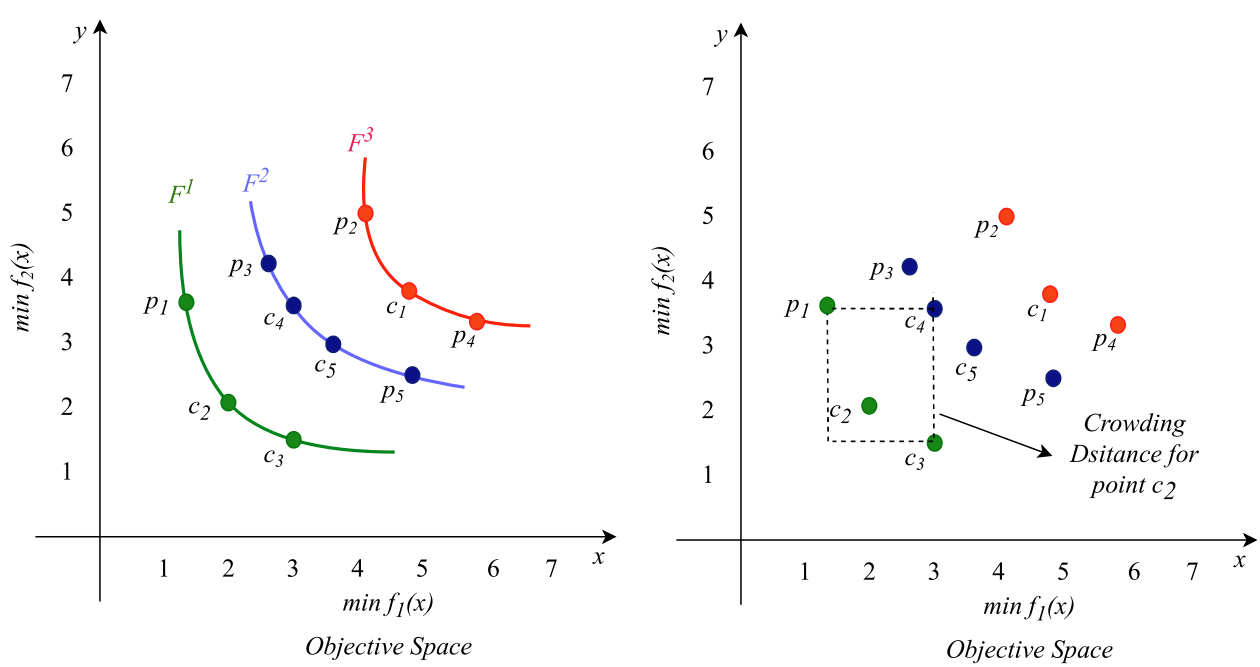

Fig. 3 Crowding distance calculation in comparison with the Pareto front-based solution calculation

individuals $P = \{p_1, p_2, p_3, p_4, p_5\}$ and five children $C = \{c_1, c_2, c_3, c_4, c_5\}$ both are combined to form $P \cup C = \{p_1, p_2, p_3, p_4, p_5, c_1, c_2, c_3, c_4, c_5\}$. Now for the combined population, domination is applied to find the Pareto front. The combined population size is usually $2P$. Consider three solutions, $F^1 = \{p_1, c_2, c_3\}$ are identified as the first non-dominated solution front. To find the next non-dominated solution front, the F^1 solution is removed $(P \cup C) - F^1 = \{p_2, p_3, p_4, p_5, c_1, c_4, c_5\}$ from the combined population. Once again, the non-domination process is repeated until all the individuals are fitted into the front F^m, where m denotes the number of fronts. The non-dominated sorting method is to find the multiple Pareto fronts in the given objective, which is depicted in Fig. 1.

The second part of the algorithm focuses on the density estimation of the solutions called crowding distance. The combined population values are initially sorted according to the individual objective function values to find the distance to the surrounding solutions. The figure depicts the crowding distance calculation for point c_2. To its proximity, three solutions exist $\{c_3, c_4, p_1\}$ and the distance is calculated for the point c_2. For the solutions in the front's extreme boundary, the distance is assigned to infinity. The overall crowding distance is calculated by summing the distance of each solution. The solution with a smaller crowding distance implicitly represents several solutions in its proximity. The solutions for the next iteration are selected based on two conditions; the solutions in the lower front are preferred over those in the higher Pareto front [32]. If both solutions are from the same front, then the minimal crowding distance solution is chosen to generate the next parent population of size P. (Fig. 3)

A selection operator based on the reference points is proposed to maintain the diversity in a population [33]. It is adapted with the help of detailed or well-spread reference points in the solution space [34]. The working model of NSGA-III and its computation of reference points are discussed. A series of distributed reference points of G dimensions are also generated. p is a number that is used for division, and it is generated by the end-user. To deploy the reference points over the normalized hyperplane with the interception of one for each axis is given by Das and Dennis [35]. The total

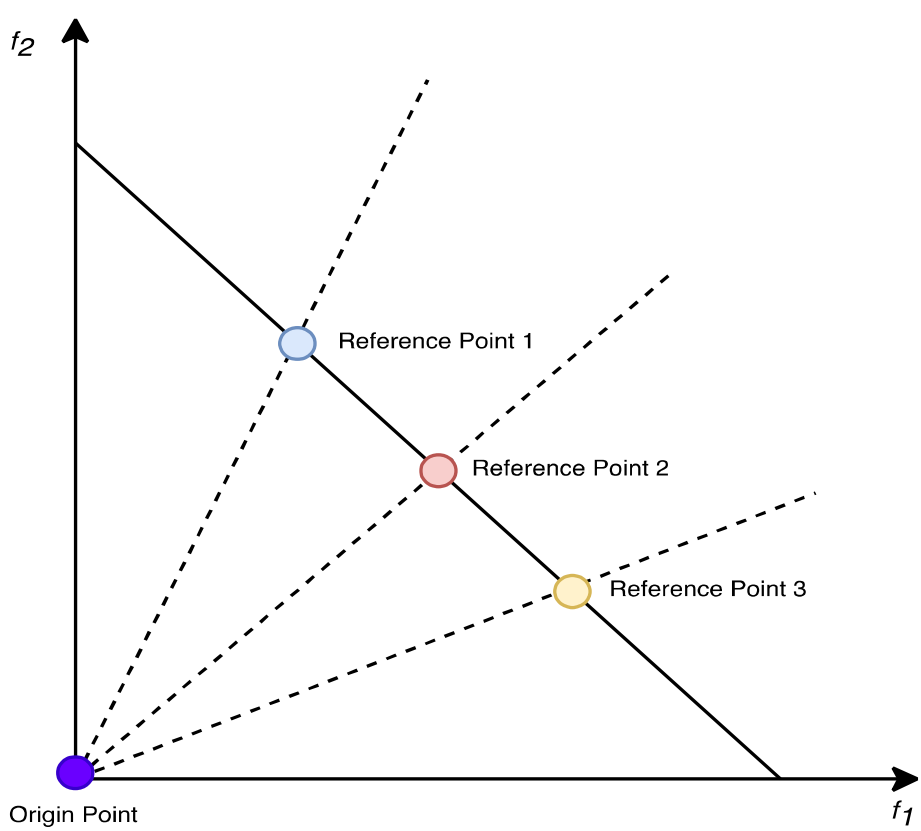

Fig. 4 The rays from origin to reference points in a two-objective solution space

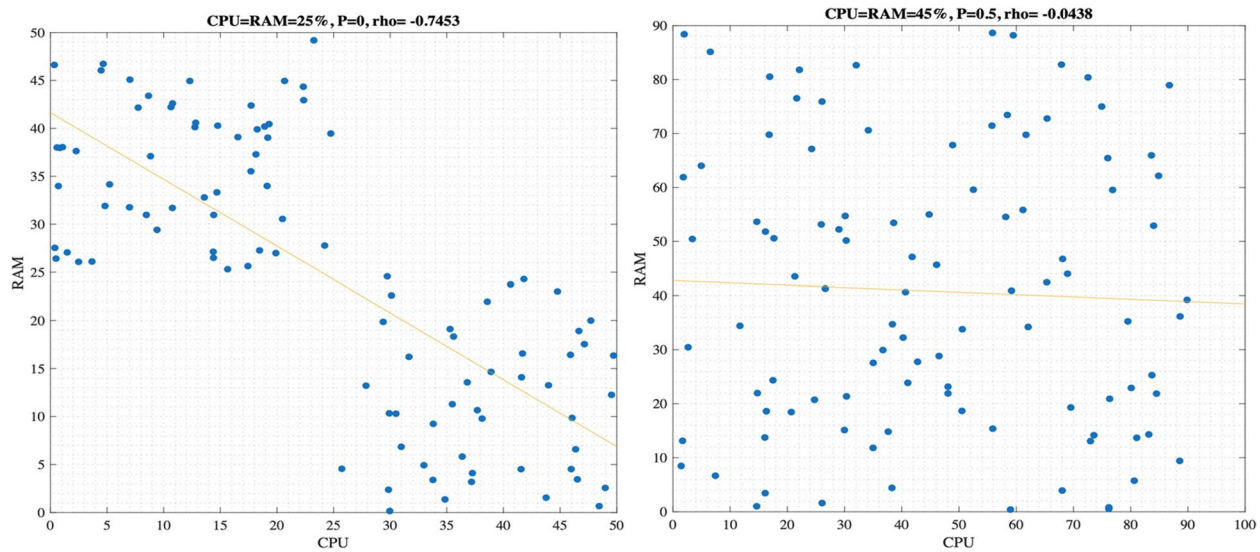

Fig. 5 Distribution of statistically generated datasets with different $\overline{R_{CPU}}$, $\overline{R_{RAM}}$ and P values

number of reference points (H) generation can be done by

$$H = \binom{G + p - 1}{p} \tag{6}$$

These reference points are distributed uniformly to maintain diversity among the population. The direction of the reference points is denoted as a ray that starts from the origin and passes through the reference points, as given in Fig. 4.

NSGA III algorithm balance between exploration and exploitation using the process called Niching. Niching ensures that the algorithm does not converge to the local optimum. Niching works based on the principle of fitness sharing. If a solution I in the Pareto optimal front is close to another solution j (d_ij ≤ R), then the fitness value of all the solutions is shared among them using the equation below. The critical parameter that decides the fitness sharing is the radius R. Sh(d_ij) Eq. 7 represents the sharing value between the solution pair, such that we have n solution. The sharing values are added as shown in Eq. 8. The final fitness values for all solutions closer to each are calculated as given in Eq. 9.

$$Sh(d_{ij}) = \begin{cases} 1 - \frac{d_{ij}}{R} & d_{ij} \le R \\ 0 & otherwise \end{cases} \tag{7}$$

$$nc_i = Sh(d_{i1}) + Sh(d_{i2}) + \cdots + Sh(d_{in}) \tag{8}$$

$$f' = \frac{f}{nc_i} \tag{9}$$

Initialize: Current iteration ($t = 1$), Maximum Iteration ($MaxIt$), Number of Population (N_{pop}), Objective Function $f()$, $Pop*.cost = \infty$, Number of Objectives $nObj = 3$, Algorithmic Parameters: Reference Points R_n, crossover C_{rate} and Mutation Rate M_{rate}, Number of VMs N_{vm}.
Generate the initial population of size (N_{pop}), with each individual of size N_{vm} using $randperm()$ method
for $i = 1$ to N_{pop} do
    Evaluate objective values of individuals using the objective function $f()$
Normalize the objective values in reference to the updated ideal vector $Z*$ eq.8.
Use (ASF) eq.9 to convert the objective to a scalar value.
Apply Non-Dominated Sorting for the population, as explained in Figure 3.
Associate reference pint R_n to each solution in an objective space for diversity
for t to $MaxIt$ **do**
    for 1 to ($C_{rate} * N_{pop}$) **do**
        Randomly choose two individuals, apply crossover
        Evaluate objective values of individuals using the objective function $f()$
    Pop_c contains the individual generate via crossover
    for 1 to ($M_{rate} * N_{pop}$) **do**
        Randomly choose two individuals, apply mutation
        Evaluate objective values of individuals using the objective function $f()$
    Pop_m contains the individual generated via mutation
    Merge the population Pop, Pop_c, Pop_m
    Normalize the objective values in reference to the updated ideal vector $Z*$ eq.7.
    Use (ASF) eq.9 to convert the objective vector to a scalar value.
    Apply Non-Dominated Sorting for the population, as explained in Figure 3.
    Associate reference point Rn to each solution in objective space for diversity.
    Store the solution at the front F^1
Return F^1

Algorithm 1. NSGA III algorithm for virtual machine placement

After the initialization of the population, recombination, mutation, and crossover procedures [34], the size of the merged population depends on mutation and crossover percentage; for elite individual preservation, a non-dominated sorting model is used. Each level of individuals is sorted by crowding distance in NSGA-II. It has been replaced in NSGA-III with reference direction-based niching. Before this operation,

the objectives are normalized with the below formulae. The equation updates the ideal point Z^* during each iteration. The NSGA II-based tchebychef scalarization method is replaced with the achievement scalarization function (ASF), as mentioned in Eq. 8, to convert objective values from vector to scalar values [36].

$$f_g'(x) = f_g(x) - z_g^{min} \tag{10}$$

$$ASF(x, w) = \frac{zmax_{g=1}^{G} f_g'(x)}{w_g} \tag{11}$$

$$f_g^n(x) = \frac{f_g'(x)}{a_g - z_g^{min}} \tag{12}$$

where ASF is defined as the extreme points in each objective axis, z_g^{min} is the ideal value $f_g^n(x)$ is the normalized objective function value. The pseudocode of the proposed model is given in Algorithm 1.

Table 2 Experimental results of $\overline{R_{CPU}} = \overline{R_{RAM}}$ =25% for 100 and 200 VM instances

25%		100 Instances		200 Instances	
Corr	ALG	ONVG	SP	ONVG	SP
-0.753	VEGA	15.26	0.70	14.78	0.67
	MOGA	17.34	0.69	15.37	0.67
	SPEA	16.73	0.65	16.49	0.56
	NSGA-II	20.74	0.55	18.96	0.46
	NSGA-III	28.51	0.28	26.56	0.22
-0.362	VEGA	17.07	0.79	15.49	0.77
	MOGA	16.17	0.66	16.80	0.63
	SPEA	17.84	0.60	17.60	0.52
	NSGA-II	18.99	0.57	19.64	0.46
	NSGA-III	30.38	0.25	30.43	0.24
-0.054	VEGA	17.66	0.70	15.71	0.69
	MOGA	15.94	0.60	16.43	0.56
	SPEA	16.81	0.59	17.37	0.50
	NSGA-II	22.66	0.48	20.78	0.44
	NSGA-III	30.67	0.22	30.95	0.24
0.37	VEGA	14.71	0.68	14.77	0.60
	MOGA	16.45	0.53	16.17	0.49
	SPEA	16.07	0.55	15.90	0.46
	NSGA-II	18.15	0.49	18.63	0.40
	NSGA-III	26.45	0.27	26.46	0.22
0.752	VEGA	19.20	0.62	18.17	0.58
	MOGA	19.86	0.50	18.87	0.41
	SPEA	19.35	0.47	19.05	0.38
	NSGA-II	22.50	0.46	21.19	0.38
	NSGA-III	33.44	0.26	32.53	0.20

Table 3 Experimental results of $\overline{R_{CPU}} = \overline{R_{RAM}}$ =35% for 100 and 200 VM instances

35%		100 Instances		200 Instances	
Corr	ALG	ONVG	SP	ONVG	SP
-0.755	VEGA	19.05	0.78	18.23	0.68
	MOGA	19.01	0.69	18.33	0.62
	SPEA	18.19	0.69	16.90	0.56
	NSGA-II	23.61	0.54	21.73	0.52
	NSGA-III	31.25	0.30	29.40	0.20
-0.372	VEGA	18.07	0.75	18.24	0.69
	MOGA	21.01	0.67	19.68	0.58
	SPEA	20.07	0.59	20.27	0.51
	NSGA-II	20.34	0.48	21.27	0.44
	NSGA-III	30.45	0.28	29.11	0.26
-0.062	VEGA	19.11	0.68	18.83	0.60
	MOGA	18.80	0.60	18.37	0.56
	SPEA	21.05	0.54	19.99	0.49
	NSGA-II	23.88	0.40	22.83	0.35
	NSGA-III	37.12	0.22	35.03	0.20
0.384	VEGA	14.53	0.72	14.91	0.64
	MOGA	18.08	0.50	15.76	0.47
	SPEA	18.85	0.54	18.46	0.47
	NSGA-II	19.75	0.45	19.85	0.43
	NSGA-III	27.95	0.28	28.61	0.19
0.753	VEGA	19.97	0.62	18.79	0.50
	MOGA	22.19	0.47	20.59	0.41
	SPEA	22.09	0.46	19.78	0.36
	NSGA-II	22.52	0.36	21.03	0.31
	NSGA-III	31.67	0.23	32.35	0.18

Experimental setup

Generating virtual machine's CPU and RAM dataset is generated statistically based on the algorithm proposed in [8, 14]. The experiment is carried out with three sets of reference values $\overline{R_{CPU}}$ and $\overline{R_{RAM}}$ (25%, 35% and 45%) respectively. When the reference value is 25% the $\overline{R_{CPU}}$ and $\overline{R_{RAM}}$ generated values are approximately ranging from 0 to 50%. The correlation values are calculated after generating the datasets to show variations in datasets. The negative correlation denotes a memory-intensive virtual machine workload, and the positive correlation denotes CPU-intensive virtual machine. The expected correlation values are achieved by varying the probability values P to (0, 25, 50, 25, 100), giving the correlation values of (-0.7, -0.3, 0.04, 0.2, 0.74). Figure 5 represents the distribution of the resources of the virtual machine for $\overline{R_{CPU}} = 25\%$, $\overline{R_{RAM}} = 25\%$, $P = 0$ (Left) and $\overline{R_{CPU}} = 45\%$, $\overline{R_{RAM}} = 45\%$, $P = 0.5$ (right). Based on the combination of values of P(0, 25, 50, 25, 100), $\overline{R_{CPU}R_{RAM}}$ (25%, 35%,45%) and N_{VM}(100,200) thirty datasets are

generated, and the corresponding correlation values are given in the column corr in Tables 1, 2 and 3.

To calculate the power consumption of the individual physical machine, the processor utilization is calculated based on$\sum_{j=1}^{N}(x_{i,j}.R_{p,j})$. An experiment performed in [16, 37] concluded that the CPU and power utilization form a linear relationship among them. When CPU utilization increases, power consumption also increases. The power consumption of a physical server is calculated based on two parameters. P_{idle} andP_{active}. P_{idle} indicates the amount of power consumed when the server is not hosted with any virtual machine. P_{active} denotes the amount of power consumption for complete physical machine utilization. Based on the experiment in [14] two values are taken where $P_{idle} = 162W$ and$P_{active} = 215W$. If the CPU utilization of the physical server is 50%, then the power utilization of the server is 188.5W.

To validate the superiority of the proposed algorithm, it is essential to utilize a statistically generated dataset that exhibits variations, thereby showcasing its performance under diverse conditions. By altering the $\overline{R_{CPU}}$, $\overline{R_{RAM}}$ and P values, different datasets can be generated. In the above figure, $\overline{R_{CPU}} = 25\%$, $\overline{R_{RAM}} = 25\%$ and $P = 0$ so that the generated request values will fall in the range > 0% and < 50%. The correlation of the dataset is calculated by generating dataset returns -0.7453 (strong negative correlation). The next figure where we use $\overline{R_{CPU}} = 45\%$, $\overline{R_{RAM}} = 45\%$ and $P = 0.5$ the CPU and RAM values ranges from > 0% and < 90%. The values $\overline{R_{CPU}}$, $\overline{R_{RAM}}$ and P are used to generate diversified datasets for experimentation.

Hosting a virtual machine in a remote server across regions will incur network transmission delay. Placing a VM at a nearer data center will have less network latency compared farther data centre. For latency, 18 regions of AWS data centers are considered. The latency is estimated using the TCPPing utility configured in the running instances in 18 regions. An average latency measured every 5 min for 15 days is considered in our research work.

Performance evaluation and discussions

To evaluate the proposed algorithm with experimented vales, two metrics specific to multi-objective algorithms are spacing [38] and Overall Non-Dominated Vector Generation (ONGV). ONGV [39] denotes the average number non dominated solutions stored in the external achieve during each iteration of the algorithm. ONGV indicates the number of better-performing solutions found during each iteration. Spacing represents the coarseness of the resolution towards the minimal objective values. If all the non-dominated solutions are closer, then the spacing value is minimal, indicating that the algorithm controls solution generation. If the spacing value is more, we can conclude that the solution is random, and the algorithm has no control over solution generation.

Table 4 Experimental results of $\overline{R_{CPU}} = \overline{R_{RAM}}$ =45% for 100 and 200 VM instances

45%		100 Instances		200 Instances	
corr	ALG	ONVG	SP	ONVG	SP
-0.756	VEGA	19.93	0.71	18.89	0.66
	MOGA	21.90	0.69	19.69	0.64
	SPEA	20.41	0.61	20.26	0.60
	NSGA-II	24.41	0.53	23.73	0.45
	NSGA-III	31.04	0.30	31.64	0.27
-0.382	VEGA	20.02	0.74	20.05	0.68
	MOGA	20.08	0.69	20.69	0.60
	SPEA	23.15	0.61	22.44	0.52
	NSGA-II	23.46	0.47	22.88	0.43
	NSGA-III	33.71	0.21	31.95	0.16
-0.059	VEGA	20.85	0.68	21.38	0.60
	MOGA	20.82	0.63	19.54	0.58
	SPEA	22.56	0.49	21.17	0.42
	NSGA-II	25.25	0.41	25.01	0.32
	NSGA-III	35.95	0.25	36.53	0.18
0.396	VEGA	17.30	0.70	16.49	0.66
	MOGA	19.14	0.54	17.96	0.47
	SPEA	21.78	0.50	19.97	0.44
	NSGA-II	20.86	0.51	20.82	0.41
	NSGA-III	29.22	0.26	29.35	0.23
0.75	VEGA	20.93	0.67	19.60	0.58
	MOGA	21.51	0.47	21.34	0.36
	SPEA	20.02	0.46	20.38	0.39
	NSGA-II	22.25	0.36	22.38	0.38
	NSGA-III	34.76	0.22	33.70	0.18

$$ONGV = \frac{Total\ number\ of\ non\ dominated\ vector\ generated}{Total\ number\ of\ iterations} \tag{13}$$

$$Sp = \sqrt{\frac{1}{|PF| - 1}\sum_{i=1}^{|PF|}(\overline{d} - d_i)^2} \tag{14}$$

The results in Tables 2, 3 and 4 are interpreted using the DMRT (Duncan Multiple Range Test) and ANOVA (Analysis of Variance), the statistical tools to show the significance between the listed algorithms. The tests are being performed for the performance indicators ONVG and SP for all three instances of RAM and CPU, respectively.

Table 5 Result analysis of 100 VM instances using ANOVA for ONVG ($\overline{R_{CPU}} = \overline{R_{RAM}} = 25\%$)

ANOVA						
Source Factor		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig**
ONVG	Between Groups	614.950	4	153.738	41.655	.000
	Within Groups	73.815	20	3.691		
	Total	688.766	24			

Table 6 Result analysis of 100 VM instances using DMRT for ONVG ($\overline{R_{CPU}} = \overline{R_{RAM}} = 25\%$)

Duncan Multiplier Range Test				
Algorithm	**N**	**Subset for alpha = 0.05**		
		1	**2**	**3**
NSGA-III	5	29.8900		
NSGA II	5		20.6080	
SPEA	5			17.3600
MOGA	5			17.1520
VEGA	5			16.7800
Sig		1.000	1.000	0.657

ANOVA

The dataset is generated statistically, and the algorithm uses the guided random approach. ANOVA and Duncan Multiplier Range test are used as statistical tests to measure the algorithm's significance. ANOVA test cannot isolate the best performing algorithm; instead, ANOVA tests if there exists a significant difference in the algorithm's performance. Using the ANOVA test, we can ensure that there is a substantial difference between the algorithm but cannot isolate better-performing algorithms. We conducted the post hoc analysis using the Duncan range test to separate the best-performing algorithm. For example, in Table 6, the Duncan multiplier range test categorises the algorithm into three homogeneous groups (three columns). The first homogeneous group denotes none of the other algorithms are performed as equivalent to the NSGA III algorithm. In the same table, SPEA, MOGA and VEGA are performing equally.

ANOVA test shows the significance between the groups or algorithms in the given sample values. In this paper, the ANOVA test is used to identify whether the results of the algorithms show significance among them. The ANOVA test was conducted at a significance level of 95%. Suppose the Sig.value is lower than the critical value ($\alpha = 0.05$). In that case, the null hypothesis (H_o) should be rejected, and the alternate hypothesis (H_1) should be accepted, indicating a significant difference among the given group of values. It allows for applying post-hoc tests, with Duncan's Multiple Range Test used in this case. Conversely, if the sig.value exceeds 0.05, the null hypothesis (H_o) should be accepted, and post-hoc tests cannot be conducted.

Duncan's multiple range test

The Duncan Multiple Range Test (DMRT) is a statistical method used to compare multiple sets' mean values. It utilizes studentized range statistics to establish numerical boundaries for classifying significant and non-significant differences between any two or more groups. The DMRT

Table 8 Result analysis of 200 VM instances using DMRT for ONVG ($\overline{R_{CPU}} = \overline{R_{RAM}} = 25\%$)

Duncan Multiplier Range Test				
Algorithm	**N**	**Subset for alpha = 0.05**		
		1	**2**	**3**
NSGA-III	5	29.3860		
NSGA II	5		19.8400	
SPEA	5			17.2820
MOGA	5			16.7280
VEGA	5			15.7840
Sig		1.000	1.000	0.193

Table 7 Result analysis of 200 VM instances using ANOVA for ONVG ($\overline{R_{CPU}} = \overline{R_{RAM}} = 25\%$)

ANOVA						
Source Factor		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig**
ONVG	Between Groups	618.993	4	154.748	55.882	.000
	Within Groups	55.384	20	2.769		
	Total	674.377	24			

Table 9 Result analysis of 100 VM instances using ANOVA for SP ($\overline{R_{CPU}} = \overline{R_{RAM}} = 25\%$)

ANOVA						
Source Factor		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig**
Spacing (SP)	Between Groups	.549	4	.137	38.865	.000
	Within Groups	.071	20	.004		
	Total	.619	24			

Table 10 Result analysis of 100 VM instances using DMRT for SP ($\overline{R_{CPU}} = \overline{R_{RAM}} = 25\%$)

Duncan Multiplier Range Test					
Algorithm	**N**	**Subset for alpha = 0.05**			
		1	**2**	**3**	**4**
NSGA-III	5	.2560			
NSGA II	5		.5100		
SPEA	5		.5720	.5720	
MOGA	5			.5960	
VEGA	5				.6980
Sig		1.000	.115	.530	1.000

ranks the sets in ascending or descending order according to the user's preference.

Tables 5 and 6 shows the ANOVA and DMRT tests of ONVG results tabulated in Table 2 for 100 instances of VM with $\overline{R_{CPU}} = \overline{R_{RAM}} = 25\%$.

Table 5 presents the statistical analysis of ANOVA, while Table 6 showcases the results of DMRT. The tabulated results are interpreted from Table 2 of ONVG of 100 VM instances for $\overline{R_{CPU}} = \overline{R_{RAM}} = 25\%$. The results from Table 5 indicate that the Sig.value is below the critical importance of 0.05, leading to the rejection of the Null Hypothesis (H_0) and acceptance of the Alternate Hypothesis (H_1). On the post-hoc analysis, DMRT NSGA-III ranks top among the existing algorithms. Three homogenous groups are formed among the algorithms, and NSGA-III shows its significance by creating a standalone group among the others.

Tables 7 and 8 shows the ANOVA and DMRT tests of ONVG results tabulated in Table 2 for 200 instances of VM with $\overline{R_{CPU}} = \overline{R_{RAM}} = 25\%$.

Table 7 presents the statistical analysis of ANOVA, while Table 8 showcases the results of DMRT. The tabulated results are interpreted from Table 2 of ONVG of 200 VM instances for $\overline{R_{CPU}} = \overline{R_{RAM}} = 25\%$. The results from Table 7 indicate that the Sig.value is below the critical importance of 0.05, leading to the rejection of the Null Hypothesis (H_0) and acceptance of the Alternate Hypothesis (H_1). On the post-hoc analysis, DMRT NSGA-III ranks top among the existing algorithms. Three homogenous groups are formed among the algorithms, and NSGA-III shows its significance by creating a standalone group among the others.

Tables 9 and 10 shows the ANOVA and DMRT tests of SP results tabulated in Table 2 for 100 instances of VM with $\overline{R_{CPU}} = \overline{R_{RAM}} = 25\%$.

Table 9 presents the statistical analysis of ANOVA, while Table 10 showcases the results of DMRT. The tabulated results are interpreted from Table 2 of SP of 100 VM instances for $\overline{R_{CPU}} = \overline{R_{RAM}} = 25\%$. The results from Table 9 indicate that the Sig.value is below the critical

Table 12 Result analysis of 200 VM instances using DMRT for SP ($\overline{R_{CPU}} = \overline{R_{RAM}} = 25\%$)

Duncan Multiplier Range Test					
Algorithm	**N**	**Subset for alpha = 0.05**			
		1	**2**	**3**	**4**
NSGA-III	5	.2240			
NSGA II	5		.4280		
SPEA	5		.4840	.4840	
MOGA	5			.5520	
VEGA	5				.6620
Sig		1.000	.207	.129	1.000

Table 11 Result analysis of 200 VM instances using ANOVA for SP ($\overline{R_{CPU}} = \overline{R_{RAM}} = 25\%$)

ANOVA						
Source Factor		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig**
Spacing (SP)	Between Groups	.530	4	.133	28.734	.000
	Within Groups	.092	20	.005		
	Total	.623	24			

Table 13 Result analysis of 100 VM instances using ANOVA for ONVG ($\overline{R_{CPU}} = \overline{R_{RAM}} = 35\%$)

ANOVA						
Source		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig**
ONVG	Between Groups	583.408	4	145.852	29.349	.000
	Within Groups	99.390	20	4.970		
	Total	682.799	24			

Table 14 Result analysis of 100 VM instances using DMRT for ONVG ($\overline{R_{CPU}} = \overline{R_{RAM}} = 35\%$)

Duncan Multiplier Range Test				
Algorithm	**N**	**Subset for alpha = 0.05**		
		1	**2**	**3**
NSGA-III	5	31.6880		
NSGA II	5		22.0200	
SPEA	5		20.0500	20.0500
MOGA	5		19.8180	19.8180
VEGA	5			18.1460
Sig		1.000	.154	.216

value of 0.05, leading to the rejection of the Null Hypothesis (H_0) and acceptance of the Alternate Hypothesis (H_1). On the post-hoc analysis, DMRT NSGA-III ranks top among the existing algorithms. Four homogenous groups are formed among the algorithms, and NSGA-III shows its significance by creating a standalone group among the others.

Tables 11 and 12 shows the ANOVA and DMRT tests of SP results tabulated in Table 2 for 200 instances of VM with $\overline{R_{CPU}} = \overline{R_{RAM}} = 25\%$.

Table 11 presents the statistical analysis of ANOVA, while Table 12 showcases the results of DMRT. The tabulated results are interpreted from Table 2 of SP of 100 VM instances for $\overline{R_{CPU}} = \overline{R_{RAM}} = 25\%$. The results from

Table 15 Result analysis of 200 VM instances using ANOVA for ONVG ($\overline{R_{CPU}} = \overline{R_{RAM}} = 35\%$)

ANOVA						
Source		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig**
ONVG	Between Groups	583.408	4	145.852	29.349	.000
	Within Groups	99.390	20	4.970		
	Total	682.799	24			

Table 16 Result analysis of 200 VM instances using DMRT for ONVG ($\overline{R_{CPU}} = \overline{R_{RAM}} = 35\%$)

Duncan Multiplier Range Test				
Algorithm	**N**	**Subset for alpha = 0.05**		
		1	**2**	**3**
NSGA-III	5	31.6880		
NSGA II	5		22.0200	
SPEA	5		20.0500	20.0500
MOGA	5		19.8180	19.8180
VEGA	5			18.1460
Sig		1.000	.154	.216

Table 11 indicate that the Sig.value is below the critical value of 0.05, leading to the rejection of the Null Hypothesis (H_0) and acceptance of the Alternate Hypothesis (H_1). On the post-hoc analysis DMRT in Table 12, NSGA-III ranks top among the existing algorithms, four homogenous groups are formed among the algorithms, and NSGA-III shows its significance by creating a standalone group among the others.

Tables 13 and 14 shows the ANOVA and DMRT tests of ONVG results tabulated in Table 3 for 100 instances of VM with $\overline{R_{CPU}} = \overline{R_{RAM}} = 35\%$.

Table 13 presents the statistical analysis of ANOVA, while Table 14 showcases the results of DMRT. The tabulated results are interpreted from Table 3 of ONVG of 100 VM instances for $\overline{R_{CPU}} = \overline{R_{RAM}} = 35\%$. The results from Table 13 indicate that the Sig.value is below the

Table 17 Result analysis of 100 VM instances using ANOVA for SP ($\overline{R_{CPU}} = \overline{R_{RAM}} = 35\%$)

ANOVA						
Source Factor		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig**
Spacing (SP)	Between Groups	.571	4	.143	26.579	.000
	Within Groups	.107	20	.005		
	Total	.679	24			

Table 18 Result analysis of 100 VM instances using DMRT for SP ($\overline{R_{CPU}} = \overline{R_{RAM}} = 35\%$)

Duncan Multiplier Range Test					
Algorithm	**N**	**Subset for alpha = 0.05**			
		1	**2**	**3**	**4**
NSGA-III	5	.2620			
NSGA II	5		.4460		
SPEA	5			.5640	
MOGA	5			.5860	
VEGA	5				.7100
Sig		1.000	1.000	.640	1.000

NSGA-III ranks top among the existing algorithms, and there are three homogenous groups are formed among the algorithms, and NSGA-III shows its significance by creating a standalone group among the others.

Tables 15 and 16 shows the ANOVA and DMRT tests of ONVG results tabulated in Table 3 for 200 instances of VM with $\overline{R_{CPU}} = \overline{R_{RAM}} = 35\%$.

Table 15 presents the statistical analysis of ANOVA, while Table 16 showcases the results of DMRT. The tabulated results are interpreted from Table 3 of ONVG of 200 VM instances for $\overline{R_{CPU}} = \overline{R_{RAM}} = 35\%$. The results from Table 15 indicate that the Sig.Value is below the critical value of 0.05, leading to the rejection of the Null Hypothesis (H_0) and acceptance of the Alternate Hypothesis (H_1). On the post-hoc analysis, DMRT in Table 16 NSGA-III ranks top among the existing algorithms, three homogenous groups are formed among the algorithms, and NSGA-III shows its significance by creating a standalone group.

Table 19 Result analysis of 200 VM instances using ANOVA for SP ($\overline{R_{CPU}} = \overline{R_{RAM}} = 35\%$)

ANOVA						
Source Factor		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig**
Spacing (SP)	Between Groups	.488	4	.122	23.023	.000
	Within Groups	.106	20	.005		
	Total	.594	24			

Table 20 Result analysis of 200 VM instances using DMRT for SP ($\overline{R_{CPU}} = \overline{R_{RAM}} = 35\%$)

Duncan Multiplier Range Test					
Algorithm	**N**	**Subset for alpha = 0.05**			
		1	**2**	**3**	**4**
NSGA-III	5	.2060			
NSGA II	5		.4100		
SPEA	5		.4780	.4780	
MOGA	5			.5280	.5280
VEGA	5				.6220
Sig		1.000	.155	.290	.055

critical value of 0.05, leading to the rejection of the Null Hypothesis (H_0) and acceptance of the Alternate Hypothesis (H_1). On the post-hoc analysis DMRT in Table 14,

Tables 17 and 18 shows the ANOVA and DMRT tests of SP results tabulated in Table 3 for 100 instances of VM with $\overline{R_{CPU}} = \overline{R_{RAM}} = 35\%$.

Table 17 presents the statistical analysis of ANOVA, while Table 18 showcases the results of DMRT. The tabulated results are interpreted from Table 3 of SP of 100 VM instances for $\overline{R_{CPU}} = \overline{R_{RAM}} = 35\%$. The results from Table 17 indicate that the Sig.Value is below the critical value of 0.05, leading to the rejection of the Null Hypothesis (H_0) and acceptance of the Alternate Hypothesis (H_1). On the post-hoc analysis DMRT in Table 18, NSGA-III ranks top among the existing algorithms. Four homogenous groups are formed among the algorithms, and

Table 21 Result analysis of 100 VM instances using ANOVA for ONVG ($\overline{R_{CPU}} = \overline{R_{RAM}} = 45\%$)

ANOVA						
Source		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig**
ONVG	Between Groups	570.996	4	142.749	45.027	.000
	Within Groups	63.406	20	3.170		
	Total	634.402	24			

Table 22 Result analysis of 100 VM instances using DMRT for ONVG ($\overline{R_{CPU}} = \overline{R_{RAM}} = 45\%$)

Duncan Multiplier Range Test				
Algorithm	**N**	**Subset for alpha = 0.05**		
		1	**2**	**3**
NSGA-III	5	32.9360		
NSGA II	5		23.2460	
SPEA	5		21.5840	21.5840
MOGA	5			20.6900
VEGA	5			19.8060
Sig		1.000	.156	.150

Table 19 presents the statistical analysis of ANOVA, while Table 20 showcases the results of DMRT. The tabulated results are interpreted from Table 3 of SP of 200 VM instances for $\overline{R_{CPU}} = \overline{R_{RAM}} = 35\%$. The results from Table 19 indicate that the Sig.Value is below the critical value of 0.05, leading to the rejection of the Null Hypothesis (H_0) and acceptance of the Alternate Hypothesis (H_1). On the post-hoc analysis DMRT in Table 20, NSGA-III ranks top among the existing algorithms. Four homogenous groups are formed among the algorithms NSGA-III shows its significance by creating a standalone group among the others.

Tables 21 and 22 shows the ANOVA and DMRT tests of ONVG results tabulated in Table 4 for 100 instances of VM with $\overline{R_{CPU}} = \overline{R_{RAM}} = 45\%$.

Table 21 presents the statistical analysis of ANOVA, while Table 22 showcases the results of DMRT. The tabulated results are interpreted from Table 4 of ONVG of 100 VM instances for $\overline{R_{CPU}} = \overline{R_{RAM}} = 45\%$. The results from Table 21 indicate that the Sig. value is below the critical value of 0.05, leading to the rejection of the Null Hypothesis (H_0) and acceptance of the Alternate Hypothesis (H_1). On the post-hoc analysis DMRT in Table 22, NSGA-III ranks top among the existing algorithms, three homogenous groups are formed among the algorithms, and NSGA-III shows its significance by creating a standalone group.

Tables 23 and 24 shows the ANOVA and DMRT tests of ONVG results tabulated in Table 4 for 200 instances of VM with $\overline{R_{CPU}} = \overline{R_{RAM}} = 45\%$.

Table 23 presents the statistical analysis of ANOVA, while Table 24 showcases the results of DMRT. The tabulated results are interpreted from Table 4 of ONVG of 200 VM instances for $\overline{R_{CPU}} = \overline{R_{RAM}} = 45\%$. The results

Table 23 Result analysis of 200 VM instances using ANOVA for ONVG ($\overline{R_{CPU}} = \overline{R_{RAM}} = 45\%$)

ANOVA						
Source		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig**
ONVG	Between Groups	605.915	4	151.479	48.937	.000
	Within Groups	61.908	20	3.095		
	Total	667.823	24			

Table 24 Result analysis of 200 VM instances using DMRT for ONVG ($\overline{R_{CPU}} = \overline{R_{RAM}} = 45\%$)

Duncan Multiplier Range Test				
Algorithm	**N**	**Subset for alpha = 0.05**		
		1	**2**	**3**
NSGA-III	5	32.6340		
NSGA II	5		22.9640	
SPEA	5		20.8440	20.8440
MOGA	5			19.8440
VEGA	5			19.2820
Sig		1.000	.071	.199

NSGA-III shows its significance by creating a standalone group among the others.

Tables 19 and 20 shows the ANOVA and DMRT tests of SP results tabulated in Table 3 for 200 instances of VM with $\overline{R_{CPU}} = \overline{R_{RAM}} = 35\%$.

Table 25 Result analysis of 100 VM instances using ANOVA for SP ($\overline{R_{CPU}} = \overline{R_{RAM}} = 45\%$)

ANOVA						
Source Factor		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig**
Spacing (SP)	Between Groups	.585	4	.146	34.172	.000
	Within Groups	.086	20	.004		
	Total	.671	24			

Table 26 Result analysis of 100 VM instances using DMRT for SP ($\overline{R_{CPU}} = \overline{R_{RAM}} = 45\%$)

Duncan Multiplier Range Test					
Algorithm	**N**	**Subset for alpha = 0.05**			
		1	**2**	**3**	**4**
NSGA-III	5	.2480			
NSGA II	5		.4560		
SPEA	5		.5340	.5340	
MOGA	5			.6040	
VEGA	5				.7000
Sig		1.000	.074	.106	1.000

Three homogenous groups are formed among the algorithms, and NSGA-III shows its significance by creating a standalone group among the others.

Tables 25 and 26 shows the ANOVA and DMRT tests of SP results tabulated in Table 4 for 100 instances of VM with $\overline{R_{CPU}} = \overline{R_{RAM}} = 45\%$.

Table 25 presents the statistical analysis of ANOVA, while Table 26 showcases the results of DMRT. The tabulated results are interpreted from Table 4 of SP of 100 VM instances for $\overline{R_{CPU}} = \overline{R_{RAM}} = 45\%$. The results from Table 25 indicate that the Sig. Value is below the critical value of 0.05, leading to the rejection of the Null Hypothesis (H_0) and acceptance of the Alternate Hypothesis (H_1). On the post-hoc analysis DMRT in Table 26, NSGA-III ranks top among the existing algorithms. Four homogenous groups are formed among the algorithms, and NSGA-III shows its significance by creating a standalone group among the others.

Tables 27 and 28 shows the ANOVA and DMRT tests of SP results tabulated in Table 4 for 200 instances of VM with $\overline{R_{CPU}} = \overline{R_{RAM}} = 45\%$.

Table 27 presents the statistical analysis of ANOVA, while Table 28 showcases the results of DMRT. The tabulated results are interpreted from Table 4 of SP of 200 VM instances for $\overline{R_{CPU}} = \overline{R_{RAM}} = 45\%$. The results from Table 27 indicate that the Sig. Value is below the critical value of 0.05, leading to the rejection of the Null Hypothesis (H_0) and acceptance of the Alternate Hypothesis (H_1). On the post-hoc analysis DMRT on Table 28, NSGA-III ranks top among the existing algorithms. Four homogenous groups are formed among the algorithms NSGA-III shows its significance by creating a standalone group among the others.

Table 27 Result analysis of 200 VM instances using ANOVA for SP ($\overline{R_{CPU}} = \overline{R_{RAM}} = 45\%$)

ANOVA						
Source Factor		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig**
Spacing (SP)	Between Groups	.524	4	.131	24.472	.000
	Within Groups	.107	20	.005		
	Total	.631	24			

Table 28 Result analysis of 200 VM instances using DMRT for SP ($\overline{R_{CPU}} = \overline{R_{RAM}} = 45\%$)

Duncan Multiplier Range Test					
Algorithm	**N**	**Subset for alpha = 0.05**			
		1	**2**	**3**	**4**
NSGA-III	5	.2040			
NSGA II	5		.3980		
SPEA	5		.4740	.4740	
MOGA	5			.5300	
VEGA	5				.6360
Sig		1.000	.116	.240	1.000

from Table 23 indicate that the Sig. Value is below the critical value of 0.05, leading to the rejection of the Null Hypothesis (H_0) and acceptance of the Alternate Hypothesis (H_1). On the post-hoc analysis DMRT in Table 24, NSGA-III ranks top among the existing algorithms.

The proposed algorithm has been tested with different VM instances, from biased towards CPU (-ve correlation coefficient) to biased RAM (+ve Correlation coefficient). The state-of-the-art algorithms are also compared on the same generated instances. For comparing the performances of the proposed system with existing algorithms in identifying VMP solutions with multiple objectives, two performance metrics were considered: Spacing (SP) and Overall Non-Dominated Vector Generation (ONGV). Due to the "tchebychef scalarization method" in NSGA-III, the Pareto optimal front solutions are identified. It is observed that the proposed algorithm performs 7% better that the existing algorithm in terms of ONVG and 12% better results in terms of spacing. ANOVA and DMRT statistical tests are used to cross-validate the results. Thus, the NSGA-III algorithm outperforms all existing algorithms in the SP and ONVG performance indicators.

Conclusion

In this paper, the NSGA-III algorithm is implemented to optimize three contradicting objectives: resource wastage, power consumption and network propagation time. The problem is formulated as a multi-objective optimization problem, and a discretized NSGA-III algorithm is implemented to find the best-performing solution in all three objectives. The results are compared with other multi-objective optimization algorithms, namely VEGA, MOGA, SPEA, and NSGA-II, regarding ONVG and Spacing performance metrics. Since the algorithm is guided randomly, we executed 30 independent runs, and the resultant values were statistically tested using ANOVA and DMRT. The statistical test shows that the significance lies among the MOEA's on VM placement, stating that the NSGA-III outperforms all the existing algorithms in all aspects. Comparing NSGA-II and SPEA, both algorithms are in the same homogenous group in many DMRT tests. Hence these two algorithms perform equally to each other. Comparing SPEA to MOGA, like NSGA-II, MOGA shares a similarly homogenous group. VEGA in-performs in all the aspects of the VM placement problem. Considering the average values of 30 independent runs, NSGA-III achieved 7% better than the existing algorithm regarding ONVG and 12% better results in terms of spacing.

Acknowledgements
The researchers would like to acknowledge Deanship of Scientific Research, Taif University for funding this work.

Authors' contributions
Conceptualization—Arunkumar Gopu; methodology- Arunkumar Gopu, Kalaipriyan Thirugnanasambandam; validation- Rajakumar R, Mamoon Rashid, Kalaipriyan Thirugnanasambandam; formal analysis- Ahmed Saeed AlGhamdi, Sultan S. Alshamrani; writing—original draft preparation- Arunkumar Gopu; writing—review and editing- K Maharajan, Mamoon Rashid; supervision- Rajakumar R; funding acquisition- Ahmed Saeed AlGhamdi. All authors have read and agreed to the submitted version of the manuscript.

Funding
The researchers would like to acknowledge the Deanship of Scientific Research, Taif University for funding this work.

Availability of data and materials
The data that support the findings of this study are available from the first author upon reasonable request.

Declarations

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 20 June 2023 Accepted: 11 August 2023
Published online: 26 August 2023

References

1. Masdari M, Zangakani M (2020) Green cloud computing using proactive virtual machine placement: challenges and issues. J Grid Comput 18(4):727–759
2. Masdari M, Gharehpasha S, Ghobaei-Arani M, Ghasemi V (2020) Bio-inspired virtual machine placement schemes in cloud computing environment: taxonomy, review, and future research directions. Clust Comput 23(4):2533–2563
3. Wei W, Wang K, Wang K, Huaxi Gu, Shen H (2020) Multi-resource balance optimization for virtual machine placement in cloud data centers. Comput Electr Eng 88:106866
4. Basu S, Kannayaram G, Ramasubbareddy S, Venkatasubbaiah C (2019) Improved genetic algorithm for monitoring of virtual machines in cloud environment. In Smart Intelligent Computing and Applications. Springer, Singapore, pp 319–326
5. Masanet E, Shehabi A, Lei N, Smith S, Koomey J (2020) Recalibrating global data center energy-use estimates. Science 367(6481):984–986
6. Gopu A, Venkataraman N (2019) Optimal VM placement in distributed cloud environment using MOEA/D. Soft Comput 23(21):11277–11296
7. Azizi S, Zandsalimi MH, Li D (2020) An energy-efficient algorithm for virtual machine placement optimization in cloud data centers. Clust Comput 23:3421–3434
8. Aydın N, Muter İ, Birbil Şİ (2020) Multi-objective temporal bin packing problem: an application in cloud computing. Comput Oper Res 121:104959
9. Karmakar K, Banerjee S, Das RK, Khatua S (2022) Utilization aware and network I/O intensive virtual machine placement policies for cloud data center. J Netw Comput Appl 205:103442
10. Tripathi A, Pathak I, Vidyarthi DP (2020) Modified dragonfly algorithm for optimal virtual machine placement in cloud computing. J Netw Syst Manag 28:1316–1342
11. Balaji K, Sai Kiran P, Sunil Kumar M (2023) Power aware virtual machine placement in IaaS cloud using discrete firefly algorithm. Appl Nanosci 13(3):2003–2011
12. Alresheedi SS, Lu S, AbdElaziz M, Ewees AA (2019) Improved multi-objective salp swarm optimization for virtual machine placement in cloud computing. Hum-centric Comput Inf Sci 9(1):1–24
13. Nabavi SS, Gill SS, Xu M, Masdari M, Garraghan P (2022) TRACTOR: Traffic-aware and power-efficient virtual machine placement in edge-cloud data centers using artificial bee colony optimization. Int J Commun Syst 35(1):e4747
14. Gao Y, Guan H, Qi Z, Hou Y, Liu L (2013) A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J Comput Syst Sci 79(8):1230–1242
15. Liu XF, Zhan ZH, Deng JD, Li Y, Gu T, Zhang J (2016) An energy efficient ant colony system for virtual machine placement in cloud computing. IEEE Trans Evol Comput 22(1):113–128

16. Zhao H, Wang J, Liu F, Wang Q, Zhang W, Zheng Q (2018) Power-aware and performance-guaranteed virtual machine placement in the cloud. IEEE Trans Parallel Distrib Syst 29(6):1385–1400
17. Kuppusamy P, Kumari NMJ, Alghamdi WY, Alyami H, Ramalingam R, Javed AR, Rashid M (2022) Job scheduling problem in fog-cloud-based environment using reinforced social spider optimization. J Cloud Comput 11(1):99
18. Xing H, Zhu J, Qu R, Dai P, Luo S, Iqbal MA (2022) An ACO for energy-efficient and traffic-aware virtual machine placement in cloud computing. Swarm Evol Comput 68:101012
19. Mirjalili S, Mirjalili S (2019) Genetic algorithm. Evolutionary algorithms and neural networks: theory and applications. pp 43–55
20. Katoch S, Chauhan SS, Kumar V (2021) A review on genetic algorithm: past, present, and future. Multimed Tools Appl 80:8091–8126
21. Sharma NK, Reddy GRM (2016) Multi-objective energy efficient virtual machines allocation at the cloud data center. IEEE Trans Serv Comput 12(1):158–171
22. Liu C, Shen C, Li S, Wang S (2014) A new evolutionary multi-objective algorithm to virtual machine placement in virtualized data center. In 2014 IEEE 5th International Conference on Software Engineering and Service Science. IEEE, Beijing, pp 272–275
23. Wang X, Xing H, Yang H (2019) On multicast-oriented virtual network function placement: a modified genetic algorithm. In Signal and Information Processing, Networking and Computers: Proceedings of the 5th International Conference on Signal and Information Processing, Networking and Computers (ICSINC). Springer, Singapore, pp 420–428
24. Kumar M, Dubey K, Singh S, Kumar Samriya J, Gill SS (2023) Experimental performance analysis of cloud resource allocation framework using spider monkey optimization algorithm. Concurr Comput 35(2):e7469
25. Saif FA, Latip R, Hanapi ZM, Shafinah K (2023) Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing. IEEE Access 11:20635–20646. https://doi.org/10.1109/ACCESS.2023.3241240
26. Kumar M, Kishor A, Abawajy J, Agarwal P, Singh A, Zomaya AY (2021) ARPS: An autonomic resource provisioning and scheduling framework for cloud platforms. IEEE Trans Sustain Comput 7(2):386–399
27. Kumar M, Samriya JK, Dubey K, Gill SS (2023) QoS-aware resource scheduling using whale optimization algorithm for microservice applications. Software: Practice and Experience
28. Kumar M, Sharma SC (2020) PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing. Neural Comput Appl 32:12103–12126
29. Kumar M, Sharma SC, Goel S, Mishra SK, Husain A (2020) Autonomic cloud resource provisioning and scheduling using meta-heuristic algorithm. Neural Comput Appl 32:18285–18303
30. Blank J, Deb K, Roy PC (2019) Investigating the normalization procedure of NSGA-III. In International Conference on Evolutionary Multi-Criterion Optimization. Springer, Cham, pp 229–240
31. Deb K, Jain H (2013) An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. IEEE Trans Evol Comput 18(4):577–601
32. Ishibuchi H, Imada R, Setoguchi Y, Nojima Y (2018) Reference point specification in inverted generational distance for triangular linear Pareto front. IEEE Trans Evol Comput 22(6):961–975
33. Luo W, Qiao Y, Lin X, Xu P, Preuss M (2020) Hybridizing niching, particle swarm optimization, and evolution strategy for multimodal optimization. IEEE Transactions on Cybernetics
34. Koohestani B (2020) A crossover operator for improving the efficiency of permutation-based genetic algorithms. Expert Syst Appl 151:113381
35. Das, Dennis JE (1998) Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM J Optim 8(3):631–657
36. Bekhit M, Fathalla A, Eldesouky E, Salah A (2023) Multi-objective VNF Placement Optimization with NSGA-III. In Proceedings of the 2023 International Conference on Advances in Computing Research (ACR'23). Springer Nature Switzerland, Cham, pp 481–493
37. Pang P, Chen Q, Zeng D, Li C, Leng J, Zheng W, Guo M (2020) Sturgeon: Preference-aware co-location for improving utilization of power constrained computers. In 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE, New Orleans, pp 718–727
38. Laszczyk M, Myszkowski PB (2019) Survey of quality measures for multi-objective optimization: Construction of complementary set of multi-objective quality measures. Swarm Evol Comput 48:109–133
39. Knowles J, Corne D (2002) On metrics for comparing nondominated sets. In Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), IEEE, Honolulu, Vol. 1, pp 711–716

Publisher's Note