Introduction

With the rapid increase in smart mobile devices, computationally intensive applications like online gaming, video conferencing, and 3D modeling are becoming more prevalent. However, these mobile devices generally possess limited resources, such as battery life and local CPU capabilities, resulting in suboptimal computation experiences. Mobile-edge computing (MEC) has emerged as a promising solution to address this challenge. By offloading computational tasks to nearby MEC servers, the quality of the computational experience can be greatly improved, especially in terms of reducing device energy consumption and minimizing execution delays. However, the diverse features of edge computing, coupled with random user requests and limited resources, present significant challenges in optimizing task scheduling. This issue is critical and cannot be disregarded. Efficient task scheduling is essential for enhancing system performance, effectively balancing the load to minimize network overhead, maximizing resource utilization, and managing energy consumption[1]. The primary function of task scheduling is to allocate tasks to suitable resources, ensuring that task execution meets quality of service (QoS) standards[2]. Despite the considerable benefits of edge computing, task scheduling encounters difficulties due to its dynamic nature, task configuration, and resource demands. These factors influence QoS optimization, requiring adjustments to parameters and the appropriate allocation of resources in edge computing[3]. The primary goal of optimization is to improve the objective function during task scheduling. Various optimization metrics include delay, completion time, energy consumption, execution time, and cost. Task scheduling can be categorized as static (offline) or dynamic (online) based on environmental characteristics. In offline scheduling, the scheduler must consider task parameters such as resources, optimization objectives, and QoS constraints. In online scheduling, task parameters and resources automatically adapt to environmental changes[4]. Task scheduling is considered NP-complete, hence meta-heuristic algorithms are employed to find approximate solutions close to the optimal[5]. Common metaheuristics for task scheduling include Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Simulated Annealing (SA), and Ant Colony Optimization (ACO). However, these metaheuristic algorithms involve various search processes. Issues of randomness limited global search capabilities, and low convergence rates in later iterations frequently cause meta-heuristic algorithms to become trapped in local optima[6]. Additionally, there is often an imbalance between global and local search capabilities[7]. To address these challenges, researchers have increasingly turned to the Grey Wolf Optimizer (GWO) due to its effectiveness in overcoming many limitations of other meta-heuristic algorithms. The GWO mechanism requires only the position of a single vector, which results in lower memory usage compared to the Particle Swarm Optimization (PSO) algorithm. Furthermore, GWO relies on the three best solutions to avoid falling into local optima, unlike PSO, which selects only the single best solution among all particles[8]. The GWO algorithm also has fewer parameters compared to Genetic Algorithms (GA), leading to reduced complexity, shorter computation time, and lower energy consumption. It is more dynamic than Ant Colony Optimization, which does not consider the dynamic nature of computing resources[9]. The GWO algorithm, a meta-heuristic approach introduced by [10] is based on the natural behavior of grey wolves in packs and their hunting techniques, following a social hierarchy. Recently, it has gained significant popularity compared to other meta-heuristic algorithms due to its effective convergence during execution, minimal parameter requirements, reduced complexity, lower energy consumption, and ease of implementation[11]. GWO has been adopted in various fields to solve numerous problems, including optimization, classification, economic and power dispatch, and capacitated vehicle routing.

In recent years, there has been a growing interest among researchers in tackling multi-objective problems in task scheduling, especially within the realm of edge computing, due to the distinct characteristics of MEC servers. The two primary objectives that significantly impact performance are execution time and energy consumption. While executing tasks on edge servers reduces execution time due to the shorter distance, it concurrently increases the energy consumption of user devices. Consequently, it is essential to develop a tradeoff strategy that balances execution time and energy consumption. The main contributions of this study are as follows

Related works

The author explores the joint task offloading and energy scheduling issues within a multi-tier edge computing system, where edge servers in the FET can harness green energy. This study introduces the integration of harvested green energy into edge servers in the management of a multi-tier edge computing system, with the objective of reducing system costs related to latency, energy consumption, and cloud rental fees. Furthermore, the author proposes a novel approximated dynamic optimization framework designed to tackle the uncertainty of system information. This framework leverages only the current system information to minimize system costs while maintaining the stability of the harvested energy buffer queues[12]. An optimal task workflow scheduling scheme is proposed for mobile devices, employing the dynamic voltage and frequency scaling technique in conjunction with the whale optimization algorithm. This study considers three critical factors: task execution location, task execution order, and the operating voltage and frequency of mobile devices. By addressing these factors, the study effectively balances performance and energy consumption through the joint optimization of task completion time and energy usage. Extensive simulation results have demonstrated and confirmed the scheme's exceptional efficiency and cost-effectiveness, providing viable solutions for similar optimization problems in mobile cloud computing[13]. The author employs a Markov decision process approach to tackle this problem, scheduling computation tasks based on the queueing state of the task buffer, the execution state of the local processing unit, and the condition of the transmission unit by examining the average delay of each task and the average power consumption of the mobile device, the scheduling process is optimized[14]. The author explores the problem of minimizing energy costs through the combined consideration of VM migration, task allocation, and green energy scheduling, and confirms its NP-hardness. To manage the computational complexity, a heuristic algorithm is proposed to approximate the optimal solution. Extensive simulations reveal that the proposed algorithm efficiently reduces brown energy consumption and performs closely to the optimal solution[15]. The author examines the task scheduling issue within ad-hoc based mobile edge computing and introduces a mechanism designed to optimize overhead. This task scheduling mechanism takes into account opportunity consumption, energy consumption, time delay, and monetary cost, aiming to minimize the overall overhead for mobile devices[16]. The author introduces a collaborative task scheduling strategy for IoT-assisted edge computing. In this framework, an edge node determines the optimal IoT devices for task offloading based on execution time and energy consumption. Concurrently, each IoT device schedules the execution of offloaded tasks while considering its local task workload. Experimental results demonstrate that the proposed scheme not only achieves near-optimal task throughput but also outperforms other scheduling algorithms in terms of meeting deadlines for time-critical tasks, while ensuring that local task deadlines within IoT devices are maintained[17]

**SYSTEM MODEL**

In an edge multi-user, multi-server environment, computational tasks are generated by various edge devices, such as smartphones and user terminals. These tasks are managed by an advanced task scheduling system designed to efficiently allocate resources. The system calculates the Estimated Processing Time (EPT) for each task on each virtual machine using the formula where EPTij=​ is the task length in millions of instructions (MI) and represents the processing power of the Virtual machine. To optimize task assignments, the system employs an improved Grey Wolf Optimizer combined with a dimension learning-based (IGWODLB) strategy. This advanced optimization method effectively explores the vast solution space to find the best task-to-Virtual machine assignments, addressing the combinatorial complexity of possible configurations. Based on these optimized results, tasks are then allocated to the appropriate VM. During task execution, continuous Quality of Service (QoS) monitoring ensures that the system meets performance metrics, such as execution time and energy consumption. This involves dynamically scaling resources and adjusting allocations to maintain optimal performance and efficient resource utilization. The QoS monitoring component provides real-time feedback to the task scheduling process, allowing for informed adjustments. This feedback loop enables the system to adapt to changing conditions and maintain high service quality. By integrating these elements—task generation from edge devices, EPT calculation, IGWODLB-based optimization, task assignment to VM, and continuous QoS monitoring with dynamic adjustments—the task scheduling system achieves efficient and effective resource management in an edge multi-user, multi-server environment. This comprehensive approach ensures that computational tasks are executed promptly and efficiently while maintaining the desired quality of service.

Problem Description

In this section, we present the mathematical formulation of the task scheduling problem for an edge multi-user, multi-server environment. We assume there are numerous physical servers in the data center with varying specifications such as storage capacity, RAM size, CPU cores, and network bandwidth. These servers can be scaled up or down to meet Quality of Service (QoS) requirements [9] . Consider = as a collection of virtual machines in a MEC server. The processing capabilities of each are measured in (millions of instructions per second). Let represent a set of application tasks provided by edge device users to be executed on the MEC servers’ collection of . Given the large number of tasks and virtual machines, the task scheduling problem inherently involves a vast solution search space. This makes finding the optimal scheduling configuration challenging due to the combinatorial explosion of possible task-to- assignments. The task length of each task ​ is measured in millions of instructions. The Estimated processing time ) is used to track the time required to perform a specific task (service) on different virtual machines [36]. refers to the of on computed as follows:

(1)

Where represents the length of task and denotes the processing power of virtual machine, the goal of our study is to improve the Quality of Service (QoS) in terms of execution time and energy efficiency. To manage the extensive solution search space, we incorporate a dimension learning-based hunting (DLH) search strategy in conjunction with the grey wolf optimizer. This method employs dimension learning to effectively search for optimal solutions within the vast combinatorial space, thereby improving the efficiency and effectiveness of the scheduling process. Therefore, effective job mapping is necessary to achieve shorter execution times, Execution time is defined as the total time needed to complete all computational tasks. The execution time (ET) is calculated as

(2)

The total energy used by computing resources to complete all tasks is referred to as energy consumption, and minimizing this is crucial for enhancing system performance and ensuring the desired Quality of Service (QoS). The energy consumed by a includes the energy used during both its active and idle states. Studies indicate that a virtual machine consumes approximately 65% of its total energy while in the idle state. Therefore, the energy consumption for a specific can be calculated as follows:

(3)

Where represents the total execution time of , denotes the energy consumed by in its idle state, and indicates the energy consumed in its active state. Consequently, the total energy consumption of aedge system can be calculated as follows:

(4)

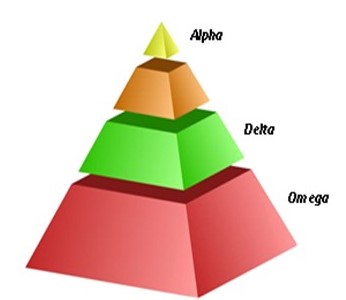
Where represents the total number of virtual machines. Given that energy consumption and execution time significantly influence the overall performance of mobile edge computing (MEC), our primary objective is to minimize execution time while simultaneously reducing total energy consumption. Consequently, our task scheduling problem becomes a multi-objective optimization problem, with the fitness function defined as follows:

(5)

Where represents the balancing parameter between the factors in the fitness function. Hence, our task scheduling objective is to find a schedule that minimizes

Proposed Improved Grey Wolf Optimizer (I-GWODLH)

The proposed IGWODLH algorithm is an improved version of the original GWO algorithm, integrating a Dimension Learning-based Hunting (DLH) search strategy to tackle multiple objectives. It is noteworthy that this marks the first introduction of the IGWODLH algorithm in edge computing. The grey wolf optimizer (GWO) algorithm is inspired by the social leadership and hunting behaviour of grey wolves in nature. In the GWO algorithm, three leader wolves, named α, β, and δ, are considered the best solutions and guide the rest of the wolves, known as ω wolves, towards promising areas to find the global solution. The wolf hunting process comprises three main steps: encircling, hunting, and attacking the prey.



* **Encircling:** The process of encircling the prey by the grey wolves can be modelled using Equations (6) and (7)

(6)

(7)

The signifies the computational distance between the prey and the grey wolf, whereas the denotes the coefficient vector. In this context, represents the position of the grey wolf at a specific time interval , and indicates the location of the prey during the same time interval. The coefficient vectors C and A are calculated using Equations (8) and (9)

(8)

) (9)

Where and are random vectors within the range [0,1], and the elements of vector decrease linearly from 2 to 0 over the course of the iterations, as described by Equation (10).

) (10)

denotes the current iteration, while represents the maximum number of iterations.

**Hunting:** To mathematically represent the hunting behaviour of wolves, it is assumed that , , and have superior knowledge about the prey's location. Consequently, the other wolves, ω, are compelled to follow these three best solutions, , β, and δ. The hunting behaviour is described by Equations (11-12).

(11)

,

, (12)

Where , represent the top three solutions at iteration. The coefficients are computed as in Equation (8), and are defined by Equation (11).

(13)

**Attacking:** The hunting process concludes when the prey stops moving, prompting the wolves to initiate an attack. Mathematically, this is controlled by the value of a, which linearly decreases over the iterations to balance exploration and exploitation. As described in Equation (5), a is updated in each iteration, ranging from 2 to 0. According to Emary et al. (2017), the first half of the iterations focus on exploration, followed by a smooth transition to exploitation for the second half. During this phase, wolves adjust their positions to any random point between the prey's location and their current positions.

**Improved Grey Wolf Optimizer (I-GWO)**

In the GWO algorithm, α, β, and δ lead the ω wolves towards promising areas in the search space to find the optimal solution. However, this approach can result in getting trapped in locally optimal solutions and reducing population diversity, causing the GWO to converge prematurely. To address these issues, we propose an improved Grey Wolf Optimizer (I-GWO). The improvements involve a new search strategy, including a selection and updating step, which are highlighted with dashed lines in the I-GWO flowchart shown in Figure 3. The I-GWO algorithm comprises three phases: initializing, movement, and selection and updating, as follows.

**Initializing phase:** In this phase, N wolves are randomly distributed within the search space over a specified range [li, uj] according to Equation (14).

(14)

The position of the i-th wolf in the t-th iteration is represented as a vector of real values, = { ), where D is the problem's dimensionality. The entire population of wolves is stored in a matrix, Pop, with rows and columns. The fitness value of is calculated using the fitness function, .

**Movement phase:** Beyond group hunting, individual hunting is another intriguing social behaviour of grey wolves (MacNulty et al., 2007), which inspires the enhancement of the GWO. The I-GWO incorporates an additional movement strategy known as the dimension learning-based hunting (DLH) search strategy. In DLH, each wolf learns from its neighbours to become another candidate for the new position of . The following steps outline how the canonical GWO and DLH search strategies generate two different candidates.

**Canonical GWO search strategy:** As described in Section 3, in GWO, the top three wolves from Pop are designated as and . The linearly decreasing coefficient a, along with coefficients and , are calculated using Equations (8-10). The encircling of the prey is determined by considering the positions of , and using Equations (11) and (12). Finally, the first candidate for the new position of wolf, named Xi-GWO(t+1), is calculated using Equation (13).

**Dimension learning-based (DLB) search strategy:** In the original GWO, each wolf's new position is generated with the assistance of the three leader wolves from Pop. This method can result in slow convergence, early loss of population diversity, and entrapment in local optima. To address these issues, the proposed DLB search strategy incorporates individual hunting, where wolves learn from their neighbours.

In the DLB search strategy, each dimension of the new position of wolf is calculated using Equation (12), where the wolf learns from its various neighbours and a randomly selected wolf from population. In addition to , the DLH search strategy generates another candidate for the new position of wolf , named . First, a radius is calculated using the Euclidean distance between the current position of and the candidate position as described by Equation (15).

(15)

Then, the neighbours of , denoted by , are constructed using Equation (11) with respect to radius , where represents the Euclidean distance between .

(16)

Once the neighbourhood of is constructed, multi-neighbor learning is performed using Equation (12). In this process, the d-th dimension of is calculated using the dimension of a randomly selected neighbour from and a randomly selected wolf from population

(17)

**Selecting and Updating Phase:** In this phase, the superior candidate is chosen by comparing the fitness values of the two candidates, Xi-GWO(t+1) and Xi-DLH(t+1), using Equation (13).

(18)

Then, to update the new position of , if the fitness value of the selected candidate is less than that of, is updated with the selected candidate. Otherwise, remains unchanged in the population.

Finally, after completing this procedure for all individuals, the iteration counter (iter) is incremented by one, and the search continues until the predefined number of iterations (Maxiter) is reached.

|  |  |
| --- | --- |
|  | **Algorithm 1:** The Improved Grey Wolf Optimizer (I-GWO) |
|  | **Input:** N, D, MaxIter  **Output:** |
|  | **Begin** |
|  | Initializing (Randomly distributing N entities in the search space and calculating their fitness). |
|  | **For** iter = 2 to Maxiter |
|  | Find |
|  | **For** |
|  | Compute by using Eq. (12) |
|  | Computing using Eg. (13) |
|  | Calculating by Eq. (15) |
|  | Constructing neighbourhood with radius by Eg (116) |
|  | **For** |
|  |  |
|  | **End for** |
|  | Selecting best ( |
|  | Updating population |
|  | **End for** |
|  | **End for** |
|  | **Return** and |
|  | **End** |

[1] L. Chen, K. Guo, G. Fan, C. Wang, and S. Song, “Resource Constrained Profit Optimization Method for Task Scheduling in Edge Cloud,” *IEEE Access*, vol. 8, pp. 118638–118652, 2020, doi: 10.1109/ACCESS.2020.3000985.

[2] P. Varshney and Y. Simmhan, “Characterizing application scheduling on edge, fog, and cloud computing resources,” *Softw. - Pract. Exp.*, vol. 50, no. 5, pp. 558–595, 2020, doi: 10.1002/spe.2699.

[3] I. M. Ali, K. M. Sallam, N. Moustafa, R. Chakraborty, M. Ryan, and K. K. R. Choo, “An Automated Task Scheduling Model Using Non-Dominated Sorting Genetic Algorithm II for Fog-Cloud Systems,” *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2294–2308, 2022, doi: 10.1109/TCC.2020.3032386.

[4] R. O. Aburukba, T. Landolsi, and D. Omer, “A heuristic scheduling approach for fog-cloud computing environment with stationary IoT devices,” *J. Netw. Comput. Appl.*, vol. 180, 2021, doi: 10.1016/j.jnca.2021.102994.

[5] M. Akbari, H. Rashidi, and S. H. Alizadeh, “An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems,” *Eng. Appl. Artif. Intell.*, vol. 61, no. March, pp. 35–46, 2017, doi: 10.1016/j.engappai.2017.02.013.

[6] Z. Deng, Z. Yan, H. Huang, and H. Shen, “Energy-aware task scheduling on heterogeneous computing systems with time constraint,” *IEEE Access*, vol. 8, pp. 23936–23950, 2020, doi: 10.1109/ACCESS.2020.2970166.

[7] P. Paknejad, R. Khorsand, and M. Ramezanpour, “Chaotic improved PICEA-g-based multi-objective optimization for workflow scheduling in cloud environment,” *Futur. Gener. Comput. Syst.*, vol. 117, pp. 12–28, 2021, doi: 10.1016/j.future.2020.11.002.

[8] H. Faris, I. Aljarah, M. A. Al-Betar, and S. Mirjalili, “Grey wolf optimizer: a review of recent variants and applications,” *Neural Comput. Appl.*, vol. 30, no. 2, pp. 413–435, 2018, doi: 10.1007/s00521-017-3272-5.

[9] D. Alsadie, “TSMGWo: Optimizing task schedule using multi-objectives grey Wolf optimizer for cloud data centers,” *IEEE Access*, vol. 9, pp. 37707–37725, 2021, doi: 10.1109/ACCESS.2021.3063723.

[10] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey Wolf Optimizer,” *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[11] Y. Yang, B. Yang, S. Wang, T. Jin, and S. Li, “An enhanced multi-objective grey wolf optimizer for service composition in cloud manufacturing,” *Appl. Soft Comput. J.*, vol. 87, 2020, doi: 10.1016/j.asoc.2019.106003.

[12] T. Offloading *et al.*, “GreenEdge : Joint Green Energy Scheduling and Dynamic,” pp. 1–14.

[13] H. Peng, W. S. Wen, M. L. Tseng, and L. L. Li, “Joint optimization method for task scheduling time and energy consumption in mobile cloud computing environment,” *Appl. Soft Comput. J.*, vol. 80, no. 2019, pp. 534–545, 2019, doi: 10.1016/j.asoc.2019.04.027.

[14] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, “Delay-optimal computation task scheduling for mobile-edge computing systems,” *IEEE Int. Symp. Inf. Theory - Proc.*, vol. 2016-August, pp. 1451–1455, 2016, doi: 10.1109/ISIT.2016.7541539.

[15] L. Gu, J. Cai, D. Zeng, Y. Zhang, H. Jin, and W. Dai, “Energy efficient task allocation and energy scheduling in green energy powered edge computing,” *Futur. Gener. Comput. Syst.*, vol. 95, pp. 89–99, 2019, doi: 10.1016/j.future.2018.12.062.

[16] L. I. Tianze, W. U. Muqing, Z. Min, and L. Wenxing, “An Overhead-Optimizing Task Scheduling Strategy for Ad-hoc Based Mobile Edge Computing,” pp. 5609–5622, 2017.

[17] Y. Kim, C. Song, H. Han, H. Jung, and S. Kang, “Collaborative Task Scheduling for IoT-Assisted Edge Computing,” *IEEE Access*, vol. 8, pp. 216593–216606, 2020, doi: 10.1109/ACCESS.2020.3041872.