Computers and Electrical Engineering 119 (2024) 109506

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Enhanced virtual machine migration for energy sustainability optimization in cloud computing through knowledge acquisition

Doraid Seddiki [a], Francisco Javier Maldonado Carrascosa [a,*], Sebastián García Galán [a], Manuel Valverde Ibáñez [a], Tomasz Marciniak [b], Nicolás Ruiz Reyes [a]

[a] Telecommunications Engineering Department, Linares Higher Polytechnic School, Jaén University, Avenida de la Universidad S/N, 23700, Linares, Jaén, Spain

[b] Institute of Telecommunications and Computer Sciences, Bydgoszcz University of Science and Technology, Profesora Sylwestra Kaliskiego 7, 85-796 Bydgoszcz, Poland

ARTICLE INFO

Keywords:
Cloud computing
Virtual machine migration
Scheduling
Follow the renewable
Expert systems

ABSTRACT

Cloud computing has revolutionized the way businesses and organizations manage their computational workloads. However, the massive data centers that support cloud services consume a lot of energy, making energy sustainability a critical concern. To address this challenge, this article introduces an innovative approach to optimize energy consumption in cloud computing environments through knowledge acquisition. The proposed method uses a Knowledge Acquisition version of the Gray Wolf Optimizer (KAGWO) algorithm to collect data on the availability and use of renewable energy within data centers, contributing to improved energy sustainability in cloud computing. The proposed KAGWO method is introduced to provide a systematic approach for addressing complex problems by integrating knowledge and global optimization principles, enhancing decision-making processes with fewer configuration parameters. This article conducts a comparative analysis between the KAGWO algorithm and the Knowledge Acquisition with a Swarm Intelligence Approach (KASIA) and a Genetic Algorithm (Pittsburgh) to highlight the benefits and advantages of the former. By comparing the performance of KAGWO, Pittsburgh and KASIA in terms of energy sustainability, the study offers valuable insights into the effectiveness of knowledge-acquisition-based algorithms for optimizing renewable energy usage in cloud computing environments. The results demonstrate that the KAGWO algorithm outperforms KASIA and Pittsburgh by offering more accurate and data acquisition capabilities, resulting in enhanced energy sustainability. Overall, this study demonstrates substantial KAGWO performance improvements ranging from 0.53% to 5.23% over previous paper baselines, with particular significance found in slightly outperforming KASIA and Pittsburgh new results in small, medium and large scenarios.

1. Introduction

Swarm Intelligence (SI) belongs to a discipline within the field of Artificial Intelligence, in which the fundamental principles of its meta-heuristic algorithms are inspired by the natural behavior exhibited by various biological systems, such as ants, wolves and other analogous entities [1,2]. These systems are based on the behavior of a decentralized and self-organized population of

* Corresponding author.
E-mail addresses: ds000025@red.ujaen.es (D. Seddiki), fjmaldon@ujaen.es (F.J. Maldonado Carrascosa), sgalan@ujaen.es (S. García Galán), mvalver@ujaen.es (M. Valverde Ibáñez), tomasz.marciniak@pbs.edu.pl (T. Marciniak), nicolas@ujaen.es (N. Ruiz Reyes).

https://doi.org/10.1016/j.compeleceng.2024.109506
Received 29 November 2023; Received in revised form 20 May 2024; Accepted 18 July 2024
Available online 26 July 2024

agents that move and interact with each other to perform a given task with a specific goal and exhibit global behavior. These systems find applicability in numerous engineering problems, covering areas such as routing, scheduling, data mining, classification, optimization, etc. In this work, the problem of sustainability optimization in cloud computing is treated as paramount due to its critical implications across environmental, economic and societal domains. By addressing energy sustainability within cloud computing, this paper aims to mitigate the environmental impact associated with energy-intensive data centers, reduce operational costs through efficient resource utilization and contribute to a greener and more sustainable future for the digital infrastructure upon which modern society relies. As an example of algorithm options used in this field, the Ant Colony Optimization (ACO) [1], the Gray Wolf Optimizer (GWO) [2] and the Particle Swarm Optimization (PSO) [3] and other methodologies that take advantage of biological behavior of nature can be considered.

A previous study compared the performance of different meta-scheduling algorithms for optimizing the renewable energy consumption in four geographically distributed data centers using virtual machine (VM) migration [4]. The study found that the best algorithm was a Fuzzy Rule-Based System (FRBS) that used a knowledge base (KB) of general purpose and non-optimized rules. This algorithm achieved more than 10% improvement in renewable energy usage compared to the Static Threshold (THR), Inter Quartile Range (IQR), Local Regression (LR) and Local Regression Robust (LRR) algorithms, and also outperformed the algorithm proposed in [5] by a low margin.

The present study aims to explore the potential improvement of the FRBS solution by using knowledge acquisition algorithms based on the field of machine learning known as SI [6]. Again, the challenge consists of sustainable energy optimization in terms of increased renewable energy consumption by employing VM migration. In this regard, two algorithms are proposed: Knowledge Acquisition with a Swarm Intelligence Approach (KASIA) [6], Pittsburgh Genetic Algorithm [7] and Knowledge Acquisition over Gray Wolf Optimizer (KAGWO), which is a novel approach proposed in this work that is based on the adaptation of Gray Wolf Optimizer (GWO) algorithm for knowledge enhancement in FRBS. Therefore, KASIA and KAGWO are swarm-based approaches and use fuzzy rule bases (RB) as particles and wolves, respectively, in order to learn better KBs and consequently improve the performance of the multi-cloud data center in terms of sustainability optimization.

For each approach, an evaluation is performed to determine whether any improvement over the results obtained by the previous expert knowledge can be achieved [2]. To evaluate potential improvements, a custom fitness function is employed to evaluate the performance of each potential solution obtained through both knowledge acquisition algorithms, representing candidate KBs. The fitness function provides several simulation-based results as output, including: total energy consumed during the simulation period, renewable energy consumption for the current KB, number of migrations performed and percentage of renewable energy over total consumed energy.

It should be noted that the aforementioned fitness function is implemented through the evolution of a custom CloudSim simulator developed specifically for this study. This simulator facilitates the simulation of different broker strategies using KBs generated by the two knowledge acquisition algorithms, as well as accommodating multiple resources architectures and diverse Cloud Data Center (CDC) builds. The variations cover distinct host and VMs configurations, along with various selection and migration policies.

In this article, we present the key innovations of our research as outlined below:

- In this paper, the KAGWO algorithm is introduced as an optimization framework tailored for maximizing renewable energy utilization in cloud environments. This framework extends the capabilities of the GWO algorithm through knowledge acquisition techniques.
- Through extensive comparative analysis, the performance of KAGWO against previous baseline and KASIA, another algorithm dedicated to optimizing renewable energy utilization in cloud infrastructures, is evaluated.
- The KAGWO algorithm is characterized by fewer configuration parameters, ensuring greater simplicity and ease of use than other algorithms.

In the following sections of this paper, Background and Related Works will be presented, focusing on two key areas: sustainable VM migration and the GWO algorithm. Next, the paper will delve into Knowledge Acquisition Approaches, covering KASIA, Pittsburgh, GWO, and the new KAGWO approach. Moving forward, the Experimental Results section will show the simulation outcomes obtained from the cloud simulator. Finally, the conclusions will be drawn in the last section of the paper, summarizing the key findings from this research.

2. Background and related works

In recent years, the rapid expansion of cloud computing infrastructures has raised concerns about their significant energy consumption and environmental impact [8]. As data centers continue to grow in size and complexity, there is a pressing need for energy-efficient strategies to ensure sustainability in cloud computing environments. VM migration, the process of moving VMs across physical servers or data centers, has emerged as a promising technique to optimize resource utilization and reduce energy consumption [4] and computing offloading [9].

The concept of VM migration involves the dynamic reallocation of VMs to underutilized servers or data centers, while ensuring the continuity of services and compliance with Quality of Service (QoS) requirements. By consolidating VMs onto fewer active servers and shutting down inactive ones, VM migration enables efficient utilization of computational resources, leading to energy savings and increased sustainability [10]. Several approaches have been proposed for VM migration in the context of sustainability. These approaches typically consider factors such as CPU utilization [11], power consumption [12], workload patterns [13], and

QoS requirements [14] to make informed migration decisions. The goal is to strike a balance between resource optimization and energy efficiency, ensuring that the migration process does not compromise the performance or availability of cloud services.

A key consideration in migrating VM for sustainability is the selection of appropriate migration algorithms or techniques. Traditional approaches, such as threshold-based migration and load balancing algorithms, have been widely used but may not fully address the energy sustainability requirements of modern cloud computing environments [15]. Therefore, researchers have explored advanced optimization techniques, machine learning algorithms, and intelligent decision-making methods to improve the effectiveness and efficiency of VM migration [16]. In addition, the selection of target data centers for VM migration plays a crucial role in achieving sustainability goals [17]. Some data centers may have greener energy sources or higher energy efficiency ratings, making them more suitable for hosting migrated VMs. Therefore, the decision of where to migrate VMs must take into account factors such as energy source mix, carbon emissions, and environmental regulations to maximize the overall sustainability impact [18].

In recent years, research efforts have focused on developing innovative approaches that integrate sustainability considerations into VM migration processes. These approaches leverage techniques such as machine learning, optimization algorithms, and intelligent resource management to achieve energy-efficient and environmentally friendly VM migration [19]. By incorporating sustainability metrics and objectives into the decision-making process, these approaches aim to achieve a balance between resource optimization, energy efficiency, and environmental impact. In conclusion, VM migration among data centers is a critical aspect of achieving sustainability in cloud computing environments. By adopting intelligent migration algorithms, considering the sustainability attributes of the target data centers and integrating sustainability metrics into decision-making processes, cloud providers can optimize resource utilization, reduce energy consumption, and minimize their environmental footprint. Ongoing research in this area aims to develop robust and efficient VM migration strategies that align with sustainability goals, ultimately contributing to a more sustainable cloud computing ecosystem [20].

In recent times, cloud computing has become an indispensable platform for providing on-demand services to users in various industries. However, one of the main challenges of cloud computing is the efficient scheduling of tasks among hosts or data centers [21,22]. To address this complexity, researchers have turned to metaheuristic algorithms, which have shown promise in efficiently optimizing various aspects of cloud computing [23]. One notable metaheuristic approach that has gained significant attention is the Gray Wolf Optimization (GWO) algorithm. Inspired by the hunting behavior of gray wolves, GWO has proven its effectiveness in addressing continuous optimization problems [24]. By mimicking the social hierarchy and hunting strategies of gray wolves, GWO has demonstrated its potential to find optimal solutions in cloud-related problems, selected for its features and for the simple parameter configuration it offers. Hence, the research community has been actively exploring new variations and improvements to the GWO algorithm to address specific challenges in cloud computing. For instance, Gayathri [25] proposed the Improved GWO (IGWO) algorithm, enhanced by chaotic theory, to prevent the algorithm from converging to local optima and accelerate the convergence rate. The proposed IGWO algorithm demonstrated superior performance in terms of energy efficiency, cost and makespan compared to alternative methods. Devi and Kumar [26] focused on reducing energy consumption in CDCs while avoiding Service Level Agreement (SLA) violations. They presented a VM allocation technique that uses an Improvised Gray Wolf Optimization (IGWO) algorithm to allocate VMs to hosts based on their energy consumption and CPU utilization. The proposed technique significantly improved energy efficiency and SLA compliance, contributing to a green computing environment. Ansari et al. [27] addressed the VM Placement (VMP) problem in CDCs and proposed a Modified GWO (MGWO) algorithm. This approach aimed to balance energy consumption and memory resource waste during VM placement. The MGWO algorithm outperformed existing greedy algorithms, achieving a remarkable tradeoff between energy efficiency and memory resource waste. In the context of containerized cloud computing, Patra et al. [28] addressed load balancing to ensure even distribution of workload across available servers. They introduced a GWO-based Simulated Annealing approach (GWO-SA) that considered the deadline miss rate as an additional criterion for load balancing. The proposed GWO-SA outperformed genetic and PSO algorithms in terms of load variation and makespan. Additionally, Jain and Sharma [29] proposed an Improved Quantum Salp Swarm Algorithm (IQSSA) and a Quantum-inspired Salp Swarm GWO (QSSGWA) for task scheduling in cloud computing. These quantum-inspired hybrid algorithms effectively met SLAs, user's QoS requirements and increased the service provider's profit through efficient resource utilization. The results of their experiments showcased the superior performance of the proposed algorithms compared to various metaheuristics, and QSSGWA demonstrated the best results on different performance metrics.

These research studies collectively demonstrate the ongoing efforts to improve the capabilities of GWO-based approaches to address critical challenges within cloud computing environments. By leveraging the strengths of GWO and incorporating innovative enhancements, these studies have contributed to the advancement of cloud computing techniques, leading to more energy-efficient, cost-effective, and responsive cloud services.

Taking into account the above ideas, highlighting the ability to obtain remarkable results with few configuration parameters, the KAGWO algorithm based on the GWO algorithm has been developed.

3. Knowledge acquisition approaches

3.1. KASIA approach

In the KASIA framework, fuzzy RBs are treated as individual knowledge entities, undergoing evaluation and evolution using SI. Specifically, the PSO algorithm is adopted as the basis for acquiring fuzzy RBs. The system operates based on the social behavior exhibited by interacting individuals within a swarm. The standard procedure involves generating a swarm of RBs composed of multiple particles, where each particle, represented by matrix P, consists of rows corresponding to individual fuzzy rules within

their respective RB particle. Each rule follows the Mamdani codification and covers antecedents, consequent and connectives. The initialization of the particles includes assigning a velocity V to each one, which generates rule modifications throughout the steps of the algorithm. Subsequently, the best positions are determined for each particle individually and for the entire swarm. Notably, the evaluation process in KASIA corresponds to the evaluation of an RB, considering the constraints for the algorithm variables. Here is the structural representation for particle i:

$$P_i = \begin{bmatrix} a^i_{1,1} & a^i_{1,2} & \cdots & a^i_{1,n} & b^i_1 & c^i_1 \\ a^i_{2,1} & a^i_{2,2} & \cdots & a^i_{2,n} & b^i_2 & c^i_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ a^i_{m,1} & a^i_{m,2} & \cdots & a^i_{m,n} & b^i_m & c^i_m \end{bmatrix} \tag{1}$$

where m is the number of rules within the KB, while n denotes the number of input variables. The symbols a and b are used to represent the antecedents and consequences (in this case a single consequent is considered) of the system, respectively. Additionally, c serves as a representation of the AND/OR connector within the framework.

The membership functions of each variable are sequentially numbered, starting from 1. A value of 0 means the absence of the variable in the rule, while negative values denote the complementary set. Rules with all antecedents as 0 cannot maintain fuzzy rule coherence. The consequences can be set to 0 to represent rule absence. Moreover, a connector value of 1 signifies AND, while 2 indicates OR. Membership functions remain constant during learning, preserving rule interpretability. Therefore, the goal is to obtain a matrix that represents high-quality fuzzy rules for an FRBS. To ensure rule quality, constraints are applied to each element in the position matrix, covering antecedents, consequences, or connectors. Furthermore, it is essential to account the process of updating the swarm position. To achieve this, the velocity matrix within this approach is introduced. At each iteration, the velocity is represented as a matrix covering the particle dimensions for each particle, denoted as V_i:

$$V_i = \begin{bmatrix} v^i_{1,1} & v^i_{1,2} & \cdots & v^i_{1,n} & v^i_{1,n+1} & v^i_{1,n+2} \\ v^i_{2,1} & v^i_{2,2} & \cdots & v^i_{2,n} & v^i_{2,n+1} & v^i_{2,n+2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ v^i_{m,1} & v^i_{m,2} & \cdots & v^i_{m,n} & v^i_{m,n+1} & v^i_{m,n+2} \end{bmatrix} \tag{2}$$

After defining and evaluating the variables, the velocity and position of the particles are updated based on both individual and social experiences. The velocity update considers the inertia of the particle ω, the best RB obtained by the particle P^B and the best position found by the swarm G^B. The velocity update is formulated as:

$$V(t+1) = \omega \otimes V(t) \oplus c_1 r_1 \otimes (P^B(t) - P(t)) \oplus c_2 r_2 \otimes (G^B(t) - P(t)) \tag{3}$$

where the weight factors c_1 and c_2, representing competitive and cooperative social factors, respectively, remain constant throughout the algorithm. Meanwhile, r_1 and r_2 are randomly generated numbers within the interval [0,1]. The particle position update is formulated as follows:

$$P(t+1) = P(t) \oplus V(t+1) \tag{4}$$

Algorithm 1 KASIA Procedure.

```
Swarm initialization: N_particles, N_rules, N_iter, inertial weight ω, c1 and c2 factors.
Random setting of RB-Swarm position P.
Random setting of velocity V.
Apply P and V constraints.
Initialize Gbest and Pbest.
while N_iter do
    while N_particles do
        Update P.
        Apply constraints to P.
        Evaluate particle.
        Particles++.
    end while
    Update Gbest.
    while N_particles do
        Update Pbest.
        Update V.
        Apply constraints to V.
        Particles++.
    end while
    iter++.
end while
Return: Gbest.
```

The KASIA algorithm, as presented in Algorithm 1, is a metaheuristic optimization approach that uses a swarm of particles to solve a given problem. It begins by initializing the swarm with random positions and velocities. Through iterative steps, the algorithm

updates the positions and velocities of the particles while evaluating their fitness based on a predefined objective function, such as makespan or percentage of renewable energy. The best positions discovered by the particles are stored as the global best and personal best. Upon reaching a specific number of iterations, the algorithm returns the global best position, which represents the best solution obtained by the swarm. In general, KASIA combines exploration of the search space through particle position updates and exploitation of promising regions to find an optimal or near-optimal solution.

3.2. Pittsburgh approach

The Pittsburgh genetic approach 2, as described by Smith [7], integrates evolutionary principles, fuzzy logic, and a population-centric framework to address complex problems like task scheduling in cloud computing. In this strategy, the genetic system treats entire RBs as chromosomes, with each representing a complete RB. Genetic operators are applied to evolve populations of RBs across iterations. At the end of the optimization process, the most optimal RB is identified for incorporation into the fuzzy system. In each generation, all RBs undergo evaluation within the fuzzy system's operational domain. This evaluation is facilitated by an assessment system that measures each RB's effectiveness based on its fitness in the optimized environment. RBs are then ranked according to performance, with higher-ranked ones undergoing crossover to generate new RBs. Additionally, top-performing RBs may undergo mutations. Meanwhile, the least effective RBs in the generation are replaced by the newly evolved ones, with the replacement rate determining the extent of substitution. Through this iterative process, RBs demonstrating improved fitness criteria across generations are progressively refined, ultimately leading to the selection of the most optimal RB to serve as the cornerstone of the fuzzy system in the final generation.

Algorithm 2 Pittsburgh algorithm.

```
Initialization: N_Population, N_Rules, N_Iter, Crossover_Rate, Mutation_Rate_Init, Selection_Rate, Replacement_Rate.
Random setting of RB Population P.
Initialize G^B.
while N_Iter do
    Update Mutation_Prob with Mutation_Prob = Mutation_Rate_Init * exp(−5t / N_iter)
    while N_Particles do
        Evaluate fitness (renewable energy).
        Particles++.
    end while
    Update G^B.
    while N_Particles·Replacement_Rate do
        Generate Q offspring:
        Apply crossover to P→Q.
        Apply mutation to Q→Q.
        Apply constraints to Q.
    end while
    Update part of P with Q.
    iter++.
end while
Return: G^B
```

3.3. GWO and KAGWO approaches

The GWO is inspired by two fundamental aspects of wolf pack behavior: social hierarchy and hunting mechanisms. The social hierarchy is characterized by a strict dominance structure, where alphas occupy the highest rank and are responsible for making decisions that influence the entire group. Betas, as second-rank members, play a supporting role in the decision-making process, assisting alphas in various tasks. Deltas occupy the third rank in the hierarchy, and lastly, the omegas occupy the lowest rank, requiring submission to the authority of the other wolf ranks within the pack. This hierarchical organization ensures efficient coordination and cooperation among the wolves, contributing to their successful hunting efforts and their overall survival as a cohesive unit.

In their hunting strategy, a pack of wolves often surrounds their prey using a triangular formation, as shown in Fig. 1. The alpha, the most dominant member, assumes the frontal position, with betas and deltas flanking the sides, while omega wolves position themselves in supporting positions. As they approach their prey, wolves coordinate their movements to create a confined space, preventing the prey from escaping. Through synchronized teamwork and communication, they drive their prey towards the center of the triangle, making it difficult to find an opening to flee. With the prey effectively surrounded, the wolves seize the moment to launch a successful capture, with the alpha or other dominant members leading the takedown while the rest of the pack provides support. This tactical approach exemplifies the remarkable adaptability, intelligence, and cooperative nature of wolves as highly skilled predators.

The algorithm takes a metaphorical perspective, making the search space seem like a vast landscape characterized by peaks and valleys, each representing potential solutions. Within this terrain, the algorithm explores various scenarios, imagining where wolves might locate elusive prey and fervently pursue it, ultimately seeking the most optimal solution. In a mathematical modeling of the social hierarchy of the pack, the classification of the positions of the wolves is established, with alpha (α) occupying the most

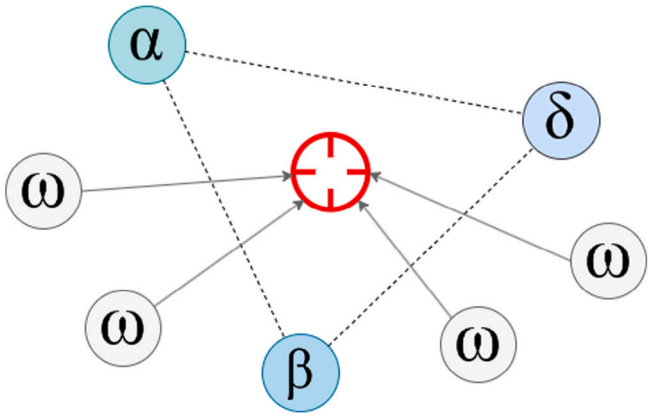

Fig. 1. Hunting mechanism of pack of gray wolves towards prey.

superior position as the best solution. It is closely followed by beta (β) which occupies the second-best position, while the delta (δ) takes the third position. All remaining wolves are collectively categorized as omegas (ω), which represent the lowest rank in the hierarchy and therefore the worst-best solutions. This hierarchical representation allows for a clear and structured understanding of the wolves' roles within the pack, aiding the development of sophisticated algorithms that mimic and leverage their social dynamics for optimization and decision-making processes.

The underlying assumption is that the optimization algorithm primarily relies on the three leading solutions: alpha, beta, and delta, forming a triangular configuration around the prey. Throughout the hunting process, as the search progresses, should a new best solution emerge, the positions of the previous alpha, beta, delta, and omegas will be promptly updated in response. This dynamic updating mechanism ensures that the algorithm remains adaptive and responsive to evolving search conditions, continually refining the approach to achieve the most efficient and effective optimization outcomes.

In the mathematical modeling of the hunting mechanisms, the main objective is to achieve a strategic encirclement of the prey. To accomplish this, two key equations are considered:

$$\vec{D} = |\vec{C}\vec{X}_p(t) - \vec{X}(t)| \tag{5}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A}\vec{D} \tag{6}$$

The first equation calculates the distance vector $\vec{D}$ that represents the distance between the wolf and the prey. The distance vector is obtained by multiplying the coefficient vector $\vec{C}$ by the $\vec{X}_p(t)$ vector that represents the position of the prey at iteration t, minus the position of the wolf in the same iteration represented by the vector $\vec{X}(t)$. The second equation updates the position of the wolf for the next iteration, denoted as $\vec{X}(t+1)$. This updated position is determined by subtracting the product of another coefficient vector $\vec{A}$ and the distance vector $\vec{D}$ calculated from the previous equation. This updating process allows the wolf to adapt its position dynamically, gradually approaching the prey while maintaining an encirclement strategy. These equations work together to establish a dynamic and strategic approach for the wolf to surround its prey during the hunting process. By continually adjusting its position based on the distance vector, the wolf can optimize its movements, increasing the probability of successfully capturing the prey. The coefficient vectors are obtained as follows:

$$\vec{A} = 2\vec{a}\vec{r_1} - \vec{a} \tag{7}$$

$$\vec{C} = 2\vec{r_2} \tag{8}$$

$$\vec{a} = (2-t)\frac{2}{maxIterations} \tag{9}$$

where $\vec{a}$ is a coefficient vector that undergoes linear decrease from 2 to 0 across all iterations, with the intention of reducing the magnitude of the movements proportional to the number of iterations. This reduction is influenced by the random vectors $\vec{r_1}$ and $\vec{r_2}$, which help the algorithm to converge as it approaches the final stages of the optimization process when $\vec{a}$ approaches 0. By incorporating this decreasing coefficient, the algorithm strategically adapts its moves and balances the effects of randomness, ultimately facilitating convergence towards an optimal solution.

The vectors $\vec{r_1}$ and $\vec{r_2}$ are random vectors, with their values uniformly distributed within the range [0,1]. These two parameters play a fundamental role in giving the GWO algorithm a stochastic nature. The inclusion of these random vectors introduces uncertainty and variability in the algorithm behavior during the search process. As a result, the algorithm exhibits probabilistic and random characteristics, allowing it to explore diverse regions of the search space and avoid being confined to a single trajectory. This stochasticity improves the adaptability of the algorithm and facilitates its ability to discover promising solutions in complex

optimization landscapes. It is essential to emphasize that the use of vectors in the equations improves abstraction and generality of the algorithm, allowing it to adapt effectively to various dimensions and ranges of the problem. This vector-based approach allows the algorithm to be flexible and applicable in various problem domains, providing a versatile and adaptable framework to address a wide range of optimization challenges.

The algorithm employs a pack of multiple wolves to conduct the hunting process, searching for the best solution. Among wolves, alpha, beta, and delta represent the three best solutions. These solutions are essential for locating the prey and updating the positions of the remaining wolves (omegas) based on their estimated positions. The algorithm assumes that the optimal solution is close to the alpha, beta and delta solutions. Consequently, computational resources are mainly allocated to these three solutions due to their higher potential, reflecting the stochastic behavior of the GWO algorithm. As a result, three specific distance equations are used for alpha, beta, and delta, adapting the optimization process to these most promising solutions:

$$\vec{D}_\alpha = |\vec{C}_1 \vec{X}_\alpha - \vec{X}| \tag{10}$$

$$\vec{D}_\beta = |\vec{C}_2 \vec{X}_\beta - \vec{X}| \tag{11}$$

$$\vec{D}_\delta = |\vec{C}_3 \vec{X}_\delta - \vec{X}| \tag{12}$$

Taking advantage of the values of these distance vectors, the algorithm determines the positions it should occupy, considering three different scenarios: only following alpha, only following beta or only following delta. Using the specific equations for each scenario, the algorithm defines three specific positions corresponding to each situation:

$$\vec{X}_1 = \vec{X}_\alpha - \vec{a}_1 \vec{D}_\alpha \tag{13}$$

$$\vec{X}_2 = \vec{X}_\beta - \vec{a}_2 \vec{D}_\beta \tag{14}$$

$$\vec{X}_3 = \vec{X}_\delta - \vec{a}_3 \vec{D}_\delta \tag{15}$$

And to consider them all, the algorithm averages their values as follows:

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{16}$$

This averaging process ensures that the algorithm appropriately weighs the influence of each individual position, leading to a comprehensive and balanced approach that benefits from the combined knowledge of alpha, beta and delta solutions.

To strike a balance between exploration and exploitation phases and to avoid local optimal solutions, the algorithm incorporates the vector $\vec{A}$. This vector varies within the range $[-2a, 2a]$, determined by a random vector $\vec{r}_1$ and linearly decreases with respect to the vector $\vec{a}$, which in turn decreases from [2,0] over time. When $-1<\vec{A}<1$, the wolf position is considered "within" the distance $\vec{D}_i$ which means an exploitation/convergence state in the search process. On the contrary, when $\vec{A}>1$ or $\vec{A}<-1$ the wolf position is considered "outside" the distance $\vec{D}_i$, indicative of an exploration state, leading the algorithm to move away from the prey to avoid local optimal solutions. In essence, the positions of alpha, beta and delta govern the search for the prey (the best solution), dynamically diverging stochastically for exploration and converging for exploitation, guided by the values of the vector $\vec{A}$.

Another crucial parameter that accentuates exploration is the coefficient vector $\vec{C}$, which encompasses random values within the range [0, 2]. This parameter plays a pivotal role in enhancing the algorithm's stochastic nature, since vector $\vec{A}$ might become trapped in local optimal solutions over time. By introducing random and independent variations through vector $\vec{C}$, the algorithm effectively balances the influence of the prey (the current solution) in defining the distance vector. When $\vec{C} > 1$, the wolves are drawn closer to the prey, while $\vec{C} < 1$ pushes them away from it. Importantly, the changes in $\vec{C}$ are not dependent on the current iteration, furnishing an additional stochastic mechanism to emphasize exploration. This not only occurs during the initial iterations like in vector $\vec{A}$ but persists throughout all iterations, effectively avoiding stagnation of local optima.

3.3.1. Knowledge acquisition over Gray Wolf optimizer

This article proposes the KAGWO Algorithm 3, which is an adaptation of the GWO for knowledge acquisition. The purpose of this algorithm is to serve as the learning mechanism for an FRBS-based expert system to generate an optimized RB for renewable energy consumption in a geographically distant modular CDCs distribution. As in the KASIA algorithm, the intelligence used to obtain an output for a given input lies in the potential KB used. In the following paragraphs, a formal mathematical description will be provided by adding the knowledge acquisition part to the GWO algorithm. In this study, every RB that composes the KB is considered a wolf (solution) which is represented by a matrix of $[m \times n + 2]$ scalars where each row represents the mathematical encoding of a fuzzy rule, m is the number of rules, n is the number of inputs (antecedents) and the two remaining parameters are the output (consequent) and the logical operator to complete the fuzzy rule. The equations presented below are those suggested by the authors for KAGWO.

$$P_i = \begin{bmatrix} ra^i_{1,1} & ra^i_{1,2} & \cdots & ra^i_{1,n} & rc^i_1 & ro^i_1 \\ ra^i_{2,1} & ra^i_{2,2} & \cdots & ra^i_{2,n} & rc^i_2 & ro^i_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ ra^i_{m,1} & ra^i_{m,2} & \cdots & ra^i_{m,n} & rc^i_m & ro^i_m \end{bmatrix} \tag{17}$$

Algorithm 3 KAGWO Procedure.

```
Pack initialization: coefficient vectors A, C, a, r1, r2, N_iter, N_wolves, N_simulations.
Random setting of prey position C.
Random setting of wolf position X_p.
Apply variable constraints.
Initialize D and X.
Obtain α, β and δ wolves positions.
while N_simulations do
    while N_iter do
        while N_wolves do
            Update X_p.
            Apply X_p constraints.
            wolves++
        end while
        Update coefficient vectors A, C and a.
        Calculate D and X.
        Evaluate pack of wolves.
        Update α, β and δ wolves positions.
        Apply position constraints.
        iter++
    end while
    sim++.
end while
Return: Best α wolf position.
```

In the particle P_i (the RB matrix), $ra_{m,n}$ represents the encoded fuzzy input value for the rule antecedent n of the rule m, where the $rc_{m,1}$ represents the encoded fuzzy input value for the rule consequent of the m rule, and at last, the $ro_{m,1}$ represents the encoded fuzzy input value for the rule logical operator of the rule m. The anatomy of the RB rules is the same as in Table 1. Each row of the particle matrix is composed of seven elements that represent the fuzzy rule. The first five elements represents the coded values of all the considered input variables which are the CDC efficiency, host's computation capacity and the computational availability and finally the VMs computational maximum needs and actual needs. The value of these entries can vary depending on the granularity of the universe of discourse, as follows:

$$ra_{m,n} \in [-MF_a, +MF_a] \tag{18}$$

where MF_a is the number of antecedent membership functions.

The sixth element represents the encoded value for the rule consequent and it will vary in the following interval:

$$rc_{m,n} \in [-MF_c, +MF_c] \tag{19}$$

where MF_c is the number of consequent membership functions.

The seventh and last element represent the logical operator for the rule, and its value is either 1 for AND or 2 for OR:

$$ro_{m,n} \in [1, 2] \tag{20}$$

Although the initial generated particles could belong to the allowed boundaries of the search space, the next matrix updates could eventually lead to an incoherent matrix whose elements reach values outside the permitted limits of the search space. To avoid these situations, the former values are constrained to ensure the coherence of the RB matrices during the execution of the algorithm. The constraints are described by the following equations:

$$ra_{m,n} = \begin{cases} -fa(MF_a) & if \quad ra_{m,n} < -fa(MF_a) \\ +fa(MF_a) & if \quad ra_{m,n} > +fa(MF_a) \end{cases} \tag{21}$$

$$rc_{m,n} = \begin{cases} -fc(MF_c) & if \quad rc_{m,n} < -fc(MF_c) \\ +fc(MF_c) & if \quad rc_{m,n} > +fc(MF_c) \end{cases} \tag{22}$$

$$ro_{m,n} = \begin{cases} 1 & if \quad ro_{m,n} < 1 \\ 2 & if \quad ro_{m,n} > 2 \end{cases} \tag{23}$$

where $fa(MF_a)$ represents the coding value of the membership function of the antecedent and $fc(MF_c)$ represents the coding value of the membership function of the consequent. Finally, any value outside the range for the value of the operator $ro_{m,n}$ will fit within the limits established by (23).

Furthermore, the KAGWO algorithm considers the search space as a landscape with peaks and valleys, then considers different possible solutions: where the wolves could find the prey they are hunting (the best solution). In the mathematical modeling of the social hierarchy of the pack, the best solutions involve the α, β, δ and ω wolves, as in the GWO algorithm. The assumption is also the same: the optimization will be influenced mainly by the first three solutions which are alpha, beta and delta, creating a triangle around the prey. If during the hunting process (search) a new best solution occur, the previous alpha, beta, delta and omegas will be updated. On the other hand, in the mathematical modeling of the hunting mechanisms, the idea is to encircle the prey and Eqs. (5)

and (6) are considered to obtain the distance D and the X position update, respectively, using the following $X_p(t)$ and $X_w(t)$ matrices for the prey and wolf positions, respectively. The coefficients are obtained like in the GWO algorithm. The A, C, and a coefficients are computed using the (7), (8) and (9) equations, respectively.

$$X_p(t) = \begin{bmatrix} ra^p_{1,1} & ra^p_{1,2} & \cdots & ra^p_{1,n} & rc^p_1 & ro^p_1 \\ ra^p_{2,1} & ra^p_{2,2} & \cdots & ra^p_{2,n} & rc^p_2 & ro^i_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ ra^p_{m,1} & ra^p_{m,2} & \cdots & ra^p_{m,n} & rc^p_m & ro^p_m \end{bmatrix} \tag{24}$$

$$X_w(t) = \begin{bmatrix} ra^w_{1,1} & ra^w_{1,2} & \cdots & ra^w_{1,n} & rc^w_1 & ro^w_1 \\ ra^w_{2,1} & ra^w_{2,2} & \cdots & ra^w_{2,n} & rc^w_2 & ro^w_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ ra^w_{m,1} & ra^w_{m,2} & \cdots & ra^w_{m,n} & rc^w_m & ro^w_m \end{bmatrix} \tag{25}$$

In order to perform the hunting of the prey (to find the best solution), the algorithm uses a pack of multiple wolves to obtain alpha, beta and delta which are the best three solutions. The algorithm will use them to find the position of the prey and also to update the position of the other wolves (omegas) based on the estimated position of these three. It is assumed that the best solution will be around the first three solutions alpha, beta and delta and the computational resources will be allocated to these solutions instead of the others because they would be the most promising (this represents the stochastic behavior of the GWO algorithm). This means that there will be three particularization of the distance equation for alpha, beta and delta, as in the GWO algorithm. The three distances are computed using the Eqs. (10), (11) and (12) and the values of α, β, δ and prey positions are described below:

$$X_\alpha(t) = \begin{bmatrix} ra^\alpha_{1,1} & ra^\alpha_{1,2} & \cdots & ra^\alpha_{1,n} & rc^\alpha_1 & ro^\alpha_1 \\ ra^\alpha_{2,1} & ra^\alpha_{2,2} & \cdots & ra^\alpha_{2,n} & rc^\alpha_2 & ro^\alpha_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ ra^\alpha_{m,1} & ra^\alpha_{m,2} & \cdots & ra^\alpha_{m,n} & rc^\alpha_m & ro^\alpha_m \end{bmatrix} \tag{26}$$

$$X_\beta(t) = \begin{bmatrix} ra^\beta_{1,1} & ra^\beta_{1,2} & \cdots & ra^\beta_{1,n} & rc^\beta_1 & ro^\beta_1 \\ ra^\beta_{2,1} & ra^\beta_{2,2} & \cdots & ra^\beta_{2,n} & rc^\beta_2 & ro^\beta_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ ra^\beta_{m,1} & ra^\beta_{m,2} & \cdots & ra^\beta_{m,n} & rc^\beta_m & ro^\beta_m \end{bmatrix} \tag{27}$$

$$X_\delta(t) = \begin{bmatrix} ra^\delta_{1,1} & ra^\delta_{1,2} & \cdots & ra^\delta_{1,n} & rc^\delta_1 & ro^\delta_1 \\ ra^\delta_{2,1} & ra^\delta_{2,2} & \cdots & ra^\delta_{2,n} & rc^\delta_2 & ro^\delta_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ ra^\delta_{m,1} & ra^\delta_{m,2} & \cdots & ra^\delta_{m,n} & rc^\delta_m & ro^\delta_m \end{bmatrix} \tag{28}$$

$$X(t) = \begin{bmatrix} ra_{1,1} & ra_{1,2} & \cdots & ra_{1,n} & rc_1 & ro_1 \\ ra_{2,1} & ra_{2,2} & \cdots & ra_{2,n} & rc_2 & ro_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ ra_{m,1} & ra_{m,2} & \cdots & ra_{m,n} & rc_m & ro_m \end{bmatrix} \tag{29}$$

And based on the values of these distance matrices, the algorithm finds the position where it is supposed to follow only alpha, only beta or only delta. Using the particularization of position equation, three positions are defined as in the GWO algorithm using the (13), (14) and (15) equations, and to consider them all, the KAGWO makes use of the (16) equation to average them. Finally, to avoid local optima and balance the KAGWO algorithm between the phases of exploration and exploitation, the A, C and a coefficients have particular constraints as in the GWO algorithm. In the context of gray wolf hunting, the exploration phase involves wolves venturing into new territories, typically surrounding areas encircled by omega wolves, in search of potential prey and resources. This phase allows for the discovery of new hunting grounds and potential targets. Conversely, the exploitation phase centers on wolves focusing their efforts within established territories, particularly within the core triangle consisting of alpha, beta, and delta wolves. This phase emphasizes the utilization and optimization of known resources, often targeting specific prey populations. The proposed method impacts these phases by strategically managing the distribution and movement of wolves, potentially increasing efficiency in prey acquisition and minimizing resource depletion within core territories while simultaneously encouraging exploration to maintain ecological balance and adaptability.

The time complexity of the KAGWO algorithm can be expressed as above. This breakdown outlines the computational cost of different components within the algorithm. Firstly, the initialization phase $O(noP)$ sets up the initial population of solutions. The main iterative loop, executed for $O(maxIter)$ times, dominates the overall time complexity. Within each iteration, the algorithm incurs costs associated with updating the population of solutions $O(noP)$, evaluating their fitness $O(f(n))$, and performing sorting operations $O(noP \times log(noP))$ to maintain the population. This breakdown provides insight into the computational demands of the KAGWO algorithm, with each element contributing to the overall time complexity.

$$O_{total} = O(noP) + O(maxIter) \times [O(noP) + O(f(n)) + O(noP \times log(noP))] \tag{30}$$

Table 1
Features for cloud system description.

Variable	Description
Renewable Availability (CDC-RA)	Renewable energy supplied to the CDC
Host Computational Capacity (HCC)	Maximum computational capacity (CC) in MIPS
Host Computational Availability (HCA)	Remaining CC of the host in MIPS after holding other VMs
VM Maximum Computational Needs (VM-MCN)	Maximum needs of the VM in MIPS
VM Current Computational Needs (VM-CCN)	Remaining computational needs in MIPS for the VM

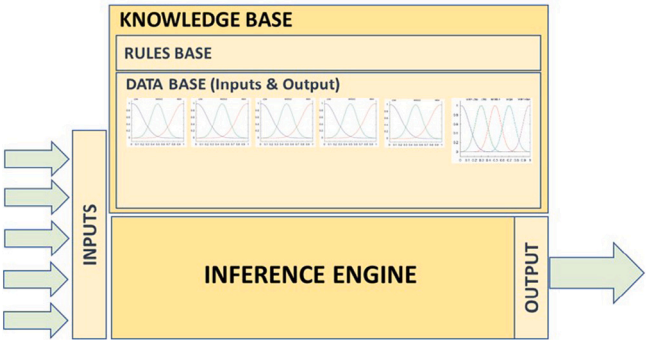

Fig. 2. FRBS meta-scheduling system.

4. Experimental results

4.1. Simulation framework

The input variables in the Table 1 represent crucial factors that impact the effectiveness of a CDC host. The CDC Renewable Availability (CDC-RA) indicates the proportion of renewable energy supplied to the CDC. The Host Computational Capacity (HCC) represents the maximum computational capacity in MIPS, while the Host Computational Availability (HCA) shows the remaining computational capacity after allocating resources to other VMs. On the other hand, the VM Maximum Computational Needs (VM-MCN) signify the maximum computational requirements of a VM, and the VM Current Computational Needs (VM-CCN) reflect the remaining computational needs of a VM after executing its current tasks. Additionally, the output variable, Host Suitability for CDC (Host-Suitability), serves as an essential metric to evaluate the overall suitability of the host to accommodate VM workloads, considering factors such as renewable energy availability, computational capacity and VM resource allocation. This evaluation contributes to the efficiency, sustainability and performance of the CDC.

In the presented Fig. 2, the FRBS meta-scheduling system employs fuzzification and defuzzification stages, utilizing membership functions to facilitate decision-making during the scheduling process. The input variables are represented by three fuzzy sets (low, medium and high), while the output is represented by five fuzzy sets (very unsuitable, unsuitable, suitable, very suitable and extremely suitable). Gaussian functions are chosen in this study to define fuzzy sets due to their efficiency and simplicity. The formulation of the Gaussian function is represented as follows:

$$\mu_i^{X_m} = \frac{1}{\sqrt{2\pi\sigma_i^{2X_m}}} exp\left(\frac{-(z-\tau_i^{X_m})^2}{2\sigma_i^{2X_m}}\right),\ \{z \in \mathbb{R},\ z \leq 1\} \tag{31}$$

where $\tau_i^{X_m}$ and $\sigma_i^{X_m}$ denote the mean and the standard deviation, respectively. Here, z denotes the independent variable describing the feature, and m corresponds to the current feature. The selection of Gaussian functions is driven by their advantageous property of having an extended area of influence, covering the entire universe of discourse. This feature allows the system to make valuable contributions across a wide range of system conditions. In a fuzzy system, each system variable value is associated with a membership degree related to a fuzzy set, assigned with normalized values ranging from 0 (complete exclusion of the variable) to 1 (full membership of the variable). Intermediate values indicate partial membership to the set, enabling precise and flexible reasoning within the fuzzy framework.

In this paper, the KBs are conformed of seven rules, since the work adopts identical rule structure used in a previous work in order to maintain consistency and leverage its proven effectiveness in addressing the problem at hand. The rule within the KBs have the anatomy shown in Table 2:

The rules are composed of eight variables, comprising the five input variables and the output of the FRBS system, the weight variable, and the logical operator used in the rule. To ensure the rules remain in a stable position, these variables are subject to specific constraints given by upper and lower bounds.

Table 2
Fuzzy rule anatomy.

Variable	Upper bounds	Lower bounds	Rules/KB	Rule anatomy
8	[3 3 3 3 3 5 1 2]	[1 1 1 1 1 1 1 1]	7	Inputs(5) - Output - Weight - Logical Operator

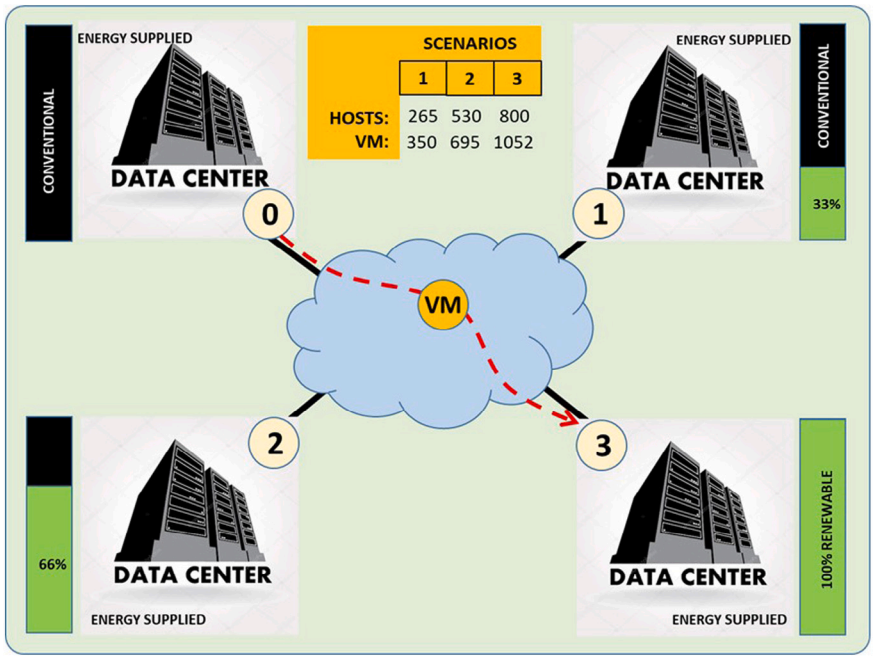

Fig. 3. General meta-scheduler structure.

In the initialization phase, the KBs are created randomly, ensuring adherence to the universe of discourse for both input and output variables. For the input variables, there are three possible values, along with the "not" or complementary operator, hence there are six distinct values per input. The output variable encompasses five different values, as well as their complementary counterparts, resulting in ten possible output values. In all rules, the weight remains constant, with a value of 1, signifying equal importance among them. Additionally, the logical operators that combine the input values have two potential values: AND and OR, allowing for different rule combinations during the initialization process.

4.2. Simulation scenarios

The simulation scenarios are carried out using an enhanced CloudSim simulator presented in [4], which allows for VM migration across four datacenters as presented in Fig. 3. These datacenters are equipped with hosts of two types:

- HP ProLiant ML110 G4 (Xeon 3040, dual-core, 1.8 GHz, 4 GB RAM, and 1Gbps of Bandwidth)
- HP ProLiant ML110 G5 (Xeon 3075, dual-core, 2.6 GHz, 4 GB RAM, and 1Gbps of Bandwidth)

The simulations are conducted using real traces from the Standard Performance Evaluation Corporation (SPEC) [30], ensuring realistic and representative workload scenarios.

Within the datacenters, the distribution of renewable energy varies across four modes: 0%, 33%, 66%, and 100% of available renewable energy inside each datacenter as depicted in Fig. 4. This variation is observed throughout the 4-hour simulation time-lapse, enabling an investigation of the impact of different renewable energy levels on the performance of the cloud infrastructure.

The simulation involves four datacenters, each representing distinct geographic regions, enabling the study of distributed cloud infrastructure behavior, including data locality and network latency considerations. Each datacenter consists of a varying number of hosts: 265, 530, or 800 physical machines, with specific characteristics detailed in Table 3, such as processing elements (PEs), RAM, bandwidth (BW), MIPS, and storage capacity.

VMs are distributed across the four datacenters, with a total of 350, 695, or 1052 VMs. Each VM possesses different features, including PEs, RAM, BW, MIPS, and size, as specified in Table 4. The cloud simulation configuration includes a workload represented by cloudlets, which can be many cases, depending on the tasks within the workload. Each cloudlet is characterized by its PEs and length, as presented in Table 5.

During each fitness function execution, a simulation is conducted based on one of the sub-scenarios outlined in Table 6. This simulation encompasses all interactions between hosts and VMs, along with migrations between the four CDCs. The obtained results

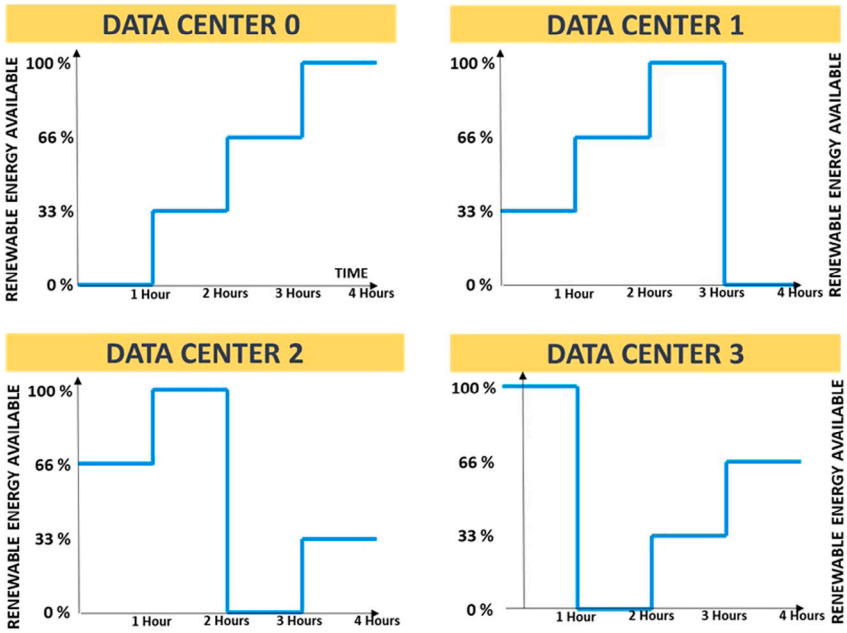

Fig. 4. Dynamic availability of renewable energy in CDCs.

Table 3
Host configuration.

Variable	Value
PEs	2
RAM (MB)	4,096
BW (MB)	1,000,000
MIPS	2,660
Storage (MB)	1,000,000

Table 4
VM configuration.

Variable	Value
PEs	1
RAM (MB)	613/870/1,740
BW (MB)	1,000
MIPS	500/1,000/2,000/2,500
Size (MB)	2,500

Table 5
Cloudlet configuration.

Variable	Value
PEs	1
Length	36,000,000

include the total energy consumed over the simulation period of four hours, the renewable energy consumed for the current KB, the number of migrations performed, and the percentage of renewable energy over the total energy consumed. Subsequently, in the following simulations, the best KB will be determined based on the percentage of renewable energy achieved among all possible solutions (particles/wolves). The KB demonstrating the highest percentage of renewable energy will be deemed the most favorable and selected as the best solution.

The experiments were conducted based on three distinct scenarios as stated in [4] and in Table 6, each characterized by different sizes: small, medium, and big. In Scenario 1, the simulation included 256 hosts, 350 VMs, and 500, 1500, or 3000 cloudlets. Scenario 2 comprised 530 hosts, 695 VMs, and 1000, 2000, or 5000 cloudlets. Lastly, Scenario 3 involved 800 hosts, 1052 VMs, and 1500, 5000, or 10,000 cloudlets. Within each scenario, three sub-scenarios were initiated, corresponding to small, medium, and big simulations, with each sub-scenario representing varying numbers of cloudlets as shown in Table 6, thus having nine

Table 6
Scenarios of simulation based on Hosts, VMs and Cloudlets.

Scenario	Hosts	VMs	Cloudlets
1 (Small)	265	350	500/1,500/3,000
2 (Medium)	530	695	1,000/2,000/5,000
3 (Large)	800	1,052	1,500/5,000/10,000

Table 7
KASIA parameter configuration.

Simulations	Particles	Iterations	Initial weight	Final weight	c_1	c_2
30	64	50	0.9	0.2	2	2

Table 8
Pittsburgh parameter configuration.

Simulations	Particles	Iterations	Crossover rate	Initial mutation rate	Selection rate	Replacement rate
30	64	50	0.8	0.1	0.8	0.8

Table 9
KAGWO parameter configuration.

Simulations	Particles	Iterations
30	64	50

simulation scenarios. The hosts were evenly distributed across four distant data centers, ensuring equitable representation of the cloud infrastructure. These diverse scenarios and simulation variations allowed for a comprehensive evaluation of the system's performance under varying workload sizes and infrastructure capacities.

The configuration of KASIA, Pittsburgh and KAGWO algorithms are presented in the Tables 7–9. The simulations were conducted through 30 repetitions, and each simulation utilized 50 iterations to optimize the performance. For KASIA, Pittsburgh and KAGWO, a total of 64 particles/wolves were employed, specifically designed to utilize multiple cores effectively. The approaches leverages 64 cores from the simulation machine using multithreading, enabling concurrent evaluation of the renewable energy consumed fitness function for a given KB. This approach optimizes CPU usage across all cores, and extensive testing revealed that 64 particles offered the best results.

In KASIA, the weight used had a maximum value of 0.9, which decreased exponentially to 0.2 during the optimization process. The social coefficients were both set to 2, as they delivered the most favorable outcomes based on several tests. On the other hand, The KAGWO configuration is considered better than KASIA's and Pittsburgh's due to its simplicity and efficiency, as it requires fewer parameters for optimization. While KASIA and Pittsburgh necessitate careful tuning of various parameters, such as weight, social or genetic coefficients, KAGWO's straightforward setup with only the number of simulations, wolves, and iterations significantly streamlines the configuration process. This advantage allows for easier implementation and reduces the risk of potential parameter-related complexities and errors, making KAGWO a more practical and preferred choice for optimization tasks.

4.3. Simulation results

The objective of this research is to validate the efficacy of the KAGWO algorithm, which belongs to the SI field of machine learning, in enhancing a baseline solution established by an expert-developed set of rules utilized within a FRBS. For each of the nine scenarios, 30 simulations were performed for KASIA, Pittsburgh and KAGWO. Table 10 illustrates the mean and standard deviation of all simulations for each scenario, encompassing both algorithms and the results of the previous paper [4].

In evaluating the performance of renewable energy utilization across diverse cloud computing scenarios, the KAGWO algorithm consistently emerges as the superior solution in comparison to both the baseline results from [4] and the KASIA algorithm. In the small scenario (256 Hosts, 350 VMs), KAGWO showcased notable improvements, achieving a 3.51% increase in renewable energy utilization with 500 cloudlets, while Pittsburgh also demonstrated competitive results with a 3.41% improvement. This trend continued with larger cloudlet counts, with KAGWO reaching a 4.57% improvement, slightly surpassing Pittsburgh's improvement of 4.37%. Simultaneously, KASIA demonstrated respectable improvements; however, KAGWO consistently outperformed KASIA in terms of percentage improvement. Transitioning to the medium scenario (530 Hosts, 695 VMs), KAGWO continued to exhibit commendable performance, achieving a maximum improvement of 5.23% with 2000 cloudlets, while Pittsburgh maintained its competitiveness with a 5.21% improvement. In comparison, KASIA showcased competitive improvements, reaching a maximum of 5.19% with 2000 cloudlets. Despite KASIA's noteworthy improvements, KAGWO maintained an edge in terms of percentage increase in renewable energy utilization. Even in the large scenario (800 Hosts, 1052 VMs), KAGWO consistently outperformed the baseline, with a 1.34% improvement in renewable energy utilization with 5000 cloudlets, while Pittsburgh showcased a 1.28% improvement. While KASIA achieved a 1.31% increase with the same number of cloudlets, KAGWO demonstrated a slightly more significant

Table 10
Results comparison between the Baseline results, KASIA, Pittsburgh and KAGWO: Percentage of renewable energy used.

Hosts	VMs	Cloudlets	[4] Results	KASIA	KASIA Imp.	Pitts.	Pitts. Imp.	KAGWO	KAGWO Imp.
256	350	500	63.31 ± 0.13	65.49 ± 0.02	2.18 (3.45%)	65.47 ± 0.03	2.16 (3.41%)	65.53 ± 0.02	2.22 (3.51%)
256	350	1,500	62.75 ± 0.11	65.51 ± 0.02	2.76 (4.40%)	65.49 ± 0.02	2.74(4.37%)	65.52 ± 0.02	2.77 (4.42%)
256	350	3,000	62.68 ± 0.15	65.50 ± 0.02	2.82 (4.50%)	65.50 ± 0.02	2.82 (4.50%)	65.54 ± 0.02	2.86 (4.57%)
530	695	1,000	63.71 ± 0.07	66.69 ± 0.02	2.98 (4.68%)	66.69 ± 0.02	2.98 (4.68%)	66.73 ± 0.01	3.02 (4.74%)
530	695	2,000	63.40 ± 0.08	66.69 ± 0.01	3.29 (5.19%)	66.70 ± 0.02	3.30 (5.21%)	66.72 ± 0.01	3.32 (5.23%)
530	695	5,000	63.65 ± 0.09	66.69 ± 0.01	3.04 (4.77%)	66.68 ± 0.01	3.03 (4.76%)	66.72 ± 0.01	3.07 (4.83%)
800	1,052	1,500	64.36 ± 0.05	64.67 ± 0.02	0.31 (0.49%)	64.66 ± 0.02	0.30 (0.47%)	64.70 ± 0.01	0.34 (0.53%)
800	1,052	5,000	63.85 ± 0.07	64.68 ± 0.02	0.83 (1.31%)	64.67 ± 0.02	0.82 (1.28%)	64.70 ± 0.02	0.85 (1.34%)
800	1,052	10,000	64.28 ± 0.09	64.69 ± 0.01	0.41 (0.64%)	64.67 ± 0.01	0.39 (0.61%)	64.72 ± 0.02	0.44 (0.68%)

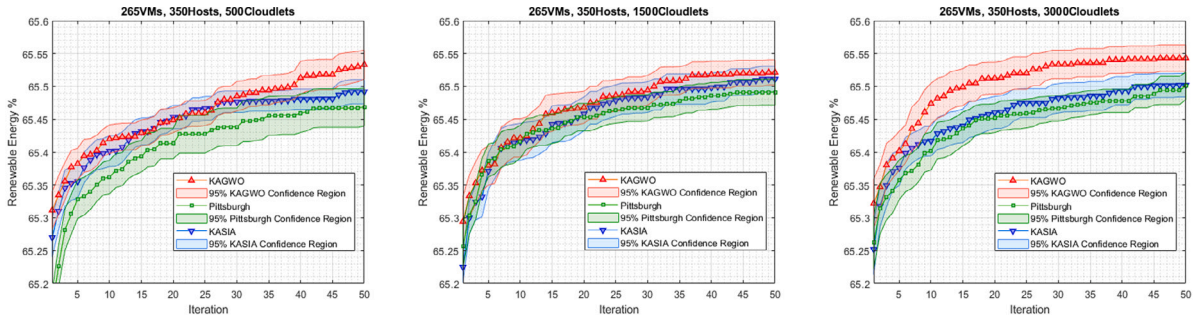

Fig. 5. Evolution of KASIA, Pitts. and KAGWO through knowledge acquisition in the small scenario (1st, 2nd and 3rd sub-scenarios).

improvement. These comprehensive findings underscore the robust and reliable nature of the KAGWO algorithm, positioning it as a standout choice for optimizing renewable energy usage across a spectrum of cloud computing scenarios.

The superiority of the KAGWO approach extends beyond its performance in renewable energy utilization; it is also evident in terms of parameters configuration, making it a more straightforward and practical choice for implementation. KAGWO achieves its impressive results with a leaner set of configuration parameters compared to the more intricate configurations of KASIA and Pittsburgh. The simplicity of KAGWO's parameter setup not only streamlines the implementation process but also reduces the potential for complexity-related errors. This efficiency in configuration makes KAGWO a more accessible and user-friendly solution, further highlighting its practical advantages over KASIA and Pittsburgh. In essence, the KAGWO approach not only excels in performance metrics but also offers a more straightforward and user-friendly experience, making it an appealing choice for those seeking both effectiveness and ease of implementation.

The graphical comparison of KASIA, Pittsburgh and KAGWO across the nine sub-scenarios reveals distinct convergence patterns and performance dynamics as seen in the Figs. 5, 6, and 7. In the first sub-scenario, KASIA and Pittsburgh demonstrates an early convergence, but KAGWO exhibits a tendency for continual improvement through iterations, ultimately surpassing KASIA and Pittsburgh. The superiority of KAGWO becomes statistically evident as the 95% confidence regions do not overlap, emphasizing its consistent outperformance. Conversely, in the second sub-scenario, KASIA, Pittsburgh and KAGWO exhibit rapid convergence, with their results overlapping. The third sub-scenario witnesses fast convergence for all algorithms, but KAGWO stands out with superior results, as indicated by non-overlapping confidence regions. A similar trend is observed in the fourth, fifth, and sixth sub-scenarios, where KAGWO consistently outperforms KASIA and Pittsburgh after surpassing its convergence point, except in the 5th sub-scenario. The seventh sub-scenario mirrors this pattern. In the eighth sub-scenario, KAGWO, Pittsburgh and KASIA exhibit similar results, with overlapping confidence regions. The ninth sub-scenario replicates the convergence dynamics of the first, with KAGWO showcasing superiority over KASIA and Pittsburgh. This comprehensive graphical analysis underscores the nuanced performance variations across scenarios, establishing KAGWO as a consistently effective and robust solution in comparison to KASIA and Pittsburgh. KAGWO consistently demonstrates superior performance across a spectrum of scenarios, excelling particularly in the exploration phase, where its initial iterations consistently outshine those of KASIA and Pittsburgh. This early advantage lays a solid foundation for subsequent iterations, facilitating a more effective exploitation phase. As KAGWO progresses, its final iterations consistently outperform KASIA and Pittsburgh, showcasing its capacity for robust exploitation and optimization. In essence, KAGWO's strength lies in its ability to not only explore the solution space effectively from the outset but also to exploit discovered solutions more efficiently as it evolves, making it a formidable choice across various optimization tasks.

Table 11 presents the results of an ANOVA test conducted on data from various scenarios, comparing different hosts, VMs, and cloudlets configurations. Significance levels at $\alpha = 0.05$ were used. It indicates the p-values for comparisons between algorithms across all scenarios, comparing KAGWO vs KASIA and KAGWO vs Pittsburgh, along with whether one scenario was significantly better than the other based on the p-value. This additional statistical analysis provides valuable insights into the performance variations across different configurations, aiding in the comprehensive understanding of the studied system.

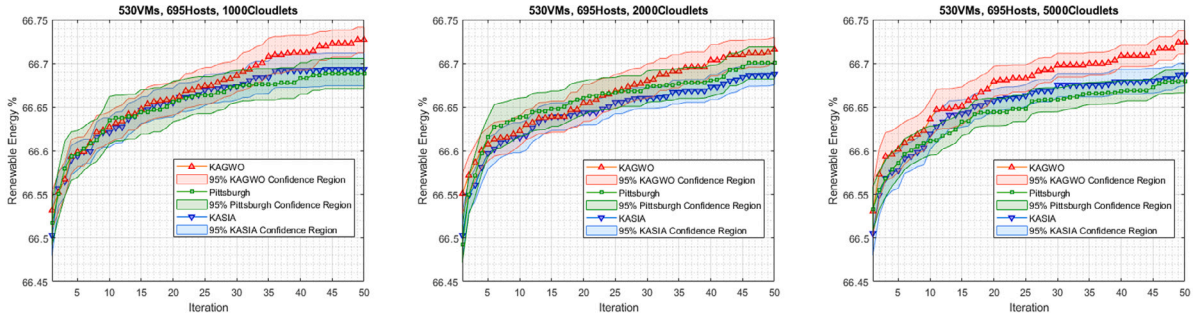

Fig. 6. Evolution of KASIA, Pitts. and KAGWO through knowledge acquisition in the medium scenario (4th, 5th and 6th sub-scenarios).

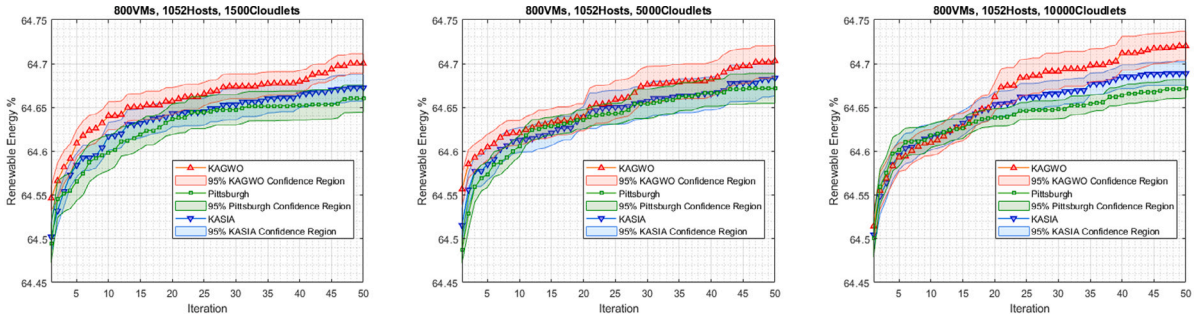

Fig. 7. Evolution of KASIA, Pitts. and KAGWO through knowledge acquisition in the large scenario (7th, 8th and 9th sub-scenarios).

Table 11
ANOVA test conducted on data across all scenarios using a significance level $\alpha = 0.05$.

Scenarios			KAGWO vs KASIA		KAGWO vs Pittsburgh	
Hosts	VMs	Cloudlets	ANOVA p-value	Significantly better	ANOVA p-value	Significantly better
256	350	500	0.006521	✓	0.000993	✓
256	350	1,500	0.466500	✗	0.036145	✓
256	350	3,000	0.004118	✓	0.005937	✓
530	695	1,000	0.007370	✓	0.001578	✓
530	695	2,000	0.004290	✓	0.199050	✗
530	695	5,000	0.000282	✓	0.000022	✓
800	1,052	1,500	0.004947	✓	0.000142	✓
800	1,052	5,000	0.157650	✗	0.015717	✓
800	1,052	10,000	0.004931	✓	0.000009	✓

The primary aim of this research was to investigate the capabilities of KAGWO, a SI-based algorithm, in improving the results of a FRBS that utilized a KB provided by an expert. The comparison was made with other approaches, including KASIA and Pittsburgh, which had demonstrated efficacy in similar applications. The KAGWO results obtained from the previous section showcased a significant improvement, ranging between 3.51% and 5.23% for the first two simulation scenarios. However, KAGWO exhibited certain limitations in the third scenario, showing only a modest improvement ranging between 0.53% and 1.34%. Further comparisons were conducted, and it was revealed that KASIA and Pittsburgh delivered similar, but worse, results to KAGWO, with analogous limitations in the latter scenario.

These findings suggest intriguing applications for small and medium and large-scale architectures, where the utilization of renewable energy can be optimized through the implementation of KAGWO for broker strategies in VM migrations across distant CDCs. By harnessing KAGWO's capabilities, significant improvements in energy costs can be achieved, making it a valuable tool for enhancing energy efficiency in cloud computing environments.

5. Conclusions and future actions

The pursuit of sustainable and eco-friendly practices in cloud computing gave rise to innovative approaches for optimizing resources. In this context, the new KAGWO approach was presented as a promising methodology. The results obtained from the application of KAGWO demonstrated noticeable improvements comparable to those achieved by KASIA and Pittsburgh, with a slight enhancement observed in the last sub-scenario. The improvements over the baseline results presented in [4] ranged from

0.53% to 5.23%. Although the improvement in the third scenario was not significant in the overall context, it showed slightly better performance than baseline results. The algorithms exhibited similar characteristics, but the advantage of KAGWO lay in its ability to achieve comparable results with fewer configuration parameters. This made KAGWO a more attractive choice for optimization tasks, as it simplified the implementation process and reduced the risk of potential parameter-related complexities.

Moving forward, it would be intriguing to explore the potential of other machine learning algorithms, such as neural networks, to further enhance the results presented in this study. Investigating the applicability of neural networks in this context may lead to even more robust and efficient optimization strategies for CDCs. Additionally, as part of future research, it would be beneficial to conduct more extensive comparisons between KAGWO and KASIA, considering various aspects such as makespan, convergence speed, scalability, and computational efficiency. Further investigation into the behavior and performance of these algorithms in larger-scale cloud environments may provide valuable insights for practical implementation. Moreover, it would be valuable to explore the generalization capabilities of KAGWO in different cloud scenarios and real world applications. Analyzing its performance under diverse conditions and varying workload scenarios can offer a more comprehensive understanding of its potential advantages and limitations.

CRediT authorship contribution statement

Doraid Seddiki: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Francisco Javier Maldonado Carrascosa:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing. **Sebastián García Galán:** Conceptualization, Formal analysis, Investigation, Funding acquisition, Project administration, Validation, Visualization, Resources, Supervision, Writing – original draft, Writing – review & editing. **Manuel Valverde Ibáñez:** Conceptualization, Investigation, Resources, Supervision, Visualization, Writing – review & editing. **Tomasz Marciniak:** Conceptualization, Investigation, Supervision, Visualization, Writing – review & editing. **Nicolás Ruiz Reyes:** Conceptualization, Investigation, Project administration, Resources, Supervision, Visualization, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Sebastan Garcia Galan reports financial support was provided by Government of Andalusia. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work has been supported by the Research Project P18-RT-4046, funded by the Andalusian Government in Spain.

References

[1] Akhtar A. Evolution of ant colony optimization algorithm–A brief literature review. Neural Evol Comput 2019;4:4–8. http://dx.doi.org/10.48550/arXiv.1908.08007, arXiv preprint.
[2] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Adv Eng Softw 2014;69:46–61. http://dx.doi.org/10.1016/j.advengsoft.2013.12.007.
[3] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of iCNN'95-international conference on neural networks, vol. 4. IEEE; 1995, p. 1942–8. http://dx.doi.org/10.1109/ICNN.1995.488968.
[4] Seddiki D, Galán SG, Expósito JEM, Ibañez MV, Marciniak T, Rocío J, et al. Sustainable expert virtual machine migration in dynamic clouds. Comput Electr Eng 2022;102:108257. http://dx.doi.org/10.1016/j.compeleceng.2022.108257.
[5] Seddiki D, Galán SG, Expósito EM, Ibañez MV, Marciniak T, De Prado RJP. Sustainability-based Framework for virtual machines migration among cloud data centers. In: 2021 15th international conference on signal processing and communication systems. ICSPCS, IEEE; 2021, p. 1–8. http://dx.doi.org/10.1109/ICSPCS53099.2021.9660237.
[6] García-Galán S, Prado RP, Expósito JEM. Swarm fuzzy systems: knowledge acquisition in fuzzy systems and its applications in grid computing. IEEE Trans Knowl Data Eng 2013;26(7):1791–804. http://dx.doi.org/10.1109/TKDE.2013.118.
[7] Smith SF. A learning system based on genetic adaptive algorithms. University of Pittsburgh; 1980.
[8] Bharany S, Sharma S, Khalaf OI, Abdulsahib GM, Al Humaimeedy AS, Aldhyani TH, et al. A systematic survey on energy-efficient techniques in sustainable cloud computing. Sustainability 2022;14(10):6256. http://dx.doi.org/10.3390/su14106256.
[9] Taheri-abed S, Eftekhari Moghadam AM, Rezvani MH. Machine learning-based computation offloading in edge and fog: a systematic review. Cluster Comput 2023;26(5):3113–44. http://dx.doi.org/10.1007/s10586-023-04100-z.
[10] Magotra B, Malhotra D, Dogra AK. Adaptive computational solutions to energy efficiency in cloud computing environment using VM consolidation. Arch Comput Methods Eng 2023;30(3):1789–818. http://dx.doi.org/10.1007/s11831-022-09852-2.
[11] Awad M, Kara N, Leivadeas A. Utilization prediction-based VM consolidation approach. J Parallel Distrib Comput 2022;170:24–38. http://dx.doi.org/10.1016/j.jpdc.2022.08.001.
[12] Arshad U, Aleem M, Srivastava G, Lin JC-W. Utilizing power consumption and SLA violations using dynamic VM consolidation in cloud data centers. Renew Sustain Energy Rev 2022;167:112782. http://dx.doi.org/10.1016/j.rser.2022.112782.

[13] Patel YS, Bedi J. MAG-D: A multivariate attention network based approach for cloud workload forecasting. Future Gener Comput Syst 2023;142:376–92. http://dx.doi.org/10.1016/j.future.2023.01.002.
[14] Rezakhani M, Sarrafzadeh-Ghadimi N, Entezari-Maleki R, Sousa L, Movaghar A. Energy-aware QoS-based dynamic virtual machine consolidation approach based on RL and ANN. Cluster Comput 2023;1–17. http://dx.doi.org/10.1007/s10586-023-03983-2.
[15] Gures E, Shayea I, Ergen M, Azmi MH, El-Saleh AA. Machine learning-based load balancing algorithms in future heterogeneous networks: A survey. IEEE Access 2022;10:37689–717. http://dx.doi.org/10.1109/ACCESS.2022.3161511.
[16] Supreeth S, Patil K. VM scheduling for efficient dynamically migrated virtual machines (VMS-EDMVM) in cloud computing environment. KSII Trans Internet Inf Syst 2022;16(6).
[17] Saxena D, Singh AK, Lee C-N, Buyya R. A sustainable and secure load management model for green cloud data centres. Sci Rep 2023;13(1):491. http://dx.doi.org/10.1038/s41598-023-27703-3.
[18] Vatsal S, Verma SB. Virtual machine migration based algorithmic approach for safeguarding environmental sustainability by renewable energy usage maximization in Cloud data centres. Int J Inf Technol 2023;1–16. http://dx.doi.org/10.1007/s41870-023-01478-2.
[19] Kumar Y, Kaul S, Hu Y-C. Machine learning for energy-resource allocation, workflow scheduling and live migration in cloud computing: State-of-the-art survey. Sustain Comput Inform Syst 2022;36:100780. http://dx.doi.org/10.1016/j.suscom.2022.100780.
[20] Khaleel MI. Synergies between resource sustainability and energy performance of cloud servers: The role of virtual machine repacking approach. Comput Electr Eng 2023;106:108568. http://dx.doi.org/10.1016/j.compeleceng.2022.108568.
[21] Song Y, Li C, Tian L, Song H. A reinforcement learning based job scheduling algorithm for heterogeneous computing environment. Comput Electr Eng 2023;107:108653. http://dx.doi.org/10.1016/j.compeleceng.2023.108653.
[22] Prado R, García-Galán S, Yuste AJ, Expósito JM. Genetic fuzzy rule-based scheduling system for grid computing in virtual organizations. Soft Comput 2011;15:1255–71. http://dx.doi.org/10.1007/s00500-010-0660-5.
[23] Shingne H, Shriram R. Heuristic deep learning scheduling in cloud for resource-intensive internet of things systems. Comput Electr Eng 2023;108:108652. http://dx.doi.org/10.1016/j.compeleceng.2023.108652.
[24] Li H, Lv T, Shui Y, Zhang J, Zhang H, Zhao H, et al. An Improved grey wolf optimizer with weighting functions and its application to Unmanned Aerial Vehicles path planning. Comput Electr Eng 2023;111:108893. http://dx.doi.org/10.1016/j.compeleceng.2023.108893.
[25] Gayathri B. Gray wolf optimisation based energy efficient green cloud computing. J Algebr Stat 2022;13(1):932–40.
[26] Devi NN, Kumar SV. SLAV mitigation and energy-efficient VM allocation technique using improvised grey wolf optimization algorithm for cloud computing. In: 2022 8th international conference on advanced computing and communication systems, vol. 1. ICACCS, IEEE; 2022, p. 155–60. http://dx.doi.org/10.1109/ICACCS54159.2022.9785078.
[27] Ansari A, Asghari M, Gorgin S, Rahmati D. A modified grey wolf optimization for energy efficiency and resource wastage balancing in cloud data-centers. In: 2020 10th international conference on computer and knowledge engineering. ICCKE, IEEE; 2020, p. 392–8. http://dx.doi.org/10.1109/ICCKE50421.2020.9303615.
[28] Patra MK, Misra S, Sahoo B, Turuk AK. GWO-based simulated annealing approach for load balancing in cloud for hosting container as a service. Appl Sci 2022;12(21):11115. http://dx.doi.org/10.3390/app122111115.
[29] Jain R, Sharma N. A quantum inspired hybrid SSA–GWO algorithm for SLA based task scheduling to improve QoS parameter in cloud computing. Cluster Comput 2022;1–24. http://dx.doi.org/10.1007/s10586-022-03740-x.
[30] Standard Performance Evaluation Corporation (SPEC). 2023, http://www.spec.org/index.html. [Accessed 28 November 2023].