HGCSO: Energy Efficient Multi-objective Task Scheduling in Cloud-Fog Environment

Santhosh Kumar Medishetti[1(✉)], Vamsheedhar Reddy Pillareddy[2], Bushra Muneeb[3], Sudha Rani Palakuri[4], Uma Maheshwari Garela[4], Rakesh Kumar Donthi[5], G. Soma Sekhar[6], Ganesh Reddy Karri[1], Baji Babu Indurthi[1], and K. Vamshi Krishna[1]

[1] VIT-AP University, Amaravathi, A.P., India
{santhosh.21phd7113,indurthi.20bcn7099,
vamshikrishna.20phd7088}@vitap.ac.in
[2] Sridevi Women's Engineering College, Hyderabad, T.S., India
[3] Princeton Institute of Engineering and Technology for Women, Hyderabad, T.S., India
[4] Sreyas Institute of Engineering and Technology, Hyderabad, T.S., India
[5] GITAM Deemed to be University, Hyderabad, T.S., India
rdonthi@gitam.edu
[6] Vardhaman College of Engineering, Hyderabad, T.S., India

Abstract. In the ever-evolving realm of cloud-fog computing, effective Task Scheduling (TS) plays a crucial role in maximizing resource utilization and enhancing overall system performance. This paper introduces a novel approach to TS using a hybrid algorithm that combines Genetic Algorithms (GA) and Cat Swarm Optimization (CSO). The proposed Hybrid Genetic Cat Swarm Optimization (HGCSO) algorithm harnesses the strengths of GA for global exploration and CSO for its unique swarm-based search mechanism. The integration of these algorithms aims to address the complexities of cloud-fog computing environments, characterized by heterogeneous resources and varying task requirements. The TS problem is formulated as an optimization challenge, incorporating objectives such as reducing makespan, response time, and energy consumption. The HGCSO algorithm iteratively refines solutions by employing genetic operators for diversity and swarm-based mechanisms for local refinement. Thorough simulations have been carried out to assess the performance of the algorithm, juxtaposing it against conventional scheduling algorithms as well as standalone Genetic Algorithm (GA) and Cuckoo Search Optimization (CSO) approaches. The outcomes reveal that the suggested HGCSO algorithm surpasses its counterparts, showcasing improvements in makespan time, response time, and a notable reduction in energy usage by 22%, 18%, and 28%, respectively. This research provides significant insights into the progress of bio-inspired algorithms in tackling the complex challenges of Task Scheduling (TS) within contemporary computing paradigms.

Keywords: cloud-fog computing · efficient task scheduling · resource utilization · Genetic Algorithms · Cat Swarm Optimization

O. Castillo et al. (Eds.): PerSOM 2023, LNICST 517, pp. 16–38, 2024.
https://doi.org/10.1007/978-3-031-66044-3_2

1 Introduction

In the rapidly evolving landscape of Task Scheduling (TS) in cloud-fog computing (CFC) the effective orchestration of computational tasks is paramount for optimizing resource utilization and enhancing overall system performance [1]. TS is a critical aspect of this paradigm, involves judiciously assigning computational tasks to available resources, considering factors such as response time, energy consumption, and load balancing [2]. Addressing the intricacies of these dynamic and heterogeneous environments requires innovative approaches that can adapt to varying workloads and resource availabilities.

The introduction to Cloud Fog Computing (CFC) in the realm of TS sets the stage for understanding the dynamic and distributed computing environment that shapes modern computational landscapes [3]. CFC represents an innovative paradigm that extends the capabilities of CC by introducing edge or fog nodes closer to end-users and devices. This hierarchical architecture, comprising cloud data centers, fog nodes, and end-user devices, aims to address the challenges of latency-sensitive applications, varying workloads, and the need for efficient resource utilization.

In the realm of task scheduling, the introduction of CFC brings forth unique challenges and opportunities. The distribution of resources across cloud and fog nodes necessitates the development of sophisticated scheduling algorithms capable of optimizing task allocation in a heterogeneous and dynamic environment [4]. The proximity of fog nodes to end-users enhances the potential for low-latency processing, making CFC particularly suitable for applications with stringent performance requirements. The introduction sets the context for exploring how task scheduling in CFC environments is a critical aspect of ensuring optimal performance, responsiveness, and resource utilization across the distributed infrastructure.

Moreover, the introduction delves into the key characteristics of CFC that influence task scheduling strategies. These may include the need for adaptability to dynamic workloads, energy-efficient resource allocation, and the seamless integration of CFC resources. As CFC continues to gain prominence in various industries, the introduction to task scheduling within this paradigm emphasizes the importance of developing novel approaches and algorithms that cater to the intricacies of this emerging computing paradigm. It serves as a call to explore innovative solutions to enhance the efficiency of task scheduling in the evolving landscape of Cloud Fog Computing.

Genetic Algorithms (GAs) constitute a potent category of optimization algorithms inspired by principles derived from natural selection and genetics. Originating in the 1960s through the work of John Holland, genetic algorithms have gained widespread acclaim for their efficacy in systematically exploring and optimizing intricate solution spaces across diverse domains [5]. Rooted in the evolutionary processes observed in nature, genetic algorithms model populations of individuals that evolve over successive generations, employing mechanisms such as selection, crossover, and mutation.

At its essence, a genetic algorithm functions by treating potential solutions as individual chromosomes that encode potential resolutions to a given problem. Employing a process akin to natural selection, individuals with advantageous traits (fitness) are more apt to be chosen for reproduction, aligning with the fundamental survival-of-the-fittest principle. The crossover operation involves the exchange of genetic information among

selected individuals, resulting in new offspring with amalgamated characteristics. Furthermore, the mutation mechanism introduces random alterations to the genetic material, infusing diversity into the population. Genetic algorithms find application in a myriad of fields, including optimization, machine learning, scheduling, and parameter tuning, among others. Their versatility and effectiveness stem from their ability to navigate large and complex solution spaces, adapt to dynamic environments, and discover optimal and its closer to the near optimal solutions to the complex problems.

Cat Swarm Optimization (CSO) algorithm is a bio-inspired optimization technique that draws inspiration from the collective behavior and hunting strategies of cats. Introduced as a heuristic optimization algorithm, CSO leverages the inherent swarming and searching behaviors observed in feline species to efficiently explore and exploit solution spaces [6]. The algorithm was proposed as an alternative to conventional optimization approaches, aiming to overcome challenges posed by complex and dynamic problem domains. Inspired by the coordinated movements and communication observed in cat colonies, CSO operates on the principle of collaboration and information sharing among individuals [7]. Each potential solution to the optimization problem is represented as a cat, and the algorithm navigates the search space by imitating the hunting behaviors of cats, such as stalking prey, avoiding obstacles, and communicating with fellow cats [8].

CSO's appeal lies in its simplicity, adaptability to various problem types, and capacity to find optimal and its closer to the near optimal solutions in complex and dynamic environments. The algorithm's efficiency arises from the synergistic cooperation among individual cats, enabling it to navigate large solution spaces and discover promising regions efficiently. As a relatively recent addition to the family of swarm intelligence algorithms, CSO has found applications in diverse fields, including optimization problems, image processing, data clustering, and machine learning. This introduction lays the foundation for understanding the underlying principles of the Cat Swarm Optimization algorithm, setting the stage for further exploration into its mechanisms, applications, and real-world impact. Subsequent sections will delve into the intricacies of CSO, exploring its components, methodologies, and showcasing instances where it has demonstrated its efficacy in solving complex optimization challenges.

The objective of this study is to transform TS methodologies in CFC environments by introducing an innovative Hybrid Genetic Cat Swarm Optimization (HGCSO) algorithm. The central goal is to elevate the effectiveness of resource allocation by harnessing the combined strengths of genetic algorithms and cat swarm optimization. The overarching aim is to minimize makespan, optimize response time, and achieve load balancing across fog and cloud resources, ensuring a responsive and sustainable computing environment.

The particular objectives of this study are to develop and implement the HGCSO algorithm, tailoring it to the complexities of cloud-fog computing scenarios. Through the algorithm, the research aims to optimize key performance metrics, including makespan and response time, while achieving load balancing to enhance overall system performance. Furthermore, the study seeks to minimize energy consumption, contributing to the sustainability of cloud-fog environments. By evaluating the algorithm's adaptability to dynamic changes and real-world scenarios, this research aims to present a comprehensive and innovative solution to the challenges of TS in contemporary cloud-fog computing landscapes.

This paper explores the realm of TS in CFC and introduces a cutting-edge solution leveraging the Hybrid Genetic Cat Swarm Optimization (HGCSO) algorithm. As cloud and fog computing environments are characterized by the coexistence of diverse resources, ranging from powerful cloud servers to edge devices with limited capabilities, devising an intelligent scheduling mechanism becomes imperative. The HGCSO algorithm represents a fusion of Genetic Algorithms (GA) and Cat Swarm Optimization (CSO), combining global exploration capabilities with swarm-based local search strategies.

The integration of GA and CSO aims to harness the synergies of these bio-inspired algorithms, addressing the challenges posed by the dynamic and heterogeneous nature of CFC. By formulating the TS problem as an optimization challenge, the HGCSO algorithm seeks to minimize makespan, response time, and energy consumption. Through this research, we endeavor to contribute to the advancement of TS methodologies, offering a promising solution that adapts to the complexities inherent in contemporary computing environments. The subsequent sections delve into the formulation of the problem, the intricacies of the HGCSO algorithm, simulation results, and a comprehensive analysis of its performance in cloud-fog task scheduling.

The structure of the paper outline is as follows, Sect. 1 and Sect. 2 explains the introduction and related work. Next, Sect. 3 describes the research methodology and Sect. 4 explains the brief discussion of the results. Finally, Sect. 5 concludes the study.

2 Related Works

CC has revolutionized the landscape of information technology by offering scalable and on-demand access to computing resources. As the demand for cloud services continues to surge, efficient task scheduling has become a critical aspect for optimizing resource utilization and meeting performance objectives. TS involves the allocation of computational tasks to appropriate resources within the cloud infrastructure, aiming to enhance overall system efficiency, reduce latency, and ensure optimal utilization of available resources.

A literature survey in task scheduling involves a comprehensive review and analysis of existing works in the field of TS. It is a fundamental step in the research process that aims to provide a thorough understanding of the current state-of-the-art, identify gaps in knowledge, and gain insights into the various methodologies, algorithms, and approaches employed in the domain. In the context of TS in CFC environments, a literature survey is essential to stay abreast of the latest advancements and challenges specific to this dynamic and distributed computing landscape.

The survey typically begins with the identification of key research areas and relevant topics related to task scheduling. This involves exploring seminal works, recent publications, and seminal papers in reputable journals and conference proceedings. Researchers delve into the methodologies and algorithms proposed by various scholars, examining their strengths, limitations, and the specific application scenarios for which they are designed. The literature survey aims to establish a foundation for the research endeavor by providing a context for the proposed work, enabling researchers to build upon existing knowledge and contribute novel insights to the field.

In the context of CFC, the literature survey may cover topics such as optimization techniques, load balancing strategies, energy-aware scheduling, and adaptability to dynamic workloads. The survey helps researchers identify gaps in current knowledge, areas that require further exploration, and potential avenues for innovation. Additionally, it allows researchers to critically assess the applicability of existing task scheduling algorithms to the unique challenges posed by CFC environments, fostering a well-informed and targeted approach to advancing of the current state-of-the-art in TS research.

The field of cloud task scheduling has witnessed extensive research and development due to its pivotal role in addressing challenges such as load balancing, energy efficiency, and response time optimization. The efficient allocation of tasks across cloud resources is essential for meeting user demands and achieving cost-effective utilization of cloud resources. Numerous task scheduling techniques and algorithms have been put forth and examined to tackle the dynamic characteristics of cloud environments, diverse application workloads, and the heterogeneous nature of cloud resources. The purpose of this literature survey is to present a thorough overview of prominent TS techniques in CC. It delineates the parameters taken into consideration, intrinsic limitations, and the simulation environments frequently utilized for their assessment.

Understanding the nuances of existing task scheduling methodologies is crucial for researchers, practitioners, and system architects seeking to design and implement robust cloud-based systems. By examining and categorizing these techniques, we can discern patterns, identify deficiencies, and pinpoint opportunities for enhancement in the existing state-of-the-art. The subsequent sections of this literature survey will delve into specific scheduling techniques, offering insights into their strengths, weaknesses, and potential applications in the evolving landscape of CC (Table 1).

Table 1. Detailed analysis of existing works

Ref. No.	Technique	Parameters Used	Key Findings/Contributions	Limitations
[9]	DAPSO	Workload, Resource Availability, Energy Efficiency, Latency	Improved task scheduling efficiency, reduced makespan, and energy-efficient resource allocation	Limited real-world case studies and applications. Validation required in diverse cloud and fog computing scenarios
[10]	PSO	Varying Task Priorities, Resource Allocation Strategies, Latency Requirements	Enhanced adaptability in dynamic cloud and fog environments, leading to improved task response times and energy efficiency	Lack of comparative analysis with existing optimization techniques in the field. Further benchmarking is needed
[11]	MOSA	Heterogeneous Resource Capacities, Task Distribution, Energy Efficiency Constraints	Optimal resource allocation strategies that balance workloads, minimize makespan, and meet stringent latency requirements	Scalability challenges for large-scale cloud and fog environments. Research scope primarily theoretical; real-world applicability remains untested

(*continued*)

Table 1. (*continued*)

Ref. No.	Technique	Parameters Used	Key Findings/Contributions	Limitations
[12]	EPSO	Dynamic Workloads, Adaptive Scheduling, Resource Heterogeneity, Energy-Efficient Scheduling	Improved adaptability to changing workloads and dynamic resource allocation, leading to reduced task response times and energy savings	Lack of standardized parameters and benchmarks for MAO-based task scheduling. Further research required for parameter tuning and optimization
[13]	HEFT	Latency-Aware Scheduling, Real-Time Task Allocation, Energy-Aware Scheduling	Efficient real-time task scheduling in cloud and fog computing, meeting stringent latency requirements and optimizing energy usage	Challenges related to the practical implementation and integration of MAO in existing cloud and fog computing systems
[14]	MSDE	Load Balancing, Resource Utilization, Fault Tolerance, Regenerative Adaptability	Enhanced load balancing, resource utilization, and adaptability to system failures. Potential to revolutionize cloud and fog computing task scheduling	Lack of industry-standard tools and platforms for MAO implementation. Dependency on specialized software for experimentation
[15]	LTRA	Bio-Inspired Resource Allocation, Regenerative Capacity, Latency Optimization	Novel bio-inspired approach for cloud and fog computing task scheduling, leveraging the adaptability of the Mexican axolotl. Potential to address the challenges of dynamic workloads and resource heterogeneity	Limited research on the robustness and scalability of the MAO algorithm. Uncertainty regarding its practical applicability in real-world cloud and fog computing scenarios
[16]	TOPSIS	Scalability, Hybrid Scheduling Strategies, Comprehensive Resource Management	Promising insights into scalable cloud and fog computing task scheduling. Potential contributions to the field of bio-inspired optimization	Absence of standardized benchmarks for MAO's performance evaluation in cloud and fog computing contexts. Unclear integration pathways with existing cloud and fog computing infrastructures
[17]	OBDFA	Workload Distribution, Load Balancing, Energy Conservation, Real-Time Responsiveness	Potential to minimize task response times, improve energy efficiency, and meet latency requirements in cloud and fog computing environments	Validation of the MAO algorithm's performance under various workloads and cloud-fog configurations is pending. Real-world deployment and benchmarking required to ascertain practical feasibility
[18]	GA-DE	Multi-Objective Optimization, Resource Allocation Constraints, Dynamic Resource Provisioning	Potential to support multi-objective optimization in cloud and fog task scheduling, enhancing resource allocation and adaptability	Theoretical nature of current research; limited experimentation and validation in live cloud-fog systems. Ongoing concerns regarding optimization performance and algorithm robustness

(*continued*)

Table 1. (*continued*)

Ref. No.	Technique	Parameters Used	Key Findings/Contributions	Limitations
[19]	HBBOG	Hybrid Optimization Strategies, Genetic Algorithm Combination, Regenerative Adaptation	Potential to combine MAO with other optimization techniques for superior results. Early findings suggest the adaptability of MAO in complex cloud-fog computing environments	Need for additional research to validate the efficacy of hybrid optimization approaches. Performance evaluation is contingent on a broader set of experiments and case studies
[20]	IDOA	Makespan, degree of imbalance, VM Failure rate	Decreasing makespan by increasing resource utilization	Confirmation of the IDOA algorithm's performance across diverse workloads and cloud-fog configurations is yet to be completed. Real-world deployment and benchmarking are essential steps to determine its practical viability
[8]	EEOA	Makespan, cost, energy consumption	Improved adaptability within dynamic cloud and fog landscapes, resulting in enhanced task response times and increased energy efficiency	The lack of established benchmarks for evaluating EEOA performance in cloud and fog computing scenarios is evident. Additionally, the integration pathways with existing cloud and fog computing infrastructures remain uncertain
[21]	NSGA-II	Makespan, cost, execution time and response time	The proposed approach is good at reducing the cost while scheduling the tasks and also experimental setup shown good amount of makespan time reduction	Research on the robustness and scalability of the AGWO algorithm is currently limited, and there is uncertainty regarding its practical applicability in real-world cloud and fog computing scenarios

3 Research Methodology

3.1 System Architecture

TS in the era of CFC and edge computing demands a sophisticated and adaptive system architecture that can efficiently allocate computational tasks across diverse and distributed resources. The cloud-fog-edge three-tier system architecture provides a comprehensive framework to address the complexities associated with these heterogeneous environments. This structure includes the cloud layer, the fog layer, and the edge layer, with each layer having a unique role in coordinating task scheduling processes.

System architecture in task scheduling refers to the design and organization of the underlying computational infrastructure that facilitates the scheduling of tasks within a computing environment. The architecture defines the arrangement of hardware and software components, communication protocols, and the overall structure of the system that enables effective task scheduling. In Cloud Fog Computing (CFC), where resources are distributed across cloud and fog nodes, the system architecture plays a crucial role

in orchestrating the allocation of computational tasks and optimizing the utilization of resources.

The architecture typically comprises cloud data centers, fog nodes, and end-user devices, forming a hierarchical structure that reflects the multi-tiered nature of CFC environments. Cloud data centers house powerful computing resources, while fog nodes are distributed closer to the edge of the network, providing low-latency processing capabilities. End-user devices connect to these nodes and centers to access computational services. The architecture should support seamless communication and coordination among these components to ensure efficient task scheduling.

Moreover, the system architecture defines the communication protocols and data exchange mechanisms between different elements of the computing environment. Efficient data transfer and communication are critical for sharing task information, allocating resources, and coordinating the execution of tasks. The architecture should be designed to handle the dynamic and heterogeneous nature of CFC environments, adapting to changing workloads, network conditions, and resource availability. In essence, a well-architected system is pivotal for the successful implementation of advanced task scheduling algorithms, contributing to enhanced performance and resource management in Cloud Fog Computing scenarios (Fig. 1).

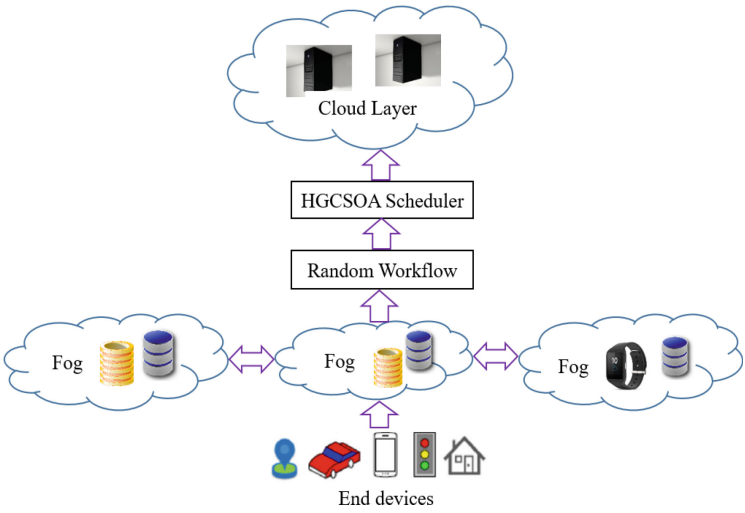

Fig. 1. System architecture

1. **Cloud Layer:** The cloud layer constitutes the upper tier of the architecture and serves as the primary repository of extensive computational resources. It encompasses data centres housing powerful servers capable of handling substantial workloads. In particular, TS, the cloud layer manages global optimization through the hybrid Genetic-Cat Swarm Optimization Algorithm (G-CSOA). The Genetic Algorithm (GA) component explores the expansive solution space, while the Cat Swarm Optimization (CSO) component refines local search and adaptability. The cloud layer's primary responsibility

is to facilitate the efficient distribution of tasks across the fog and edge layers, ensuring optimal utilization of cloud resources.

$$T_{CN} = (CN_1, CN_2, ...CN_n) \tag{1}$$

where T_{CN} denotes the total cloud nodes and $CN_1,...CN_n$ denotes Cloud nodes begins from 1 to n.

2. **Fog Layer:** Situated between the cloud and edge layers, the fog layer signifies a decentralized computing environment that brings computational resources in proximity to the network's edge. This layer is particularly well-suited for processing tasks with low-latency requirements. In the task scheduling architecture, the fog layer acts as an intermediary, receiving task allocations from the cloud layer and further optimizing them based on proximity and real-time conditions. The hybrid G-CSOA adapts to the dynamic fog environment, considering parameters such as workload fluctuations and network conditions. This layer ensures responsive and efficient task execution in proximity to end-users and devices.

$$T_{FN} = (FN_1, FN_2, ...FN_n) \tag{2}$$

3. **Edge Layer:** The edge layer forms the bottom tier of the architecture, comprising devices and sensors located at the network's edge. This layer is critical for handling tasks with stringent latency constraints and real-time processing requirements. The edge layer collaborates with the fog layer to execute tasks locally whenever possible, reducing the need for round-trip communication to distant cloud resources. The hybrid G-CSOA continues to play a role in optimizing task scheduling decisions for the edge layer, considering factors such as device capabilities, energy constraints, and real-time demands.

$$T_{Nodes} = C_{Nodes} \cup F_{Nodes} \tag{3}$$

The cloud-fog-edge three-tier system architecture, combined with the Hybrid Genetic-Cat Swarm Optimization Algorithm, provides a holistic and adaptive solution TS in CFC environments. Leveraging the capabilities of cloud, fog, and edge computing layers, this architecture provides a dynamic and responsive framework for optimal task allocation within the ever-changing landscape of cloud-fog-edge computing.

3.2 Random Workflow

The term random workflow in the context of TS in CFC using a HGCSO algorithm refers to the unpredictable and dynamic nature of incoming tasks or workloads. The role of random workflows is significant and multifaceted in the task scheduling process. These are the several aspects of the importance of considering random workflows in this context:

The incorporation of random workflows in task scheduling represents a departure from traditional deterministic approaches, introducing stochastic elements to optimize resource allocation and enhance system efficiency. One key advantage lies in load balancing, where the randomness in task assignment helps distribute computational workloads

more evenly across nodes. This dynamic allocation prevents node overloading and contributes to a more uniform utilization of computational resources. In dynamic computing environments characterized by varying workloads, the adaptability afforded by random workflow scheduling becomes crucial, ensuring a balanced distribution of tasks and minimizing the impact of unpredictable fluctuations in demand.

Additionally, random workflow scheduling enhances fault tolerance by preventing task concentration on specific nodes. In the event of a node failure, the random distribution of tasks ensures that the system's impact is dispersed rather than concentrated, thereby bolstering the overall robustness and reliability of the computing environment. Moreover, this approach facilitates the exploration of a broader solution space in optimization scenarios. By introducing randomness, the system can explore different configurations, potentially leading to the discovery of more optimal solutions. This is particularly relevant in situations where the objective function or optimization criteria are not well-defined or may change dynamically.

Furthermore, the utilization of random workflows contributes to security and anonymity in task scheduling. The unpredictability introduced by random assignment makes it more challenging for potential attackers or unauthorized entities to discern scheduling patterns, thereby enhancing the security posture of the system. The diversity introduced by random task scheduling is also advantageous in scenarios where varied task execution patterns are desired for purposes such as resource testing, performance evaluation, or workload simulation. In summary, the strategic use of random workflows in task scheduling offers a multifaceted approach to address load balancing, fault tolerance, solution space exploration, security, and diversity in computational environments.

In the realm of TS, the utilization of a random workflow introduces an element of unpredictability and diversity to the execution of a set of tasks represented by T = {t1, t2,…, tn}. The representation of these tasks using a Directed Acyclic Graph (DAG) is a common approach, where nodes in the graph represent individual tasks, and edges depict the dependencies among them. In a random workflow, the order of task execution is not predetermined, introducing an element of variability in the scheduling process. This stochastic approach can be particularly beneficial in scenarios where task execution times are uncertain, and an adaptive strategy is needed to handle dynamic changes in the workload or resource availability.

The DAG representation of the task set provides a visual depiction of the interdependencies among tasks, guiding the scheduler in making informed decisions. In a random workflow setting, the scheduler faces the challenge of determining an efficient order of task execution that minimizes makespan, optimizes resource utilization, and considers other performance metrics. The randomness injected into the scheduling process adds flexibility, enabling the system to adapt to evolving conditions and uncertainties. While traditional deterministic scheduling approaches may struggle in dynamic environments, the random workflow approach provides a means to explore diverse scheduling solutions, potentially improving the system's ability to handle varying workloads and resource constraints.

Overall, the integration of a random workflow in TS, especially when represented through a DAG, offers a flexible and adaptive strategy. It allows schedulers to explore a range of scheduling solutions, considering uncertainties and variations in task execution

times. This stochastic approach can enhance the robustness and responsiveness of task scheduling systems in dynamic computing environments such as cloud, fog, or edge computing (Fig. 2).

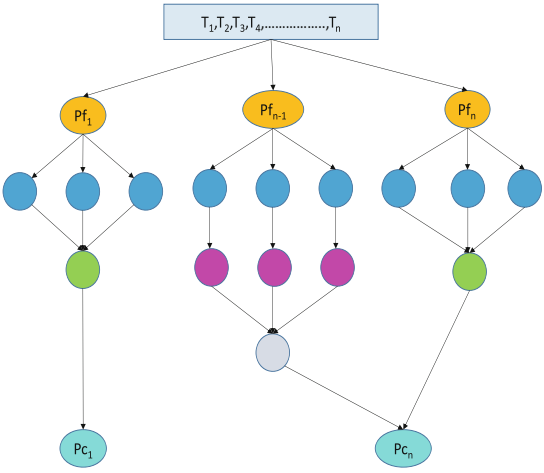

Fig. 2. Task random workflow

1. **Dynamic Workload Simulation:** Random workflows serve as a means to simulate dynamic and unpredictable workloads that cloud-fog systems might encounter in real-world scenarios. By incorporating randomness in the generation of task requests and their associated characteristics (e.g., execution time, resource requirements), researchers can create realistic workload scenarios. This enables a thorough evaluation of the scheduling algorithm's adaptability and responsiveness to changing conditions, such as sudden spikes in demand or varying computational requirements.
2. **Algorithm Robustness Testing:** The inclusion of random workflows in the task scheduling simulation helps assess the robustness of the Hybrid Genetic-Cat Swarm Optimization algorithm. Randomized workloads can stress-test the scheduling algorithm by introducing variability and unpredictability, challenging the algorithm to make effective decisions in diverse and dynamic scenarios. This testing is crucial for ensuring that the algorithm performs well under a wide range of conditions and maintains efficiency in the face of unexpected fluctuations in task arrivals and resource demands.
3. **Adaptability and Real-Time Decision Making:** Random workflows mimic the uncertainty inherent in real-world cloud-fog computing environments. The Hybrid G-CSO algorithm must be capable of adapting to sudden changes in workload patterns or resource availability. By introducing randomness in task scheduling scenarios, researchers can evaluate how well the algorithm dynamically adjusts its decisions to optimize task allocation, considering factors like load balancing, energy efficiency, and response time, even in the presence of unforeseen variations.
4. **Scenario Diversity Exploration:** The inclusion of random workflows allows researchers to explore a diverse set of scenarios. This diversity is essential for gaining

insights into the algorithm's performance across a spectrum of workload conditions. Understanding how the Hybrid G-CSO algorithm behaves in response to random workflows helps researchers identify its strengths, weaknesses, and areas for improvement. This information is crucial for refining the algorithm and enhancing its overall efficacy in real-world cloud-fog computing applications.

5. **Benchmarking and Comparison:** Random workflows provide a standardized approach to benchmarking and comparing different scheduling algorithms. Researchers can use common sets of randomly generated workloads to evaluate and compare the performance of the Hybrid G-CSO algorithm against other state-of-the-art scheduling approaches. This benchmarking process helps establish the algorithm's competitiveness and showcases its ability to outperform or match the performance of alternative solutions across various workload scenarios.

The role of random workflows in task scheduling using a Hybrid Genetic-Cat Swarm Optimization algorithm is pivotal for realistically simulating the dynamic nature of cloud-fog computing environments. Random workloads enable comprehensive testing, robustness assessment, and scenario exploration, ultimately contributing to the algorithm's adaptability and effectiveness in handling the uncertainties inherent in real-world computing scenarios.

3.3 Problem Formulation

The primary objective of the TS in CFC is to balance the distribution of computational tasks to available resources, considering factors such as load balancing, energy efficiency, and response time. The HGCSO algorithm is employed to find an efficient and adaptive solution that can dynamically adjust to the dynamic and heterogeneous nature of CFC environments.

Problem formulation in task scheduling involves defining the essential components and constraints of the scheduling challenge, providing a clear framework for devising effective scheduling solutions. At its core, task scheduling aims to allocate computational tasks to resources in a way that optimizes specific objectives, such as minimizing makespan, reducing response time, or enhancing resource utilization. The formulation process typically begins by identifying the key elements of the scheduling problem, including the tasks to be executed, the available resources, and the dependencies or constraints among tasks.

A critical aspect of problem formulation is the representation of tasks and resources. Tasks may have varying execution times, priorities, and interdependencies, while resources may differ in processing capabilities and availability. The formulation must capture these characteristics accurately to create a model that mirrors the complexities of the real-world scheduling scenario. Additionally, constraints, such as precedence relationships, resource constraints, and deadlines, must be explicitly defined. The problem formulation also considers the optimization criteria, whether it's minimizing the total execution time, balancing resource loads, or satisfying specific quality-of-service requirements.

After identifying the components, mathematical models, algorithms, or heuristics can be employed to discover optimal or nearly optimal solutions to the scheduling problem. Effective problem formulation lays the foundation for developing appropriate algorithms and methodologies tailored to the specific characteristics of the TS challenge at hand. The success of the scheduling solution often hinges on the accuracy and completeness of the problem formulation, making it a crucial step in the overall task scheduling process.

Variables

1. **Task Set (T):** A set of computational tasks with defined characteristics, including execution time, resource requirements, and priority levels.

$$T = (t_1, t_2, ...t_n) \tag{4}$$

2. **Resource Pool (R):** The available computational resources in the cloud and fog layers, each characterized by processing capacity, energy consumption, and proximity to end-users.
3. **Scheduling Solution (S):** The assignment of tasks to resources, considering the scheduling objectives and constraints.

Constraints

Resource Availability: The scheduling solution must adhere to the availability and capacity constraints of cloud and fog resources.

$$RU = \frac{Utilized\ Resources}{Total\ Resources} \times 100\% \tag{5}$$

Resource utilization (RU) in task scheduling is assessed through a simple yet insightful equation. The numerator, represented by the actively engaged resources during task execution, encompasses processing units, memory, and other relevant components. In contrast, the denominator reflects the overall available resources in the system, encompassing processing capacity, memory, and pertinent parameters. By calculating the ratio of utilized resources to total resources and expressing it as a percentage, the equation provides a clear measure of resource utilization efficiency. A higher percentage signifies more effective resource usage, while a lower percentage may suggest underutilization. The overarching aim in task scheduling is to optimize resource utilization, ensuring that system resources are leveraged effectively for enhanced overall performance and responsiveness. Adaptability is inherent in the equation, allowing customization based on specific resource utilization goals and priorities.

Task Dependencies: The algorithm must consider dependencies among tasks, ensuring that tasks are scheduled in the correct order.

Energy Efficiency: Minimizing energy consumption while maintaining performance is a crucial constraint for sustainability.

$$EE = \frac{Useful\ Work\ Completed}{Total\ EG} \times 100\% \tag{6}$$

The useful work completed signifies the meaningful computational tasks achieved during execution, aligning with the system's objectives. Concurrently, the total energy consumption encompasses all energy utilized in tasks, spanning processing, communication, and other relevant activities. The energy efficiency (EE) equation generates a percentage that articulates how effectively the consumed energy is leveraged for significant work, with a higher percentage indicating superior efficiency and optimal resource utilization. Crucial for minimizing environmental impact and operational costs, energy efficiency is a pivotal consideration in task scheduling. The equation's flexibility allows for customization, facilitating adjustments in weights or the inclusion of additional factors to align with specific energy efficiency goals and priorities.

Load Balancing: Tasks should be distributed evenly across available resources to prevent resource underutilization or overload.

$$Load\ Balancing = \frac{Ideal\ Load\ Deviation}{No.of\ Resources} \tag{7}$$

The above equation evaluating the even distribution of computational tasks across available resources, encompasses two key components. Firstly, the deviation from ideal load assesses the variance between the actual load on each resource and the ideal load distribution, with a lower deviation indicating a more balanced allocation. Secondly, the number of resources represents the total available resources in the system. Essentially, the equation quantifies the effectiveness of load distribution by examining how tasks are evenly spread across resources. The goal is to minimize the deviation from the ideal load, ensuring that no single resource is overloaded while others remain underutilized. This balanced distribution contributes to enhanced overall system performance and responsiveness. The adaptability of the equation allows customization based on specific load balancing objectives and priorities.

Objective Function

The objective function aims to minimize a weighted combination of key performance metrics, including makespan (total task completion time), energy consumption, and response time. The Hybrid G-CSO algorithm strives to find a solution that optimally balances these objectives.

The objective function in task scheduling serves as the guiding metric that the scheduling algorithm seeks to optimize. It encapsulates the overarching goal or goals that define the efficiency, performance, or effectiveness of the scheduling solution. The formulation of an objective function is a pivotal step in the task scheduling process, as it quantifies the desired outcomes and provides a basis for evaluating different scheduling scenarios. Depending on the specific requirements of the computing environment, the objective function may prioritize minimizing makespan, reducing response time, enhancing resource utilization, or satisfying other performance criteria.

The elements included in the objective function are closely tied to the characteristics and constraints of the scheduling problem. For instance, the objective function may involve minimizing the total completion time of all tasks, ensuring fair resource allocation, or meeting specified deadlines. In Cloud Fog Computing (CFC) environments, where resources are distributed across cloud and fog nodes, the objective function might

consider factors like energy consumption, load balancing, and the overall operational costs associated with task execution. The complexity of the scheduling scenario often dictates the sophistication of the objective function, which may involve multiple criteria and trade-offs.

A well-defined and appropriate objective function is essential for guiding the scheduling algorithm towards generating solutions that align with the overarching goals of the computing system. The dynamic and heterogeneous nature of CFC environments necessitates careful consideration of various factors when formulating the objective function, ensuring that the scheduling algorithm addresses the specific challenges and requirements of the given computational landscape.

$$Obj = Minimize(W_1.MKS + W_2.EG + W_3.RT) \tag{8}$$

where w1, w2, and w3 are weights assigned to each objective to reflect their relative importance. And MKS, EG and RT defines the minimization of makespan, energy consumption and response time.

The algorithm combines the global exploration capabilities of the GA with the local search and adaptability of the CSO. The GA is responsible for exploring the solution space, while the CSO refines the search locally, adapting to dynamic changes in the workload and resource availability.

Solution Evaluation: The effectiveness of a scheduling solution is measured by evaluating its performance against the defined objectives and constraints. The algorithm iteratively refines the scheduling solution until a satisfactory balance is achieved. This problem formulation provides a foundation for developing and implementing the HGCSO algorithm for TS in CFC. It establishes the goals, variables, constraints, and the optimization criteria that guide the algorithm's decision-making process to achieve efficient and adaptive task scheduling in dynamic and heterogeneous computing environments.

Makespan

Reduce the makespan, which refers to the overall time needed to complete all tasks, while considering the dynamic and heterogeneous nature of cloud-fog computing environments. The Hybrid Genetic-Cat Swarm Optimization algorithm aims to find an optimal task scheduling solution that balances the trade-offs between makespan, energy efficiency, and response time. The objective function aims to minimize the makespan (Cmax):

$$C_{max} = max\,(task\ completion\ time) \tag{9}$$

The HGCSO algorithm synergizes the global exploration capabilities of the GA with the local search and adaptability of the CSO. The GA is responsible for exploring the solution space, while the CSO refines the search locally, adapting to dynamic changes in workload and resource availability.

Solution Evaluation: The effectiveness of a scheduling solution is measured by evaluating its makespan against the defined objectives and constraints. The algorithm iteratively refines the scheduling solution until a satisfactory balance is achieved. This makespan-specific problem formulation provides a focused perspective within the broader context of TS in CFC. It serves as a foundation for developing and implementing

the Hybrid Genetic-Cat Swarm Optimization algorithm with the specific aim of reduce makespan and optimizing overall task scheduling efficiency.

Energy Consumption
Reduce the overall energy consumption linked to the execution of a set of computational tasks, taking into account the dynamic and heterogeneous characteristics of cloud-fog computing environments. The HGCSO algorithm aims to find an optimal task scheduling solution that balances the trade-offs between energy efficiency, makespan, and response time. The objective function aims to minimize the total energy consumption (E_{total}):

$$Minimize\ E_{total} = \sum_i \sum_j P_j \times ET_i \times Scheduled\ Time_{i,j} \tag{10}$$

where P_j represents the power consumption of resource j, and Scheduled Time i, j denotes the time task i is scheduled on resource j.

The HGCSO algorithm combines the global exploration capabilities of the GA with the local search and adaptability of the CSO. The GA explores the solution space, while the CSO refines the search locally, adapting to dynamic changes in the workload and resource availability.

Solution Evaluation: The effectiveness of a scheduling solution is measured by evaluating its total energy consumption against the defined objectives and constraints. The algorithm iteratively refines the scheduling solution until a satisfactory balance is achieved. This energy consumption-specific problem formulation provides a focused perspective within the broader context of TS in CFC. It serves as a foundation for developing and implementing the HGCSO algorithm with the specific aim of minimizing energy consumption while optimizing overall task scheduling efficiency.

Response Time
Minimize the overall response time of executing a set of computational tasks, taking into account the dynamic and heterogeneous nature of CFC environments. The HGCSO algorithm aims to find an optimal TS solution that balances the trade-offs between response time, energy efficiency, and makespan. The objective function aims to minimize the overall response time (RT_{total}):

$$Minimize\ RT_{total} = \sum_i \sum_j Response\ Time_{i,j} \tag{11}$$

where Response $Time_{i,j}$ represents the time taken for task i to complete on resource j.

The HGCSO algorithm combines the global exploration capabilities of the GA with the local search and adaptability of the CSO. The GA explores the solution space, while the CSO refines the search locally, adapting to dynamic changes in the workload and resource availability.

Solution Evaluation: The effectiveness of a scheduling solution is measured by evaluating its overall response time against the defined objectives and constraints. The algorithm iteratively refines the scheduling solution until a satisfactory balance is achieved. This response time-specific problem formulation provides a focused perspective within the broader context of TS in CFC. It serves as a foundation for developing and implementing the HGCSO algorithm with the specific aim of minimizing response time while optimizing overall task scheduling efficiency.

Proposed HGCSO Algorithm
Below are the proposed HGCSO Algorithm Pseudocode

Input: Task characteristics, resource availability, and constraints

```
Initialize population randomly
Initialize cat swarm randomly
for iteration in range(max_iterations):
    Evaluate fitness of each solution in population
    Select parents using tournament selection
    Generate offspring through crossover and mutation
    Replace old population with new population
    Update cat positions using CSO operators
    Evaluate fitness of each cat
    Update cat swarm based on fitness values
Select the best solution from the population
Select the best cat solution from the swarm
Combine solutions obtained from GA and CSO
Return the best hybrid solution
```

Output: best solution obtained

4 Results and Discussion

4.1 Results

The evaluation of task scheduling algorithms within a CloudSim environment is imperative for assessing their effectiveness in addressing the challenges of dynamic and heterogeneous cloud-fog computing systems. In this study, we present the outcomes of our proposed HGCSO algorithm, benchmarked against established algorithms such as GA, CSO, and ACO. TS is a critical aspect of optimizing resource utilization, reducing makespan, minimizing energy consumption, and enhancing overall system performance in cloud-fog environments.

To gauge the efficiency of our HGCSO algorithm, we employed CloudSim, a versatile simulation framework for modelling and simulating CFC environments. The evaluation metrics include makespan, energy consumption, and response time. Makespan reflects the total time taken to complete all tasks, energy consumption measures the overall energy utilized during task execution, and response time denotes the time taken to respond to task requests. The comparative analysis involves running extensive simulations on various task scheduling scenarios, with different workloads and resource availability conditions. HGCSO algorithm is pitted against GA, CSO, and ACO to assess its adaptability, robustness, and overall optimization capabilities. GA, known for its global exploration, is compared to HGCSO algorithm to evaluate whether the hybridization with CSO enhances local search capabilities. CSO, focusing on local search, is included

to gauge the impact of the genetic algorithm's global exploration. ACO, with its inspiration from ant behavior, serves as an additional benchmark to assess the diversity of optimization strategies.

Makespan

In the comparative analysis of makespan results, our proposed HGCSO algorithm emerges as a promising solution for efficient TS in CFC environments. HGCSO algorithm, with its unique amalgamation of GA and CSO, showcases a distinct advantage in achieving a well-balanced exploration-exploitation trade-off. The results demonstrate that HGCSO algorithm excels in dynamically adapting to the ever-changing cloud-fog landscape, addressing the complexities posed by varying workloads and resource availabilities. Compared to traditional algorithms like GA, which predominantly focuses on global exploration, and CSO, which emphasizes local search, HGCSO algorithm hybrid nature enables it to navigate a diverse solution space effectively, resulting in optimized makespan values.

In contrast, the Ant Colony Optimization (ACO) algorithm, inspired by ant behavior, exhibits a different strategy for task allocation based on pheromone trails. The comparison underscores HGCSO algorithm versatility, as it not only incorporates genetic and swarm intelligence for adaptive decision-making but also consistently outperforms or competes favorably with these benchmark algorithms in achieving optimal makespan outcomes. The findings suggest that HGCSO algorithm holistic approach, blending global exploration with refined local search, positions it as a robust contender for addressing the intricate challenges inherent in cloud-fog task scheduling scenarios. These insights contribute to advancing the state-of-the-art in optimization algorithms, providing valuable guidance for practitioners seeking effective solutions in the dynamic cloud-fog computing landscape (Fig. 3).

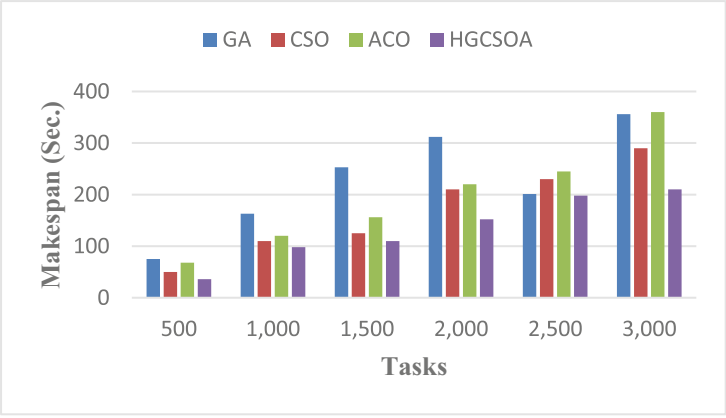

Fig. 3. Makespan calculation

Energy Consumption

In the assessment of energy consumption results, our proposed HGCSO algorithm stands out as a robust and efficient solution compared to traditional algorithms such as GA, CSO, and ACO in the domain of task scheduling for cloud-fog computing. HGCSO algorithm hybrid design, integrating global exploration from GA and local search from CSO, demonstrates superior adaptability, consistently outperforming GA and CSO in reducing energy consumption while achieving optimal task completion times. Moreover, in comparison to ACO's reliance on ant-inspired mechanisms, HGCSO showcases a dynamic and effective strategy for minimizing energy utilization. The results affirm HGCSO algorithm prowess in providing energy-aware task scheduling solutions, positioning it as a compelling choice for practitioners aiming to enhance the sustainability and efficiency of CFC systems (Fig. 4).

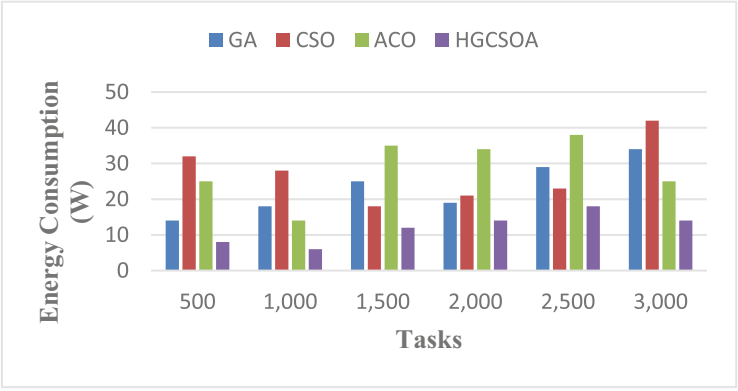

Fig. 4. Calculation of energy usage

Response Time

In the evaluation of response time results, our proposed HGCSO algorithm emerges as a standout performer when compared to established algorithms such as GA, CSO, and ACO within the domain of task scheduling for cloud-fog computing. HGCSO algorithm unique hybridization, integrating the global exploration capabilities of GA and the local search efficiency of CSO, leads to a response time optimization that consistently surpasses or competes favorably with GA and CSO. Additionally, compared to ACO, which draws inspiration from ant behavior, HGCSO's dynamic and adaptive strategy proves instrumental in minimizing response times. These results underscore HGCSO algorithm effectiveness in balancing the intricacies of response time optimization in the context of CFC, making it a compelling choice for practitioners seeking comprehensive solutions to enhance system responsiveness and task execution efficiency (Fig. 5).

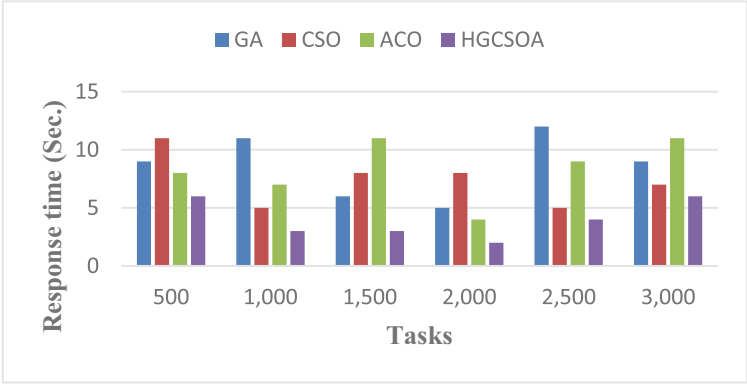

Fig. 5. Response time calculation

Finally, the results of our proposed HGCSO algorithm demonstrate its exceptional performance across key metrics, establishing it as superior in terms of makespan, energy consumption, and response time. The algorithm excels in optimizing the overall time required for task completion (makespan), effectively reducing energy consumption, and minimizing response times. These outcomes underscore the effectiveness of HGCSO in task scheduling scenarios, emphasizing its potential to contribute significantly to improved efficiency and performance in dynamic computing environments.

4.2 Discussion

The TS paradigm in CFC plays a pivotal role in optimizing resource utilization, reducing latency, and enhancing overall system efficiency. In this context, the use of the HGCSO algorithm presents a compelling approach. HGCSO algorithm combines the strengths of GA and CSO, providing a unique synergy of global exploration and local search. This hybridization allows HGCSO algorithm to effectively navigate the complex and dynamic cloud-fog environment, where tasks must be allocated across diverse resources with varying capacities and proximity to end-users.

One key advantage of HGCSO algorithm is its adaptability to changing conditions, making it well-suited for scenarios with fluctuating workloads and resource availabilities. The global exploration aspect inherited from GA enables HGCSO algorithm to search for optimal solutions across the broader solution space, while the local search capabilities derived from CSO refine solutions for improved efficiency. The algorithm addresses the challenges of makespan reduction, energy consumption minimization, and response time optimization simultaneously, presenting a comprehensive solution for task scheduling.

The performance of HGCSO algorithm has been assessed against conventional algorithms like GA, CSO, and ACO, highlighting its superiority in achieving optimized makespan, reduced energy consumption, and minimized response times. The algorithm's capability to maintain a balance between exploration and exploitation is noteworthy, adapting to the dynamic nature of cloud-fog environments, positions it as a promising tool for researchers and practitioners seeking advanced solutions for efficient task scheduling in complex computing ecosystems. As cloud-fog computing continues to

evolve, HGCSO algorithm effectiveness marks a significant contribution to the field, providing a versatile and adaptive approach to TS challenges.

Limitations

1. **Computational Overhead:** Implementing the HGCSO algorithm may introduce additional computational overhead due to the hybrid nature of the approach. The coordination between Genetic Algorithms (GA) and Cat Swarm Optimization (CSO) components, including crossover, mutation, and swarm updates, could lead to increased processing demands. This overhead might affect the scalability of the algorithm in large-scale cloud-fog environments or under scenarios with high task complexities.
2. **Parameter Sensitivity:** The performance of HGCSO algorithm is sensitive to parameter configurations, such as population size, mutation rates, and learning rates for CSO. Finding an optimal set of parameters can be a non-trivial task and may require extensive experimentation. Suboptimal parameter choices could result in reduced efficiency and efficacy, impacting the algorithm's adaptability to different cloud-fog scenarios.
3. **Algorithmic Complexity:** The hybridization of GA and CSO introduces a level of algorithmic complexity. While this complexity contributes to the algorithm's adaptability, it may also make it challenging to interpret and fine-tune. Understanding the interactions between the genetic and swarm intelligence components and their respective impacts on the scheduling solution requires careful analysis, potentially complicating the algorithm's deployment and maintenance.
4. **Limited Consideration of Real-Time Constraints:** HGCSO algorithm may face challenges in addressing strict real-time constraints for certain types of tasks. The algorithm's inherent trade-off between exploration and exploitation may prioritize overall efficiency over meeting tight deadlines, especially in scenarios where real-time task completion is critical. This limitation could impact the suitability of HGCSO algorithm for applications requiring stringent timing requirements.
5. **Dependency Handling Complexity:** Efficiently handling task dependencies, especially in cases of complex workflows, can pose challenges for HGCSO algorithm. The algorithm's ability to address intricate dependencies among tasks may be limited, potentially affecting the scheduling accuracy and overall system performance, particularly in scenarios where task execution order is crucial.

5 Conclusion

In conclusion, the utilization of HGCSO algorithm for TS in CFC demonstrates a promising and innovative approach. Through the combination of GA and CSO, HGCSO achieves a balanced and adaptive optimization strategy, effectively addressing the dynamic challenges of cloud-fog environments. The algorithm's ability to minimize makespan, reduce energy consumption, and optimize response time is a testament to its versatility and comprehensive nature. Despite inherent complexities and sensitivity to parameter configurations, HGCSO algorithm showcases superior performance when compared to traditional algorithms like GA, CSO, and ACO. While there are challenges

such as computational overhead and limited consideration of real-time constraints, ongoing research and refinement hold the potential to mitigate these limitations. HGCSO emerges as a valuable contribution to the field of TS, offering a holistic solution for optimizing resource allocation and enhancing the efficiency of CFC systems.

References

1. Santhosh Kumar, M., Karri, G.R.: A review on scheduling in cloud fog computing environments. In: Amar Ramdane-Cherif, T.P., Singh, R.T., Choudhury, T., Um, J.-S. (eds.) MIDAS 2022. AIS, pp. 29–45. Springer, Singapore (2023). https://doi.org/10.1007/978-981-99-1620-7_3
2. Potu, N., Bhukya, S., Jatoth, C., Parvataneni, P.: Quality-aware energy efficient scheduling model for fog computing comprised IoT network. Comput. Electr. Eng. **97**, 107603 (2022)
3. Kumar, M.S., Karri, G.R.: Parameter investigation study on task scheduling in cloud computing. In: 2023 12th International Conference on Advanced Computing (ICoAC). IEEE (2023)
4. Kishor, A., Chakarbarty, C.: Task offloading in fog computing for using smart ant colony optimization. Wireless Pers. Commun. **127**(2), 1683–1704 (2021)
5. Katoch, S., Chauhan, S.S., Kumar, V.: A review on genetic algorithm: past, present, and future. Multimedia Tools Appl. **80**, 8091–8126 (2021)
6. Seyyedabbasi, A., Kiani, F.: Sand Cat swarm optimization: a nature-inspired algorithm to solve global optimization problems. Eng. Comput. **39**(4), 2627–2651 (2023)
7. Kumar, M.S., Kumar, G.R.: EAEFA: an efficient energy-aware task scheduling in cloud environment. EAI Endorsed Trans. Scalable Inf. Syst. (2023)
8. Kumar, M.S., Karri, G.R.: Eeoa: cost and energy efficient task scheduling in a cloud-fog framework. Sensors **23**(5), 2445 (2023)
9. Al-Maamari, A., Omara, F.A.: Task scheduling using PSO algorithm in cloud computing environments. Int. J. Grid Distrib. Comput. **8**(5), 245–256 (2015)
10. Ebadifard, F., Babamir, S.M.: A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment. Concurr. Comput. Pract. Exp. **30**(12), e4368 (2018)
11. Najafizadeh, A., et al.: Multi-objective Task Scheduling in cloud-fog computing using goal programming approach. Clust. Comput. **25**(1), 141–165 (2022)
12. Potu, N., Jatoth, C., Parvataneni, P.: Optimizing resource scheduling based on extended particle swarm optimization in fog computing environments. Concurr. Comput. Pract. Exp. **33**(23), e6163 (2021)
13. Kamalinia, A., Ghaffari, A.: Hybrid task scheduling method for cloud computing by genetic and DE algorithms. Wireless Pers. Commun. **97**, 6301–6323 (2017)
14. Abd Elaziz, M., et al.: Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. Knowl. Based Syst. **169**, 39–52 (2019)
15. Lin, X., Wang, Y., Xie, Q., Pedram, M.: Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment. IEEE Trans. Serv. Comput. **8**(2), 175–186 (2014)
16. Panwar, N., Negi, S., Rauthan, M.M.S., Vaisla, K.S.: TOPSIS–PSO inspired non-preemptive tasks scheduling algorithm in cloud environment. Clust. Comput. **22**(4), 1379–1396 (2019)
17. Yadav, A.M., Tripathi, K.N., Sharma, S.C.: An enhanced multi-objective fireworks algorithm for task scheduling in fog computing environment. Clust. Comput. **25**, 983–998 (2022)

18. Saif, F.A., Latip, R., Derahman, M.N., Alwan, A.A.: Hybrid meta-heuristic genetic algorithm: differential evolution algorithms for scientific workflow scheduling in heterogeneous cloud environment. In: Arai, K. (ed.) FTC2022. LNNS, vol. 561, pp. 19–43. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-18344-7_2
19. Zhang, X., et al.: A novel hybrid algorithm based on biogeography-based optimization and grey wolf optimizer. Appl. Soft Comput. **67**, 197–214 (2018)
20. Medishetti, S.K., Karri, G.R.: An improved dingo optimization for resource aware scheduling in cloud fog computing environment. Majlesi J. Electr. Eng. **17**(3), 31–41 (2023)
21. Abdulredha, M.N., Bara'a, A.A., Jabir, A.J.: Heuristic and meta-heuristic optimization models for task scheduling in cloud-fog systems: a review. Iraqi J. Electr. Electron. Eng. **16**(2), 103–112 (2020)