ISSN: 0067-2904

Power-Efficient Virtual Machine Placement in Cloud Datacenters using Heuristic Assisted Enhanced Discrete Particle Swarm Optimization

Seema A. Alsaidy*, Nada A. Z. Abdullah
Computer Science, College of Science, University of Baghdad, Baghdad, Iraq

Received: 2/11/2021 Accepted: 16/12/2021 Published: 30/10/2022

Abstract
The increase in cloud computing services and the large-scale construction of data centers led to excessive power consumption. Datacenters contain a large number of servers where the major power consumption takes place. An efficient virtual machine placement algorithm is substantial to attain energy consumption minimization and improve resource utilization through reducing the number of operating servers. In this paper, an enhanced discrete particle swarm optimization (EDPSO) is proposed. The enhancement of the discrete PSO algorithm is achieved through modifying the velocity update equation to bound the resultant particles and ensuring feasibility. Furthermore, EDPSO is assisted by two heuristic algorithms random first fit (RFF) and random best fit (RBF) to produce hydride algorithms termed RFF-EDPSO and RBF-EDPSO. The proposed algorithms are evaluated and compared with recent algorithms to minimize power consumption. Simulation results showed the effective performance of RFF-EDPSO and RBF-EDPSO in minimizing the number of operating servers.

Keywords: Particle swarm optimization, Virtual machine placement, Cloud computing, Metaheuristic algorithms.

وضع جهاز افتراضي موفر للطاقة في مراكز البيانات السحابية باستخدام تحسين مجرى الجسيمات المنفصلة المعزز المحسن

سيماء عبدالله صاحب*, ندا عبد الزهرة عبدالله
قسم علوم الحاسبات,كلية العلوم, جامعة بغداد, بغداد, العراق

الخلاصة
أدت الزيادة الهائلة في خدمات الحوسبة السحابية والبناء واسع النطاق لمراكز البيانات إلى استهلاك مفرط للطاقة. تضم مراكز البيانات عددًا كبيرًا من الخوادم حيث يتم استهلاك الطاقة بشكل كبير. تعد خوارزمية وضع الآلة الافتراضية الفعالة أمرًا جوهريًا لتحقيق تقليل استهلاك الطاقة وتحسين استخدام الموارد من خلال تقليل عدد الخوادم العاملة. في هذا البحث ، تم اقتراح تحسين محسن لسرب الجسيمات المنفصل (EDPSO). يتم تحسين خوارزمية PSO المنفصلة من خلال تعديل معادلة تحديث السرعة لربط الجسيمات الناتجة وضمان الجدوى. علاوة على ذلك ، يتم مساعدة EDPSO من خلال خوارزميات إرشادية عشوائية أول ملائمة (RFF) وأفضل ملاءمة عشوائية (RBF) لإنتاج خوارزميات هيدريد تسمى RFF-EDPSO و RBF-EDPSO. يتم تقييم

*Email: seema.alsaidy@gmail.com

الخوارزميات المقترحة ومقارنتها مع الخوارزميات الحديثة لتقليل استهلاك الطاقة. تظهر نتائج المحاكاة الأداء الفعال لـ RFF–EDPSO و RBF–EDPSO في تقليل عدد خوادم التشغيل.

1. Introduction

Recently, cloud computing has gained the attention of many global enterprises and cloud users due to its significant impact on global IT technology. Cloud computing can broadly be defined as an emerging technology based on the internet where cloud users can access shareable computing resources. Those resources are provisioned to cloud users on-demand rules. Cloud computing has overcome the drawback of local computation and replaced it with remote computing. With this facility, cloud users do not need to become involved in what happens behind the scene to provide computing services.

Datacenter (DC) is where cloud physical resources, termed physical machines (PMs), are hosted and provide services. Due to the excessive use of the internet, new virtualization technology has been developed to be the essential concept of cloud computing functionality [1]. Virtualization is the process of splitting physical machines into multiple virtual computing units known as virtual machines (VMs). Each VM can provide a service at a time individually. As a result of virtualization, throughput, cost reduction and performance will be improved.

Demands on cloud computing resources, are increasing and dynamically fluctuating with cloud users' demands. Managing these resources between different cloud users became a critical issue. Therefore, it is necessary to develop an efficient resource management procedure to sustain both user and provider satisfaction. Efficient resource management can be accomplished by placing VMs on PMs effectively and optimally. Providing an efficient placement technique for mapping VMs on PMs has a significant impact on both cloud providers and users. From the user’s perspective, the service is improved. While from the provider's perspective, profits, energy, and labor productivity are enhanced [2].

Mapping virtual machines to physical machines are termed Virtual Machine Placement (VMP) problem. This optimization problem is considered an NP-hard problem where different VMs are placed on different PMs to satisfy a certain objective(s). One of the main objectives of an efficient VMP algorithm is to reduce data center power consumption [3]. In 2021, according to statistical analysis of several data centers worldwide, 2,670 data centers are located in the U.S., 452 in the U.K., Germany ranked third with 443, while China has 416 [4].

According to studies produced by the department of energy in the US, the power consumed by cloud data centers is about 1.5 from the global power consumption of the U.S. and this percentage has increased every year [5]. The high energy consumption of data centers results in environmental pollution through carbon emission (CO2) [6]. Servers and cooling systems are the main components that consume energy in data centers. The efficient VMP technique, can minimize the number of operating servers through optimal resource utilization, as result, reduce power consumption. In cloud computing, VMP techniques can be classified based on the complexity of the algorithm as a heuristic, meta-heuristic and hybrid virtual machine placement [7][8]. Examples of heuristic algorithms used for VMP are Best Fit (BF), Best Fit Decreasing (BFD), First Fit (FF), First Fit Decreasing (FFD), Next Fit (NF), and Any Fit (AF) [9]. Since VMP is an NP-hard optimization problem, a meta-heuristic is the optimal choice for such a combinatorial optimization problem. Such meta-heuristic algorithms include Genetic Algorithm (GA) [10], Particle Swarm Optimization (PSO) [11], Grey Wolf Optimization (GWO) [12][13], Slap Swarm Optimization (SSD) [14]. While hybridized

algorithms use a combination of two algorithms such algorithms are: genetic and tube search algorithms [5], genetic and BF algorithms [10].
The conventional PSO algorithm is commonly used in a continuous optimization problem with design vectors having real numbers. However, the most common optimization problems are discrete, and an example of such problems is VMP in the cloud environment.
In this paper, an enhanced discrete PSO (EDPSO) algorithm is proposed. Enhancing the discrete PSO algorithm is achieved through modifying the velocity update equation to bound the resultant particles and ensuring feasibility. Furthermore, EDPSO is assisted by two heuristic algorithms: random first fit (RFF) and random best fit (RBF). These are used to produce hydride algorithms termed RFF-EDPSO and RBF-EDPSO. The main contributions of this paper are:

▪Formulating the problem of heterogeneous virtual machine (VMs) placement and defining the mathematical modelling for optimal resource usage with minimum energy consumption.
▪Enhancing the discrete PSO to adopt such discrete optimization problem, by modifying the velocity and position update equations to validate the solution and reduce the penalty of replacing infeasible solutions.
▪Utilizing heuristic algorithms (RFF and RBF) to improve the initialization starting search points of PSO particles. In addition, the RFF and RBF are used to replace infeasible solutions during the update process.
▪Evaluating the performance of the two proposed algorithms (RFF-EDPSO and RBF-EDPSO) and comparing it with two recent and algorithms VMP. The evaluation is performed based on the total energy consumption of the operating servers.
The rest of this paper is organized as follows: In Section 2, the most relevant related works are presented. The mathematical modelling and formulation of the VMP problem are described in Section 3. The proposed algorithm (EDPSO) and its hybrid versions (RFF-EDPSO and RBF-EDPSO) are explained in Section 4. The simulation results of the proposed algorithms are discussed in Section 5. lastly, conclusions are presented in Section 6.

2. Related Work

The major issue that attracts many researchers' attention and concerns in cloud computing is data center energy consumption management [6]. Managing such critical issue is significantly affected by exploring the optimal VM placement. The VMP placement problem involves mapping a set of VMs to an available server called PMs subject to specific constraints. The aim behind efficient mapping is to optimize one or more objectives. Those optimization objectives measure the VMP technique effectiveness. Commonly, these optimization criteria are categorized into two demands based on cloud service: cloud users' demands and cloud service provider’s demands. The user’s demands are makespan, response time, cost. On the other hand, minimizing energy consumption, improving throughput and reliability are the provider’s demands [15].

Many researchers and companies attempt to solve the VMP problem using various methods and algorithms. Those algorithms can be classified as: heuristic, meta-heuristic, and hybrid algorithms. Ammar et al. [16] proposed a heuristic algorithm called balanced resource consumption and imbalance VM with minimum migration time (BRC-IBMMT) to balance resource consumption with minimum resource wastage and migration time. Mollamotalebi and Hajireza [17] proposed another heuristic algorithm to minimize energy and SLAV rates called Energy-reduction and SLAV-reduction dynamic run-time replacement (ESDR). In addition, Moges and Abebe [9] suggested a modified heuristic bin-packing algorithm called Medium-Fit (MF) by providing an efficient balance between energy and SLA violation.

Meta-heuristic algorithms have also been suggested to solve the VMP problem. For example, Soltanshahi et al. [18] proposed the Krill Herd algorithm to minimize power consumption and SLAV. Also, a multi-objective algorithm was presented by Satpathy et al. [19], to optimize data center power consumption and resource wastage called crow search VM placement (CSAVMP). On the other hand, some researchers used a hybridization between meta-heuristic algorithms. Abohamama and Hamouda [10] proposed a hybrid genetic and BF algorithm (IGA- POP) to optimize usage of power and resource wastage. While Zhao et al. [5] employed a tabu search algorithm as a mutation operator to improve the exploitation search-ability of GA and produced a hybrid algorithm called (GATA).

Particle swarm optimization (PSO) is one of the common and widely used nature-inspired meta-heuristic algorithms. Simplicity, accuracy, and speed are what makes PSO preferable for researchers. PSO was initially designed for continuous optimizing problems. However, in problems with discrete domains, PSO needs to be modified to mimic such discrete optimization problems. Examples of discrete problems in cloud computing are task scheduling and virtual machine placement. There are many techniques used to convert PSO from continuous to discrete. Rezaee and Jasni [20] presented eight procedures for handling discrete problems in PSO such as rounding-off, trinary PSO, and penalty approach. Tasgetiren et al. [21] used another approach for transforming continuous value to discrete using the small positive value (SPV) rule. This procedure depends on three operations: sorting the continuous values, replacing the vector with its ranking values, and binding the solution using mod operation.

Wu and Tsai [22] suggested an approach called local optimal list instead of using self-influence and adding a round-off procedure for PSO modification. Furthermore, Kashan and Karimi [23] used different operators for generating a new position and velocity. The three operators were: subtract, multiply, and add operators. Miao et al. [24] presented a new PSO-based static load balancing algorithm called adaptive Pbest discrete PSO (APDPSO) algorithm. The proposed algorithm is used to balance the load for the satisfactory performance of large-scale parallel and distributed simulations. Ibrahim et al. [11] proposed a Power-Aware technique using a discrete version of PSO named PAPSO to determine the near-optimal placement for migrated VMs. Different than the above-mentioned algorithms, Chitra et al. [25] offered a discrete PSO called Jumping PSO (JPSO) to update the position vector directly without velocity calculation. The particle is updated by jumping from its current position to a new one according to the influence of the personal and social experience.

3. VMP Problem Formulation

The VMP problem in cloud computing can be defined as the assignment or mapping of virtual machines (VMs) on servers or physical machines (PMs). This problem is considered a multi-dimensional bin packing problem where the bins represent the servers and the items are the VMs. The dimensions of VMs represent the requirements of resources, such as (CPU, Memory, Bandwidth, etc.) that need to be provided by the bins (PMs). The optimal mapping should meet all VMs' demands (resources requirements) and minimize the number of active servers to reduce the energy consumption of the data center. This problem can be modeled as follows:

1. The set of virtual machines that need to be mapped is defined by $VM = \{vm_1, vm_2, \dots, vm_{N_{vm}}\}$, where N_{vm} is the total number of virtual machines. Each VM (vm_i) in **VM** has D-dimensional of resources such that $vm_i = \{rv_i^1, rv_i^2, \dots., rv_i^D\}$, where rv_i^d represents the amount of resource type-d required by the i-th VM. The number of resource dimensions of each VM is D.
2. The set of physical machines required for VMs deployment is $PM =$

$\{pm_1, pm_2, \ldots, pm_{N_{pm}}\}$, where $N_{pm}$is the total number of physical machines. Each PM (pm_j) in **PM** can provide the same number of resources (D-dimensional of resources) such that $pm_j = \{rp_j^1, rp_j^2, \ldots, rp_j^D\}$. rp_j^d represent the available amount of resource type-*d* supplied by the j-th PM.

3. Given the sets of VMs and PMs, the objective is to find an optimal placement or mapping set M from VM to PM such that $M = \{m_1, m_2, \ldots, m_{N_{vm}}\}$ where $1 \leq m_i \leq N_{pm}$.
4. The mapping set M, should **s**atisfy the following constraints:

- A virtual machine (vm_i**)** should be placed only on one server (pm_j) at a time, i.e.:

$$\sum_{j=1}^{N_{pm}} x_{ij} = 1 \qquad \text{where } x_{ij} = \begin{cases} 1 & \boldsymbol{vm_i} \text{ is assigned to } \boldsymbol{pm_j} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

- For a given resource of type (*d*) in sever (pm_j), the total corresponding resource demands of the assigned VMs to that server should be less than or equal to the resource capacity. i.e.:

$$\sum_{i=1}^{N_{vm}} x_{ij} rv_i^d \leq rp_j^d \qquad \begin{matrix} \forall\, d \in \{1,2,\ldots,D\} \\ \forall\, j \in \{1,2,\ldots,N_{pm}\} \end{matrix} \tag{2}$$

The optimal placement can be achieved based on one or more of the objectives: minimizing power consumption, reducing resource wastage, and/or maximizing resource usage balance.

3.1. Power Consumption Modeling

Globally, cloud data centers consume a massive amount of energy. According to the United States data center energy usage report, data centers consumed an estimated 70 billion kWh representing about 1.8% of total U.S. electricity consumption. In 2020 and based on historical trends, it is estimated that the U.S. data centers are projected to consume approximately 73 billion kWh [26]. Servers are the major entities of energy consumption in data centers. All server resources (CPU, memory, disk, bandwidth, etc.) contribute to energy consumption, however, CPU resource is considered the main contributor [27]. The power consumption is directly proportional to CPU utilization and can be modeled as a linear relationship [5]. In general, servers can have one of three energy consumption modes: sleep, busy, and idle. In sleep mode, the server is shut down with zero energy consumption. In this mode, the server has no assigned VMs to serve. When the server is on with a fully utilized CPU, the server enters the busy mode and has the maximum energy consumption E^max. When the server is on with a CPU utilization approaching zero, the server is in the idle mode consuming an energy of (E^min) equal to 60-70% from E^max [15]. The total energy consumption E_T of the data center is given by:

$$E_T = \sum_{j=1}^{N_{pm}} y_j E_j \tag{3}$$

where,

$$y_j = \begin{cases} 1 & \text{Server j is ON} \\ 0 & \text{Server j is OFF} \end{cases}, \tag{4}$$

$$E_j = E_j^{min} + \left(E_j^{max} - E_j^{min}\right) U_j^{d_{cpu}}, \tag{5}$$

$$U_j^{d_{cpu}} = \frac{\sum_{i=1}^{N_{vm}} x_{ij}\, rv_i^{d_{cpu}}}{rp_j^{d_{cpu}}} \tag{6}$$

In equation (4), y_j is a binary variable indicating whether server j is powered on or not. E_j

is the energy consumed by the j-th server. $U_j^{d_{cpu}}$ is the CPU utilization of server j, which is equal to the summation of all VMs CPU utilization assigned to server j. Equation (6) can be generalized to calculate the utilization of any resource in server j (U_j^d) by setting $d_{cpu} = d$ and the set of server utilization from D resources is defined by $U_j = \{U_j^1, U_j^2, \dots\dots, U_j^D\}$.

3.2. Resource Wastage Modeling

An optimal mapping is significantly affected by the efficient resource utilization of servers. Wastage is one of the metrics used to evaluate the effectiveness of a given mapping. It determines the total amount of the unused resources of a server in all dimensions. Maximizing resource utilization in a balanced manner has an effective effect in reducing the wastage of resources. Thus, minimizing the number of active servers which in turn reduces data center energy consumption. The D-dimensional wastage of server j (W_j) can be calculated as follows:

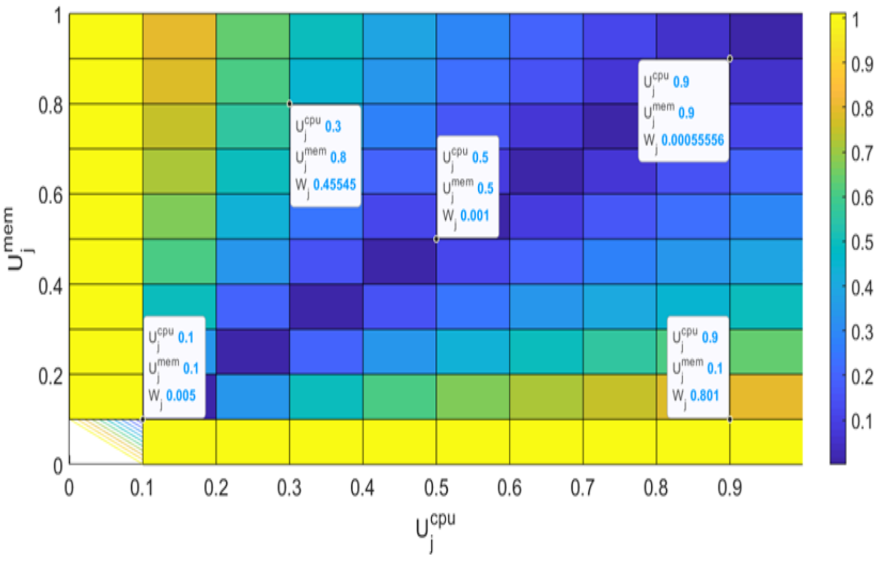
$$W_j = \frac{U_{max} - U_{min} + \epsilon}{\sum_{d=1}^{D} U_j^d} = \frac{L_{max} - L_{min} + \epsilon}{\sum_{d=1}^{D} U_j^d} \quad (7)$$

where

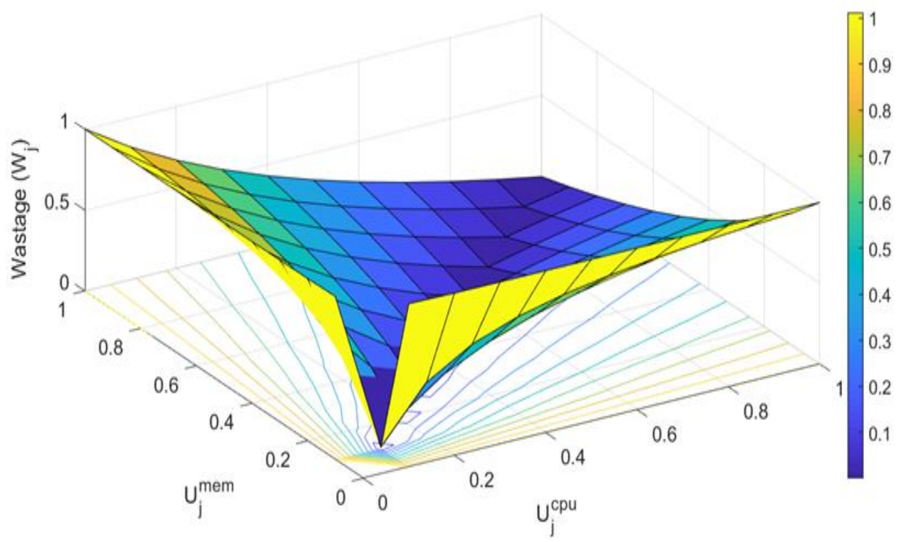
$$U_{max} = \max\{U_j^1, U_j^2, \dots\dots, U_j^D\}, \quad (8)$$
$$U_{min} = \min\{U_j^1, U_j^2, \dots\dots, U_j^D\} \quad (9)$$
$$L_{max} = 1 - U_{min}, \quad (10)$$
$$L_{min} = 1 - U_{max} \quad (11)$$

In equation (7), the numerator represents the balance usage among all resources, which is equivalent to the difference between the maximum and minimum utilizations. While in the denominator, the sum of all resource utilizations is used to reflect the total amount of resource usage. The larger the sum of resource utilizations, the lower the wastage. The epsilon parameter (ϵ) is a small positive real number added to distinguish which server wastage is greater when all resources utilizations are equal and hence, balanced in usage. L is the complement utilization of U.

To illustrate the behavior of the wastage function concerning different resource utilizations a two-dimensional resource server case is used. Figure 1 depicts the graph of equation (7) setting D=2 and ϵ=0.001.



a) Top view



b) 3-D view

Figure -1 This figure depicts 2-D wastage function graph with two views: a) Top view, b) 3-D view.

The server j is considered to have two resources utilizations CPU (U_j^{cpu}) and memory (U_j^{mem}). Both utilizations are varied within the range [0-1]. As can be seen from the wastage plot, the graph is symmetric about the line $U_j^{cpu} = U_j^{mem}$. On the line of symmetry, the wastage decreases as both utilizations increase evenly. This is due to the increase of the denominator term in equation (7) with a fixed constant numerator equal to ϵ. As the utilization points (U_j^{cpu}, U_j^{mem}) roll away from symmetry line, the wastage of the resources increases approaching maximum wastage at the boundaries. This increase in wastage is due to the unbalance resource utilizations represented by the numerator term ($U_{max} - U_{min}$). The minimum wastage value can be achieved when both resources are fully utilized and this value equals to $W_j^{min} = \frac{\epsilon}{D} = 0.0005$.

Figure 2 shows different cases of VMs allocation on one PM. In case 1, the server contains only two VMs, the residual area of resources represented by the yellow rectangle, shows that there is a large unused amount of CPU and memory resources. The residual resources are almost equal due to balance placement. Although there is a wastage in both resources, it is still possible to add more VMs. However, in case 2, the memory resource is fully utilized while only about 50% of CPU resource is utilized. In this case, the wastage is worse as there is no viable way to allocate any VM in the future. In case 3, a near-optimal mapping is achieved through satisfying both balance and full resources occupation resulting in a small wastage.

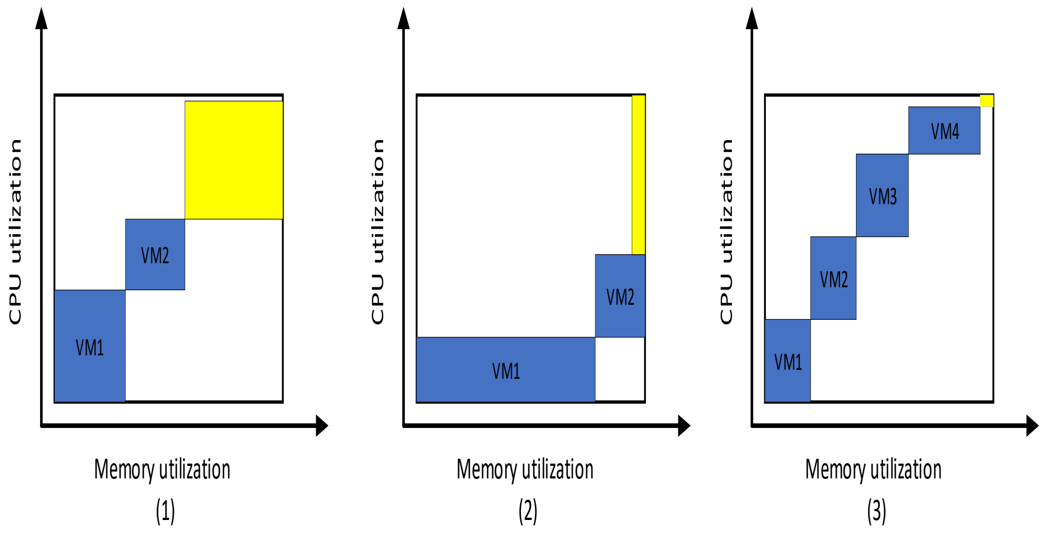


Figure 2: This figure depicts wastage in different VMP cases.

3.3. Multi-resource Balance Modeling

The other metric used for VMP is to achieve a balance resource usage. Balance resource utilization can highly influence wastage reduction and hence, minimizing energy consumption.

In data centers, servers include milt-dimensional resources such as (CPU, memory, storage, bandwidth, etc.). The diversity of resources requirements in VMs needed for allocation on servers, result in unbalance of server resources utilizations. Balancing multi-resources utilizations is one of the major challenges for cloud providers. Efficient balance mapping takes into consideration avoidance of fragmentation problem between servers' resources. Fragmentation is due to the intensive usage of one resource that prevents other VM allocation even if other resources are underloaded. If fragments among resources are minimized, maximum resources balance can be obtained. A three-dimensional balance index has been introduced in [28] to quantify the equilibrium state of multi-resources usage. The balance index of server j (B_j) can be generalized to D-dimensional as:

$$B_j = D\left(B_j^U + B_j^L\right)/2 \tag{12}$$

Where

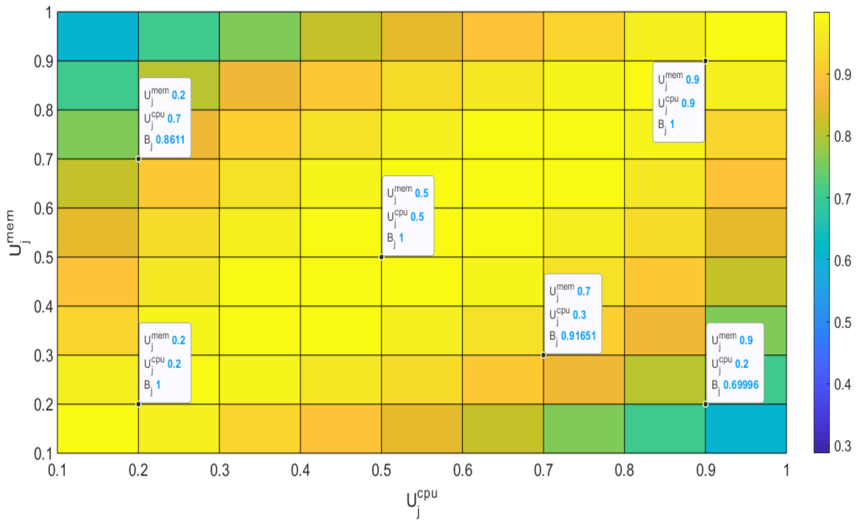
$$B_j^U = \frac{\sqrt[D]{\prod_{d=1}^{D} U_j^d}}{\epsilon + \sum_{d=1}^{D} U_j^d}, \tag{13}$$

$$B_j^L = \frac{\sqrt[D]{\prod_{d=1}^{D} L_j^d}}{\epsilon + \sum_{d=1}^{D} L_j^d}, \tag{14}$$

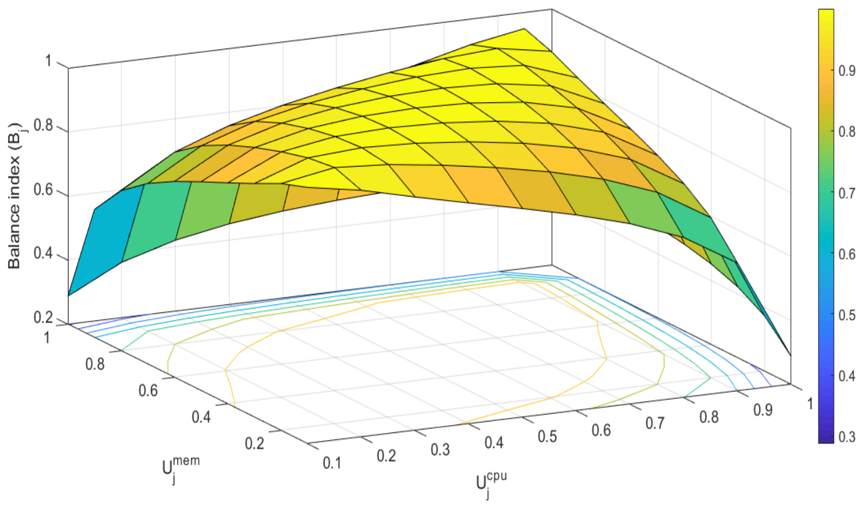
$$L_j^d = 1 - U_j^d \tag{15}$$

B_j is affected by the balance of both utilized and non-utilized resources $B_j^U$and B_j^L respectively. The parameter ϵ is added to avoid division by zero at 0% (empty server) and 100% (All fully utilized servers). L_j^d is the complement utilization of resource d in server j. In the perfect balance case, ${U_j}^1 = {U_j}^2 = {U_j}^d = \cdots = {U_j}^D$ and ${L_j}^1 = {L_j}^2 = {L_j}^d = \cdots = {L_j}^D$ in this case $B_j^u = B_j^L = 1/D$ as $\epsilon \to 0$.

To illustrate the behavior of the balance index as a function of the D-dimensional utilizations, a server with a 2-dimensional resource will be considered. Figure 3 depicts the graph of the balance index B_j by setting D=2 and ϵ=0.001. The server j is considered to have two resources utilization: CPU (U_j^{cpu}) and memory (U_j^{mem}). Both utilizations are varied within the range [0-1].



a) Top view



b) 3-D view

Figure 3: This figure depicts the graph of a 2-D Balance index function with two views: a) Top view, b) 3-D view.

To show the advantage of using balance index we have the flowing two mapping cases (a) and (b) shown in Figure 4.

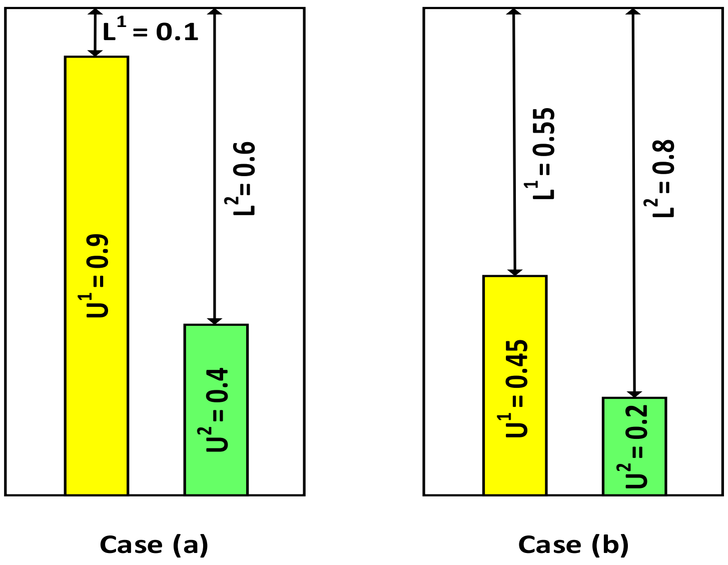


Figure 4: This figure depicts balance index with two mapping cases: case (a), case (b).

In Figure 4, case (a) has $U^1 = 0.9$, $U^2 = 0.4$, $L^1 = 0.1$, and $L^2 = 0.6$ and case (b) has $U^1 = 0.45$, $U^2 = 0.2$, $L^1 = 0.55$, and $L^2 = 0.8$. According to equation (7), the wastage index of the case (a) is comparable to that of case (b) ($W_a = 0.385$ and $W_b = 0.386$). However, according to equation (12), the balance index of the case (b) is greater than that of case (a) ($B_a = 0.811$ and $B_b = 0.953$). This confirms our conception that the mapping of case (b) is better than that of the case (a).

4. Enhanced Discrete PSO algorithm

Particle Swarm Optimization (PSO) is one of the most widely used Swarm Intelligence and population-based techniques. PSO can be employed in solving hard_NP problems. This is due to its adaptation capabilities in large search spaces. PSO is an evolutionary technique that simulates the social behavior of species having specific activity movement in a swarm. This type of behavior is found in shoals of fish and flock of birds [29].

The mathematical modeling of the PSO algorithm is based on the interaction between members in a set of the population called a swarm. Those members are called particles or agents. The movement of each particle in the swarm is influenced by two guiders derived from self and global history search experience. Each particle represents a solution to the optimization problem. The goal of particles is to find an optimal solution within a large search space. Finding an optimal solution process is based on the interaction and cooperation between particles. Particles can share information about their current situation to update their new position. The update equation of the new particle position is given by:

$$P_k^{t+1} = P_k^t + V_k^{t+1} \tag{16}$$

Where P_k^{t+1} is the updated position of the k-th particle in the swarm at iteration $(t+1)$. Accordingly, the new velocity of the updated particle position (V_k^{t+1}) is defined by:

$$V_k^{t+1} = \omega V_k^t + c_1\, r_1\left(P_k^{best} - P_k^t\right) + c_2\, r_2(P^{gbest} - P_k^t) \tag{17}$$

The updated velocity V_k^{t+1} is affected by three components termed inertia, personal influence, and social influence. ω is an inertia factor used to control or balance the exploring

and exploiting of the PSO algorithm. P_k^{best} is the best position that k-th particle reached so far. The last term represents the component that has a global effect on all particles termed as the social influence component. The best global position found between all best positions particles is termed as P^{gbest} . c_1 and c_2 are acceleration constants, r_1 and r_2 are random numbers between (0 and 1).
In the virtual machine placement (VMP) problem, the PSO parameters are defined as follows: $S = \{P_1, P_1, \ldots., P_{N_s}\}$, where P_k is the k-th particle in the swarm and represents a solution. N_s is the total number of particles. The structure of each particle is represented by $P_k = \{p_k^1, p_k^2, \ldots, p_k^{N_{vm}}\}$ of size $(1 \times N_{vm})$, where p_k^i is a server index holding the i-th VM and can have an integer value within the interval $[1, N_{pm}]$. One of the optimizations goals or objectives is to minimize the maximum value inside P_k, which represent the total number of powered servers.

Similar to the swarm matrix, the velocity is represented as $V = \{V_1, V_2, \ldots, V_{N_s}\}$, and each particle velocity is given by $V_k = \{v_k^1, v_k^2, \ldots, v_k^{N_{vm}}\}$ of size $(1 \times N_{vm})$, where v_k^i is corresponding velocity value of p_k^i. The general PSO algorithm is configured to optimize problems with continuous real number solutions. However, in cloud computing, virtual machine placement (VMP) is a discrete optimization problem. Therefore, a modification of the original PSO is required to solve such a problem.

Many researchers exert potential efforts to adjust the original PSO for different discrete optimization problems. For example, in the application of cloud computing, a binary PSO is adopted for task scheduling [15]. Another approach for converting PSO to discrete using truncated to the nearest integer [20]. In scheduling parallel machines, [23] use different operators, for example: *Subtract, multiply,* and *add operators* to convert regular PSO to discrete one.

Figure 5 shows an example of a VMP solution (a swarm particle) with $N_{vm} = N_{pm} = 7$. In this example, the upper row represents the VM index number and the lower row represents the PM index number.

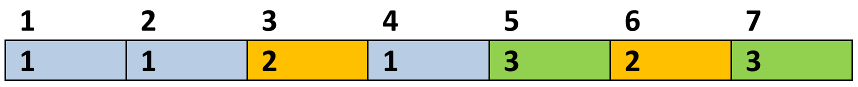


Figure 5: This figure depicts the particle structure in a swarm.

Thus, for example vm_1, vm_2, and vm_4 are mapped to pm_1while vm_3, and vm_6 are mapped to pm_2. Here, the capacities of the server resources constraints are taken into consideration in mapping. When updating this particle using PSO, the solution stripe may become invalid having a PM index number greater than N_{pm}. If such a case occurs as shown in Figure 6, where vm_7 is assigned to pm_8, the new stripe becomes invalid (infeasible solution).

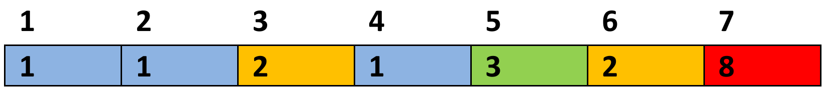


Figure 6: This figure depicts the invalid particle.

The proposed enhanced discrete PSO, solves this type of an invalid situation by using mod operation in the original PSO velocity equation to become:

$$V_k^{t+1} = V_k^t + b1_k\left(P_k^{best} - P_k^t\right) + b2_k(P^{gbest} - P_k^t) \tag{18}$$

Equation (18) can be detailed in vector form as:

$$\begin{bmatrix} v_{k,1}^{t+1} \\ v_{k,2}^{t+1} \\ \vdots \\ v_{k,Nvm}^{t+1} \end{bmatrix} = \begin{bmatrix} v_{k,1}^{t} \\ v_{k,2}^{t} \\ \vdots \\ v_{k,Nvm}^{t} \end{bmatrix} + \begin{bmatrix} b1_{k,1}^{t} \\ b1_{k,2}^{t} \\ \vdots \\ b1_{k,Nvm}^{t} \end{bmatrix} \begin{bmatrix} p_{k,1}^{pbest} & & p_{k,1}^{t} \\ p_{k,2}^{pbest} & - & p_{k,2}^{t} \\ \vdots & & \vdots \\ p_{k,Nvm}^{pbest} & & p_{k,Nvm}^{t} \end{bmatrix} + \begin{bmatrix} b2_{k,1}^{t} \\ b2_{k,2}^{t} \\ \vdots \\ b2_{k,Nvm}^{t} \end{bmatrix} \begin{bmatrix} p_{1}^{gbest} & & p_{k,1}^{t} \\ p_{2}^{pbest} & - & p_{k,2}^{t} \\ \vdots & & \vdots \\ p_{Nvm}^{gbest} & & p_{k,Nvm}^{t} \end{bmatrix} \tag{19}$$

After modifying the velocity, it must be validated using the mode operation as follows:

$$v_{k,i}^{*(t+1)} = \begin{cases} \mathrm{mod}\left(v_{k,i}^{t+1}, N_{pm}\right) & if\ v_{k,i}^{t+1} \neq nN_{pm}\ , n = \{\mp 1, \mp 2, \mp 3, \ldots\ldots\} \\ N_{pm} & otherwise \end{cases} \tag{20}$$

Equation (20) shows the use of mod operation of the generated velocity. It will ensure that the new velocity index $v_{k,i}^{(t+1)}$ values fall within the range $1 \leq v_{k,1} \leq N_{pm}$. This operation is equivalent to bounding the velocity values to $v_{min} = 1$ and $v_{max} = N_{pm}$.
Similarly, for the position update, the modified velocity obtained from equation (20) is used as follows:

$$P_{k,i}^{(t+1)} = P_{k,i}^{(t)} + V_{k,i}^{*(t+1)} \tag{21}$$

After modifying the position, it must be validated using the mode operation as follows:

$$p_{k,i}^{*(t+1)} = \begin{cases} \mathrm{mod}\left(p_{k,i}^{t+1}, N_{pm}\right) & if\ p_{k,i}^{t+1} \neq nN_{pm}\ , n = \{1, 2, 3, \ldots\} \\ N_{pm} & otherwise \end{cases} \tag{22}$$

Equation (22) can be written in another form to validate the position as follows:

$$p_{k,i}^{*(t+1)} = \lfloor \frac{p_{k,i}^{(t)}}{N_{pm} + 1} \rfloor + mod\left(p_{k,i}^{(t)}, N_{pm} + 1\right) \tag{23}$$

A feasible solution should also satisfy the resource demands of the assigned VMs according to equation (2). At this point, the solution is checked for feasibility. If it's not feasible (resource demands are greater than the available server recourse capacity), the solution must be replaced with a feasible mapping. Several approaches can be used for infeasible solution replacement. One approach is to use random mapping, which also requires a feasibility test for the newly generated particle. This approach suffers from time consumption affecting the speed of the algorithm convergence. Another approach is to use random permutation between 1 to N_{pm}. This will guarantee the generation of a feasible solution; however, it will utilize all servers and hence, affect the power consumption.

A more efficient approach for infeasible particle replacement is to use a heuristic algorithm with random sorting. Two heuristic algorithms with random sorting are used: Random First Fit (RFF), and Random Best Fit (RBF). Furthermore, RFF and RBF are hybridized with EDPSO to improve the initialization through good starting points instead of random mapping. Utilizing heuristic algorithms in the PSO initialization phase shows a significant improvement on the algorithm performance and convergence [15]. In regular BF and FF, all particles (solutions) have the same starting point. While, with RFF and RBF, a diversity of starting points population can be produced improving the exploring phase of the algorithm.

The pseudo-codes of RFF-EDPSO and RBF-EDPSO are presented in algorithm 1. In algorithm 1, the inputs are the sets of VMs and PMs that need to be mapped as in lines 2 and 3. The output of the algorithm will hold the optimal VMP vector designated by (M^***)** in line5. In line7, the parameters of the proposed PSO are initialized (number of particles and iterations). In lines 8 and 9, the initial position of particles is initiated using a heuristic algorithm (RFF or RBF) instead of randomness. The Fitness of each particle is calculated and saved to be compared with the best solution found so far corresponding to the current position in lines 12 and 13. It is worth noting that the fitness calculation depends on the objective function being Energy, wastage, and/or balance index as given by equations (3), (7), and/or (12) respectively.

Algorithm 1: Pseudo-code of RFF-EDPSO and RBF-EDPSO algorithms

```
1   Input:
2   Set of VMs (VM),
3   Set of PMs (PM),

4   Output:
5   Optimal allocation (mapping) vector ($M^*$) (VM to PM) of size 1 × Nvm

6   Start:
7   Initialize EDPSO parameters
8   Initialize particles according to RFF algorithm for RFF-EDPSO, or
9   Initialize particles according to RBF algorithm for RBF-EDPSO.

10  For $i_t$=1 to no. of iterations
11  For k=1 to no. of particles
12  Calculate fitness value for each particle using equation (Energy)
13  Find the best particle so far ($P_k^{best}$).
14  Update velocity and position for the current particle according to equations (x and x).
15  Test if the updated new particle is a feasible solution
16  If (not feasible)
17        generate new solution using RFF or RBF
18  End
19   End
20  Find the global best particle among all best particles ($P^{gbest}$).
21  End
22  Set the Optimal allocation vector ($M^*$) to the global best particle ($M^* = P^{gbest}$)
23
```

In line 14, the velocity and position are updated according to equations (20) and (22). In line 16, 17 feasibility checking are performed after velocity and position updating according to equations (20) and (22). If the particle is infeasible, a new solution is generated using algorithms RFF or RBF in line 18. Finally, in line 21 the optimal global best solution is found as the best mapping between all other best solutions to be the optimal allocation vector (M^*) in line 23.

5. Simulation Results and Discussion

The implementation of all experiments used to evaluate the proposed algorithm was performed using MATLAB software installed on a PC with Intel(R) Core (TM) i5-10210U CPU @ 1.60GHz, RAM 12.0 GB. The operating system was Windows 10 Pro. The performance of the proposed algorithm (EDPSO) was evaluated and compared with other related bin packing and swarm-based algorithms. A synthesis VMs configuration was chosen for simulation and based on Amazon EC2, which is one of the most common cloud service providers. The CPU and memory requirements of each VM were generated randomly from the following set: [1, 2, 4, 6, 8, 16, 32, 48, 64, 96, 128]. All servers were assumed to be homogeneous having equal CPU and memory capacities of size 128. The worst-case scenario, at which $N_{pm} = N_{vm}$ was considered in all simulation experiments.

Figure 7 shows the simulation method structure of the algorithms chosen for comparison that were classified into heuristic (BF, BFD, FF, FFD) and meta-heuristic algorithms (RFF-EDPSO, RBF-EDPSO, PAPSO, IGA-POP).

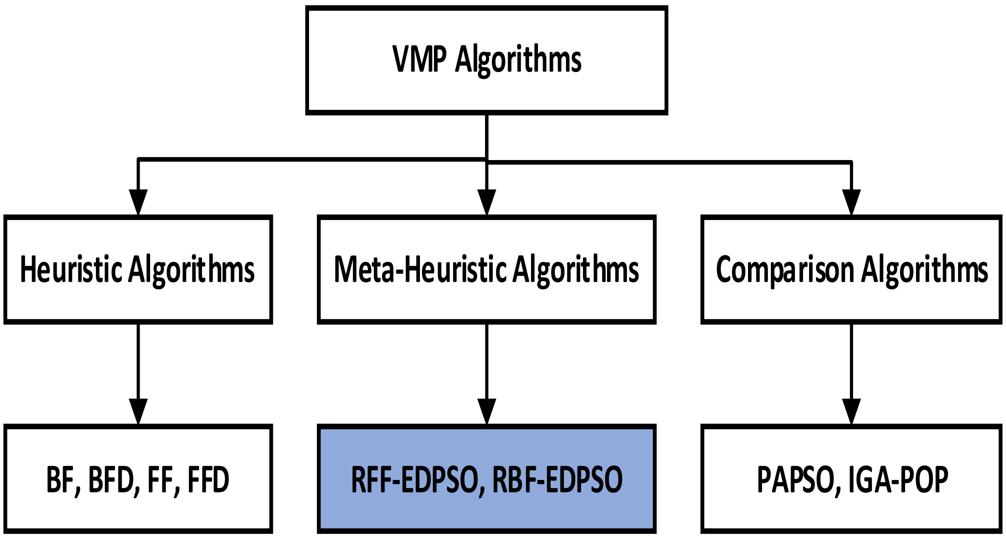


Figure 7 :This figure depicts VMP Algorithms Simulation Structure.

The PSO algorithm parameters are defined in Table 1.

Table 1: PSO parameters

Parameter	Value
Number of Iterations	100
Population Size	50
Velocity and Position bounds	[0, N_{pm}]

The performance of the two proposed algorithms (RFF-EDPSO and RBF-EDPSO) is based on the energy consumption metric (E) defined by equation (3). The energy consumption of the servers at the busy and idle modes were set to 117 and 86 respectively and it's linearly related to CPU utilization. In this section two simulation scenarios were performed to evaluate the proposed algorithms as follows:

<u>Scenario 1:</u> In this scenario, the two proposed algorithms were compared to heuristic algorithms (BF, BFD, FF, FFD). The comparison was conducted using three different cloud environments termed light-load (LL), medium-load (ML), and heavy-load (HL). In these environments (LL, ML, and HL), the number of VMs were varied in ranges; [20-100] with step 20, [200-400] with step 50, and [500-900] with step 100, respectively. Figures 8, 9, and 10 show the bar chart performance comparison between the algorithms in terms of energy consumption in LL, ML, and HL environments, respectively. Also, Tables 2, 3, and 4 depict the corresponding numerical value for Figures 8, 9, and 10. As can be seen from Figure 8, as the number of VMs increase more energy is consumed by all algorithms.

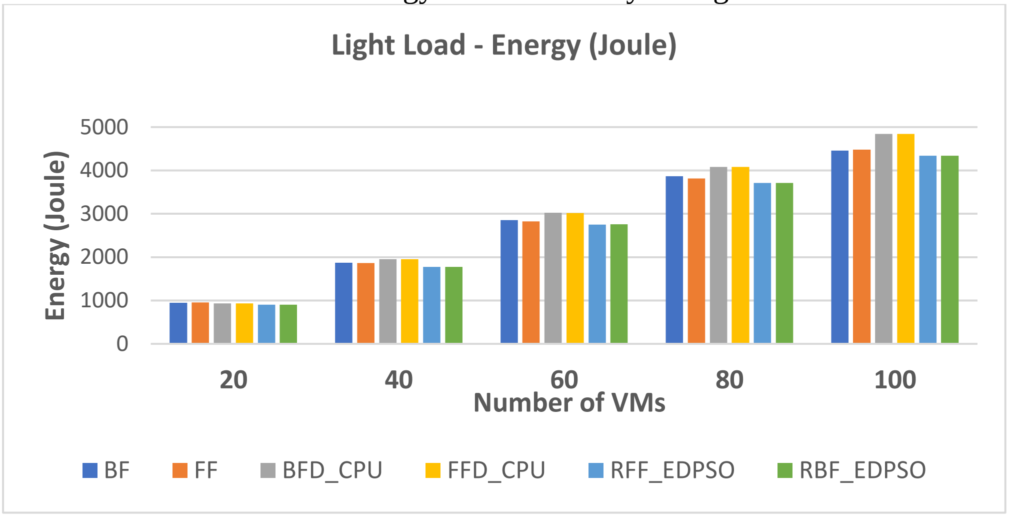


Figure 8: This figure depicts the energy consumption comparison between VMP algorithms using LL in scenario 1.

The two proposed algorithms outperform the other algorithms through consuming the least energy. The proposed algorithms have the same energy consumption results in all light load cases except in case 60, where RFF-EDPSO gains a better result than the others. Also, BFD-CPU and FFD-CPU have the same result and represent the two worst algorithms in energy consumption.

Table 2: Energy consumption performance comparison in LL (scenario 1)

Metric	Algorithms	Number of VMs				
		20	**40**	**60**	**80**	**100**
Energy Consumption (Joule)	BF	948.058	1872.202	2852.856	3867.869	4460.827
	FF	956.658	1863.602	2827.056	3816.269	4478.027
	BFD-CPU	930.858	1949.602	3024.856	4082.869	4839.227
	FFD-CPU	930.858	1949.602	3016.256	4082.869	4839.227
	RFF-EDPSO	**905.058**	**1777.602**	**2749.656**	**3713.069**	**4340.427**
	RBF-EDPSO	**905.058**	**1777.602**	2758.256	**3713.069**	**4340.427**

For the ML environment, as shown in Figure 9, RBF-EDPSO has the best performance among all other algorithms by achieving minimum energy consumption. RFF-EDPSO obtained a comparable result to RBF-EDPSO and ranked as second best algorithm. While the FFD-CPU algorithm has the highest energy consumption values at all medium load cases and hence ranked last.

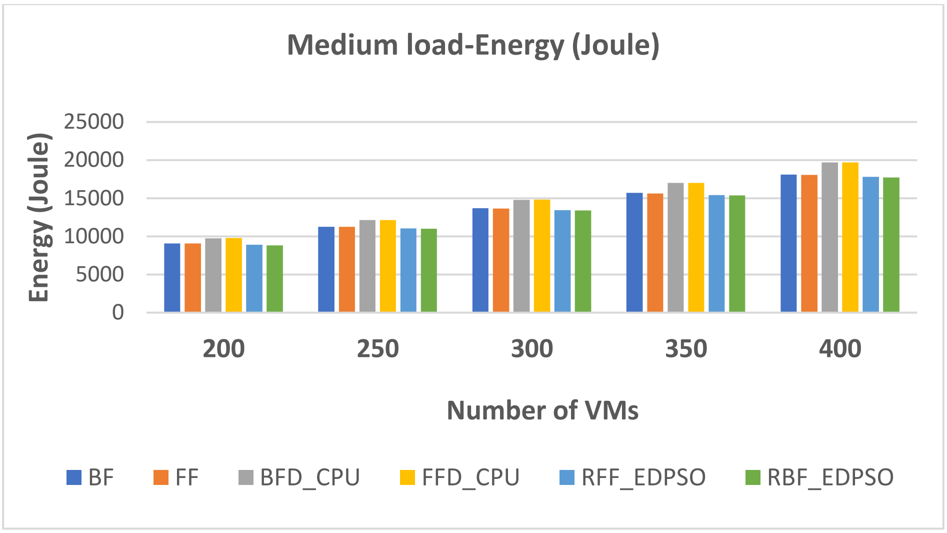


Figure 9: This figure depicts energy consumption comparison between VMP algorithms using ML in scenario 1

Table 3: Energy consumption performance comparison in ML (scenario 1)

Metric	Algorithm	Number of VMs				
		200	**250**	**300**	**350**	**400**
Energy Consumption (Joule)	BF	9073.145	11254.390	13694.288	15713.981	18086.859
	FF	9055.945	11237.190	13659.888	15645.181	18052.459
	BFD-CPU	9761.145	12122.990	14803.688	16995.381	19695.059
	FFD-CPU	9769.745	12122.990	14812.288	17012.581	19695.059
	RFF-EDPSO	8909.745	11039.390	13427.688	15430.181	17820.259
	RBF-EDPSO	8840.945	10987.790	13393.288	15361.381	17734.259

The performance comparison using heavy load environments is shown in Figure 10.

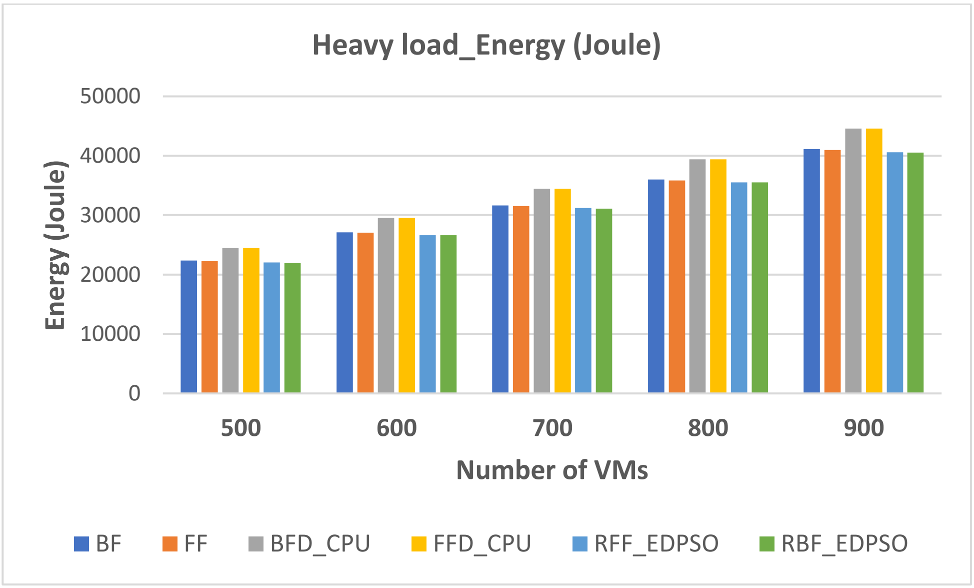


Figure 10: This figure depicts energy consumption comparison between VMP algorithms using HL in scenario 1

It can be observed from Figure 10, that RBF-EDPSO also consumes the least amount of energy in all cases. Similarly, in medium load case, RFF-EDPSO ranked the second-best performance among the algorithms. While the worst values of energy consumption were obtained by the FFD-CPU algorithm.

Table 4: Energy consumption performance comparison in HL (scenario 1)

Metric	Algorithm	Number of VMs				
		500	600	700	800	900
Energy Consumption (Joule)	BF	22379.430	27112.341	31649.539	36031.193	41125.143
	FF	22250.430	27034.941	31546.339	35850.593	40987.543
	BFD-CPU	24452.030	29554.741	34427.339	39393.793	44599.543
	FFD-CPU	24452.030	29563.341	34435.939	39419.593	44608.143
	RFF-EDPSO	22026.830	26647.941	31185.139	35549.593	40591.943
	RBF-EDPSO	**21932.230**	**26604.941**	**31124.939**	**35497.993**	**40531.743**

Scenario 2: In this scenario, the two proposed methods were compared with recent meta-heuristic algorithms (IGA-POP [10] and PAPSO [11]). To attain a fair comparison between the algorithms, simulation experiments were conducted using the same environment and parameters. Here, three different numbers of VMs from several types of environment loads were taken, they were: **60**, **300**, **700**. Similar to scenario 1, the total energy consumption is considered for comparison as an objective to optimize the VMP process. Figure 11 shows the comparison of energy consumption between algorithms.

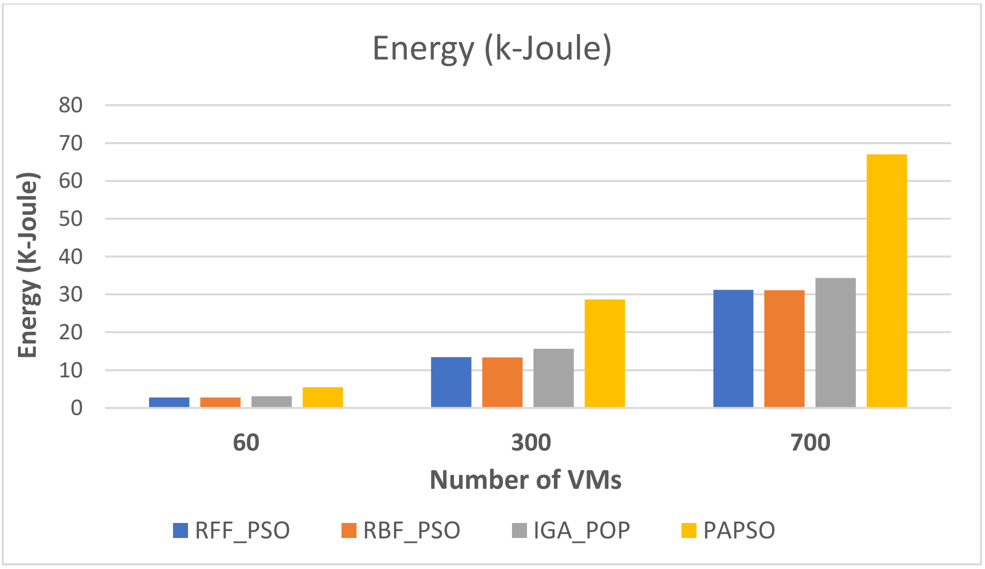


Figure 11: This figure depicts the energy consumption comparison between VMP algorithms in scenario 2.

As can be seen from Figure 11, the RFF-EDPSO and RBF-EDPSO achieved optimum energy consumption values. In case 60, RFF-EDPSO achieves the least energy consumption values among the other algorithm while, in cases 300 and 700, RBF-EDPSO has the minimum energy consumption values. The proposed algorithms overcome the other comparative algorithms in all cases. Table 5, summarize the numerical comparison results.

Table 5: Energy consumption performance comparison in (Scenario 2)

Metric	**Algorithms**	**Number of VMs**		
		60	**300**	**700**
Energy Consumption (k-Joule)	RFF-EDPSO	**2749.656**	13427.688	31185.139
	RBF-EDPSO	2758.256	**13393.288**	**31124.939**
	IGA_POP	3076.456	15629.288	34375.739
	PAPSO	5510.256	28692.688	66995.539

6. Conclusion

In this paper, Enhanced Discrete PSO algorithms (EDPSO) are proposed with heuristic assistance for the virtual machine placement problem (VMP). The proposed EDPSO algorithms are hybridized with two heuristic algorithms: random first fit (RFF) and random best fit (RBF) and termed as RFF-EDPSO and RBF-EDPSO. Enhancing the discrete PSO algorithm is achieved through modifying the velocity update equation to bound the resultant particles and ensuring feasibility.

Furthermore, EDPSO is assisted by heuristic algorithms for two benefits. First, initializing the population swarm using RFF and RBF, which can significantly improve the convergence speed and performance of the algorithm. Second, reducing the effect of infeasible solutions replacement during the particles update phase. Compared to recent literature algorithms, simulation results demonstrate the effective performance of the proposed RFF-EDPSO and RBF-EDPSO algorithms in terms of energy consumption.

References

[1] W. Zhang, X. Chen, J. Jiang, “A multi-objective optimization method of initial virtual machine fault-tolerant placement for star topological data centers of cloud systems”, *Tsinghua Science Technology* , Vol. 26 ,no. 2021, pp. 95–111,2021.

[2] C. Wei, Z.H. Hu, Y.G. Wang, “Exact algorithms for energy-efficient virtual machine placement in data centers”, *Future Generation Computer Systems*, Vol.106, no.3, pp. 77–91,2020.

[3] C. Jin, X. Bai, C. Yang, W. Mao, X. Xu, “A review of power consumption models of servers in data centers”, *Appl. Energy.* , Vol.265, no.2, pp.114806-114810,2020.

[4] Walmart stores worldwide, by country 2021 | Statista, Thomas Alsop. 2021.

[5] D.M. Zhao, J.T. Zhou, K. Li, “An energy-aware algorithm for virtual machine placement in cloud computing”, *IEEE Access.*, vol.7, no. 2019, pp. 55659–55668, 2019.

[6] Y. Gu, C. Budati, “Energy-aware workflow scheduling and optimization in clouds using bat algorithm”, *Future Generation Computer Systems*, vol.113, no. 2, pp. 106–112, 2020.

[7] M.A. Elaziz, S. Xiong, K.P.N. Jayasena, L. Li, “Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution”, *Knowledge-Based Syst*em, Vol.169, no.2019, pp. 39–52, 2019.

[8] M. Xu, W. Tian, R. Buyya,” A survey on load balancing algorithms for virtual machines placement in cloud computing”, *Concurrency Computing*, vol.29, no.7, pp. 1–16, 2017.

[9] F.F. Moges, S.L. Abebe, “Energy-aware VM placement algorithms for the Open Stack Neat consolidation framework”, *Journal of Cloud Computing*, vol. 8, no.4,pp. 1-10,2019.

[10] A.S. Abohamama, E. Hamouda, “A hybrid energy–Aware virtual machine placement algorithm for cloud environments”, *Expert System Application*, Vol.150, No.3, pp.1-7,2020.

[11] A. Ibrahim, M. Noshy, H.A. Ali, M. Badawy, “PAPSO: A power-aware VM placement technique based on particle swarm optimization”, *IEEE Access.*, Vol.8, no.2, pp20-30,2020.

[12] P. Singh, M.A. Rizvi, “Virtual machine selection strategy based on grey Wolf optimizer in cloud environment: A study”, *8th International Conference of Communication Systems and Network Technology CSNT*, pp. 108–112, 2018. https://doi.org/10.1109/CSNT.2018.8820290.

[13] A. Al-Moalmi, J. Luo, A. Salah, K. Li, “Optimal virtual machine placement based on grey wolf optimization”, *Electronic*, Vol.8, no.2, pp. 1–22,2019.

https://doi.org/10.3390/electronics8030283.

[14] S.S. Alresheedi, S. Lu, M. Abd Elaziz, A.A. Ewees, "Improved multiobjective salp swarm optimization for virtual machine placement in cloud computing", *Human-Centric Computer and Information Science*, vol.9, no.2019, pp. 1–24 ,2019. https://doi.org/10.1186/s13673-019-0174-9.

[15] S.A. Alsaidy, A.D. Abbood, M.A. Sahib, "Heuristic initialization of PSO task scheduling algorithm in cloud computing", *Journal King Saud University – Computer and Information Sciences*, vol.34 no.6,pp.2370-2382, 2022.

[16] A.M. Ammar, J. Luo, Z. Tang, O. Wajdy, "Intra-Balance Virtual Machine Placement for Effective Reduction in Energy Consumption and SLA Violation," *IEEE Access*., Vol.7,pp 72387–72402,2019.

[17] M. Mollamotalebi, S. Hajireza, "Multi-objective dynamic management of virtual machines in cloud environments", *Journal of Cloud Computing,* Vol.6 , no.16 pp1-13,2017. https://doi.org/10.1186/s13677-017-0086-z.

[18] M. Soltanshahi, R. Asemi, N. Shafiei, "Energy-aware virtual machines allocation by krill herd algorithm in cloud data centers," *Heliyon,* Vol.5, pp.3–8,2019. https://doi.org/10.1016/j.heliyon.2019.e02066.

[19] A. Satpathy, S.K. Addya, A.K. Turuk, B. Majhi, G. Sahoo, Crow search based virtual machine placement strategy in cloud data centers with live migration, *Computer and Electronic Engineering* ,Vol.69, pp. 334–350,2017. https://doi.org/10.1016/j.compeleceng.2017.12.032.

[20] A. Rezaee Jordehi, J. Jasni, "Particle swarm optimisation for discrete optimization problems: a review", *Artificial Intelligent,* Rev. vol. 43, pp. 243–258,2015. https://doi.org/10.1007/s10462-012-9373-8.

[21] M.F. Tasgetiren, Y.C. Liang, M. Sevkli, G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flow shop sequencing problem", *Eur. J. Oper. Res.,* vol.177, pp. 1930–1947,2005.
https://doi.org/10.1016/j.ejor.2005.12.024.

[22] W.C. Wu, M.S. Tsai," Application of enhanced integer coded particle swarm optimization for distribution system feeder reconfiguration,", *IEEE Trans. Power Systems,* Vol.26, pp.1591–1599, 2011. https://doi.org/10.1109/TPWRS.2010.2094212.

[23] A.H. Kashan, B. Karimi, "A discrete particle swarm optimization algorithm for scheduling parallel machines,", *Computer. Ind. Eng.* , Vol.56, pp. 216–223,2009.
https://doi.org/10.1016/j.cie.2008.05.007.

[24] Z. Miao, P. Yong, Y. Mei, Y. Quanjun, X. Xu, "A discrete PSO-based static load balancing algorithm for distributed simulations in a cloud environment,", *Future Generation Computer System,* Vol.115, pp.497–516, 2021. https://doi.org/10.1016/j.future.2020.09.016.

[25] S. Chitra, B. Madhusudhanan, G.R. Sakthidharan, P. Saravanan, "Local minima jump PSO for workflow scheduling in cloud computing environments", *Lecture Notes, Electronic Eng.* 279 LNEE pp.1225–1234, 2014. https://doi.org/10.1007/978-3-642-41674-3_170.

[26] A. Shehabi, S.J. Smith, D.A. Sartor, R.E. Brown, M. Herrlin, J.G. Koomey, E.R. Masanet, N. Horner, I.L. Azevedo, W. Lintner, "United States Data Center Energy Usage Report", Berkeley Lab. , 2016.

[27] S.K. Mishra, D. Puthal, B. Sahoo, P.P. Jayaraman, S. Jun, A.Y. Zomaya, R. Ranjan," Energy-efficient VM-placement in cloud data center", *Sustain. Computer Informatics System,* vol. 20, pp. 48–55, 2018. https://doi.org/10.1016/j.suscom.2018.01.002.

[28] W. Wei, K. Wang, K. Wang, H. Gu, H. Shen, "Multi-resource balance optimization for virtual machine placement in cloud data centers,", *Computer Electronic Engineering* , vol.88, 2020.
106866. https://doi.org/10.1016/j.compeleceng.2020.106866.

[29] M. Sardaraz, M. Tahir, "A hybrid algorithm for scheduling scientific workflows in cloud computing," , *IEEE Access.,* vol.7, pp.186137–186146,2019.
https://doi.org/10.1109/ACCESS.2019.2961106.