EAI Endorsed Transactions
on Energy Web
Research Article EAI.EU

Optimizing Energy Efficiency in Cloud Data center using an Enhanced Dualist Algorithm with Improved Exploration

Amol C. Adamuthe[1*], Vrushabh D. Kupwade[2]

[1,2] Department of Information Technology, Rajarambapu Institute of Technology, Shivaji University, Sakhrale, MS – 415414, India

Abstract

Investigation for minimizing energy consumption in data centers is increasing due to their heavy usage. In a data center, virtual machine placement is a procedure that maps virtual machines to physical machines. VMP problem is a complex combinatorial optimization problem with various constraints. In literature, the VMP problem is investigated with different objectives. In this paper, the problem is formulated as a single-objective optimization problem with the goal of minimizing energy consumption in cloud data centers. A metaheuristic evolutionary algorithm called the Duelist algorithm is designed to solve the VMP problem. Two variations are proposed with modifications in the winner's learning strategy. The proposed strategy improved the exploration capability of the Duelist algorithm. The performance of proposed variations is tested using 15 datasets with varying problem sizes. Performance is evaluated using the best, mean, standard deviation, success rate, acceleration rate and convergence speed. Variation 1 and variation 2 are better than the basic Duelist algorithm on all measures.

Keywords: Energy efficiency in cloud, virtual machine placement, duelist algorithm

Received on 29 June 2023, accepted on 26 September 2024, published on 03 October 2024

doi: 10.4108/ew.3498

1. Introduction

Cloud computing is a model for delivering computing resources over the Internet, allowing users to access and use computational power, storage, networking and software applications on-demand. These cloud services are available in different forms known as types of cloud services. The main cloud services are Software-as-a-Service (SaaS), Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS). A cloud computing service paradigm called Infrastructure as a Service (IaaS) gives consumers access to virtualized computer resources including storage, networking, and processing power online. IaaS allows users to rent these resources from a cloud service provider on a pay-per-use basis, enabling them to scale up or down their infrastructure needs based on their requirements [1]. The IaaS model offers numerous benefits to both data center providers and end-users, benefiting in cost savings and increased flexibility. Computing resources are sold by data center providers to a large number of customers, resulting in significant revenue generation opportunities. At the same time, end-users are able to purchase computing resources at lower costs than what would be required to maintain private infrastructure. However, in order to maximize profits, cloud providers must manage their resources and reduce deployed computer resources to the extent possible.

The major IT infrastructure for cloud computing, which primarily consists of IT equipment for data processing, storage, and communication is called data center. To comply with the demands of cloud users, data centers operate continuously with a multitude of active hosts, servers, networking equipment and storage devices. The extensive use of energy in various operations is a

*Corresponding author. Email: amol.admuthe@gmail.com

significant challenge for data centers [2]. According to estimates, approximately 3% of the world's electricity is currently consumed by data centers and that share is predicted to rise to 4% by year 2030. The typical hyperscale operation uses 20 to 50 MW of power annually, which could possibly power 37,000 households. Experts at Vertiv anticipate that in year 2023, this will lead to more governmental scrutiny [3]. Since year 2015, the amount of electricity used in data centers in Ireland has more than tripled and by year 2021, it took up 14% of all electricity used. By year 2025, it is anticipated that Denmark's data center energy use will have tripled, making up about 7% of the nation's electrical consumption [4]. The data center industry overview reveals that 10% of global IT organizations will go server-less before year 2023. According to forecasting data, upto year 2025 the data center industry will consume 20 percent of the total energy. Spending on IT data center will reach $222 billion in year 2023 [5]. According to IEA [4], data center workload in year 2015 was 180 million and was 650 million in year 2021 which significantly increased by +260% and data center energy usage in year 2015 was 200 TWh and was 220-320 TWh in year 2021 which increased by +10-60%. The energy consumption in cloud data centers is important issue from environmental point of view. According to reports, the operation of data center servers is responsible for 0.5% of the world's CO2 emissions.

As a result, the research for minimising data center power consumption and the Virtual Machine Placement (VMP) problem in cloud infrastructure increased. Researchers have examined a variety of strategies to lower data centers' energy usage. The management of servers or physical machines can be made easier by applying virtualization technology. In the context of data centers, virtual machine placement, also known as server consolidation, refers to the process of mapping multiple virtual machines (VMs) onto physical machines (PMs) and allowing them to share resources like CPU, storage, bandwidth and memory. This technique enables efficient utilization of resources and is a critical component of data center management. VMP problem is a complex optimization problem in cloud computing due to the dynamic and unpredictable nature of cloud workloads and the numerous constraints involved in placing virtual machines on physical hosts. The problem involves optimizing multiple objectives such as minimizing resource wastage, energy consumption and achieving load balancing. VMP problem is NP, which means it is computationally challenging to discover the optimal solution in a reasonable length of time. The VMP is regarded as a strategy for effectively managing physical machine usage to lower the overall number of active physical machines in the data centers [6]. The original concept involved mapping virtual machines onto a number of operational servers that are energy-efficient, followed by the deactivation of idle or underutilized hosts [7]. About 66% energy consumption is reduced by switching off inactive PMs [8]. Several studies discussed the complexity of the VMP problem. Despite the benefits of virtual machine placement, there are significant challenges that must be addressed. A primary concern is determining the optimal allocation or placement of virtual machines onto physical machines in a manner that minimizes the amount of energy used by data center. This is a crucial research problem that requires careful consideration. The primary aim of virtual machine placement algorithms is to achieve an optimal allocation of virtual machines to physical machines while considering specific design objectives.

The Duelist algorithm is a population-based evolutionary algorithm where Duelist make improvements to win a fight. Every individual in a population is referred to as a duelist. To determine who wins and losses, each duelist engages in a fight with another duelist. Both winners and losers have different ways of improving themselves. The winner improves himself by learning from past mistakes. The loser learns from the winner thereby trying to gain some of the good skills of the winner. Some duelists will emerge as the best solutions to a set of challenges after numerous improvements and duels [9, 10].

The main objectives for our study are as follows:

- Design a Duelist algorithm to solve the cloud data center's virtual machine placement problem to minimize energy consumption.
- Improve performance of Duelist algorithm with better exploration by new learning strategies of winner improvement.
- Fine-tune the algorithm-specific parameters of Duelist algorithms.

The contribution of this paper is improvements in Duelist algorithm for solving VMP by adapting new learning strategies. The proposed variations reduced physical machines' energy usage.

Section 2 presents the related work done in the area of virtual machine placement problem and Duelist algorithms. Virtual machine placement problem with objectives, assumptions and constraints is presented in section 3. Duelist algorithm, proposed variations, solution representation and fitness function is presented in section 4. Section 5 is about experimental details, results and discussion. Section 6 is about the conclusion.

2. Related Work

This section presents work done related to virtual machine placement and duelist algorithm. Different objectives have been investigated in various studies of VMP, such as enhancing resource utilization, minimizing network traffic, achieving optimal placement, reducing power consumption, maximizing economical revenue, utilization rate and quality of service. Deterministic algorithms [11–13] and more intelligent metaheuristic algorithms [14, 15] are few of the approaches that have been suggested to deal with the VMP problem.

Constraint programming based exact allocation and migration algorithms are used to solve VMP problems. These algorithms are proficient in decreasing the number of active physical machines (APMs) and lowering

migration costs. However, a major disadvantage of constraint-based algorithms is their extended search time [16]. Min-cut hierarchical clustering algorithms aimed to optimize the MLU by utilizing CPU, memory and bandwidth. One of the drawback is the higher cost of VM migration [17]. Paper [18] addressed the VMP problem as a linear programming (LP) problem. Their approach focused on minimizing energy and hardware costs by formulating linear programming formulations for server consolidation problems. The paper [19] focuses on load balancing and unbalancing techniques in cluster-based systems to achieve both power and performance optimization. It discusses the trade-offs between these two objectives and proposes a dynamic balancing approach that adapts to workload changes and takes into account the power consumption of each node in the cluster. Paper [20] reduced energy usage by switching off unused servers and used a statistical flip-flop filter to address the virtual machine placement problem.

Stochastic bin packing algorithms based on group packing techniques proposed as a solution for virtual machine placement problems. These algorithms utilize random variables to forecast future bandwidth, which in turn helps in reducing the number of physical machines in data centres that are active [21]. The authors propose a dynamic consolidation approach that combines two heuristics, the first-fit decreasing (FFD) and best-fit decreasing (BFD) algorithms. They also introduce a power-aware load balancing technique called Local Regression (LR) that is used to determine the optimal utilization of the available resources. Additionally, the paper presents a Hybrid Threshold-Min-Migration (HTMM) algorithm that is designed to minimize the number of active servers while ensuring that the service level agreements (SLAs) are met. [22-23]. In literature, investigations are also done based on the Multiple Knapsack Problem and Best Fit Algorithm [24-25]. The paper [26] discusses the challenges associated with server-storage virtualization and proposes a load balancing approach to address them. The proposed approach dynamically migrates virtual machines and storage resources to balance the workload across servers. The paper [27] proposes a novel virtual placement algorithm for virtual computing that aims to improve the placement of virtual machines (VMs) on physical servers to optimize the resource utilization and performance of the virtualized environment. The proposed algorithm takes into account the resource requirements and load of VMs, as well as the resource availability of physical servers to determine the optimal placement of VMs. Papers [28-29] used greedy algorithms to solve the problem.

Various metaheuristic algorithms are proposed in the literature to solve the VMP problem, including simulated annealing [30] and ant colony optimization [14]. These algorithms aim to optimize the VM placement by iteratively improving solutions based on a set of rules or heuristics. The use of metaheuristic algorithms provides a promising approach for solving the VMP problem, as they are able to overcome some of the limitations of exact algorithms in terms of solution quality and computation time. The Genetic Algorithm Based Approach (GABA), which addresses multiobjective optimisation, is used to handle VMP problems [31]. Multi-objective virtual machine placement in a cloud environment focusing on maximising profits, balancing loads, and minimising resource wastage is presented. Performance of three algorithms namely GA, NSGA, and NSGA-II presented [32]. Paper proposed a multi-objective optimization approach using a non-dominated sorting genetic algorithm-II (NSGA-II) to solve the virtual machine placement problem with objective maximize profits, minimize resource wastage and balance the load among physical machines. However, the proposed approach was evaluated only for small-sized problems having up to 60 VMs [33]. With the recent survey, virtual machine placement problem in cloud data center is classified in three ways as resource type, considered VM set and objectives [34]. To solve the power efficiency challenges, the Virtual Machine Placement Framework towards the Power Efficiency of Sustainable Cloud Environment (MV-PESC) technique is recommended [35]. Paper [36] presented an algorithm, FPNSO that optimizes resource usage and reduces energy consumption and carbon emissions in cloud data centers. The algorithm employs FPO and NSGA-II to find suitable physical server for assigning virtual machines. The algorithm is examined by Google Cluster Dataset (GCD). Results showed significant improvements in power consumption, carbon emissions and resource utilization compared to existing approaches. Paper [37] presented a new approach to improve the energy efficiency of VMP problem in cloud data centers. The proposed approach reduced the computational complexity of the genetic algorithm (GA) and accelerates its execution time by using a new data structure and an alternative fitness function. Paper [38] discussed the importance of efficient virtual machine (VM) placement in cloud computing, which aims to reduce energy consumption and increase resource usage. It introduces an enhanced version of the ABCSO algorithm that integrates an adaptive concept, combining the capabilities of Artificial Bee Colony (ABC) and Cat Swarm Optimization (CSO). In the paper [39], the authors propose an Ant Colony Optimization (ACO) algorithm for optimizing virtual machine placement in cloud computing environments. The algorithm aims to improve energy efficiency and consider traffic-awareness to enhance overall system performance. It offers a promising approach for effective resource allocation and management in cloud computing, addressing the challenges of energy consumption and network traffic in virtualized environments. Paper [40] provided a comprehensive review of virtual machine placement methods using metaheuristic algorithms in a cloud environment. It discusses the advantages and disadvantages of various methods and highlights the need for further research in this area.

3. Virtual machine placement problem

The The process of server virtualization involves mounting virtual machines on a collection of physical machines. This process is called virtual machine placement. VMP problem is a key challenge in cloud computing that involves determining the optimal allocation of virtual machines (VMs) to physical servers, also known as hosts or physical machine (PM) or nodes, within a cloud data center. The goal of VMP problem is to achieve efficient utilization of resources such as CPU, storage, memory and network bandwidth while minimizing operational cost and meeting performance requirements of the applications running on the VMs. VMP problem is a complex optimization problem with various constraints. Different objectives have been investigated in various studies, such as load balancing, response time, minimizing number of servers used, throughput, process maximum requests of customers, maximizing resource utilization and minimizing resource wastage, minimizing energy consumption and carbon footprint, maximizing profits and minimizing operational costs, minimizing the number of virtual machine migrations, maximizing the availability and reliability of the virtual machines [22, 33-35, 41]. In this study, the primary goal is to reduce energy consumption in cloud data centers by considering CPU as a promising resource.

The VMP problem is addressed in this experiment as a bin packing problem, and the formulation of the problem is identical to that in [41]. The formulation of VMP problem is presented as, given a set of VMs and a set of PMs in a cloud data center. The goal is to assign VMs to PM with satisfying given constraints to achieve reduction in total energy usage of the data center.

The following are the constraints,

- All virtual machines must be allocated to physical machine.
- Only one physical machine should be allocated to each virtual machine.
- Physical machine must possess sufficient resources for allocated VMs.

The virtual machine placement problem is formulated as,

$$Minimize\ f(p) = \sum_{j=1}^{Q} \alpha_j \times \left\{ \left(P\_max_j^{busy} - P\_min_j^{idle} \right) \times P_{ut_j}^{cpu} + P\_min_j^{idle} \right\} \quad (1)$$

$$P_{ut_j}^{cpu} = \sum_{i=1}^{V} \beta_{ij} \times \frac{CPU\_V_i}{CPU\_P_j} \quad (2)$$

Where,

i € {1,2,3,……,V} and j € {1,2,3,……,Q}

$f(p)$ is total energy consumed by data center due to PMs

α_j show whether or not j[th] physical machine contains virtual machine

$P\_max_j^{busy}$ maximum energy consumption of PM

$P\_min_j^{idle}$ minimum energy consumption of PM

$P_{ut_j}^{cpu}$ is CPU utilization ratio of j[th] physical machine

β_{ij} is a binary that shows whether or not a virtual machine is assigned to a physical machine

CPU_V_i is virtual machine's CPU demand

CPU_P_j is physical machine's CPU capacity

4. Duelist algorithm and proposed variations for VMP

The Duelist algorithm is metaheuristic evolutionary optimization algorithm. It uses the principles of genetic algorithms and is designed to mimic the fighting and learning capabilities of humans. It is based on the concept of 'dueling,' where two potential solutions are pitted against each other and evaluated based on their fitness. Based on the fighting capabilities and luck of the duelist the champions, winner and loser are determined. Each of these categories have their own way of enhancement. Champion trains a new duelist which is similar as himself, winner learns from himself to be more advance and loser learns from winner from whom it lost the duel. Then post qualification is done where worst duelists are eliminated to maintain the pool of duelist. This process is repeated until an optimal solution is found [9,10]. Figure 1 shows flowchart of basic duelist algorithm. Several real world problems in different domains have been solved by using duelist algorithm in refinery crude preheat train cleaning scheduling [42], optimization of oil production in CO2 enhanced oil recovery, operating conditions of steam injection in enhanced oil recovery [43]. This algorithm is hybridized with other algorithms to solve the problems such as response surface methodology along with Duelist algorithm for optimizing the dimensional surface quality and material removal rate in turing [44], Duelist algorithm along with Killer-Whale and Rain-Water algorithms for optimization of energy efficiency and conservation in green building design [45], using Duelist algorithm for optimization of PID controller tuning parameters for multivariable system [46]. Optimization of petlyuk distillation column design [27].

Following are the steps of duelist algorithm.

a) Registration of duelist candidate: It is an initialization step. 1D array is used to register each duelist in a duelist set. In a duelist algorithm, it is referred to as a skillset.

b) Pre-Qualification: Each duelist must pass a pre-qualification test to assess their fighting capacity based on their skill set.

c) Determination of champion's board: In order to keep the best duelist in the game, the board of champions is determined. Each champion will train a new duelist which will have the same capabilities as the champion. The game's champion would be replaced by these new duelists, who would then participate in the subsequent duel.

d) Duel between each duelist: The fight between the duelists is set randomly. Each duelist will participate in the fight utilizing their fighting skills and luck to determine who will win and who will lose. Duelist A wins if his fighting ability combined with his luck is greater than duelist B's, and vice versa. To avoid local optimum, the luck of the duelist is completely determined by random function.
e) Duelist's improvement: Each duelist is categorized into winner and loser after the fight. Each loser is trained by winner. Learning means that the loser may acquire a portion of the winner's array or skill set. The second method is for the winners. Each winner would develop their own skills, where it mutates to be more advance.
f) Elimination: Due to certain new duelists joining the game, an elimination is necessary to maintain the predetermined number of duelists. Based on each duelist's individual duelling abilities, they are eliminated. The duelist who performs the worst in a duel is eliminated.

Figure 2 displays the solution representation used to solve the virtual machine placement problem. The term "VM" is virtual machine and "PM" is physical machine. The solution is represented by a one-dimensional array. The number of virtual machines determines the size of the solution. Consider there are 7 virtual machines and 3 physical machines. The representation of solution is given as,

PM_3	PM_1	PM_1	PM_2	PM_3	PM_2	PM_2
VM_1	VM_2	VM_3	VM_4	VM_5	VM_6	VM_7

Figure 2. Solution Representation

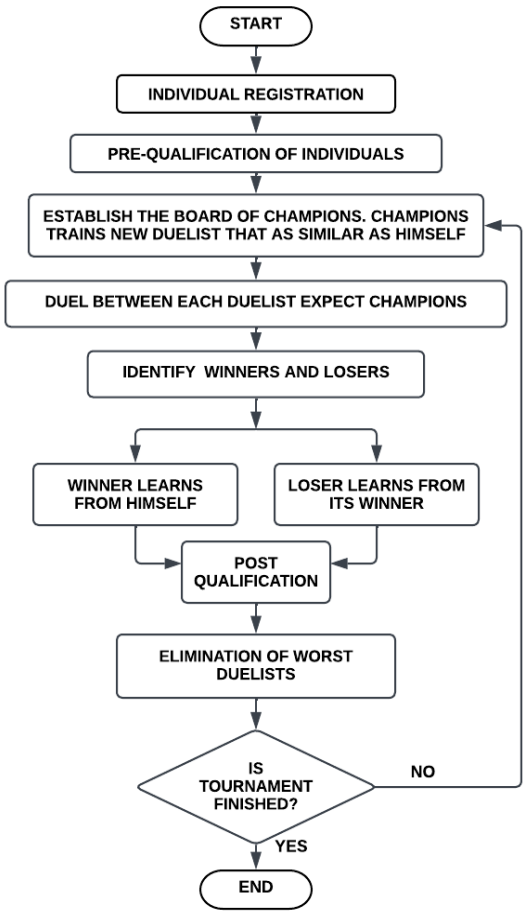

Figure 1. Flowchart of basic Duelist algorithm

Pseudocode for Duelist algorithm for VMP
Input: *no_pm, vm_instance_req_cpu, no_pm, pm_cpu, pm_energy_busy_cpu*
Output: *Minimum fitness value (energy)*
Initialization: *Registration of duelists* *Randomly initialize the solution matrix* *Pre-qualification of registered duelist using fitness (objective) function using equation 1*
Procedure: Begin *While generation< max_generation+1 do* *Determine board of champions (best duelists based on fitness value)* *Champion will train the duelist* *for i=1: (no_of_champions)* *for j = 1: no_vm* *if random(0,1) < mutation_probability* *matrix[j,i]=random (1,no_pm)* *fi* *end*

```
    end

    Dueling procedure
    While i< no_vm do
      If (i== no_vm-1)
         winner_loser[i]=1
      else:
         compute battle score for each of the two duelist. based on fitness and luck
          battlescore[i]=fitness *(1+(luck+(random (0,1)* luck)))
          battlescore[i+1]=fitness *(1+(luck+(random (0,1)* luck)))
        compare battle score and indicate winner and loser
         if(battlescore[i]> battlescore[i+1]):
                    winner_loser[i]=1
                   winner_loser[i+1]=0
        else:
                   winner_loser[i]=0
                   winner_loser[i+1]=1
         fi
      fi
    od

     Duelist improvement procedure
    for j = 0:no_vm:
        for i=0:population size:
           if winner_loser[i]==1:
              if random(0,1)<innovation probability:
                    train[i,j]=random(1, no_pm)
           else:
              if random (0,1)<learning probability:
                    if (i%2==0):
                       train[i,j] = matrix[i+1,j]
                    else:
                       train[i,j] = matrix[i-1,j]
                 fi
            fi
       end
     end

    Elimination of worst duelists
  od
End
```

Pseudocode for VMP objective function
Input: no_pm, vm_instance_req_cpu, no_pm, pm_cpu, pm_energy_busy_cpu
Output: *Minimum fitness value (energy)*
*function f(*g):* *c = g*

```
    fitness_value = 0.0
    temp_pm_cpu = pm_cpu[:]
    for vm, pm in enumerate(c):
        temp_pm_cpu[pm-1] = temp_pm_cpu[pm-1] - vm_instance_req_cpu[vm]
    for i in temp_pm_cpu:
        if i <= 0:
            fitness_value = 100000000
            return fitness_value
    ph_has_vm = [0 for i in range(1, no_pm+1)]
    rows, cols = (no_vm, no_pm)
    vm_assign_to_pm = [[0 for i in range(cols)] for j in range(rows)]
    k = 0
    for j :1, no_pm+1:
        for i : 0,no_vm:
            if c[i] != j and 0:
                ph_has_vm[k] = 0
            elseif c[i] == j:
                ph_has_vm[k] = 1
        k += 1
    for j ; 0,no_pm:
        for i:0:no_vm:
            if c[i] == j+1:
                vm_assign_to_pm[i][j] = 1
    calculate_p_idle()
    for j:0,no_pm:
        cpu_utilization = 0.0
        for i:0:no_vm:
            cpu_utilization += (vm_assign_to_pm[i][j] * vm_instance_req_cpu[i])
                        pm_cpu_utilization[j] = (cpu_utilization / pm_cpu[j])
    for j:0,no_pm:
            fitness_value += ph_has_vm[j] * (((pm_energy_busy_cpu[j] - pm_energy_idle_cpu[j]) *
            pm_cpu_utilization[j]) + pm_energy_idle_cpu[j])

    return fitness_value

function calculate_p_idle():
    for j:0,no_pm:
        pm_energy_idle_cpu[j] = 0.6 * pm_energy_busy_cpu[j]
```

Pseudocode for duelist algorithm and proposed fitness for solving virtual machine placement is presented. In the algorithm no_vm is number of virtual machines, vm_instance_req_cpu is CPU values of virtual machines, no_pm is number of physical machines, pm_cpu is CPU values of physical machines, pm_energy_busy_cpu is and CPU values at busy state of physical machine. The algorithm involves a set of duelist solutions that compete with each other through a series of duels to improve their virtual machine placement solutions. The algorithm starts by randomly initializing the solution matrix and pre-qualifying the duelist solutions based on their fitness score. The duelist solutions then engage in a dueling procedure where they compete based on their fitness and luck. The winner of each duel has the opportunity to improve its solution by mutating its innovating its placement solution. Loser learns from winner by transferring the information. Again post qualification is done and based on fitness value duelist who performed worst are eliminated. This process continues till last iterations.

Proper balance of exploration and exploitation is important for any evolutionary algorithm. Improper balance leads to premature convergence. In the basic Duelist algorithm, the solution improvement is done with updating on winner and loser. Each loser is trained by winner. Learning means that the loser may acquire a portion of the winner's array or skill set. It works for exploring the new search space. The second updating method is for the winners. Each winner would develop their own skills, where it mutates to be more advance. It works like exploitation of search space. Paper presents two variations of Duelist algorithm. Figure 3, shows the flowchart for proposed variations in Duelist algorithm that focused on Duelist's improvement. In proposed variation 1, winner learns from best champion to get more skills. In variation 2, winner learns from group of champion also known as board of champions.

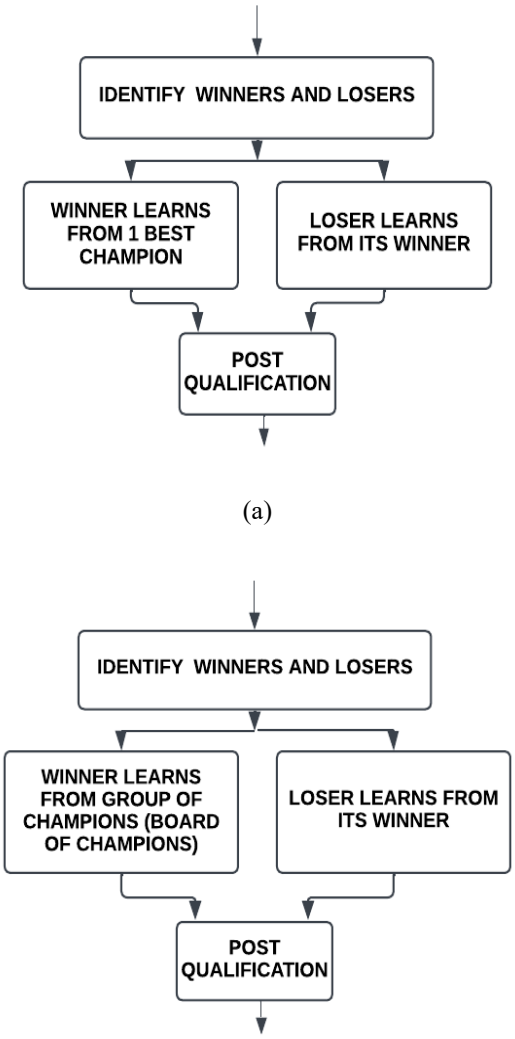

Figure 3. Flowchart of Duelist's improvement step with proposed (a) variation 1 (b) variation 2

5. Experimental details, Results and Discussion

5.1. Simulation details

This section presents simulation details required for conducting the experiments to examine the performance of algorithms and various datasets used for virtual machine placement problem. The performance of duelist algorithm and proposed variations tested with 15 different datasets. These datasets are generated using a python program which generates a dataset for the problem with different characteristics. Datasets have a diverse number of VMs and PMs. Based on the number of physical and virtual machines these datasets are classified as small, medium, and large datasets. Maximum number of PMs are 450 and VMs are 600. The minimum number of PMs are 35 and minimum number of VMs are 35. Virtual machine CPU values considered are between 50 and 150 to generate the dataset. By meeting the restriction of having smaller Vcpu values than the available Pcpu capacities, these VM resources are produced. Two python programs are written to generate dataset. Inputs are generated randomly by the first programme, and it creates random CPU values for virtual machines within a specific range. Five datasets are generated by this method. Remaining ten are generated by the second program which generates inputs by normal distribution. It generates distributed VM's CPU values normally. Some of parameter's values are set fixed as per the standards.

Table 1 represents specifications for virtual machine and physical machines with their values. The unit of CPU is Million Instructions per Second abbreviated as MIPS. Table 2 describes dataset details used in experiments. Total 15 datasets are used for evaluating performance of the algorithms, these datasets are categorized as small, medium and large datasets based on number of physical and virtual machines. It also describes method of generation whether the dataset is randomly generated or generated by normal distribution.

Table 1. VM and PM specifications

PM CPU	1000 (MIPS)
VM CPU Usage	50-150 (In MIPS)
PM CPU at busy rate	250 (MIPS)

Table 2. Dataset description

Dataset	No. of PM:VM	Size	Method of generation
1	35:50	Small	Random
2	40:100		Random
3	50:120		Random
4	70:150		Random
5	100:200	Medium	Random
6	150:200		Normal
7	150:300		Normal
8	200:300		Normal
9	250:300		Normal
10	290:400		Normal
11	300:450		Normal
12	350:500	Large	Normal
13	400:500		Normal
14	450:600		Normal
15	500:700		Normal

For experimentation, heterogeneous virtual machines and homogeneous physical machines are used. The number of VMs, virtual machine's CPU values, number of physical

machines, physical machine's CPU values and CPU values at busy state of physical machine are given as input to the algorithm. The input file for the dataset has the information as given in figure 4.

```
120     Number of VMs
…       CPU values of 120 virtual machines

50      Number of PMs
…       CPU values of 50 PMs
…       Pbusy values of PMs
```

Figure 4. Sample input file

The duelist algorithm and its variations are executed using the 'Python' programming language. The algorithm programs executed on system with an Intel(R) Core(TM) i5-6300U CPU -2.50GHz - 2.40 GHz and 8 GB RAM.

5.2. Results and Discussion

The performance of each of the three algorithms, Basic Duelist algorithm (DA), variation 1, and variation 2 for all 15 dataset instances are examined. Luck coefficient is luck factor of duelist used to introduce randomness, mutation probability is used in champion training. Based on innovation probability winner learns and based on learning probability loser learns from winner. The parameter values considered are number of 100 generation, 100 population, 0.01 luck coefficient, 0.1 mutation probability, 0.3 innovation probability, 0.6 learning probability and 5 number of champions. All algorithms have executed 10 times by using same parameter values and results are obtained.

For evaluation of the performance of all the algorithms, this paper considers best, mean and standard deviation values. Best fitness value is the optimal value generated by the program among all 10 executions. Mean is average of fitness value of all population obtained at 100th iteration and standard deviation is standard deviation of that population. Paper [48] have provided six matrices for assessing effectiveness of algorithms. The success rate (SR), the number of function evaluations (NFEs), convergence graph, the improvement and acceleration rate (AR). Following equations are used to calculate the results.

$$Success\ rate = \frac{(number\ of\ times\ value\ obtained\ is\ greater\ than\ average\ of\ all\ execution)}{number\ of\ times\ algorithm\ executed} \quad (3)$$

$$NFE = Population \times Number\ of\ generations\ needed\ for\ convergence \quad (4)$$

$$Acceleration\ rate = \frac{NFE\ of\ proposed\ variation}{NFE\ of\ basic\ DA} \quad (5)$$

$$Improvement = \left(NEF_{other} - NEF_{proposed}\right) \times \frac{100}{NEF_{other}} \quad (6)$$

Table 3 show the performance of three algorithms based on best, mean and standard deviation on 15 datasets. For all three categories of dataset, the proposed variations 1 and 2 outperforms the basic Duelist algorithm. Variation 2 gives reasonably good outcomes than variation 1. It shows that variation 1 and variation 2 obtained an average 41.45 % and 50.36% improvement with respect to basic DA respectively. In literature, it is reported that performance of some evolutionary reduces with increase in problem size and problem complexity. Results in table 3 show that the performance improvement in proposed Duelist variations are consistent. There is no performance degradation in Duelist variation1 and 2 with respect to problem size.

Table 3. Best, mean, standard deviation and improvements in algorithm

Dataset		Basic DA	Variation 1	Variation 2	Improvement in variation 1 (in %)	Improvement in variation 2 (in %)
1	Best	2681	2231	**1931**	54	60
	Mean	2960	2381	**2081**		
	SD	56	0	0		
2	Best	5023	4123	**3973**	55	50
	Mean	5578	4273	**4213**		
	SD	77	0	73		
3	Best	6600	5250	**4950**	42	50
	Mean	7282	5395	**5250**		
	SD	77	33	0		
4	Best	8861	6461	**6161**	33	33
	Mean	9737	6854	**6311**		
	SD	58	94	0		

5	Best	12747	10047	**9897**	21	30
	Mean	13776	10424	**10173**		
	SD	76	102	69		
6	Best	17951	13751	**13151**	56	80
	Mean	19163	13971	**13712**		
	SD	117	129	146		
7	Best	20175	16575	**15225**	17	36
	Mean	21324	17124	**15766**		
	SD	90	150	157		
8	Best	23025	17625	**17325**	47	43
	Mean	24540	18567	**18096**		
	SD	153	172	207		
9	Best	25275	19125	**17475**	52	57
	Mean	27126	20293	**18636**		
	SD	172	208	258		
10	Best	32880	25530	**22830**	12	34
	Mean	34498	26382	**24057**		
	SD	64	199	233		
11	Best	35699	30749	**28649**	51	61
	Mean	37472	31824	**29828**		
	SD	189	258	255		
12	Best	40883	34883	**33233**	47	65
	Mean	42839	36247	**34395**		
	SD	103	311	234		
13	Best	43133	36983	**32784**	48	57
	Mean	45496	38498	**34001**		
	SD	74	267	277		
14	Best	50857	43507	**41557**	31	45
	Mean	53116	44854	**42458**		
	SD	155	279	277		
15	Best	48981	45682	**44632**	50	50
	Mean	50974	47135	**45864**		
	SD	87	373	280		

Table 4. Average, best value and success rate

Dataset	Category	Basic DA			Variation 1			Variation 2		
		Average	Best	SR	Average	Best	SR	Average	Best	SR
1	Small	2801	2681	40	2231	2231	100	**2051**	1931	60
2	Small	5203	5023	60	4198	4190	60	**4153**	3973	60
3	Small	6690	6600	40	5220	5220	60	**5100**	4950	60
4	Small	8951	8861	40	6771	6461	60	**6581**	6161	40
5	Medium	13017	13017	40	10647	10047	60	**10009**	9897	60
6	Medium	18101	17951	60	14163	13701	60	**13901**	13151	60
7	Medium	20265	20175	40	16755	16125	60	**15225**	16125	60
8	Medium	23226	23025	40	18555	17625	40	**18135**	17325	80
9	Medium	25575	25225	60	19515	19125	60	**17475**	19005	60
10	Medium	32850	32580	60	27030	25530	60	**25490**	22830	60
11	Medium	35849	35699	40	31949	30749	40	**29549**	28649	60

12	Large	40973	40883	40	35633	34883	60	**33533**	33233	60
13	Large	43343	43133	40	37853	36983	40	**35123**	32784	60
14	Large	51307	50857	60	44527	43507	60	**42757**	41557	60
15	Large	49192	48981	60	46582	45682	60	**45921**	44632	60

Table 5. Comparison of results using function evaluations and acceleration rate

Dataset	Function evaluations				Acceleration rate wrt basic DA	
	Base fitness value	Basic DA	Variation1	Variation 2	Variation 1	Variation 2
1	2681	8700	4000	3600	2.17	2.41
2	5023	9800	4400	4900	2.22	2.00
3	6600	5000	2900	2500	1.72	2.00
4	8861	6800	4500	4500	1.51	1.51
5	13017	5500	4300	3800	1.27	1.44
6	17951	9100	4000	1800	2.28	5.05
7	20175	4700	3900	3000	1.20	1.50
8	23025	6700	3500	3800	1.91	1.76
9	25225	9100	4300	3900	2.11	2.33
10	33030	5000	4400	3300	1.13	1.56
11	35699	9300	4500	3600	2.06	2.58
12	40883	6300	3300	2200	1.91	2.86
13	43133	7600	3900	3200	1.95	2.38
14	50857	6000	4100	3300	1.46	1.89
15	48981	7800	3900	3900	2.00	2.00

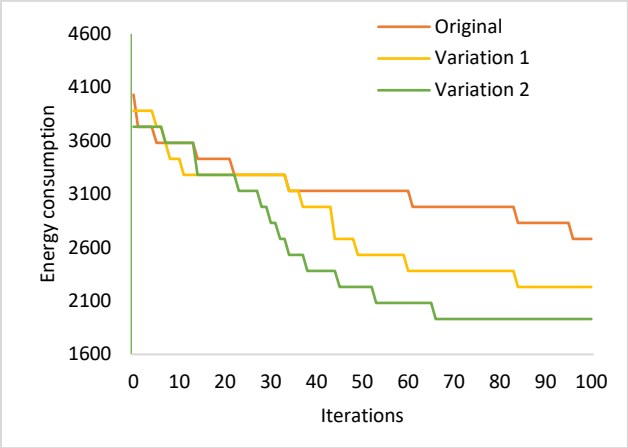

Figure 5. Convergence curve of three algorithms for dataset 1

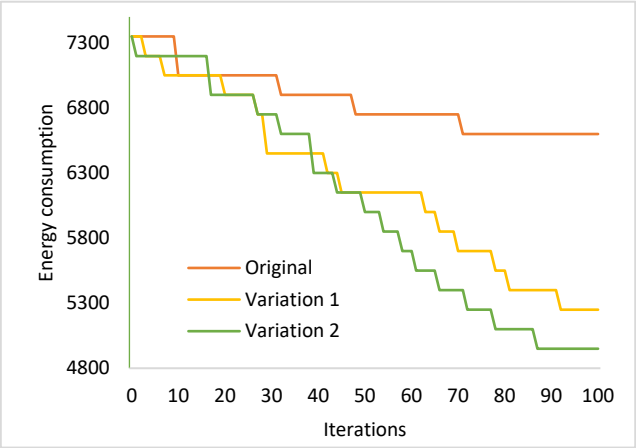

Figure 6. Convergence curve of three algorithms for dataset 3

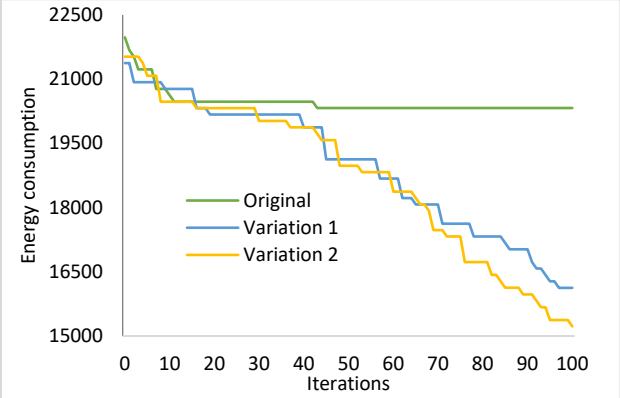

Figure 7. Convergence curve of three algorithms for dataset 7

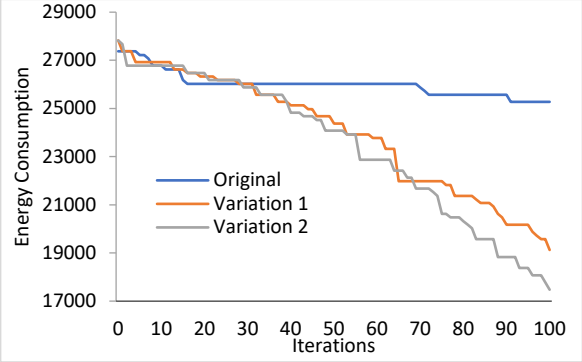

Figure 8. Convergence curve of three algorithms for dataset 9

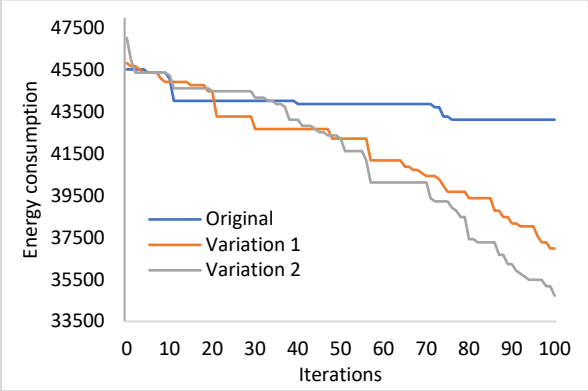

Figure 9. Convergence curve of three algorithms for dataset 13

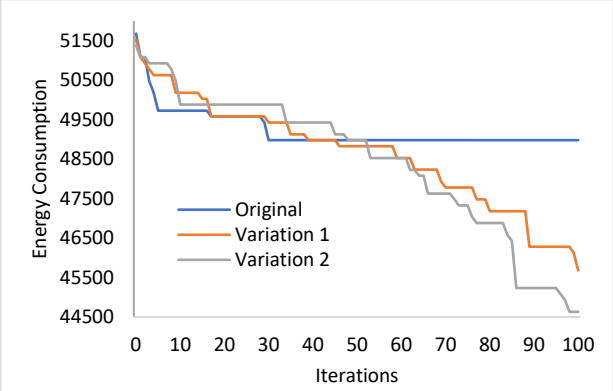

Figure 10. Convergence curve of three algorithms for dataset 15

Figure 5 and 6 show the convergence curves of two small datasets. Figure 7 and 8 show the convergence curves for two medium datasets. Figure 9 and 10 show the convergence curves for two large datasets. The results show that proposed variations of Duelist algorithm outperforms basic Duelist algorithm. Proposed variations show faster convergence than basic Duelist algorithms for all six datasets with varying sizes. The basic Duelist algorithm stuck into local optima. The proposed two variations show consistent progress towards optimal solution with fast convergence speed.

Duelist algorithm and its variations have different parameters as discussed in above section. In literature it is reported that algorithms specific parameters values have significant impact on the performance of algorithm. This sub-section presents results of impact of mutation probability, innovation probability, learning probability and luck coefficient parameter values of Duelist algorithm and proposed variations. In this investigation, number of generations, population size and number of champions are kept constant. Number of generations, population size and number of champions are 100, 100 and 5 respectively. For fine tuning the parameters we have taken one dataset from small, medium and large category and experimented with different parameter values. Table 6, 7 and 8 shows the results of parameter tuning on dataset number 2, 8 and 13 respectively. Results show that for all selected datasets of varying size, proposed variations of Duelist algorithms performs better than basic Duelist algorithm. Four values are tested for luck parameter. There is no significant change in performance of basic Duelist algorithm for tested luck parameter value. For proposed two variations the 0.2 luck value gives best results. Mutation probability value 0.05 gives better results for all the three algorithms. Lower mutation probability is better for algorithm convergence. Higher value of innovation probability is not suitable for performance of all three algorithms

Table 6 Results of parameter tuning on dataset number 2

Parameter	Parameter value	Basic DA	Variation 1	Variation 2
Luck	0.01	5233	4183	**4063**
	0.07	5173	4213	**4093**
	0.15	5173	4333	**4033**
	0.2	5293	4183	**3973**
Mutation probability	0.05	4903	4153	**4123**
	0.1	5293	4183	**3973**
	0.2	5323	4273	**4243**
	0.3	5353	4243	**4223**
Innovation probability	0.2	5203	4153	**4123**
	0.3	5203	4273	**3973**
	0.45	5293	4453	**4303**
	0.6	5233	4513	**4423**
Learning probability	0.5	5263	4243	**4063**
	0.6	5203	4273	**3973**
	0.75	5203	4183	**4243**
	0.9	5053	4543	**4153**

Table 7 Results of parameter tuning on dataset number 8

Parameter	Parameter value	Basic DA	Variation 1	Variation 2
Luck	0.01	23355	18645	**18585**
	0.07	23175	17805	**17025**
	0.15	23145	**17565**	18165
	0.2	23175	17265	**16815**
Mutation probability	0.05	22575	16455	**16215**
	0.1	23265	16725	**16715**
	0.2	23385	17865	**17845**
	0.3	23385	**18525**	19125
Innovation probability	0.2	22485	17115	**16665**
	0.3	22575	16455	**16215**
	0.45	22785	16425	**16335**
	0.6	22515	**18255**	18295
Learning probability	0.5	22725	16935	**16185**
	0.6	22575	16455	**16215**
	0.75	22605	**16485**	16515
	0.9	21555	**17775**	17925

Table 8 Results of parameter tuning on dataset number 13

Parameter	Parameter value	Basic DA	Variation 1	Variation 2
Luck	0.01	43553	38393	**36593**
	0.07	43613	**34913**	36023
	0.15	43463	34734	**34013**
	0.2	43253	34703	**33413**
Mutation probability	0.05	42653	33113	**33053**
	0.1	43463	34703	**33413**
	0.2	43733	**35333**	36623
	0.3	44004	**34493**	36653
Innovation probability	0.2	42443	**33923**	35123
	0.3	42653	33113	**32303**
	0.45	43104	32513	**32093**
	0.6	43133	34223	**34193**
Learning probability	0.5	42533	33413	**33413**
	0.6	42653	33113	**32303**
	0.75	41783	33713	**33453**
	0.9	41873	34973	**34553**

5. Conclusion

The paper presents virtual machine placement problem in cloud computing with the objective of minimization of energy consumption in cloud data centers. Energy usage is strongly impacted by the allocation of virtual machines to physical machines. This paper presents an enhanced dualist algorithm with improved exploration for optimizing energy efficiency in virtual machine placement. The Duelist algorithm and its two variations are designed to solve the problem. The proposed variations focused on increasing exploration of the Duelist algorithm by changing the learning strategy of the winner. In the basic Duelist algorithm, the winner learns from himself to be more advance. In the first variation, winner learns from one best champion and in the second variation, winner learns from a group of champions.

Fifteen datasets with varying sizes are generated using random and normal distribution. The datasets are categorized into small, medium and large datasets. The proposed variations shows proper balance in exploration and exploitation which avoided premature convergence and shows consistent improvements. For all three categories of the dataset, the proposed variations 1 and 2 outperform the basic Duelist algorithm. Variation 2 gives reasonably good outcomes than variation 1. It shows that variation 1 and variation 2 obtained an average of 41.45 % and 50.36% improvement with respect to basic DA respectively. Variation 1 and variation 2 have an average success rate of more than 58% for all 15 datasets. The success rate of proposed variations for small, medium and large datasets is better than the basic Duelist algorithm. Results of the acceleration rate show that the proposed variations convergence speed is higher than the basic Duelist algorithm. Proposed variations show faster convergence than basic Duelist algorithms for all six datasets with varying sizes. The basic Duelist algorithm is stuck into local optima. The proposed two variations show consistent progress towards an optimal solution with fast convergence speed.

References

[1] Buyya R, Broberg J, Goscinski AM, editors. Cloud computing: Principles and paradigms. John Wiley & Sons; 2010 Dec 17.

[2] Shuja J, Gani A, Shamshirband S, Ahmad RW, Bilal K. Sustainable cloud data centers: a survey of enabling techniques and technologies. Renewable and Sustainable Energy Reviews. 2016 Sep 1;62:195-214.

[3] Energy Efficiency Predictions for Data Centres in 2023. (2022, December 30). Data Centre Magazine. Retrieved from

https://www.datacentremagazine.com/articles/efficiency-to-loom-large-for-data-centre-industry-in-2023
[4] IEA. (2022). Data Centres and Data Transmission Networks. IEA, Paris. Retrieved from https://www.iea.org/reports/data-centres-and-data-transmission-networks
[5] 15 Crucial Data Center Statistics to Know in 2023. (n.d.). Techjury. Retrieved from https://techjury.net/blog/data-center-statistics/
[6] Speitkamp B, Bichler M. A mathematical programming approach for server consolidation problems in virtualized data centers. IEEE Transactions on services computing. 2010 May 20;3(4):266-78.
[7] Tang M, Pan S. A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers. Neural processing letters. 2015 Apr;41:211-21.
[8] Chen G, He W, Liu J, Nath S, Rigas L, Xiao L, Zhao F. Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services. InNSDI 2008 Apr 16 (Vol. 8, pp. 337-350).
[9] Biyanto TR, Fibrianto HY, Nugroho G, Hatta AM, Listijorini E, Budiati T, Huda H. Duelist algorithm: an algorithm inspired by how duelist improve their capabilities in a duel. InAdvances in Swarm Intelligence: 7th International Conference, ICSI 2016, Bali, Indonesia, June 25-30, 2016, Proceedings, Part I 7 2016 (pp. 39-47). Springer International Publishing.
[10] Ruki Biyanto T, Yernias Fibrianto H, Nugroho G, Listijorini E, Budiati T, Huda H. Duelist Algorithm: An Algorithm Inspired by How Duelist Improve Their Capabilities in a Duel. arXiv e-prints. 2015 Dec:arXiv-1512.
[11] Chaisiri S, Lee BS, Niyato D. Optimal virtual machine placement across multiple cloud providers. In2009 IEEE Asia-Pacific Services Computing Conference (APSCC) 2009 Dec 7 (pp. 103-110). IEEE.
[12] Alicherry M, Lakshman TV. Optimizing data access latencies in cloud systems by intelligent virtual machine placement. In2013 Proceedings IEEE INFOCOM 2013 Apr 14 (pp. 647-655). IEEE.
[13] Dang HT, Hermenier F. Higher SLA satisfaction in datacenters with continuous VM placement constraints. InProceedings of the 9th workshop on hot topics in dependable systems 2013 Nov 3 (pp. 1-6).
[14] Qin Y, Wang H, Zhu F, Zhai L. A multi-objective ant colony system algorithm for virtual machine placement in traffic intense data centers. IEEE access. 2018 Oct 9;6:58912-23.
[15] Abdel-Basset M, Abdle-Fatah L, Sangaiah AK. An improved Lévy based whale optimization algorithm for bandwidth-efficient virtual machine placement in cloud computing environment. Cluster Computing. 2019 Jul;22:8319-34.
[16] Ghribi C, Hadji M, Zeghlache D. Energy efficient vm scheduling for cloud data centers: Exact allocation and migration algorithms. In2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing 2013 May 13 (pp. 671-678). IEEE.
[17] Dong J, Wang H, Cheng S. Energy-performance tradeoffs in IaaS cloud with virtual machine scheduling. China communications. 2015 Feb;12(2):155-66.
[18] Speitkamp B, Bichler M. A mathematical programming approach for server consolidation problems in virtualized data centers. IEEE Transactions on services computing. 2010 May 20;3(4):266-78.
[19] Pinheiro E, Bianchini R, Carrera EV, Heath T. Load balancing and unbalancing for power and performance in cluster-based systems. Rutgers University; 2001.
[20] Chase JS, Anderson DC, Thakar PN, Vahdat AM, Doyle RP. Managing energy and server resources in hosting centers. ACM SIGOPS operating systems review. 2001 Oct 21;35(5):103-16.
[21] Wang M, Meng X, Zhang L. Consolidating virtual machines with dynamic bandwidth demand in data centers. In2011 Proceedings IEEE INFOCOM 2011 Apr 10 (pp. 71-75). IEEE.
[22] Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future generation computer systems. 2012 May 1;28(5):755-68.
[23] Xia B, Tan Z. Tighter bounds of the First Fit algorithm for the bin-packing problem. Discrete Applied Mathematics. 2010 Aug 6;158(15):1668-75.
[24] Fidanova S, Fidanova S. Multiple Knapsack Problem. Ant Colony Optimization and Applications. 2021:9-18.
[25] Song Y, Zhang C, Fang Y. Multiple multidimensional knapsack problem and its applications in cognitive radio networks. InMILCOM 2008-2008 IEEE Military Communications Conference 2008 Nov 16 (pp. 1-7). IEEE.
[26] Singh A, Korupolu M, Mohapatra D. Server-storage virtualization: integration and load balancing in data centers. InSC'08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing 2008 Nov 15 (pp. 1-12). IEEE.
[27] Mohamadi E, Karimi M, Heikalabad SR. A Novel Virtual Placement in Virtual Computing. Australian Journal of Basic and Applied Scienes, Australia. 2011;5(10):1149-555.
[28] Wood T, Shenoy P, Venkataramani A, Yousif M. Sandpiper: Black-box and gray-box resource management for virtual machines. Computer Networks. 2009 Dec 3;53(17):2923-38.
[29] Salehi MA, Krishna PR, Deepak KS, Buyya R. Preemption-aware energy management in virtualized data centers. In2012 IEEE Fifth International Conference on Cloud Computing 2012 Jun 24 (pp. 844-851). IEEE.
[30] Wu Y, Tang M, Fraser W. A simulated annealing algorithm for energy efficient virtual machine placement. In2012 IEEE international conference on systems, man, and cybernetics (SMC) 2012 Oct 14 (pp. 1245-1250). IEEE.
[31] Mi H, Wang H, Yin G, Zhou Y, Shi D, Yuan L. Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In2010 IEEE International Conference on Services Computing 2010 Jul 5 (pp. 514-521). IEEE.
[32] Adamuthe AC, Patil JT. Differential evolution algorithm for optimizing virtual machine placement problem in cloud computing. International Journal of Intelligent Systems and Applications. 2018 Jul 1;10(7):58.
[33] Adamuthe AC, Pandharpatte RM, Thampi GT. Multiobjective virtual machine placement in cloud environment. In2013 international conference on cloud & ubiquitous computing & emerging technologies 2013 Nov 15 (pp. 8-13). IEEE.
[34] Mann ZÁ. Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms. Acm Computing Surveys (CSUR). 2015 Aug 10;48(1):1-34.
[35] Singh AK, Swain SR, Lee CN. A metaheuristic virtual machine placement framework toward power efficiency of sustainable cloud environment. Soft Computing. 2023 Apr;27(7):3817-28.

[36] Singh AK, Swain SR, Saxena D, Lee CN. A bio-inspired virtual machine placement toward sustainable cloud resource management. IEEE Systems Journal. 2023 Mar 13.
[37] Ding Z, Tian YC, Wang YG, Zhang WZ, Yu ZG. Accelerated computation of the genetic algorithm for energy-efficient virtual machine placement in data centers. Neural Computing and Applications. 2023 Mar;35(7):5421-36.
[38] Pushpa R, Siddappa M. Adaptive Hybrid Optimization Based Virtual Machine Placement in Cloud Computing. In2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT) 2022 Jan 20 (pp. 1-9). IEEE.
[39] Xing H, Zhu J, Qu R, Dai P, Luo S, Iqbal MA. An ACO for energy-efficient and traffic-aware virtual machine placement in cloud computing. Swarm and Evolutionary Computation. 2022 Feb 1;68:101012.
[40] Alsadie D. Virtual Machine Placement Methods using Metaheuristic Algorithms in a Cloud Environment-A Comprehensive Review. International Journal of Computer Science & Network Security. 2022;22(4):147-58.
[41] Abohamama AS, Hamouda E. A hybrid energy–aware virtual machine placement algorithm for cloud environments. Expert Systems with Applications. 2020 Jul 15;150:113306.
[42] Biyanto TR, Ramasamy M, Jameran AB, Fibrianto HY. Thermal and hydraulic impacts consideration in refinery crude preheat train cleaning scheduling using recent stochastic optimization methods. Applied Thermal Engineering. 2016 Sep 5;108:1436-50.
[43] Biyanto TR, Irawan S, Ginting HJ, Fitri AI. Operating Conditions Optimization of Steam Injection in Enhanced Oil Recovery Using Duelist Algorithm. International Journal of Industrial and Manufacturing Engineering. 2017 Jan 23;11(2):272-7.
[44] LASHIN MM, GAAFER AM, AL NEMER GN. OPTIMIZATION OF DIMENSIONAL, SURFACE QUALITY AND MATERIAL REMOVAL RATE IN TURNING USING RESPONSE SURFACE METHODOLOGY AND DUELIST ALGORITHM.
[45] Biyanto TR, Syamsi MN, Fibrianto HY, Afdanny N, Rahman AH, Gunawan KS, Pratama JA, Malwindasari A, Abdillah AI, Bethiana TN, Putra YA. Optimization of energy efficiency and conservation in green building design using Duelist, Killer-Whale and Rain-Water Algorithms. InIOP conference series: materials science and engineering 2017 Nov (Vol. 267, No. 1, p. 012036). IOP Publishing.
[46] Biyanto TR, Sehamat N, Sordi NA, Zabiri H. Optimization of PID controller tuning parameters for multivariable system using Duelist algorithm. InIOP Conference Series: Materials Science and Engineering 2018 Dec 1 (Vol. 458, No. 1, p. 012053). IOP Publishing.
[47] Biyanto TR, Rahman JA, Laila HN, Abdurrakhman A, Darwito PA. Techno economic optimization of petlyuk distillation column design using duelist algorithm. Procedia engineering. 2017 Jan 1;170:520-7.
[48] Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, Tiwari S. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. KanGAL report. 2005 May 30;2005005(2005):2005.