Enhanced Virtual Machine Placement in Cloud Data Centers: Combinations of Fuzzy Logic with Reinforcement Learning and Biogeography-Based Optimization (BBO) Algorithms

Arezoo Ghasemi

Qazvin Islamic Azad University

Abolfazl Toroghi Haghighat (✉ haghighat@qiau.ac.ir)

Qazvin Islamic Azad University <https://orcid.org/0000-0003-4500-274X>

Amin Keshavarzi

: Islamic Azad University Marvdasht

Research Article

Keywords: Cloud computing, Fuzzy logic, Machine learning, Load balancing, Multi objectives, Virtual machine placement

Posted Date: January 3rd, 2022

DOI: <https://doi.org/10.21203/rs.3.rs-1034028/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.
[Read Full License](#)

Enhanced Virtual Machine Placement in Cloud Data Centers: Combinations of Fuzzy Logic with Reinforcement Learning and Biogeography-based Optimization (BBO) Algorithms

Arezoo Ghasemi ·
Abolfazl Toroghi Haghighat ·
Amin Keshavarzi

Abstract The process of mapping Virtual Machines (VMs) to Physical Machines (PMs), which is defined as VM placement, affects Cloud Data Centers (DCs) performance. To enhance the performance, optimal placement of VMs regarding conflicting objectives has been proposed in some research, such as Multi-Objective VM reBalance (MOVMrB) and Reinforcement Learning VM reBalance (RLVMrB) in recent years. The MOVMrB algorithm is based on the BBO meta-heuristic algorithm and the RLVMrB algorithm inspired by reinforcement learning, which in both of them the non-dominance method is used to evaluate generated solutions. Although this approach reaches acceptable results, it fails to consider other solutions which are optimal regarding all objectives, when it meets the best solution based on one of these objectives. In this paper, we propose two enhanced multi-objective algorithms, Fuzzy-RLVMrB and Fuzzy-MOVMrB, that are able to consider all objectives when evaluating candidate solutions in solution space. All four algorithms aim to balance the load between VMs in terms of processor, bandwidth, and memory as well as horizontal and vertical load balance. We simulated all algorithms using the CloudSim simulator and compared them in terms of horizontal and vertical load balance and execution time. The simulation results show that Fuzzy-RLVMrB and Fuzzy-MOVMrB algorithms outperform RLVMrB and MOVMrB algorithms in terms of vertical load balancing and horizontal load

Arezoo Ghasemi
Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran. E-mail: arezoo.qasemi214@gmail.com

Abolfazl Toroghi Haghighat (*Corresponding author)
Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran. E-mail: haghighat@qiau.ac.ir

Amin Keshavarzi
Department of Computer Engineering, Marvdasht Branch, Islamic Azad University, Marvdasht, Iran. E-mail: keshavarzi@miau.ac.ir

balancing. Also, the RLVMrB and Fuzzy-RLVMrB algorithms are better in execution time than the MOVMrB and Fuzzy-MOVMrB algorithms.

Keywords Cloud computing · Fuzzy logic · Machine learning · Load balancing · Multi objectives · Virtual machine placement

1 Introduction

Cloud computing, as a distributed computing model, provides everything as a service and aims to scale down the cost for both customers and providers [1]. In this regard, virtual machine (VM) placement, which is the mapping of VMs to Host Machine (HM), is vitally important. The optimal placement of VM makes data centers (DCs) more efficient, and cuts providers' expenditure. Even though lots of research has been conducted on optimal VM placement, most of them have considered only one goal, such as reducing energy consumption, load balancing, increasing resource efficiency, preventing Service Level Agreement (SLA) violations, and increasing Quality of Service (QoS) [2]. However, in VM placement, multiple of the mentioned goals should be considered simultaneously, which means we encounter a multi-objective problem.

In each multi-objective problem, first the candidate solutions are generated using different algorithms, and then they are compared to select the optimum solution. In the case of VM placement, MOVMrB [3] and RLVMrB (our previous work) [4] are two multi-objective methods, which demonstrated exemplary performance. In MOVMrB, generating candidate solutions was conducted using the BBO algorithm, and in RLVMrB, reinforcement learning algorithm was applied. However, in a multi-objective problem, the evaluation method, which is used for comparing different candidate solutions based on various parameters affects final results [4][5][6]. Therefore, in this paper, we propose two new algorithms, which are called Fuzzy-MOVMrB and Fuzzy-RLVMrB. These algorithms are the fuzzy versions of MOVMrB and RLVMrB algorithms, which in them we apply a customized fuzzy logic method to evaluate the generated solutions in VM placement problem. The proposed methods improve load balancing in terms of vertical load balancing and horizontal load balancing, which in the first, load imbalance between HMs is considered, and in the second, load imbalance between each HM.

In contrast to Boolean logic, which in it, variables have either a value of one (true) or zero (false), Fuzzy logic is a multi-value logic in which the value of variables (degree of correctness of variables) can take any number in the range of $[0, 1]$. A Fuzzy system consists of four main components: fuzzifier, knowledge base, inference engine, and de-fuzzifier [7]. In the fuzzifier part, the inputs are converted to fuzzy sets. Then, in the inference engine part, the degree of compliance of these fuzzy sets with the rules predefined for the system is measured. Finally, in the de-fuzzifier part, the results of the inference engine are converted to a numerical value [7].

The remaining parts of this paper proceed as follows: In section two, the literature review is investigated. The third section is concerned with the proposed

methods, Fuzzy-RLVMB and Fuzzy-MOVMB. In the fourth part, test settings and their results are analyzing. Finally, the conclusion gives a summary of the findings and future work.

2 Related work

In this section, the literature on VM placement is reviewed. First, the related works are investigated, and then the proposed algorithms are compared with all related works.

Li et al. [3] proposed a multi-objective algorithm called MOVMB for VM placement, which aimed to balance the load between HMs (horizontal load balance) and the load in each HM (vertical load balance). In both cases, load balancing has been conducted regarding the CPU, bandwidth, and memory. In this work, the BBO algorithm was used to find the best solution in the solution space. To this, in the first, two separate population sets were generated, a population with the aim of balancing the horizontal load and a population with the aim of balancing the vertical load. In each population, there existed three solutions, and information was exchanged between the two population sets. They used the non-dominance algorithm to evaluate the generated solutions.

RLVMB [4] was another work that aimed to balance horizontal and vertical loads. In this work, a multi-objective version of the reinforcement learning algorithm was proposed for VM placement. In the proposed method, the learner agent generated a new solution in each learning episode using three actions, including random selection, mutation, and crossover. The reward, which was received for each performed action, was calculated as a weighted sum of four parameters of horizontal and vertical load imbalance in terms of processor, bandwidth, and memory. For evaluating the generated solutions, the non-dominance algorithms was applied in this work too.

Azizi et al. [8] proposed a VM placement algorithm called MinPR, which aimed to reduce resource wastage and energy consumption. They applied a method called the Resource Usage Factor to minimize power consumption by turning off the HMs. They ranked HMs based on performance so that higher-performance HMs are a priority. In [7], a VM placement algorithm was proposed, which aimed to reduce energy consumption and increase resource utilization. To this, the best-fit-decreasing algorithm based on fuzzy logic was used, which its inputs were energy consumption and resource utilization. They simulated their proposed method using the CloudSime simulator. The evaluation results showed that they were able to improve resource wastage and energy consumption by 30% compared to heuristic algorithms.

Gharehpasha et al. [9] proposed a multi-objective VM placement algorithm with the aim of reducing energy consumption and resource wastage. To achieve these goals, they used a combination of a multi-objective version of the whale optimization algorithm and a chaotic function-based multi-verse optimizer. The proposed method was able to prevent the increase in VM migration too.

In [10] the authors suggested a VM placement algorithm with to reduce energy consumption and waste of resources using the Particle Swarm Optimization algorithm, which was called multi-PSO. They formulated the VM placement problem as a multi-objective bin packing problem. TPVMP was an energy and traffic-aware VM placement algorithm, which was proposed in [11]. The purpose of this algorithm was to reduce energy consumption and improve communication performance by reducing the cost of VM traffic. In addition, this algorithm minimize network congestion in heterogeneous DCs, while power consumption was unchanged. HMs are also identified as overloaded, and VMs are selected for migration with different immigration policies.

In 2020, Farzaei et al. [12] introduced a multi-objective genetic algorithm-based VM placement method to reduce energy consumption and resource waste, which consider the consumed traffic to raise the efficiency and quality of service (QoS). Their proposed algorithm outperformed FFD and ACO algorithms. The experiment results also showed that this algorithm achieved better performance in large search spaces . In 2020, Parvizi et al. [13] demonstrated that a genetic-based VM placement algorithm (NSGA-III) could reduce energy consumption and the number of active HMs. They compared their proposed method with algorithms such as FFD and the exact mathematical approach in terms of critical criteria such as algorithm execution time, resource wastage, and energy consumption. The results depicted the superiority of the proposed algorithm .

Liu et al. [14] introduced a multi-objective algorithm called NSGGA for VM placement. This algorithm tried to reduce the number of active HMs, and also minimize communication traffic. Another goal of this method was to balance multidimensional resources in DCs. VMPMBBO was a multi-objective VM placement algorithm introduced by Zhang et al. in 2014. VMPMBBO considered the placement problem as a complex system, and tried to minimize both waste of resources and energy consumption. Their experiment showed better and more efficient convergence. In addition, they used a resource wastage model that considered bandwidth, memory, and CPU usage. Their power consumption model also considered the processor efficiency of the servers, and idle servers were shut down to prevent further power consumption.

Gao et al.[15] proposed a multi-objective VM placement algorithm to reduce energy consumption and resource wastage using an ant colony algorithm called VMPACS. They formulated the VM placement problem as a bin packing problem and considered CPU and memory resources in their work. In [16], the authors proposed a multi-objective VM placement algorithm called Island NSGA II. Their goal was trading off energy consumption, resource wastage, and quality of service. Regarding these objectives, they formulated the VM placement problem as a bin packing problem and used the NSGA II algorithm. Wang et al. [17] proposed a VM placement algorithm called IGA with to maximize resource utilization, reducing traffic load, and multidimensional balancing. For this purpose, they used an enhanced version of the genetic algorithm and an elitist strategy. They combined the values of all three objectives and performed the evaluation.

In [18], a multi-objective VM placement algorithm for homogeneous and heterogeneous data centers was proposed. They considered the problem of VM placement as an integer linear programming model with the goals of increasing the number of hosted VMs, and reducing resource waste and the number of active HMs. They tried to increase customer satisfaction by lowering DC costs. To this, they proposed two methods, which in the first, the VM placement problem was calculated for each goal in order of priority, while in the second, the goals were considered a weighted sum. The simulation results showed that the second method reduced the number of active HMs by up to 30%.

Table 1 depicts an analytical comparison of related work and our proposed methods. As shown in this table, we compare all methods in terms of the applied approach, objectives, simulation tools, and the year of the related publication.

3 The proposed methodology

While in single-objective problems, solutions are evaluated based on only one criterion, and finding the best solution is straightforward, the evaluation of generated solutions is widely considered to be the most critical challenge of multi-objective problems [3]. To address this challenge, in this paper, we apply the fuzzy logic to evaluate the generated solutions in the VM replacement problem. In the rest of this section, we first formulate the VM replacement problem, then explain the proposed fuzzy system. After that, we propose two new algorithms, which are called Fuzzy-MOVMrB and Fuzzy-RLVMrB. These algorithms are the fuzzy versions of MOVMrB and RLVMrB, respectively.

3.1 Problem formulation

In RLVMrB and MOVMrB algorithms, all VMs in the DC are mapped to HMs. Therefore, the output of these algorithms is a matrix with the size of $m*n$, in which m and n represent the number of VMs, and the number of HMs in the DC, respectively. Both RLVMrB and MOVMrB are multi-objective and objectives in both of them are the same. To find the optimum placement, the output matrix in these algorithms is evaluated based on the following criteria:

1. CPU load imbalance between HMs (horizontal load balance)
2. Memory load imbalance between HMs (horizontal load balance)
3. Bandwidth load imbalance between HMs (horizontal load balance)
4. Internal load imbalance in each single HM (vertical load balance)

For each generated mapping matrix in RLVMrB and MOVMrB algorithms, these four parameters are calculated, and the mapping matrices are evaluated using the non-dominance algorithm. To assess the generated solutions using the non-dominance algorithm, one solution is considered better than the other, when all the goals in that solution are better than the other (one solution overcomes the other). Therefore, this algorithm fails to consider a trade-off

Table 1 Classification of introduced works

Reference	Name	Method	Objectives	Simulation	Year
[3]	MOVMrB	BBO	Load balancing	Cloudsim	2017
[4]	RLVMrB	Reinforcement Learning	Load balancing	Cloudsim	2020
[8]	MinPR	Resource Usage Factor	minimize resource wastage and energy consumption	C++	2020
[7]	fuzzyBFD	best-fit-decreasing based on fuzzy logic whale optimization and a multi-verse optimizer	minimize resource wastage and energy consumption	Cloudsim	2019
[9]			minimize resource wastage and energy consumption	Matlab	2020
[10]		PSO	minimize resource wastage and energy consumption	Java	2018
[11]	TPVMP		minimize energy consumption and traffic cost	Cloudsim	2014
[12]		Genetic algorithm	minimize resource wastage, energy consumption, traffic and QoS	Cloudsim	2020
[13]	MO-VMP	Genetic algorithm	minimize energy consumption	Cloudsim	2020
[14]	NSGGA	Genetic algorithm	minimize energy consumption and traffic cost		2014
[19]	VMPMBBO	BBO	minimize resource wastage and energy consumption	Matlab	2016
[15]	VMPACS	ACO	minimize resource wastage, energy consumption and QoS	Java	2013
[16]	Island NSGA II	Genetic algorithm	minimize resource wastage, energy consumption and QoS	Cloudsim	2020
[17]	IGA	Genetic algorithm	minimize resource wastage and balancing		2013
[18]			minimize resource wastage and number of active HMs and increasing the number of hosted VMs	Julia Language	2021
-	Fuzzy-RLVMrB	Reinforcement Learning and Fuzzy system	Load balancing	Cloudsim	-
-	Fuzzy-MOVMrB	BBo and Fuzzy system	Load balancing	Cloudsim	-

between objectives, when evaluates candidate solutions. In this paper, we feed the above-mentioned parameters as the inputs to the fuzzy system. The output of the fuzzy system determines the desirability of generated mapping matrix. With regards to a multi-objective minimization problem, we shall compare

two solutions, A and B, through the different perspectives of x, y, and z, using the non-dominance algorithm, each bearing equal importance to us. Based on Table /ref2, Solution B is superior to A in Objectives X and Y, and Solution A is superior to B in Objective Z. By using the non-dominance algorithm to evaluate two solutions A and B, this algorithm cannot decide between solution A and B, since solution B cannot overcome A (all three goals are not overcome by solution A), whereas solution B can be considered superior.

The four parameters, which are used for evaluating the output matrix can be divided into two general categories: load imbalance among HMs (in terms of processor, bandwidth, and memory) and load imbalance within each HM. In the following, we will explain how to calculate these parameters.

Table 2 Example

Objectives	A solution	B solution
x	0.71	0.62
y	0.32	0.21
z	0.37	0.38

3.2 Objectives calculation

Equations 1 and 2 are used in the RLVMrB, MOVMrB, Fuzzy-RLVMrB, and Fuzzy-MOVMrB algorithms to calculate the horizontal and vertical load imbalances. Equation 1 is used to calculate the load imbalance between HMs (horizontal load imbalance) in terms of processor, memory, and bandwidth. Equation 2 is used to calculate the load imbalance within each HM (vertical load imbalance). Equations 1 and 2 are based on the theory of coefficient of variance (CV). CV theory is defined as the ratio of the standard deviation to the mean, which indicates the degree of dispersion of resource consumption relative to the mean. CV is one of the normalization methods, so the result is a dimensionless number. Table 3 depicts all symbols and notations, which are used in these equations [4][3][20].

$$unevenness(r) = \sqrt{\frac{1}{M} \cdot \sum_{j=1}^M \left(\frac{L_j^r}{\bar{L}^r} - 1 \right)^2} \quad (1)$$

$$disequilibrium(j) = \sqrt{\frac{1}{R} \cdot \sum_{r=1}^R \left(\frac{L_j^r - \bar{L}_j}{\bar{L}_j} \right)^2} \quad (2)$$

Table 3 Symbols in equations

Variables	Description	Equation
M	Total Number of HMs	
N	Total Number of VMs	
r	The load of resource (CPU, bandwidth, and memory)	
R	The number of resources	$R = 3$
L_j^r	The load of resource r in the HM	$L_j^r = \sum_{i=1}^N l_i^r \cdot x_{i,j}$
l_i^r	The CPU, memory, and bandwidth load of the ith VM	
$\bar{L}^r$	The average load of resource r	$\frac{1}{M} \sum_{j=1}^M \sum_{i=1}^N l_i^r \cdot x_{i,j}$
$x_{i,j}$	$x_{i,j} = 1$ if ith VM is placed on jth HM otherwise $x_{i,j} = 0$	
$\bar{L}^j$	Average load of all resources in jth HM	$\bar{L}^j = \frac{1}{R} \sum_{r=1}^R \sum_{i=1}^N l_i^r \cdot x_{i,j}$

3.3 Fuzzy system

As mentioned in previous sections, a fuzzy system has four functional blocks: fuzzifier, knowledge base, inference engine, and de-fuzzifier [21] [22]. In the following, we will describe each component.

3.3.1 Fuzzifier

Numerical data feed as inputs to Fuzzy system. As explained earlier, we apply the CV criterion to calculate horizontal and vertical load imbalance. Because CV is a normalization method, all four inputs of the fuzzy system are numbers between zero and one, which should be converted to fuzzy sets. In this regard, for each input, a fuzzy set is defined using a trapezoidal membership function with three linguistic variables: good, medium, and bad. The output of the fuzzy system is also considered as a fuzzy set with a trapezoidal membership function, which has five linguistic variables very good, good, medium, bad, and very bad.

For each numeric input (x) in each membership function, the membership degree ($\mu(x)$) should be calculated using Equation 3 is applied in this regard. Parameters a, b, c, and d in Equation 3 are the four scalar parameters related to four trapezoidal vertices in the membership functions.

$$\mu(x) = \begin{cases} 0 & \text{if } x \leq a \\ \frac{x-a}{b-a} & \text{if } a \leq x \leq b \\ 1 & \text{if } b \leq x \leq c \\ \frac{d-x}{d-c} & \text{if } c \leq x \leq d \\ 0 & \text{if } x \geq d \end{cases} \quad (3)$$

Figure 1 and 2 depicts the membership function for input and output, respectively, where the x-axis represents the input values, and the y-axis shows membership degree $\mu(x)$. Because the frequency of most input values is close to zero, the range of membership functions is considered asymmetric.

Fig. 1 Defined membership functions for the inputs

Fig. 2 Defined membership functions for the output

3.3.2 Knowledge base

This part of the fuzzy system contains a set of rules that are defined as statements "If a and b and c and Then z ", which map fuzzy inputs to the desired fuzzy output. The number of rules is usually equal to the number of combinations of all the different states of the inputs in the fuzzy system. In our problem, there exist four fuzzy sets, and each set contains three membership functions. Therefore, the number of combinations of input states is equal to 3^4 . Therefore, the rule base in our fuzzy system contains 81 rules that do not violate each other. Table 4 depicts some of these rules.

Table 4 Rules

Number	$unev_{CPU}$	$unev_{RAM}$	$unev_{BW}$	disequilibrium	Output
1	Good	Good	Good	Good	VeryGood
2	Good	Medium	Bad	Bad	VeryBad
3	Good	Medium	Medium	Medium	Medium
4	Good	Medium	Good	Good	VeryGood
5	Good	Bad	Bad	Good	Medium
6	Medium	Good	Bad	Medium	Bad
7	Medium	Medium	Bad	Bad	Bad
8	Bad	Bad	Bad	Medium	VeryBad
9	Bad	Bad	Bad	Bad	VeryBad
10	Good	Bad	Good	Medium	Medium

3.3.3 Inference engine

In this section, the compliance of fuzzy inputs with defined rules in the rule base is calculated. The antecedent of a rule consists of smaller statements conjunct with each other by 'And' operation. The lowest membership degree

(Minimum) among the statements of each rule is considered as the output membership degree. Then, in the aggregation step, the results of each rule must be combined. For this purpose, we have used the maximum S-norm.

3.3.4 De-fuzzifier

Finally, after the aggregation step, a numeric value is expressed as the output of the fuzzy system. For this purpose, we have used the center of gravity method. Finally, we compare the generated matrices based on the output of the fuzzy system. Accordingly, any matrix with a smaller output is more desirable.

3.3.5 Fuzzy-RLVMB

As mentioned earlier, in our previous work (RLVMB), we have applied the reinforcement learning method to VM replacement problem. In this algorithm, the non-dominance method has been used to evaluate the generated matrix in each learning episode. Also, to calculate the reward of the reinforcement learning method, the weighted sum of four horizontal and vertical load balance parameters has been calculated. In the Fuzzy-RLVMB algorithm, instead of using the weighted sum, the output of the fuzzy system (described in section 3.3) is used to evaluate the generated matrices and reward the agent.

Algorithm 1 presents the general procedure of the Fuzzy-RLVMB algorithm. Accordingly, in each learning episode, first (line 12), one action is selected from the permissible actions set (random selection, crossover, and mutation). Then a new mapping matrix is generated based on the chosen action(matrix P in line 13). After that, the matrix is examined in terms of feasibility(line 14). In the next step (line 15), the four parameters of vertical and horizontal load imbalance in terms of processor, bandwidth, and memory are calculated for that matrix (by using Equations 1 and 2). After selecting the action and generating a new matrix, the environment enters a new state. We determine the new state of the environment based on four evaluated parameters (line 16). The values of those parameters are then fed to the fuzzy system to evaluate the generated matrix. The output of the fuzzy system is given as a reward to the learning agent (line 17). Fuzzy output is also used to compare the generated matrix in the current episode with the elite matrix (the best matrix obtained throughout the machine learning to the current episode). If the generated matrix is better than the elite matrix, the new matrix is replaced to the elite matrix (line 18,19, and 20). Finally, the action value table and state of the environment is updated (lines 21,22, and 23).

3.3.6 Fuzzy-MOVMB

The MOVMB algorithm is a multi-objective algorithm that tries to balance the load horizontally and vertically in terms of processor, memory, and bandwidth in the VM replacement problem. They used the BBO algorithm for this purpose, which is a population-based meta-heuristic algorithm. To achieve

Algorithm 1: Fuzzy-RLVMrB algorithm

```

1 Inputs :
2    $P(m, n)$       \\ Mapping matrix before rebalance
3    $L(3, n)^*$       \\ VM loads include CPU, memory and bandwidth
4 Output :
5    $P(m, n)^*$       \\ The output mapping matrix
6 Initialization :
7    $Q = 0$ ;        \\ Action value table
8    $P^* = P$ ;
9    $Load(4) = CostCal(P)$  Calculate four load balance criteria ;
10   $S = AssState(Load(4))$ ;
11 for  $inti = 0; i < generationLimit; i++$  do
12    $a = ChoiceAction(S)$ ;    \\ Select an action based on Boltzmann policy
13    $P = PerformAction(P, a)$  ;
14    $P = Feasibility(P)$ ;    \\ Evaluate the feasibility of the matrix
15    $Load(4) = CostCal(P)$ ;
16    $S' = AssState(Load(4))$ ;
17    $Reward = FuzzySystem(Load(4))$ ;
18   Compare( $P, P^*$ ) with Fuzzy system;
19   if  $P$  is better than  $P^*$  then
20      $P^* = P$ ;
21    $a' = Choiceofaction(S')$ ;
22    $Q_{s,a} = (1 - \alpha)Q_{s,a} + \alpha[Reward + \gamma Q_{s',a'}]$ ;
23    $S = S'$ ;
24 return  $P^*$ ;

```

these goals, this algorithm produces solutions in two separate population sets, a set to achieve horizontal load balance and a set to achieve vertical load balance. This increases the execution time of the algorithm. They also use the non-dominance algorithm to evaluate the generated solutions. In the Fuzzy version of the MOVMrB algorithm, we evaluate and compare the generated solutions using fuzzy logic as described in section 3.3. Also, because fuzzy logic is close to human decision-making and has great power, in Fuzzy-MOVMrB we can use a population set instead of creating two separate sets. In the Fuzzy-MOVMrB, the generated solutions are simultaneously compared in terms of both horizontal and vertical load balances.

Algorithm 2 depicts the general procedure of the Fuzzy-RLVMrB algorithm. Accordingly, in the first, an initial population consisting of three mapping matrices is generated (line 8). Then two of these matrices are randomly selected, and a crossover is performed between them (line 10). After that, a matrix of three initial matrices is randomly chosen to perform the mutation (line 11). In the next step, all three matrices are examined in terms of feasibility (line 12). Then the four parameters of vertical and horizontal load imbalance in terms of processor, memory, and bandwidth are calculated separately for all three matrices (line 13). In the next step to evaluate the matrices, the values of these four parameters are given to the fuzzy system. The output of the fuzzy system determines the desirability of the matrices (line 14). Then the previous

best population matrix replaces the worst current population matrix (line 15). Finally, all three matrices are compared with the elite matrix. If a matrix is better than the elite matrix, that matrix replaces the elite matrix (lines 16, 17, and 18).

Algorithm 2: Fuzzy-MOVMrB algorithm

```

1 Input :
2    $P(m, n)$       \\ Mapping matrix before rebalance
3    $L(3, n)^*$       \\ VM loads include CPU, memory and bandwidth
4 Output :
5    $P(m, n)^*$       \\ The Placement matrix after rebalance
6 Initialization :
7    $P^* = P$ ;
8    $P_j$  \\ Production of initial population sets includes three solutions
9 for  $inti = 0; i < generationLimit; i++$  do
10  Do crossover
11  Do mutation
12   $P_j = Feasibility(P)$ ;    \\ Evaluate the feasibility of all matrices
13   $Load(4) = CostCal(P_j)$ ;    \\ Calculate four load balance criteria
14   $Cost = Fuzzysystem(Load(4))$ ;
15  Exchange the worst matrix with the best previous population matrix;
16  Compare( $P_j, P^*$ ) with Fuzzy system output (Cost);
17  if  $P_j$  is better than  $P^*$  then
18     $P^* = P_j$ ;
19 return  $P^*$ ;
```

4 Experimental results

In this section, first, an overview of the simulation setting is given and then, the proposed algorithms are evaluated, and a comparison of them with the baselines are demonstrated.

4.1 Simulation setting

We have simulated four algorithms Fuzzy-RLVMrB, Fuzzy-MOVMrB, RLVMrB, and MOVMrB, using CloudSim 3.0.3 simulator. In this regard, we have altered the "optimizeAllocation" function of the 'VmAllocationPolicy' class. We considered three scenarios with different real and synthetic datasets and different data center sizes, so that we could evaluate the mentioned algorithms in different situations. Table 5 demonstrates the features of each scenario. In the DC, the scheduling policy is considered as Space Shared. In the Space Shared policy, when a CPU is assigned to a VM, it holds it as long as the VM needs the CPU. Also, HMs are considered homogeneous with 32 GB of RAM and a 32-core processor with 1000 MIPS. To execute scenarios, a system with an

Intel Core i3, a maximum frequency of 2.13 GHz processor, 6 GB of RAM, and Windows 10 OS were used.

Table 5 Scenario conditions

Scenario	Number of HM	Number of VM	Dataset
Scenario 1	80	160	Nottingham University
Scenario 2	80	160	Gaussian distribution $N(0.12, 0.05)$
Scenario 3	20	70	Gaussian distribution $N(0.12, 0.05)$

4.2 Performance analysis

This section provides the results of preliminary analysis of proposed algorithms, and a comparison of them with baselines. In the first, we compare them in terms of Horizontal and vertical load balance, and then in term of execution time.

4.2.1 Horizontal and vertical load balance

Table 6 depicts the vertical load imbalance (disequilibrium column) and horizontal load imbalance in terms of processor, memory, and bandwidth ($unev_{CPU}$, $unev_{RAM}$, and $unev_{BW}$ columns). Provided results in Table 6, have been calculated based on an average of 5 executions and in each execution with 10,000 runs. As has been demonstrated in Table 6, in the MOVMrB algorithm, in the first scenario, the horizontal imbalance parameter from the processor dimension is in an ideal state, and this has caused the non-dominance algorithm to have difficulty in decision-making. For this reason, although the vertical load imbalance is not in good condition, this matrix has been selected as the final matrix, but the Fuzzy-MOVMrB algorithm uses a fuzzy system to evaluate the generated matrices. The fuzzy system converts the input values from quantitative to qualitative based on the degree of membership, and calculates and evaluates the output according to defined rules. Therefore, it can choose a solution that its goals are acceptable. Unlike the non-dominance method, the fuzzy system works similarly to human decision-making. The same procedure applies between RLVMrB and Fuzzy-RLVMrB algorithms. The Fuzzy-RLVMrB algorithm was able to select a matrix, which has reached an acceptable level of balance in all dimensions. In scenarios two and three, the Fuzzy-RLVMrB and Fuzzy-MOVMrB algorithms could have better choices than the RLVMrB and MOVMrB algorithms.

4.2.2 Execution time

Another parameter that is important in evaluating algorithms is the algorithm execution time. Table 6 presents the execution time of all algorithms. Due to

Table 6 Simulation results

Scenario	Algorithm	<i>unev_{CPU}</i>	<i>unev_{RAM}</i>	<i>unev_{BW}</i>	disequilibrium	RunTime
Scenario1	Before	0.411	0.38	0.335	0.42	
	MOVMrB	0.0531	0.086	0.0981	0.182	23 _{min}
	Fuzzy-MOVMrB	0.0615	0.087	0.08831	0.099	14 _{min}
	RLVMrB	0.066	0.038	0.082	0.09056	3 _{min}
	Fuzzy-RLVMrB	0.0517	0.0501	0.0742	0.0952	3 _{min}
Scenario2	Before	0.421	0.412	0.512	0.417	
	MOVMrB	0.071	0.067	0.061	0.048	20 _{min}
	Fuzzy-MOVMrB	0.057	0.072	0.063	0.053	12 _{min}
	RLVMrB	0.044	0.046	0.073	0.062	1 _{min}
	Fuzzy-RLVMrB	0.046	0.055	0.0691	0.059	1 _{min}
Scenario3	Before	0.497	0.546	0.336	0.491	
	MOVMrB	0.072	0.0322	0.072	0.0506	4 _{min}
	Fuzzy-MOVMrB	0.048	0.041	0.0552	0.048	2 _{min}
	RLVMrB	0.0525	0.0356	0.0511	0.0515	27 _s
	Fuzzy-RLVMrB	0.043	0.041	0.057	0.0553	42 _s

the fact that the MOVMrB method uses the BBO Complex population-based algorithm and examines the horizontal and vertical load imbalance in two different sets, the execution time of the MOVMrB algorithm is significantly longer than other methods. In the Fuzzy-MOVMrB algorithm, due to the fact that instead of using two separate population sets, only one population set is used, the execution time is reduced compared to the MOVMrB algorithm. Because MOVMrB and Fuzzy-MOVMrB algorithms are population-based algorithms, they have a higher execution time than RLVMrB and Fuzzy-MOVMrB algorithms, which apply reinforcement learning.

5 Conclusion

Cloud DCs performance is highly correlated with VM placement. Although multi-objective methods have been vastly applied in the literature to optimal placement of VMs, the evaluation method which is used to compare candidate solutions affects final results. The non-dominance algorithm is a common method, which has been applied in the related work to evaluate candidate solutions, but it fails to consider a trade-off between objectives. In this paper, we applied fuzzy logic to evaluate output matrices in multi-objective VM replacement problems instead of the non-dominance algorithm. We proposed the Fuzzy-MOVMrB and Fuzzy-RLVMrB algorithms and compared them with the RLVMrB and MOVMrB algorithms in terms of vertical and horizontal load balance (regarding the CPU, memory, and bandwidth) and runtime. The experimental results using the CloudSim simulator depict that fuzzy logic, which is similar to human decision logic, outperforms the non-dominance algorithm in the multi-objective VM replacement problem. Indeed, the non-dominance algorithm encounters difficulty in making decisions when one of the parameters has ideal conditions, while fuzzy logic does not suffer from this issue. Also, the MOVMrB algorithm has a long execution time because it is a population-based algorithm and uses two separate population sets. But in the Fuzzy-MOVMrB

algorithm, in which only one population set is generated, execution time is less than the MOVMrB algorithm. Also, the RLVMrB and Fuzzy-RLVMrB algorithms have much less runtime than the MOVMrB and Fuzzy-MOVMrB algorithms because they are not population-based.

6 Future studies

As future work, we plan to apply fuzzy logic in other multi-objective problems that use other methods instead of non-dominance algorithms and evaluate the result. Moreover, we plan to consider DCs with heterogeneous VMs configuration.

Funding Not applicable

7 Declaration

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. A. Keshavarzi, A. T. Haghighat, M. Bohlouli, Research challenges and prospective business impacts of cloud computing: A survey, in: 2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), Vol. 02, 2013, pp. 731–736.
2. A. Keshavarzi, A. Toroghi Haghighat, M. Bohlouli, Clustering of large scale qos time series data in federated clouds using improved variable chromosome length genetic algorithm (cqga), *Expert Systems with Applications* 164 (2021) 113840.
3. R. Li, Q. Zheng, X. Li, Z. Yan, Multi-objective optimization for rebalancing virtual machine placement, *Future Generation Computer Systems* 105 (2020) 824–842.
4. A. Ghasemi, A. Toroghi Haghighat, A multi-objective load balancing algorithm for virtual machine placement in cloud data centers based on machine learning, *Computing* 102 (2020) 2049–2072.
5. M. Masdari, S. Gharehpasha, M. Ghobaei-Arani, V. Ghasemi, Bio-inspired virtual machine placement schemes in cloud computing environment: taxonomy, review, and future research directions, *Cluster Computing* (2019) 1–31.
6. A. Keshavarzi, A. T. Haghighat, M. Bohlouli, Enhanced time-aware qos prediction in multi-cloud: a hybrid k-medoids and lazy learning approach (qopc), *Computing* 102 (4) (2020) 923–949.

7. K. Braiki, H. Youssef, Fuzzy-logic-based multi-objective best-fit-decreasing virtual machine reallocation, *The Journal of Supercomputing* 76 (1) (2020) 427–454.
8. S. Azizi, D. Li, et al., An energy-efficient algorithm for virtual machine placement optimization in cloud data centers, *Cluster Computing* (2020) 1–14.
9. S. Gharehpasha, M. Masdari, A discrete chaotic multi-objective scs-alo optimization algorithm for an optimal virtual machine placement in cloud data center, *Journal of Ambient Intelligence and Humanized Computing* (2020) 1–17.
10. K. Braiki, H. Youssef, Multi-objective virtual machine placement algorithm based on particle swarm optimization, in: 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC), IEEE, 2018, pp. 279–284.
11. H. T. Vu, S. Hwang, A traffic and power-aware algorithm for virtual machine placement in cloud data center, *International Journal of Grid & Distributed Computing* 7 (1) (2014) 350–355.
12. S. Farzai, M. H. Shirvani, M. Rabbani, Multi-objective communication-aware optimization for virtual machine placement in cloud datacenters, *Sustainable Computing: Informatics and Systems* 28 (2020) 100374.
13. E. Parvizi, M. H. Rezvani, Utilization-aware energy-efficient virtual machine placement in cloud networks using nsga-iii meta-heuristic approach, *Cluster Computing* (2020) 1–23.
14. C. Liu, C. Shen, S. Li, S. Wang, 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, 2014, pp. 272–275.
15. Y. Gao, H. Guan, Z. Qi, Y. Hou, L. Liu, A multi-objective ant colony system algorithm for virtual machine placement in cloud computing, *Journal of computer and system sciences* 79 (8) (2013) 1230–1242.
16. E. Torre, J. J. Durillo, V. de Maio, P. Agrawal, S. Benedict, N. Saurabh, R. Prodan, A dynamic evolutionary multi-objective virtual machine placement heuristic for cloud data centers, *Information and Software Technology* 128 (2020) 106390.
17. S. Wang, H. Gu, G. Wu, A new approach to multi-objective virtual machine placement in virtualized data center, in: 2013 IEEE Eighth International Conference on Networking, Architecture and Storage, IEEE, 2013, pp. 331–335.
18. R. Regaieg, M. Koubaa, Z. Ales, T. Aguil, Multi-objective optimization for vm placement in homogeneous and heterogeneous cloud service provider data centers, *Computing* (2021) 1–25.
19. Q. Zheng, R. Li, X. Li, N. Shah, J. Zhang, F. Tian, K.-M. Chao, J. Li, Virtual machine consolidated placement based on multi-objective biogeography-based optimization, *Future Generation Computer Systems* 54 (2016) 95–122.

20. Q. Zheng, R. Li, X. Li, N. Shah, J. Zhang, F. Tian, K.-M. Chao, J. Li, Virtual machine consolidated placement based on multi-objective biogeography-based optimization, *Future Generation Computer Systems* 54 (2016) 95–122.
21. M. Silvana, R. Akbar, M. Audina, et al., Development of classification features of mental disorder characteristics using the fuzzy logic mamdani method, in: *2018 International Conference on Information Technology Systems and Innovation (ICITSI)*, IEEE, 2018, pp. 410–414.
22. H. S. Mondal, M. T. Hasan, T. K. Karmokar, S. Sarker, Improving quality of service in cloud computing architecture using fuzzy logic, in: *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*, IEEE, 2017, pp. 149–152.