I.J. Modern Education and Computer Science, **2023, 6, 44-55**
Published Online on December 8, 2023 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2023.06.04

Dynamic Load Balancing in Cloud Computing: A Convergence of PSO, GSA, and Fuzzy Logic within a Hybridized Metaheuristic Framework

Rajgopal K T
Department of Computer Science and Engineering, Canara Engineering College, Benjanapadavu, Karnataka, India
Email: rajgopalkt@gmail.com
ORCID iD: https://orcid.org/0009-0001-4557-1240

Abhishek S. Rao*
Department of Information Science and Engineering, Nitte (Deemed to be University), NMAM Institute of Technology, Nitte, Karnataka, India
E-mail: abhishekrao@nitte.edu.in
ORCID iD: https://orcid.org/0000-0002-3574-3571
*Corresponding Author

Ramaprasad Poojary
School of Engineering & IT, Manipal Academy of Higher Education, Dubai Campus, Dubai, UAE
Email: rama.poojary@manipaldubai.com
ORCID iD: https://orcid.org/0000-0002-4354-5639

Deepak D
Department of Information Science and Engineering, Nitte (Deemed to be University), NMAM Institute of Technology, Nitte, Karnataka, India
Email: deepak_dv@nitte.edu.in
ORCID iD: https://orcid.org/0000-0002-9869-0825

Received: 14 May, 2023; Revised: 24 August, 2023; Accepted: 13 September, 2023; Published: 08 December, 2023

Abstract: In the recent era, there has been a significant surge in the demand for cloud computing due to its versatile applications in real-time situations. Cloud computing efficiently tackles extensive computing challenges, providing a cost-effective and energy-efficient solution for cloud service providers (CSPs). However, the surge in task requests has led to an overload on cloud servers, resulting in performance degradation. To address this problem, load balancing has emerged as a favorable approach, wherein incoming tasks are allocated to the most appropriate virtual machine (VM) according to their specific needs. However, finding the optimal VM poses a challenge as it is considered a difficult problem known as NP-hard. To address this challenge, current research has widely adopted meta-heuristic approaches for solving NP-hard problems. This research introduces a novel hybrid optimization approach, integrating the particle swarm optimization algorithm (PSO) to handle optimization, the gravitational search algorithm (GSA) to improve the search process, and leveraging fuzzy logic to create an effective rule for selecting virtual machines (VMs) efficiently. The integration of PSO and GSA results in a streamlined process for updating particle velocity and position, while the utilization of fuzzy logic assists in discerning the optimal solution for individual tasks. We assess the efficacy of our suggested method by gauging its performance through various metrics, including throughput, makespan, and execution time. In terms of performance, the suggested method demonstrates commendable performance, with average load, turnaround time, and response time measuring at 0.168, 18.20 milliseconds, and 11.26 milliseconds, respectively. Furthermore, the proposed method achieves an average makespan of 92.5 milliseconds and average throughput performance of 85.75. The performance of the intended method is improved by 90.5%, 64.9%, 36.11%, 24.72%, 18.27%, 11.36%, and 5.21 in comparison to the existing techniques. The results demonstrate the efficacy of this approach through significant improvements in execution time, CPU utilization, makespan, and throughput, providing a valuable contribution to the field of cloud computing load balancing.

Index Terms: Cloud Computing, Fuzzy Logic, Gravitational Search Algorithm, Load Balancing, NP-hard problem, Particle Swarm Optimization.

1. Introduction

In recent times, there has been a remarkable surge in the acceptance of cloud computing across diverse application scenarios. This growth is realized due to advancements in technology, growth in internet use, and the capacity to solve large-scale problems. This approach introduces both hardware and software applications as accessible resources for users in the cloud. The significant advancement in information technology has resulted in the widespread acceptance and assimilation of cloud computing methodologies into internet-enabled computing systems. In these internet-based solutions, the cloud computing system shares the basis of the resource on pay-per-use such as storage, servers, networks, services, software, and computing platforms to users based on user demand [1]. The cloud computing model is adopted by major IT companies such as Apple, VMware, Red Hat, Verizon Cloud, VMware, Google, IBM, Amazon, Microsoft, Oracle, and many more. Generally, cloud computing systems are grouped into two main category's location and service-based category. In location-based categories, cloud computing systems can be public cloud, private, hybrid, or community clouds [2]. The public cloud services are reachable to the public, with the cloud infrastructure hosted by the cloud service provider. Nevertheless, the openness of public clouds to users for diverse computing tasks makes them vulnerable to potential attacks. The public cloud is the most suitable cost-effective solution. On the contrary, private cloud services are generally offered to specific organizations or users. This cloud service is considered more secure, but it increases the cost of the service [3]. The hybrid cloud is designed with the help of public and private cloud which is used for various objectives corresponding to the user or organization requirement. The community cloud is specially designed for use by multiple organizations that have shared data management models. In this cloud, a common infrastructure is used to provide the service to each organization. Conversely, cloud computing systems have the potential to be categorized according to the services they provide, which encompass Infrastructure Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [4]. In the IaaS service, the cloud providers offer basic processes such as computers, networking features, and control over the available computing resources. PaaS services do not require any basic infrastructure hence handling the basic operations is skipped by the organization in PaaS and it allows directly to focus on application deployment. SaaS cloud services focus only on the software requirement and do not consider the entire infrastructure and service management in the cloud. Additionally, cloud computing services are extended as Communication as a Service (CaaS) [5], Database as a Service (DaaS) [6], Expert as a service (EaaS) [7], Storage as a Service (SaaS) [8], Security as a Service (SECaaS) [9], Testing-as-a-service (TaaS) [10], Network as a Service (NaaS) [11], and Monitoring as a Service (MaaS) [12]. These cloud services cater to diverse applications, aligning with the specific needs of users. In the current scenario, several cloud-based applications are present such as educational purposes, health monitoring, data analysis, robotics, and many more. These applications offer direct accessibility, eliminating the need for limited knowledge about the underlying computing environment. The primary objective of these cloud computing systems is to efficiently distribute available resources, thereby ensuring optimal computing performance. This helps to minimize the computation cost, power requirement, and time to accomplish the computation task. Additionally, cloud computing is centered on the efficient distribution of computing resources worldwide, enabling tasks to be executed optimally across various data centers and enhancing the overall computing performance for users of the cloud [11]. Along with cloud computing, numerous other distribution systems exist such as grid computing, and peer-to-peer computing that can be employed for data transfer and resource-sharing facilities [13]. By employing these methods, the integration of cloud computing with these distribution mechanisms has the potential to enhance the establishment of a cost-efficient data center. This offers better business opportunities and quick task processing for cloud users. Thus, resource management is the prime task for the successful deployment of any cloud computing system. To achieve this, resource scheduling and allocation are considered efficient solutions. However, these resources are not in physical form and are offered to users in a virtual form based on their computational requirements. Cloud services provide a virtualization mechanism to offer these resources.

In a cloud computing system, the physical machine can be transformed into multiple virtual machines and these virtual machines can be employed to process multiple tasks. While processing these virtual resources, CSP plays a significant role in allocating the resources. During this process of incoming requests, some virtual machines receive a huge number of requests, and some virtual machines receive fewer requests. This scenario creates an unbalanced condition for cloud service providers which leads to a huge imbalance between users and resource utilization. This imbalance problem results in degrading the performance of cloud computing systems. During the incoming tasks, when a huge number of tasks are allocated at a VM, and its resources get exhausted then this situation is known as an overload state and here load balancing technique needs to be applied to decrease the load on a single VM. In this scenario, the tasks need to be migrated to another VM. This scheme is applied in three main steps load balancing, resource discovery, and workload migration. In general, load-balancing algorithms can be classified into two main types: static and dynamic load-balancing schemes. The static algorithm addresses the load balancing issue by considering the overall historical behavior of the system, whereas the dynamic load balancing scheme focuses on the more recent system behavior to efficiently manage the load. Similarly, the load-balancing algorithms are further categorized as centralized and distributed load-balancing approaches. The centralized algorithm consists of a central controller. These algorithms are easy to implement but the rate of failure is higher in these approaches. On the contrary, the distributed algorithms consider multiple computing nodes for load balancing.

Dynamic Load Balancing in Cloud Computing often employs various algorithms and heuristics to optimize task allocation to virtual machines (VMs). These solutions aim to enhance several critical performance metrics, including CPU utilization, execution time, makespan, and throughput. They typically use techniques like round-robin scheduling, weighted load balancing, or task migration strategies. While some of these methods have shown improvements in execution time and CPU utilization, they may struggle to simultaneously optimize makespan and throughput. Furthermore, existing solutions often lack adaptability to fluctuating workloads and may not efficiently handle heterogeneous VM configurations. Another constraint to consider is the potential need for substantial computational resources when making load balancing decisions, which could introduce additional processing demands on the cloud infrastructure. As cloud computing continues to evolve and experience increased demand for diverse applications, there is a need for more advanced and adaptable load balancing solutions that can effectively address these limitations and achieve comprehensive improvements in performance metrics. As cloud computing experiences increased demand for real-time applications, the existing infrastructure struggles to efficiently allocate incoming tasks to virtual machines (VMs). This allocation problem is particularly challenging because it falls under the category of NP-hard problems, making it computationally complex. To tackle this issue, the proposed method introduces a unique hybrid optimization approach that integrates particle swarm optimization (PSO), gravitational search algorithm (GSA), and fuzzy logic to efficiently allocate tasks to VMs. The goal is to improve the performance metrics, such as throughput, makespan, and execution time. The main objective of the proposed research is as follows:

1. To tackle the performance degradation caused by the overload on cloud servers due to an increase in task requests.
2. To explore load balancing as an effective approach for allocating incoming tasks to suitable virtual machines (VMs) based on their specific requirements.
3. To develop a cross model that combines the PSO for optimization, GSA to improve the search process, and fuzzy logic to create selection criteria for competent virtual machines.
4. To validate the efficacy of the anticipated approach by assessing metrics such as throughput, makespan, and execution time.

2. Literature Survey

In this segment, we explore existing schemes of load balancing. Here mainly we discuss optimization-based schemes that can handle the load dynamically. It is reported that the load balancing problem is an NP-hard problem. The current research scenario suggests that optimization problems play a substantial role in dealing with NP-Hard problems [14]. Currently, numerous schemes have been reported based on nature-inspired algorithms. Moreover, this article discussed the impact of optimization schemes on load-balancing problems. Ref. [15] deliberated on the issue of dynamic load balancing and its impact on a cloud computing system. This scheme adopts the osmosis theory from chemistry and develops an osmotic computing model. The main aim of this approach is to develop a hybrid metaheuristic approach by combining osmotic behaviour with a bio-inspired approach. The osmotic module helps to deploy the virtual machines. The artificial bee colony (ABC) and ant colony optimization (ACO) techniques handle dynamic environments efficiently. Based on these assumptions, a hybrid scheme is developed by using osmotic ABC and ant colony schemes. Ref. [16] developed a new load-balancing scheme by combining the firefly and improved PSO to deal with NP-hard problems in cloud-based load-balancing systems. The firefly optimization scheme is used to minimize the search space and improved PSO is used to detect the enhanced response. The improved PSO mitigates the distance error issue of PSO where it selects the global best-fit particle, which is near to the point to a line, thus, this combination helps to achieve the best solution for load balancing. Ref. [17] discussed that load-balancing algorithms distribute the loads over VMs equally which minimizes the power consumption for servers. Corresponding to the mechanism of this approach, the load and capacity of each VM are estimated and compared with a predefined threshold. If this shows that the load is exceeding the predefined threshold, then the load balancing scheme is activated. This scheme has 2 phases, first, it measures the requirement of the deciding factor for each virtual machine, and then it computes the selection factor. From all incoming tasks, it selects the best task based on the selection factor and allocates it to the most suitable VM. This approach is based on dragonfly optimization which is used to determine the optimal threshold. The optimal threshold value is obtained by using the dragonfly optimization. Ref. [18] developed a resource provisioning and load balancing approach to optimize the VM utilization and load distribution. This work presents a hybridized model by combining the heuristic and metaheuristic approach that helps to advance the overall performance of the system by minimizing the makespan and cost. The first approach considers ant colony optimization (ACO) with Predict Earliest Finish Time and another approach considers Heterogeneous Earliest Finish Time with ant colony optimization. Ref. [19] focused on long-term load balancing and developed a load-balancing resource clustering approach that deals with dynamic incoming tasks in the cloud system. Conventional optimization schemes suffer from a convergence issue. To address this issue, the current study employs the meta-heuristic Bat methodology, which enhances the convergence in resource clustering. Moreover, a dynamic task allocation approach is also developed to achieve improve the makespan and cost for the considered constraints. Ref. [20] discussed that task allocation is considered as an NP-hard problem that solves the load balancing issue by finding a suitable virtual machine with

minimum makespan and proper resource utilization. Mainly, this approach constructs an intelligent searching pattern where it identifies the best VM and assigns the tasks to the VM. To address these challenges, this research introduces a novel metaheuristic algorithm called the chaotic spider metaheuristic. This algorithm takes inspiration from social spider behavior and aims to effectively handle load-balancing problems in diverse virtual machine environments. The primary benefit of this model is that it expands the convergence performance and uses an intelligent scheme to obtain the optimal resources. Ref. [21] developed a particle swarm optimization-based strategy for task scheduling and incorporated the fuzzy logic strategy to enhance the performance of task allocation and load balancing. In the initial phase, the velocity updating process is modified and search capacity is enhanced by considering the roulette wheel selection technique. Further, mutation and crossover steps are applied to mitigate the local optima issue of PSO. Finally, a fuzzy inference system is applied to compute the fitness that considers task length, CPU speed, RAM size, and total execution time as input to the fuzzy systems. Ref. [22] discussed that the meta-heuristic approach is the most appropriate solution for the NP-hard problem, but these techniques should consider the dynamic situation in cloud computing. Based on this concept, the authors presented chicken swarm optimization (CSO) and Raven roosting optimization (RRO) to find the optimal solution for a given optimization problem. This scheme presents an ICDSF scheduling algorithm which is a combination of meta-heuristic approaches including an improved raven roosting optimization algorithm (IRRO) and the CSO algorithm. The CSO approach helps to improve the local and global search whereas IRRO is used to overcome the issue of premature convergence. Ref. [23] focused on minimizing energy consumption thus VM placement plays an important role. A few practices have been exhibited to minimize energy consumption and maximize load balancing, resource utilization, and robustness in the cloud system. The authors presented an energy efficient KnEA (EEKnEA) algorithm to accomplish these objectives. The functioning of this system is improved by incorporating the knee point-driven evolutionary algorithm (KnEA).

Considering the comprehensive literature survey on existing load balancing schemes, it is evident that the proposed research addresses critical gaps and challenges in the field. The surveyed literature showcases the prevalence of NP-hard problems in load balancing and highlights the effectiveness of metaheuristic and optimization-based approaches in dealing with these challenges. The incorporation of nature-inspired algorithms such as Artificial Bee Colony (ABC), Ant Colony Optimization (ACO), Firefly Optimization, and Particle Swarm Optimization (PSO) demonstrates the significance of leveraging bio-inspired strategies for dynamic load balancing. Moreover, the application of techniques like Fuzzy Logic, Osmotic Computing, and dynamic task allocation further highlights the adaptability required to handle real-time variations in cloud environments.

Relating these findings to the research objectives, it becomes apparent that the proposed hybrid optimization approach significantly contributes to each objective. Initially, in addressing performance degradation due to server overload, the integration of PSO, GSA, and fuzzy logic enables efficient task allocation, enhancing system responsiveness and reducing execution time. Secondly, the exploration of load balancing as an effective approach is evidenced by the comparison of numerous load balancing techniques in the literature, showing how optimization-based solutions are instrumental in achieving load distribution across VMs. Thirdly, the development of a cross model integrating PSO, GSA, and fuzzy logic resonates with the trend observed in the literature where innovative combinations of optimization algorithms and adaptive techniques yield improved solutions for load balancing. The synthesis of optimization algorithms, adaptive strategies, and real-time task allocation techniques reinforces the novel contribution of this research to the field.

3. Methodology

In this segment, our focus is on developing a new method for load balancing and task scheduling to obtain optimal resource allocation in cloud computing systems. The previous section describes the importance of metaheuristic optimization schemes. These practices are widely adopted in various applications. The PSO and genetic algorithm are known as the most promising techniques to solve optimization problems. However, achieving the required performance in terms of resource utilization, bandwidth utilization, and task completion remains a challenging task for huge cloud computing systems. In this work, we focus on these issues and present a novel approach by combining genetic algorithms and PSO. This approach mainly focuses on identifying the best processor to schedule the incoming tasks.

The methodology directly addresses the first objective by aiming to mitigate performance degradation caused by server overload due to an influx of task requests. By efficiently allocating tasks to suitable VMs, it prevents servers from becoming overwhelmed and guarantees that computational resources are optimally utilized. It incorporates advanced optimization techniques, namely Particle Swarm Optimization (PSO) and Gravitational Search Algorithm (GSA), along with the use of fuzzy logic to create selection criteria. This combination of methods enables the dynamic allocation of tasks to the most appropriate VMs, effectively achieving load balancing. By integrating PSO for optimization, GSA to boost the search process, and fuzzy logic for VM selection, it creates a robust and adaptable framework. This framework streamlines the process of updating particle velocity and position while providing intelligent decision-making for task allocation. The proposed methodology assesses its performance through key metrics, including throughput, makespan, and execution time. Demonstrating substantial improvements in these metrics compared to existing techniques, it substantiates the effectiveness of the approach and its capability to address the challenges of load balancing in cloud computing.

3.1 Problem Formulation

Mainly, the task scheduling process focuses on scheduling, distributing, and allocating the diverse types of tasks to different suitable virtual machines effectively so that the execution time can be minimized, and resources can be utilized efficiently. We consider a cloud computing system as $\mathcal{C}$ which contains several physical machines as N_{pm} and each machine contains several virtual machines as N_{vm}. A cloud system can be represented as shown in (1).

$$\mathcal{C} = \left[pm_1, pm_2, \ldots\ldots, pm_i, \ldots pm_{N_{pm}}\right] \tag{1}$$

Where pm_i denotes physical machines present in the cloud system which has several virtual machines which is denoted as (2)

$$pm_i = \left[vm_1, vm_2, \ldots\ldots, vm_k, \ldots vm_{N_{vm}}\right] \tag{2}$$

vm_k denotes the k^{th} number of virtual machines in the cloud model. The virtual machine follows the configuration as (3):

$$vm_k = [IDV_k, MIPS_k] \tag{3}$$

The VM configuration contains ID number of virtual machine and instruction processing speed which decides the capacity of virtual machine. Similarly, the task set is defined as (4):

$$\mathbb{T} = \left[T_1, T_2, \ldots T_l, \ldots, T_{N_{task}}\right] \tag{4}$$

Where N_{task} denotes the total l number of tasks are submitted by user to cloud service provider. The task configuration consists of task ID, task length, and expected completion time and task priority. The l^{th} task configuration can be expressed as (5):

$$T_l = [IDT_l, \ldots, length_l, ECT_l, \ldots, PI_l] \tag{5}$$

IDT_l represents the task ID, $length_l$ denotes the task length, PI_l is the task priority and ECT_l denotes the expected time to complete the task which is in the matrix form as $N_{task} \times N_{vm}$. This matrix can be represented as (6):

$$ECT = \begin{bmatrix} ECT_{1,1} & ECT_{1,2} & ECT_{1,3} & \ldots & ECT_{1,N_{vm}} \\ ECT_{2,1} & ECT_{2,2} & ECT_{2,3} & \ldots & ECT_{2,N_{vm}} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ ECT_{N_{task},1} & ECT_{N_{task},2} & ECT_{N_{task},3} & \ldots & ECT_{N_{task},N_{vm}} \end{bmatrix} \tag{6}$$

In this work, our main objective is to minimize the makespan by scheduling tasks to the suitable VMs such as (7):

$$ECT_{lk} = \frac{length_l}{MIPS_k}, k = 1,2,3, \ldots N_{task}, l = 1,2,3, \ldots N_{task} \tag{7}$$

Here, ECT_{lk} denotes the total required time to execute l^{th} on k^{th} where N_{vm} are the total number of VMs and N_{task} denotes the total number of tasks.

In this study, we assume that each VM has a different execution time $\mathcal{T}_i$, Memory $\mathcal{M}_i$, energy consumption E_i , and cost C. Based on these parameters, we design an objective function by considering cost and energy to allocate the task for the virtual machine. This objective function can be expressed as depicted in (8).

$$Objective\ function = \sum_{i=1}^{vm}(C_i, E_i) \tag{8}$$

Our aim to minimize this objective function for each virtual machine so that overall energy consumption can be minimized. To achieve this, we present a novel hybrid meta-heuristic approach by combining three well known approaches which are fuzzy logic, gravitational search algorithm and particle swarm optimization. The PSO helps to find the optimal solution which is further handled by GSA to improve the outcome of optimization later fuzzy logic is used to assign the rules for allocation.

3.2 Particle Swarm Optimization (PSO)

PSO draws its roots from the collective actions of birds flocking or fish schooling, presenting a population-centered optimization approach. It consists of a group of particles (potential solutions) that iteratively adjust their

positions in a multi-dimensional search space to find the optimal solution. Particles update their positions based on their previous best-known position and the best-known position within the entire swarm. PSO is used to optimize the allocation of tasks to virtual machines. Each particle represents a potential task allocation, and the swarm collectively explores the solution space to find the best task-to-VM mapping. PSO helps balance the workload dynamically by continuously adjusting the task allocation based on system conditions and task requirements.

PSO stands out as a highly promising method for tackling optimization challenges. In this approach, the optimization problem's search space is populated with particles. These particles search for the best position initially which can be near to the optimal solution or best among all particles. Each particle plays a critical role in finding the optimal solution by changing its position and velocity to find the improved position in the search space. The velocity v and position x for each particle can be obtained as shown in (9).

$$v_d^i(t+1) = \left(w \times v_i^d(t)\right) + \left(c_1 \times rand_i \times \left(pbest_d^i(t) - x_d^i(t)\right)\right) + \left(c_2 \times rand_i \times \left(gbest_d^i(t) - x_d^i(t)\right)\right)$$
$$x_d^i(t+1) = x_d^i(t) + v_d^i(t+1) \quad (9)$$

where v_i^d is the velocity of particle i in d direction, w is the inertia weight, $rand_i$ is the random number between 1 to 2, x_d^i is the current position of particle, $pbest$ is the local best position, and $gbest$ is the global best position of particle. This process is repeated until the objective function conditions are satisfied. However, for multi-objective function this approach may provide immature solution due to convergence hence we incorporate gravitational search algorithm to improve the local and global search algorithm.

3.3 Gravitational Search Algorithm (GSA)

This approach works based on Newton's law of gravity. It is a computational intelligence approach that has found applications in various domains, including dynamic cloud load balancing. According to this scheme, each particle is treated as an object which is later used to compute the fitness based on the mass of each object. According to the law of gravity, the two particles in any plane attract each other by a specific gravitational force between these objects. This gravitational force is found to be directly proportionate to the product of masses of two objects. On the contrary, this force is always inversely proportionate to the distance between particles. GSA's application in dynamic cloud load balancing leverages its ability to efficiently allocate tasks to virtual machines based on changing conditions. In this context, the fitness of particles represents the suitability of a virtual machine to handle incoming tasks. As the workload varies, GSA dynamically adjusts the allocation of tasks by computing gravitational forces between VMs, leading to an optimal task-to-VM assignment. This adaptability is crucial in cloud environments where workloads fluctuate, ensuring resource utilization efficiency and minimizing response times. It can also address the challenges of task allocation, resource optimization, and performance enhancement in cloud computing environments where task requests can vary significantly. By simulating gravitational interactions between VMs, GSA enables intelligent and real-time decisions regarding task distribution, leading to improved resource utilization, reduced energy consumption, and enhanced system performance.

Let us consider that total population is represented as $X_i = \left(x_i^1, \ldots., x_i^d, \ldots, x_i^{ND}\right)$ for $i = 1,2,3, \ldots. NP$.

For any scenario the force between particle i and j can be expressed as (10):

$$F_{ij}^d(t) = G \times \frac{M_i(t) \times M_j(t)}{R_{ij}(t) + \epsilon}\left(x_j^d(t) - x_i^d(t)\right) \quad (10)$$

Where $R_{ij}(t) = \left\|X_i(t), X_j(t)\right\|_2$, $m_i(t) = \frac{fit(t) - \max_{j\in\{1,\ldots,N\}} fit_j(t)}{\min_{j\in\{1,\ldots,N\}} fit_j(t) - \max_{j\in\{1,\ldots,N\}} fit_j(t)}$, $M_i(t) = \frac{m_i(t)}{\sum_{j=1, j\neq i}^{NP} rand_j F_{ij}^d(t)}$ and $a_i^d(t) = \frac{F_i^d(t)}{M_i(t)}$

Based on this approach, we consider the updated position from PSO and compute the force between two articles using above mentioned equation. Thus, the position and velocity of particle can be updated as (11):

$$v_{ij}^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t)$$
$$X_i^d(t+1) = X_i^d(t) + X_i^d(t+1) \quad (11)$$

3.4 Combining Fuzzy logic with a hybrid of PSO-GSA

As mentioned previously, the main idea of the proposed objective function is to consider energy and cost as the main components that have a substantial impact on system performance. First, we consider the overall migration cost as depicted in (12).

$$M_c = \frac{F_M + F_C}{2} \quad (12)$$

Here F_M denotes the movement information in VMs given as $F_M = 1/P_M \left[\sum_{i=1}^{VM_i}\left(\frac{number\ of\ movements}{Total\ VM}\right)\right]$ and F_C denotes the cost parameters for physical machine which is expressed as $F_C = \sum_{i=1}^{VM_i}\left(\frac{cost\ to\ run \times task\ memory}{VM \times PM}\right)$. Similarly, we compute the energy utilization to satisfy the objective function as (13):

$$E_i = \frac{1}{PM \times VM}\left[\sum_{i=1}^{PM}\sum_{j=1}^{VM} E_{max}A_{ij} + \left(i - A_{ij}\right)\delta_{ij}E_{max}\right] \tag{13}$$

where $\delta_{ij} = \frac{1}{2}\left[\left(\frac{CPU\ utlizaiton_{ij}}{CPU_{ij}}\right) + \left(\frac{memory\ utlizaiton_{ij}}{memory_{ij}}\right)\right]$

Corresponding to the concept of PSO, we measure the fitness of each particle, global best, and update the position and velocity of each particle in each iteration. To efficiently map the task to VM and to improve the impact of the fitness function we use the fuzzy logic approach which is considered a promising solution for uncertain scenarios. Fuzzy logic is presented as $\mu\tilde{A}(x)$ which has the value as [0,1]. The fuzzy logic works based on its rules and membership functions such as if $\mu\tilde{A}(x) = 0$ then x is not considered a member of the targeted set and if $\mu\tilde{A}(x) = 1$ then x is considered a member of the targeted set. In this research, we consider a triangular membership function that has 3 solutions as $(m1, m2, m3)$ as (minimum possible value, most possible value, and biggest possible value), respectively. In this context, our objective is to discover the most appropriate and optimal solution for the provided optimization challenge. To solve this, we consider a decision matrix as shown in (14) and (15).

$$DM = \begin{bmatrix} E_{T11} & T_{T11} & C_{T11} \\ E_{T21} & T_{T21} & C_{T21} \\ \vdots & \vdots & \vdots \\ E_{TK1} & T_{TK1} & C_{TK1} \end{bmatrix} \tag{14}$$

Where E denotes the energy, T is the time factor and C is the migration cost.

$$DM = \begin{bmatrix} E_{T11} & T_{T11} & C_{T11} \\ E_{T21} & T_{T21} & C_{T21} \\ \vdots & \vdots & \vdots \\ E_{TK1} & T_{TK1} & C_{TK1} \end{bmatrix} \tag{15}$$

Further, we normalize the decision matrix as $ND_m = \frac{S_{mn}}{\sqrt{\sum S_{mn}^2}}$. Further, we need to compute the impact of initial solution for given membership function. In this work, we have considered three membership functions as $m1, m2, m3$. The positive impact shows the closeness to the optimal decision whereas negative impact represents the poor solution for give problem. These impacts can be represented as $P^+ = (S_1^+, S_2^+, \ldots, S_m^+)$ and $N = (S_1^-, S_2^-, \ldots, S_m^-)$ as positive and negative impacts respectively. With the help of these solutions, we obtain the separation between positive and negative solutions as:

$$\begin{aligned} \mathbb{S}^+ &= \sqrt{\sum_{n=1}^{3}(S_{mn} - S_n^+)^2} \\ \mathbb{S}^- &= \sqrt{\sum_{n=1}^{3}(S_{mn} - S_n^-)^2} \end{aligned} \tag{16}$$

Then the closeness to the actual solution is measured as C= $\frac{\mathbb{S}^-}{S^+ + \mathbb{S}^-}$

After obtaining this solution, we apply a hybrid of the PSO and GSA approach to update the position and velocity of particles by considering their mass and gravitational forces between particles. With the help of this approach, we allocate the incoming to the VM which achieves the best fitness. This allocation minimizes the energy consumption and migration cost by efficiently scheduling the task in such a way that the task is accomplished in the given time frame.

4. Results

In this segment, we present the comparative examination of the suggested hybrid metaheuristic approach PGF-LB where the obtained performance of the proposed approach is compared with various state-of-art techniques such as First Come First Service (FCFS), Genetic Algorithm (GA), Round Robin (RR), and Short Jobs First (SJF). The performance is compared in terms of execution time, resource utilization makespan, and throughput. For this experiment, we have considered different types of tasks and VMs. Based on task numbers and memory, we divide the tasks and VMs as small, medium, large, and extra-large, respectively. Table 1. and Table 2. shown below highlight the task and VM configuration scenario.

Table 1. Task configuration

Types of incoming task	Number of incoming tasks	Size of incoming tasks
Small	100-200	30000-50000
Medium	400-500	50000-70000
Large	600-700	70000-100000
Extra large	800-1000	100000-200000

As discussed before, we have divided the task into four categories small, medium, large, and extra-large for each category we have a different set of tasks. The task allocation range is 100-200, 400-500, 600-700, and 800-1000, respectively. Similarly, Table 2. shows the VM configuration for each category.

Table 2. VM configuration

Type of task	Number of tasks	Size of tasks
Small	10000	5
Medium	20000	10
Large	25000	15
Extra large	35000	20

First, we compared the performance in terms of average execution time and compared the performance with other schemes. Table 3. shows the comparative analysis in terms of execution time.

Table 3. Comparative Analysis of Execution Time

Method	Avg. turnaround time	Avg. response time
FIMPSO [17]	21.09	13.58
FF-IPSO	22.13	15.21
GA	26.57	20.30
RR	41.98	30.50
IPSO	57.74	49.23
FCFS	41.87	30.84
Firefly	55.54	48.87
SJF	41.56	30.24
Proposed method	18.20	11.26

This comparison exhibits that the proposed method requires average load, turnaround time, and response time of 0.168, 18.20ms, and 11.26ms, respectively, for all configurations of tasks and VMs. Further, we gauge the performance in terms of CPU utilization for varied types of task configurations. Fig.1 shows the comparative analysis for this scenario.

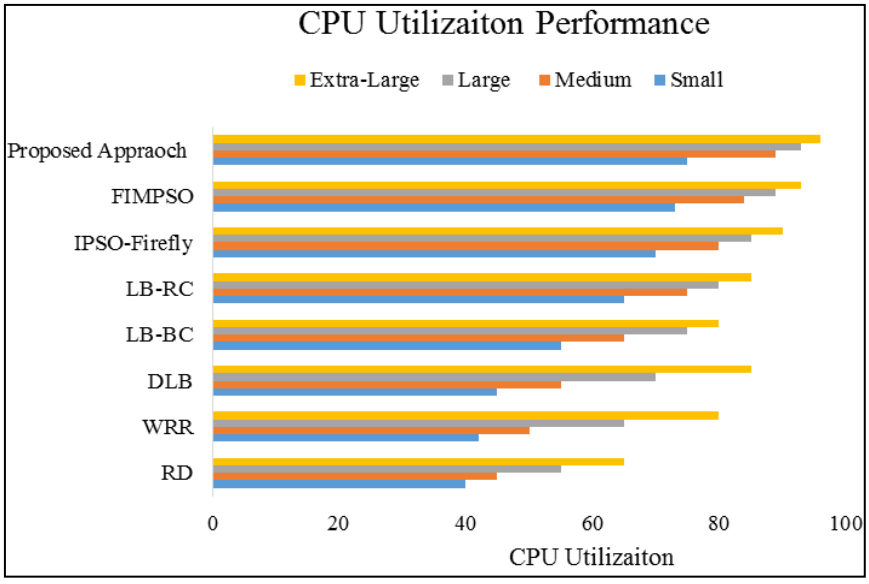

Fig.1. CPU Utilization Performance

For this scenario, we measured the average performance for different types of tasks. The average performance is obtained as 51.25, 59.25, 63.75, 68.75, 76.25, 81.25, 84.75, and 88.25 using RD, WRR, DLB, LB-BC, LB-RC, IPSO Firefly, FIMPSO, and Proposed Approach, respectively. The proposed approach shows a significant improvement in utilization performance. The comparative analysis reveals that the performance of the suggested method is improved by 72.19%, 48.94%, 38.43%, 28.36%, 15.73%, 8.61%, and 4.12 when compared with RD, WRR, DLB, LB-BC, LB-RC, IPSO Firefly, and FIMPSO. Similarly, we estimate the performance in terms of makespan for a different type of task configuration. Fig.2 portrays the comparative performance.

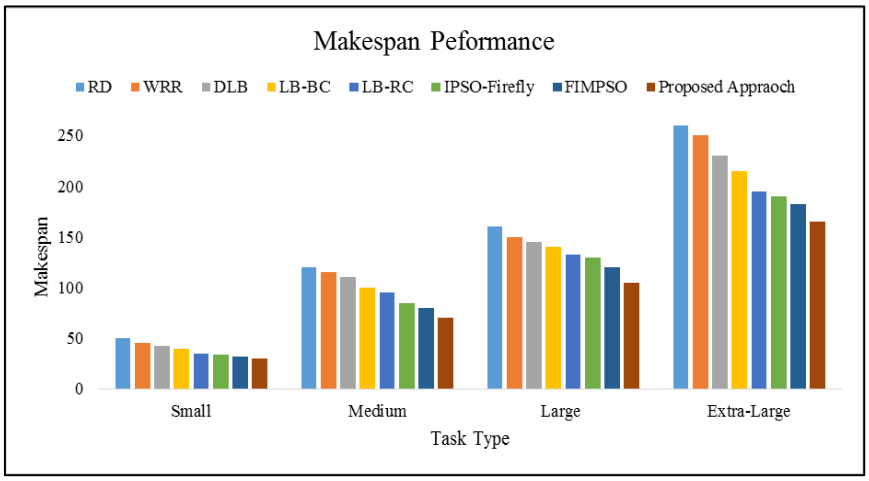

Fig.2. Makespan Performance

This comparative investigation reveals that the average makespan for different types of tasks is obtained as 147.5ms, 140ms, 131.75ms, 123.75ms, 114.25ms, 109.75ms, 103.5ms, and 92.5ms using RD, WRR, DLB, LB-BC, LB-RC, IPSO-Firefly, FIMPSO, and Proposed approach. Finally, we measure the throughput performance for the same experimental setup. The obtained performance is illustrated in Fig.3.

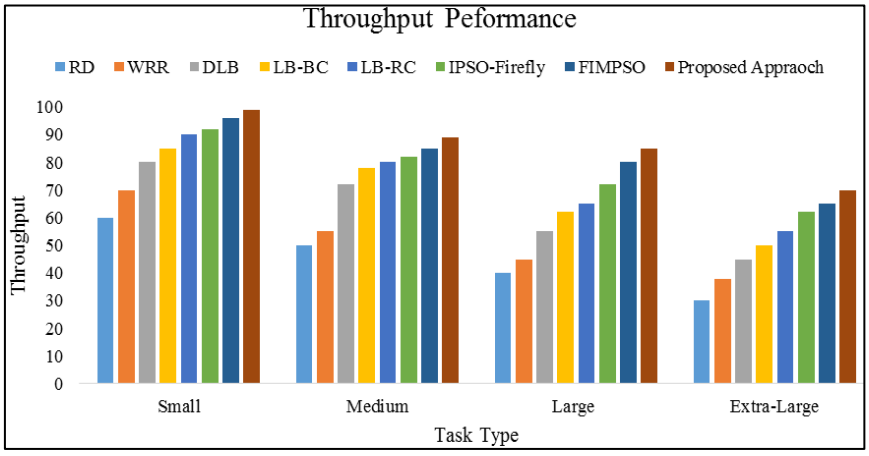

Fig.3. Throughput Performance

The average throughput performance is obtained as 45, 52, 63, 68.75, 72.5, 77, 81.5, and 85.75 using RD, WRR, DLB, LB-BC, LB-RC, IPSO-Firefly, FIMPSO, and Proposed Approach, respectively. The performance of the intended method is improved by 90.5%, 64.9%, 36.11%, 24.72%, 18.27%, 11.36%, and 5.21 in comparison to the existing techniques.

5. Discussion

The presented results offer a comprehensive comparative analysis of the proposed hybrid metaheuristic approach, PGF-LB, against several established load balancing techniques. The performance evaluation encompasses critical metrics such as execution time, resource utilization, makespan, and throughput across various task and virtual machine configurations. Notably, the proposed method demonstrates impressive results, achieving an average load, turnaround time, and response time of 0.168, 18.20ms, and 11.26ms, respectively, for all task and VM configurations. The most significant improvement, however, is observed in CPU utilization across different task configurations, with the

proposed approach outperforming other techniques by substantial margins, ranging from 4.12% to a remarkable 72.19%. Furthermore, the proposed approach substantially reduces makespan for different types of tasks compared to traditional methods, demonstrating its efficiency in task scheduling. The average throughput performance also showcases remarkable improvements, with a 5.21% to 90.5% enhancement when compared to existing techniques. These results collectively underscore the effectiveness of the PGF-LB approach in addressing the challenges of dynamic load balancing in cloud computing, providing substantial improvements in performance metrics, and making a significant contribution to the field.

The integration of PSO, GSA, and fuzzy logic within the proposed hybrid optimization approach opens the possibility of introducing novel features that can significantly improve the efficiency and efficacy of task allocation in dynamic cloud environments. One such feature could involve the introduction of a dynamic priority adjustment mechanism. This mechanism would allow the hybrid system to dynamically adjust the priority of tasks based on their characteristics, urgency, and the current state of the virtual machines. For instance, high-priority tasks could be assigned a higher weight in the fuzzy logic rules and could be given more favorable positions in the particle swarm optimization process. This would ensure that critical tasks receive a prompt and optimal allocation, thereby reducing their execution time and enhancing overall system responsiveness. Additionally, such a feature would allow the system to adapt to sudden changes in workload or task requirements, providing a more versatile and responsive load balancing solution. This dynamic priority adjustment mechanism aligns well with the proposed approach's objective of improving performance metrics like throughput, makespan, and execution time by tailoring task allocation to the definite needs of the cloud environment and workload scenarios.

6. Conclusion

The surge in demand for cloud computing in recent years has brought about numerous benefits, including versatility, cost-effectiveness, and energy efficiency. However, the accompanying surge in task requests has exposed cloud servers to the risk of performance degradation, necessitating the development of effective load balancing solutions. To tackle this challenge, this research has introduced a novel hybrid optimization approach, leveraging the strengths of particle swarm optimization (PSO), gravitational search algorithm (GSA), and fuzzy logic. The integration of these techniques has yielded remarkable results, significantly enhancing execution time, CPU utilization, makespan, and throughput. This improvement, ranging from 5.21% to an impressive 90.5% compared to existing techniques, underscores the efficacy of the proposed method in addressing the complex problem of load balancing in cloud computing. This work advances the field by presenting a comprehensive solution that combines well-established metaheuristic algorithms with the adaptability of fuzzy logic, offering a versatile approach to dynamic load balancing. It contributes a valuable tool for cloud service providers (CSPs) to optimize resource allocation, mitigate server overload, and improve overall system performance. Beyond the current state of knowledge, this research suggests new possibilities for enhancing cloud computing efficiency through advanced load balancing techniques. Future applications of this work could include its integration into cloud service platforms, providing real-world benefits to CSPs and their clients. Additionally, extensions of this research might explore further refinements of the hybrid optimization approach, accommodating even more complex cloud computing scenarios and challenges. Ultimately, this study not only addresses a pressing issue in cloud computing but also opens the door to further advancements in optimizing cloud infrastructure for the benefit of users and providers alike.

References

[1] Rodriguez, M. A., & Buyya, R. (2018). Scheduling dynamic workloads in multi-tenant scientific workflow as a service platforms. *Future Generation Computer Systems*, *79*, 739-750.

[2] Afzal, S., & Kavitha, G. (2019). Load balancing in cloud computing–A hierarchical taxonomical classification. *Journal of Cloud Computing*, *8*(1), 1-24.

[3] Havanje, N. S., Kumar, K. R. A., Shenoy, S. N., Rao, A. S., & Thimmappayya, R. K. (2022). Secure and Reliable Data Access Control Mechanism in Multi-Cloud Environment with Inter-Server Communication Security. *Suranaree Journal of Science and Technology*, *29(3).*

[4] Olokunde, T., Misra, S., & Adewumi, A. (2017, October). Quality model for evaluating platform as a service in cloud computing. In *International Conference on Information and Software Technologies* (pp. 280-291). Springer, Cham.

[5] Hussein, M. K., Mousa, M. H., & Alqarni, M. A. (2019). A placement architecture for a container as a service (CaaS) in a cloud environment. *Journal of Cloud Computing*, *8*(1), 1-15.

[6] Khan, F. A., Jamjoom, M., Ahmad, A., & Asif, M. (2022). An analytic study of architecture, security, privacy, query processing, and performance evaluation of database ‐ as ‐ a ‐ service. *Transactions on emerging telecommunications technologies*, *33*(2), e3814.

[7] Navimipour, N. J., Rahmani, A. M., Navin, A. H., & Hosseinzadeh, M. (2015). Expert Cloud: A Cloud-based framework to share the knowledge and skills of human resources. *Computers in Human Behavior*, *46*, 57-74.

[8] Gaikwad, C., Churi, B., Patil, K., & Tatwadarshi, P. N. (2017, March). Providing storage as a service on cloud using OpenStack. In *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)* (pp. 1-4). IEEE.

[9] Zargar, S. T., Takabi, H., & Iyer, J. (2019). Security-as-a-Service (SECaaS) in the cloud. Security, Privacy, and Digital Forensics in the Cloud, Chap. 9.
[10] Tung, Y. H., Lin, C. C., & Shan, H. L. (2014, April). Test as a Service: A framework for Web security TaaS service in cloud environment. In *2014 IEEE 8th International Symposium on Service Oriented System Engineering* (pp. 212-217). IEEE.
[11] Milani, A. S., & Navimipour, N. J. (2016). Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends. *Journal of Network and Computer Applications*, *71*, 86-98.
[12] Tundo, A., Mobilio, M., Orrù, M., Riganelli, O., Guzmàn, M., & Mariani, L. (2019, August). Varys: An agnostic model-driven monitoring-as-a-service framework for the cloud. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 1085-1089).
[13] Ghomi, E. J., Rahmani, A. M., & Qader, N. N. (2017). Load-balancing algorithms in cloud computing: A survey. *Journal of Network and Computer Applications*, *88*, 50-71.
[14] Milan, S. T., Rajabion, L., Ranjbar, H., & Navimipour, N. J. (2019). Nature inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments. *Computers & Operations Research*, *110*, 159-187.
[15] Gamal, M., Rizk, R., Mahdi, H., & Elnaghi, B. E. (2019). Osmotic bio-inspired load balancing algorithm in cloud computing. *IEEE Access*, *7*, 42735-42744.
[16] Devaraj, A. F. S., Elhoseny, M., Dhanasekaran, S., Lydia, E. L., & Shankar, K. (2020). Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments. *Journal of Parallel and Distributed Computing*, *142*, 36-45.
[17] Polepally, V., & Shahu Chatrapati, K. (2019). Dragonfly optimization and constraint measure-based load balancing in cloud computing. *Cluster Computing*, *22*(1), 1099-1111.
[18] Kaur, A., & Kaur, B. (2019). Load balancing optimization based on hybrid Heuristic-Metaheuristic techniques in cloud environment. *Journal of King Saud University-Computer and Information Sciences*.
[19] Adhikari, M., Nandy, S., & Amgoth, T. (2019). Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud. *Journal of Network and Computer Applications*, *128*, 64-77.
[20] Arul Xavier, V. M., & Annadurai, S. (2019). Chaotic social spider algorithm for load balance aware task scheduling in cloud computing. *Cluster Computing*, *22*(1), 287-297.
[21] Mansouri, N., Zade, B. M. H., & Javidi, M. M. (2019). Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory. *Computers & Industrial Engineering*, *130*, 597-633.
[22] Torabi, S., & Safi-Esfahani, F. (2018). A dynamic task scheduling framework based on chicken swarm and improved raven roosting optimization methods in cloud computing. *The Journal of Supercomputing*, *74*(6), 2581-2626.
[23] Ye, X., Yin, Y., & Lan, L. (2017). Energy-efficient many-objective virtual machine placement optimization in a cloud computing environment. *IEEE access*, *5*, 16006-16020.
[24] Li, J., & Dong, N. (2017, December). Gravitational search algorithm with a new technique. In *2017 13th International Conference on Computational Intelligence and Security (CIS)* (pp. 516-519). IEEE.

Authors' Profiles

Rajgopal K T received his B. E degree in Information Technology from VTU, Belgaum, M.Tech degree in Computer Network & Engineering from VTU, Belgaum, and currently pursuing his Ph.D. in Computer Science & Engineering at Nitte University, Mangalore. His major research interest is in the fields of Cloud Computing Machine Learning, and Computer Networks. He has 10 years of teaching experience and currently, he is working as an Assistant Professor in the Department of Computer Science & Engineering at Canara Engineering College, Mangalore. He has published several articles in various national and international conferences and journals. He is a lifetime member of ISTE.

Abhishek S. Rao received his B. E degree in Information Science & Engineering from Canara Engineering College, Mangalore, M.Tech degree in Computer Science and Engineering from NMAM Institute of Technology, VTU, Belagavi and his M.B.A degree in Operations Management from MIT, Pune. Currently, he is pursuing his Ph.D. in healthcare management at Visvesvaraya Technological University. His major research interest is in the fields of Cloud Computing, Machine Learning, Deep Learning, Computer Vision, and Image Processing. He has received a research grant of Rs. 3 Lakh from VGST under the RGS/F scheme. He has 2 years of industrial experience and 11 years of teaching experience. At present, he is working as Asst. Professor (Senior Grade) in the Department of Information Science & Engineering at NMAM Institute of Technology, Nitte. He has published 25 articles in various national and international conferences and journals. He is a lifetime member of ISTE and IAENG.

Ramaprasad Poojary received his B. E degree in Electronics and Communication Engineering, M.Tech degree in Digital Electronics & Communication, and Ph.D. in Dental Image Registration. His major research interest is in the fields of Machine Learning, Cloud Computing, Image Processing, and Computer Vision.

He has 20 years of teaching experience and 6 years of research experience. At present, he is working as an Associate Professor at Manipal Academy of Higher Education Dubai Campus in Dubai, UAE. He has published several articles in various national and international conferences and journals. He is a lifetime member of IEEE.

Deepak D received his B. E degree in Computer Science & Engineering from Sri Taralabalu Jagadguru Institute of Technology, Ranebennur, Karnataka - Visvesvaraya Technological University, Belagavi, M.Tech degree in Software Engineering, AMC Engineering College, Bengaluru, Karnataka- Visvesvaraya Technological University, Belagavi, and currently pursuing his Ph.D. in Computer Science & Engineering from VTU, Belgavi. His major research interest is in the fields of Machine Learning, Cloud Computing, Deep Learning. He has 11 years of teaching experience and currently, he is working as an Asst. Professor (Senior Grade) in the Department of Information Science & Engineering at NMAM Institute of Technology, Nitte. He has published several articles in various national and international conferences and journals. He is a lifetime member of ISTE.

How to cite this paper: Rajgopal K T, Abhishek S. Rao, Ramaprasad Poojary, Deepak D, "Dynamic Load Balancing in Cloud Computing: A Convergence of PSO, GSA, and Fuzzy Logic within a Hybridized Metaheuristic Framework", International Journal of Modern Education and Computer Science(IJMECS), Vol.15, No.6, pp. 44-55, 2023. DOI:10.5815/ijmecs.2023.06.04