Task Scheduling Optimization in Cloud-Fog-MCS Environment Using Genetic Algorithm and Game Theory

Ahmed R. Kadhim* and Furkan Rabee
Computer Science Department, Faculty of Computer Science and Mathematics, University of Kufa, Najaf, Iraq
Email: ahmedr.alkhafajee@uokufa.edu.iq (A.R.K.), furqan.rabee@uokufa.edu.iq (F.R.)

***Abstract*—Fog-cloud computing is a promising platform for processing Mobile Crowdsensing (MCS) tasks that come with different requirements. A fog environment is more suitable for processing time-sensitive tasks due to its proximity to the MCS layer. On the other hand. the cloud environment provides powerful resources to handle large tasks. However, due to the heterogeneity of the computing nodes, scheduling MCS tasks in a fog-cloud environment is a challenging issue. This paper presents a non-cooperative game theoretical model for the task scheduling problem of MCS tasks in the fog-cloud environment. Then, the paper presents an improved genetic algorithm to efficiently solve the problem of task scheduling game model with main enhancements including a new strategy to generate a diverse initial population, incorporating the utility function of the game theoretical model with system fitness function, and finally, the paper introduces a new strategy for population sorting and grouping with applying adaptive crossover operator to meet the specific needs of each group. This improves the exploration of the unseen regions of the search space, as well as exploiting the already-found promising solutions, ultimately leading to a faster convergence toward the optimal solution. The experimental results demonstrate that the proposed approach has better performance in terms of reducing the makespan by 26%, decreasing the energy consumption by 32.4%, decreasing total system cost by 28%, and decreasing the degree of imbalance by 21.53% as compared with other scheduling approaches such as Discrete Non-dominated Sorting Genetic Algorithm II (DNSGA-II), Grasshopper Optimization Algorithm (GOA), Grey Wolf Optimization (GWO, Time-Cost Aware Scheduling (TCaS), Moth Flame Optimization (MFO), and Bees Life Algorithm (BLA).**

***Index Terms*—cloud computing, fog computing, genetic algorithm, Mobile Crowdsensing (MCS), task scheduling, game theory**

I. Introduction

Mobile crowdsensing (MCS) has emerged as a powerful paradigm for collecting real-world data by leveraging the pervasive presence of mobile devices, such as smartphones and wearables [1]. This innovative approach enables the acquisition of real-time, fine-grained information, making it a critical component in various applications, including smart cities, environmental monitoring, healthcare, and transportation [2–4]. However, as the scale and complexity of MCS applications grow, and due to the limitation of MCS devices in terms of power resources and computational capability, they need to offload their tasks to external servers for processing. Cloud computing offers significant computational power and storage capacity to process data-intensive tasks [5], but it tends to introduce higher latency due to the distance that data needs to travel between the MCS devices and the Cloud servers [6]. This latency can be problematic for real-time applications. On the other hand, Fog computing is a distributed computing paradigm, where computational resources and storage are placed closer to the data sources of MCS devices, typically at the network edge. Fog computing is beneficial to reduce latency and improve response time, it is well-suited for tasks that require low-latency processing [7]. Fog computing faces limitations in terms of computational capacity and storage capabilities, that make it not suitable for processing data-intensive tasks. Hence, there is a need for an integrated fog-cloud architectural paradigm that combines the advantages of both fog and cloud environments.

Fog-cloud computing, an emerging paradigm, merges the advantages of cloud computing and fog computing to tackle the challenges arising from the growing demand for time-sensitive data processing with low latency and resource-intensive applications [8, 9]. Although integrated fog-cloud computing systems offer various benefits, it also presents several challenges. One of the key challenges in such systems is task scheduling [10, 11]. Task scheduling in fog-cloud systems involves allocating computational tasks to appropriate fog nodes or cloud servers, aiming to optimize the overall system performance. The problem becomes even more challenging due to the heterogeneity of resources, varying network conditions, and dynamic task requirements. Traditional centralized scheduling approaches, designed for cloud-only environments, may not be well-suited for fog-cloud scenarios due to the distributed nature and resource constraints of the fog nodes. To tackle the task scheduling problem in a fog-cloud environment, this paper proposes an approach that combines genetic algorithms and game theory. Genetic algorithms are well-known optimization techniques inspired by natural selection and genetics. They can efficiently explore the solution space and evolve towards optimal solutions by employing genetic operators such as

Manuscript received November 4, 2023; revised November 30, 2023; accepted December 3, 2023.
*Corresponding author.

crossover and mutation [12–14]. Game theory, on the other hand, is a mathematical tool for analyzing and modeling strategic interactions among rational decision-makers, often referred to as “players”. It is widely used in various fields, including economics, political science, biology, and computer science, to understand and predict how individuals or entities make decisions in competitive or cooperative situations [15]. Additionally, the versatility of game models enables the analysis of task scheduling in different computing environments efficiently. In task scheduling problems, game theory can be utilized to model the interactions between computational tasks and system resources, thereby assisting in the optimization of task scheduling [16]. In this paper, the MCS tasks scheduling problem in fog-cloud computing systems is formulated as a non-cooperative game to optimize various system parameters such as makespan, energy consumption of the nodes, and total system cost. Non-cooperative game is a branch of game theory that focuses on scenarios where players, or decision-makers, pursue their objectives independently, without any formal collaboration. Additionally, the paper seeks to address the satisfaction of MCS participants and mitigate the selfishness in terms of their profit. To tackle the problem, the paper utilizes the genetic algorithm with some useful modifications to solve the non-cooperative scheduling game. Integrating genetic algorithms with non-cooperative game theory concepts for solving task scheduling problems in the fog-cloud system can be highly effective. It allows for the optimization of individual objectives through Genetic Algorithms (GAs) while considering strategic interactions among players (MCS participants), resulting in efficient task mapping and improved performance in fog-cloud environments. The proposed Game Theory-Based Genetic Algorithm (GTGA) measures makespan, energy consumption, system cost, degree of imbalance, and throughput. GTGA simulated using the CloudSim environment.

The main contributions of this paper are summarized as follows.

- Model the task scheduling problem for MCS tasks in the fog-cloud environment as a non-cooperative game. In the proposed model, the MCS participants represent the game players, where each player aims to maximize his profit. The utility function of the game is modeled mathematically.
- Introduce a modified genetic algorithm with enhancements in the initial population, fitness function, and crossover operator to solve the scheduling game model and find a schedule that approximates optimality.
- Introduce a new approach for generating the initial population in the genetic algorithm. This approach explores the diverse locations of the search space to avoid falling in the local optima in the first iterations and to help reach the optimal solution faster.
- Present a new sorting and grouping strategy for the population intending to enhance both the exploitation and exploration of the genetic algorithm. Subsequently, the crossover operator is applied individually to each group following its specific requirements.

The remainder of this paper is structured as follows: In Section II, a review of the existing research in this area is introduced. The system model, mathematical representation, the proposed scheduling approach, and the performance metrics are introduced in Section III. Section IV provides the experimental outcomes, performance assessment, and a discussion of the findings. Finally, Section V concludes the paper.

II. Related Works

The task scheduling problem is one of the key challenging issues in fog-cloud systems. Recently, this problem has attracted significant attention due to its important role in improving the total execution time, total cost, and power consumption of such systems. This section reviews relevant research efforts and then highlights the main difference between our work and existing studies.

Hoseiny *et al.* [17] introduced a heuristic algorithm for fog-cloud computing, called priority-aware genetic algorithm (PGA). Their algorithm is designed to optimize a multi-objective function that involves a weighted combination of several factors, including the overall computation time, energy consumption, and the Percentage of Deadline-Satisfied Tasks (PDST). They consider the diverse requirements of tasks and the heterogeneity of fog and cloud nodes. Their approach combines task prioritization and a genetic algorithm to select the most suitable computing node for each task.

Elaziz *et al.* [18] presented a task scheduling method for handling Internet of Things (IoT) requests in a cloud-fog computing environment. The proposed approach, named AEOSSA (artificial ecosystem-based optimization with salp swarm algorithm), is an enhancement of the Artificial Ecosystem-Based Optimization (AEO) technique. It incorporates elements from the Salp Swarm Algorithm (SSA) to enhance its capacity for discovering optimal solutions. To assess the effectiveness of AEOSSA, the researchers conducted experiments using various real-world and synthetic datasets of different sizes.

Wang *et al.* [19] proposed a novel task scheduling method, denoted as I-FASC, for scheduling tasks with different attributes in the fog-cloud environment. Additionally, an enhanced genetic algorithm, I-FA, is presented, incorporating a firework explosion radius detection mechanism. Their proposed approach can help minimize task processing time and provide better load balancing for fog devices.

Nguyen *et al.* [20] focused on addressing the task scheduling problem within the Bag of Task (BoT) applications in a fog-cloud computing environment. They introduced a Time-Cost Aware Scheduling (TCaS) algorithm aimed at resolving this issue. The primary objective of the TCaS algorithm is to achieve an optimal balance between the time it takes to execute the tasks and the associated monetary costs within the fog-cloud system. Additionally, this algorithm demonstrates adaptability to varying user requirements, allowing for customization based on preferences such as prioritizing based on time or cost.

Alsamarai *et al*. [21] introduced a task scheduling algorithm known as the "bandwidth-deadline algorithm", designed to optimize makespan and ensure timely task completion in a cloud-fog environment. This algorithm places significant emphasis on task deadlines by prioritizing tasks with the earliest deadlines and assigning them to resources capable of completing them in the shortest time to minimize the makespan. The algorithm comprises two key components: the first part, named "fog max–cloud min", and the second part, which utilizes Ant Colony Optimization.

The primary focus of [22] is optimizing the scheduling of IoT requests in a hybrid fog-cloud computing environment to minimize latency, specifically the round-trip time (RTT) involved in processing these requests. The authors employed an integer linear program to solve and validate the model. Also, to address larger-scale problems, a modified genetic algorithms (GA) heuristic approach was developed, allowing for feasible solutions of high quality within reasonable computational time.

Ali *et al*. [23] introduced a multi-objective optimization problem for task scheduling to minimize both makespan and total costs in a fog-cloud environment. To tackle the discrete nature of this multi-objective task-scheduling problem, they presented an optimization model that employs the Discrete Non-dominated Sorting Genetic Algorithm II (DNSGA-II). This model also automates the allocation of tasks to either fog or cloud nodes. The adaptation of the NSGA-II algorithm involves the discretization of the crossover and mutation evolutionary operators, as opposed to using continuous operators that require significant computational resources and are not suitable for precise computing node allocation.

Hoseiny *et al*. [24] designed two algorithms for scheduling volunteer requests in volunteer computing systems (VCSs), named min-CCV and min-V. they formulated the task scheduling problem in fog-cloud computing as a mixed integer linear programming (MILP) with the purpose of supporting both QoS for IoT tasks and low computation and communication costs for the fog-cloud system. The primary objective of the proposed algorithms is jointly minimizing the computation, communication, and delay violation cost for the internet of things (IoT) requests.

Dabiri *et al*. [25] introduced a system model designed to address the job (set of tasks) scheduling problem within the context of cloud-fog computing. Their primary objective was to optimize both the energy consumption of the fog-cloud system and the total deadline violation time of jobs. Subsequently, the authors proposed two nature-inspired optimization techniques, namely the Grey Wolf Optimization (GWO) and the Grasshopper Optimization Algorithm (GOA), to effectively tackle the job scheduling problem within the cloud-fog environment.

Due to the complexity of the task scheduling problem which is considered an NP-hard optimization problem, most of the researchers used greedy heuristic and metaheuristic approaches to find near-optimal solutions. In the literature, some of the researchers employed metaheuristic algorithms in their works. But none of them consider the weakness of such algorithms in terms of exploration and falling in local optima. Also, most of the literature focus on the optimization of the fog-cloud system metrics only, they ignore satisfying the user requirements that aim to improve their profit in term of cost and reduce the execution time. Moreover, a few algorithms have considered a scheduling model for MCS tasks with a given deadline and cost constraints by the Fog-Cloud system. To address these gaps, this study proposed a scheduling model that guarantees meeting both the fog-cloud system and MCS participants' requirements by modeling the task scheduling problem as a non-cooperative game for the self-interested MCS participants. To solve the scheduling game, the proposed work utilized the genetic algorithm with some enhancements to help reach the near-optimal solution (Nash equilibrium point of the game model) quickly and to avoid falling in local optima during the searching process.

III. Problem Formulation

A. System Architecture

The system architecture comprises three layers called MCS, fog, and cloud layers. Each layer is composed of different devices, nodes, and servers, respectively. The architecture of the MCS-fog-cloud system is illustrated in Fig. 1.

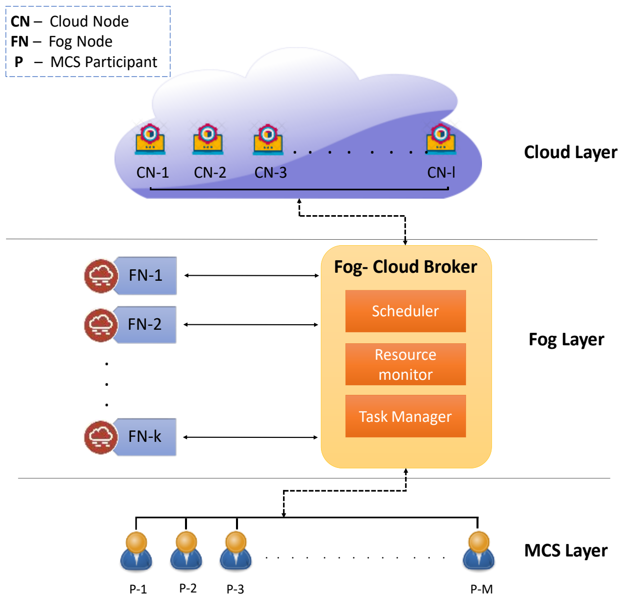


Fig. 1. Proposed MCS-Fog-Cloud architecture.

The MCS layer serves as the first tier and consists of a multitude of mobile devices owned by individuals or participants. These devices are equipped with various sensors, such as GPS, cameras, and accelerometers, enabling them to collect and contribute data about the surrounding environment to the upper (Fog and cloud) layers. Fog layer represents the second layer and serves as the intermediate tier between the MCS and Cloud layers. It comprises fog nodes, which are edge devices located closer to the end-users or MCS devices. The primary objective of this layer is to process data and execute tasks in proximity to the data source, reducing communication latency and bandwidth consumption. Fog nodes often have more computational capabilities than mobile devices, enabling them to handle more compute-intensive tasks.

The cloud computing layer represents the backend of the architecture, comprising large-scale data centers and cloud servers. This layer possesses abundant computational resources and storage capacities, making it suitable for handling resource-intensive and data-intensive tasks. The cloud layer serves as a centralized resource pool that can process tasks from both the MCS and Fog layers.

Fog broker consists of three main components: task manager, task scheduler, and resource monitor. Each component comes with different characteristics and functions, the function of the first component includes several operations. In the initial phase, it receives sensing requests from MCS data requesters, who may be individuals or organizations seeking to observe specific phenomena. Subsequently, the task manager dispatches the sensing task along with its description to the MCS participants. Ultimately, the task manager collects the completed tasks from the MCS participants and evaluates the sensed data based on criteria such as deadline, size, and duration. The resource monitor component is responsible for monitoring and evaluating the fog and cloud nodes and periodically provides the task scheduler with a report about the system resource's status. The task scheduler represents the principal component of the fog-cloud broker. This component acts as a decision-making entity responsible for intelligently distributing tasks to available fog nodes or cloud resources based on the information and data received from the task manager and resource monitor components. Fig. 2 shows the block diagram of task scheduling operations in the MCS-Fog-cloud system. Overall, In the ecosystem of mobile crowdsensing (MCS), the data flow between data requesters and MCS participants involves a dynamic exchange orchestrated by the Mobile Crowdsensing platform. Data requesters initiate the process by submitting requests to the MCS platform. The MCS platform acts as an intermediary, disseminating these requests to a pool of available MCS participants, who are individuals with mobile devices equipped to collect relevant data. The platform facilitates communication and coordination between data requesters and workers, ensuring a streamlined flow of information. Upon receiving tasks, MCS workers utilize their mobile devices to capture and transmit the requested data back to the platform. The platform schedules the data to the fog nodes and cloud servers for more processing before delivering it to the original data requester.

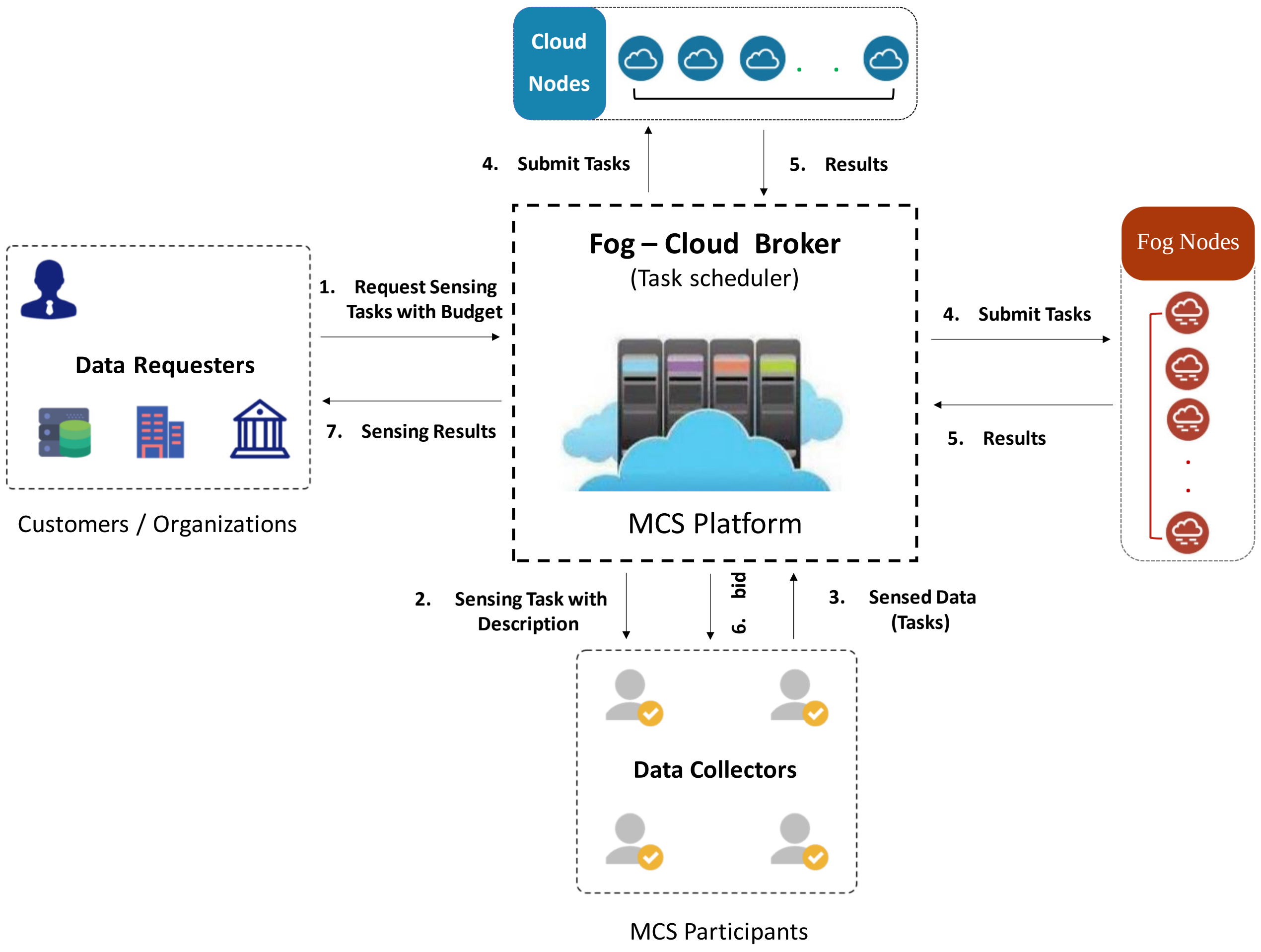


Fig. 2. Block diagram of task scheduling operations for MCS in a fog-cloud system.

B. Mathematical Formulation of the Proposed Model

In the MCS-fog-cloud architecture, first of all, the sensing tasks are released with their descriptions to the MCS participants through the MCS sensing platform. The descriptions include constraints on time to complete the sensing task, task length, and file size. Also, the description includes a price offer for MCS participants to complete the sensing task. In order to ensure the validity of the sensing data, the task manager needs to set an effective time (D) for the MCS participants to complete the sensing task, this helps to overcome the problems that come with the time-sensitive sensing tasks. Also, the task manager specifies the allowed task length which should fall in the range $[\ell_{\min}, \ell_{\max}]$, the last constraint is the file size of the task which is preferred to fall in the range $[s_{\min}, s_{\max}]$. Maintaining the task length, and size within specified ranges leads to controlling and optimizing the fog-cloud system performance in terms of execution time and cost. In addition, it benefits the MCS participants by reducing sensing time and energy consumption. Based on the response of the MCS participants to these constraints, the fog-cloud platform presents a price offer P to the participants. P represents the expected profit for MCS users to participate and accomplish the sensing task and is denoted as

$$P_{\min} < P < P_{\max} \tag{1}$$

where $P_{\min}$ represents a fixed cost to participating in the sensing platform and $P_{\max}$ represents the maximum price values that can be paid to the MCS participants and should satisfy the constraints:

$$P_{\max} < B_{\text{platform}}/\mathrm{M} \tag{2}$$

where B_{platform} represents the total budget of the sensing platform gained from the task requesters (customers or organizations). M represents the number of workers employed to sense and collect data. The fog-cloud broker receives a set of tasks (T) from the MCS participants, $T = \{T_1, T_2, \dots., T_M\}$, with each task defined by three primary parameters $T_j = (\ell_j, \underline{d}_j, s_j)$. In this context, ℓ_j indicate the task length, $\underline{d}_j$ represents the time taken to accomplish and submit the sensing task, and s_j is the task size in megabytes. Also, consider the fog-cloud system has a set (V) of N nodes $V = \{V_1, V_2, \dots, V_N\}$ where V is the combination of a set of fog nodes $F = \{F_1, F_2, \dots, F_k\}$, and set of cloud nodes (virtual machines) in cloud layer $C = \{C_1, C_2, \dots, C_l\}$, $V = F \cup C$. Each node within V is characterized by its available CPU Million instructions per second (MIPS), memory size, and storage, $V_i = (\text{MIPS}_i, \text{MEM}_i, \text{ST}_i)$. Table I shows the notation used with the formulation of the proposed model.

The system will be modeled as a non-cooperative game with complete information, where each task acts as a player in the game.

TABLE I: USED NOTATION

Notation	Description
T	Set of Tasks
F, C	Set of Fog layer nodes, set of cloud layer nodes
V	Set of Fog-Cloud system nodes
M	Number of Tasks in T
N	Number of fog-cloud nodes in V
T_j	The j^{th} task in the set T
V_i	The i^{th} node in fog-cloud system
ℓ_j	Length of T_j
$\underline{d}_j$	Time consumed to complete the sensing task T_j
D	Time allowed to complete the sensing task
MIPS_i	Million Instructions per Second of node V_i
MEM_i	The available memory size of the node V_i
ST_i	Available Storage of node V_i
P	Price offer to participate in the sensing task
$u_j(s)$	The utility function of the player T_j
E_j	The Effort level of the MCS participant
$\text{Cost}_{j,i}$	Total cost of executing T_j on V_i
$\text{ET}_{j,i}$	Execution Time of T_j on V_i
$\text{ET}v_i$	Total execution time of node V_i
C_i^{cpu}	Computing cost of node V_i
C_i^{ram}	Cost of memory usage of node V_i
C_i^{bw}	Cost of bandwidth usage of node V_i
RB_j, RM_j	Required memory, the bandwidth of task T_j
HD	Hamming Distance
$\mathcal{E}_i$	Power consumption of node V_i

In this framework, each player (MCS participant task) aims to maximize their own utility, making strategic choices based on their private information and preferences. The players make the decision by choosing a suitable strategy (selection of a suitable node for processing) to independently archive their goals. The game model is expressed as

$$G = (M, S, U) \tag{3}$$

In this context, M represents the total number of players participating in the game, while S refers to the collective set of strategies chosen by all the players, $S = \{s_1, \dots, s_{m-1}, s_m\}$, and U denotes the payoff, $U = \{u_1(s), \dots, u_{\mathrm{m}}(s)\}$, where $u_j(s)$ denotes the payoff of the player T_j. The description of the game components is explained as follows:

Players: In the task scheduling game, the players represent M tasks received from MCS participants (assuming that the tasks are independent and each task comes from a different MCS participant).

Strategies: Each task should be allocated to a suitable fog or cloud node in order to optimize the scheduling objectives, the number of possible strategies for the player T_j represent the set of fog-cloud nodes (V) available to execute their tasks, where s_j denotes the strategy profile of the player T_j.

Payoffs: In the scheduling game, each MCS participant tries to schedule their task to the node that maximizes his profit in terms of sensing cost, and execution time by selecting the best scheduling strategy. Therefore, the payoff of the player T_j is defined to be the maximizing function of the sensing price, given as

$$u_j(s) = E_j P_{\max} - \text{Cost}_{i,j} \tag{4}$$

where E_j represents an Efficiency factor, denoting the effort level of the MCS participant and reflecting the quality of sensing. It falls within the range of 0 to 1, where higher values indicate that the task sensing and submitting was done within the given deadline, and the length of the task is within the specified length by the sensing platform. This higher efficiency refers to the MCS participant's commitment to the constraints given by the sensing platform, in turn, contributes to maximizing the system performance and the profit of the MCS participant. The efficiency factor is calculated as

$$E_j = \alpha(1 - \frac{\mathrm{d}_j}{D}) + \beta \frac{\ell_j}{\ell_{\max}}, \text{ where } \alpha + \beta = 1 \quad (5)$$

The values of α, and β are selected to be 0.7, and 0.3 respectively, this indicates that prioritizing the time constraint over the task length. This prioritization aims to ensure that sensing and submitting the task to the fog-cloud broker will be done within the given deadline by the sensing platform. The second factor that affects the profit of the MCS participant is the total cost of executing his task on a specific fog or cloud node V_i. The total execution cost ($\mathrm{Cost}_{i,j}$) includes CPU execution cost, memory usage cost, and bandwidth usage cost denoted as $\mathrm{Cost}_{i,j}^{\mathrm{cpu}}$, $\mathrm{Cost}_{i,j}^{\mathrm{ram}}$, and $\mathrm{Cost}_{i,j}^{\mathrm{bw}}$, defined respectively as follows:

$$\mathrm{Cost}_{i,j} = (\mathrm{Cost}_{i,j}^{\mathrm{cpu}} + \mathrm{Cost}_{i,j}^{\mathrm{ram}} + \mathrm{Cost}_{i,j}^{\mathrm{bw}}) \quad (6)$$

$$\mathrm{Cost}_{i,j}^{\mathrm{cpu}} = \mathrm{ET}_{j,i} \times C_i^{\mathrm{cpu}} \quad (7)$$

$$\mathrm{Cost}_{i,j}^{\mathrm{ram}} = \mathrm{RB}_j \times C_i^{\mathrm{ram}} \quad (8)$$

$$\mathrm{Cost}_{i,j}^{\mathrm{bw}} = \mathrm{RM}_j \times C_i^{\mathrm{bw}} \quad (9)$$

Minimizing the total execution cost of the task T_j led to maximizing its final profit. In this context, RB_j represents the required bandwidth to upload and download the task files, RM_j the required memory, and $\mathrm{ET}_{j,i}$ refers to the execution time of the task T_j on the node V_i which is calculated as

$$\mathrm{ET}_{j,i} = \frac{\ell_j}{\mathrm{MIPS}_i} \quad (10)$$

The payoff of playing M players is defined to be an increasing function of the profit of all MCS tasks. It is noticeable that the profit of each player is affected not only by its own scheduling strategy but also by the strategic choices of other players, the total payoff function calculated as

$$U_M(S) = \sum_j u_j(s) \quad (11)$$

Nash equilibrium represents one of the principal solution concepts for game theory in general and noncooperative games in particular. Nash equilibrium is a very important concept and is a widely used solution in the N-person noncooperative game [26]. In the Nash equilibrium (NE), the strategy of each player is considered optimal when it allows the player to attain their maximum payoff while taking into account the strategies adopted by the other players. Therefore, a NE point is represented as $S^* = (s_1^*, s_2^*, \dots, s_m^*)$, where a strategy profile $S^* \in S$ is a Nash Equilibrium in a strategic form game G if and only if the condition in equation (12) is satisfied, $\forall j \in M$ and $\forall s_j \in S_j$.

$$u_j(s_j^*, s_{-j}^*) \geq u_j(s_j, s_{-j}^*) \quad (12)$$

where s_{-j}^* represent the strategies of all other players except player T_j. In this game, no player (task) can maximize his profit by changing it is own strategy only. According to the definition above, scheduling tasks are translated into searching the NE of the task scheduling game. This paper employs the GA to search the NE point of the proposed non-cooperative scheduling game. The mechanisms of the proposed GA will be described in detail in the next section.

C. Game Theory Based Genetic Algorithm

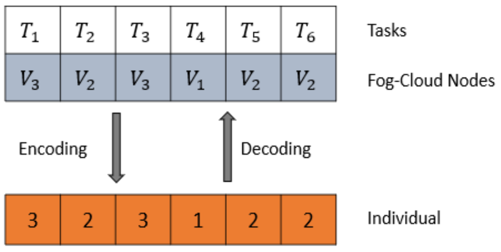
Given the NP-hard nature of the game model, a Genetic Algorithm is employed to search the NE point of this game. genetic algorithms (GAs) are search and optimization techniques inspired by the process of natural selection and evolution. It is commonly used to solve complex problems where traditional methods may be less effective. Genetic algorithms work by evolving a population of potential solutions to a problem over several generations. GAs are particularly suitable for solving optimization problems, which makes them a good choice for solving task scheduling problems in fog-cloud systems.

Based on the framework of the basic GA, three improvements are incorporated into the GA, including presenting a new way for initializing the first population to ensure distribution of the initial population across the whole search space and avoid falling in the local optima in the first iterations. The next improvement is sorting the population according to individuals' finesses and partitioning it into three equal (elite, moderate, and diversity) groups. Then performing the genetic crossover operator within each group with different crossover rates and types. The last improvement is incorporating the utility function of the game theory model with the fitness function of the genetic algorithm, this ensures optimizing the fog-cloud system performance and mitigates the selfishness of the MCS participants. The following subsections describe the representation of the GA to meet the specification of our problem, and then introduce an algorithm to generate the initial population. This is followed by presenting and discussing the selection, crossover, and mutation operators used in the proposed GTGA algorithm.

1) Encoding

GA has chromosomes, and it is essential to encode them effectively to represent the task scheduling problem. There are many encoding techniques for GA to represent task scheduling problems. Our work uses discrete encoding, also known as integer number encoding. Discrete encoding makes a vector of $1 \times M$ dimension, where M is the number of genes (tasks in our work). It shows that for each gene there is a corresponding value representing the node number. Genes have only one allele whose range is the set of fog-cloud nodes V.

As an illustrative example, suppose there are six tasks that can be denoted as $T_1, T_2, T_3, T_4, T_5,$ and T_6. Tasks have to be mapped to three nodes represented as $V_1, V,$ and V_3. Let the chromosome (Individual) be $I = 3, 2, 3, 1, 2, 2$. This implies that the T_1 is assigned to V_3. Similarly, T_2, T_3, T_4, T_5 and T_6 are mapped to V_2, V_3, V_1, V_2 and V_2, respectively. As there are three nodes, there are three pools of tasks. Here in the mapping defined for chromosome I, V_1 gets only one task, so the pools are $V_1 = \{T_4\}, V_2 = \{T_2, T_5, T_6\}$ and $V_3 = \{T_1, T_3\}$. The agenda of the proposed GTGA is to find a chromosome (solution) representing the best schedule to meet the objectives of better makespan and total execution time. Fig. 3 shows the encoding and decoding process for the integer representation of the chromosomes.



T_1 T_2 T_3 T_4 T_5 T_6
Tasks
V_3 V_2 V_3 V_1 V_2 V_2
Fog-Cloud Nodes
Encoding
Decoding
3 2 3 1 2 2
Individual

Fig. 3. Integer representation of the individuals.

2) *Initial population*

To ensure a uniform distribution of the initial population in the solution space, mitigate the centralized distribution in the local region of the search space, and enhance the diversity of the initial population, the first population can be initialized by incorporating the hamming distance technique. Hamming distance is a metric used to measure the difference between two sets of equal length, typically binary sets. However, it can also be applied to a set of integers. In the context of integers, hamming distance is used to quantify how many positions in two integer sequences differ. The hamming distance value falls in the range [0–1] by dividing the number of differences by the individual length. Hamming distance (HD) between two individuals can be calculated as [27].

$$\mathrm{HD}(I_1, I_2) = \frac{\sum_{j=1}^{M} D_j}{M} \tag{13}$$

where D_j is a boolean variable, represented as

$$D_j = \begin{cases} 1 & \text{Genes on } j\text{th position is different} \\ 0 & \text{otherwise} \end{cases}$$

For the proposed work, the first chromosome in the initial population will be generated randomly, then a threshold value $H_{\text{threshold}}$ for the allowed difference is specified. Then, the remaining chromosomes will be generated randomly by ensuring that the average hamming distance of each newly generated chromosome with all individuals in the existing population is greater than the threshold value $H_{\text{threshold}}$. The average hamming distance (AHD) is calculated as

$$\mathrm{AHD}(I_{\text{new}}) = \frac{1}{K} \sum_{k=1}^{K} \mathrm{HD}(I_{\text{new}}, I_k) \tag{14}$$

where K represents the current count of the individuals in the population during the initialization process. If the AHD value is greater than the threshold value $H_{\text{threshold}}$, the individual will be accepted and added to the initial population list, else the individual is rejected, and a new one is generated. This process will be repeated until the initial population list reaches the required size P_{size}. Algorithm 1 presents the pseudo-code of the proposed diversity-guided initialization (DGI) algorithm.

Algorithm 1: the proposed DGI algorithm

```
Input:
        $P_{\text{size}}$: population size
        N: Number of fog-cloud nodes
        M: Individual length (Number of tasks)
        $H_{\text{threshold}}$: Hamming distance threshold
Output:
        pop: Initial population list of size $P_{\text{size}}$
1.      pop= new list // list to store initial population
2.      $I_1$= Generate a Random Individual with length M
3.      pop. add (Ind$_1$)
4.      set j=2
5.      While ( $j \le P_{\text{size}}$ ) do
6.          $I_{\text{new}}$= Generate a random individual with length M
            K=J;
7.          Calculate the Average Hamming Distance (AHD) as
8.          AHD( $I_{\text{new}}$) $= \frac{1}{K}\sum_{k=1}^{K} \mathrm{HD}(I_{\text{new}}, I_k)$
9.          HD($I_{new}$ ,$I_k$) $= \frac{\sum_{j=1}^{M} D_j}{M}$
10.         If (AHD >$H_{\text{threshold}}$) do
11.             pop. add (Ind$_{\text{new}}$)
12.             j=j+1
13.         End if
14.     End while
```

3) *Fitness function*

Individuals are evaluated using the fitness function; the fitness value quantifies the quality of the solution that the individual expresses and also shows its impact on the population. In the proposed work, the fitness value is calculated by incorporating the utility function of MCS tasks in the non-cooperative game model as an optimization objective in the total fitness function. The utility function aims to manage the interactions and mitigate the selfish behavior of the MCS participants. It is calculated as a summation of the utilities of all players (tasks) as in Eq. (11). This utility should be maximized to benefit both MCS participants by maximizing their profits, and the fog-cloud system by minimizing the total execution time and cost. On the other hand, the second objective included in the total fitness function is makespn. This objective should be minimized to optimize the fog-cloud system performance in terms of total execution time. To maintain these two conflicting objectives, the total fitness function F_{total} is introduced to be a maximizing function for the two objectives as

$$F_{\text{total}} = w \frac{\text{min}-\text{makespan}}{\text{makespan(I)}} + (1-w) \frac{U_M(S)}{\max\{U_M(S)\}} \tag{15}$$

where the $\text{min}-\text{makespan}$ represent the minimum makespan obtained across all individuals (solutions) in the current population, and makespan(I) refers to the makespan of the current individual. Also, $\max\{U_M(S)\}$ refers to the maximum utility obtained from all individuals, while $U_M(S)$ represent the utility function of the current solution. w represent a coefficient to balance the influence of the fitness function and the utility function. This weight determines the relative importance of the two functions. It should be a value between 0 and 1, The value w coefficient is tested under different conditions in the simulation, and selected to be 0.5.

4) *Population sorting and grouping*

Exploration and exploitation represent the fundamental principles that characterize the capabilities of evolutionary algorithms (EAs). To enhance the exploitation and exploration of genetic algorithms and to maintain a good balance between them, we propose a population sorting and grouping strategy. Instead of applying the genetic operators (selection, crossover, and mutation) to the entire population, we divide the population into three equal groups based on the fitness of the individuals. This grouping strategy allows the genetic operators to be performed according to the specific needs of each group, thereby improving the exploration of the unseen regions of the search space, as well as exploiting the already-found promising solutions. The three groups are described as follows:

Elite group: This group contains individuals with the highest fitness values. They represent the top-performing solutions in the current population.

Moderate group: The Moderate group consists of individuals with moderate fitness levels. These individuals introduce potential solution but may benefit from increased genetic diversity to explore new solution spaces

Diversity group: The diversity group comprises individuals with lower fitness values. These individuals may hold a high rate of unexplored regions of the search space. Fig. 4 explains the process of sorting and grouping the population.

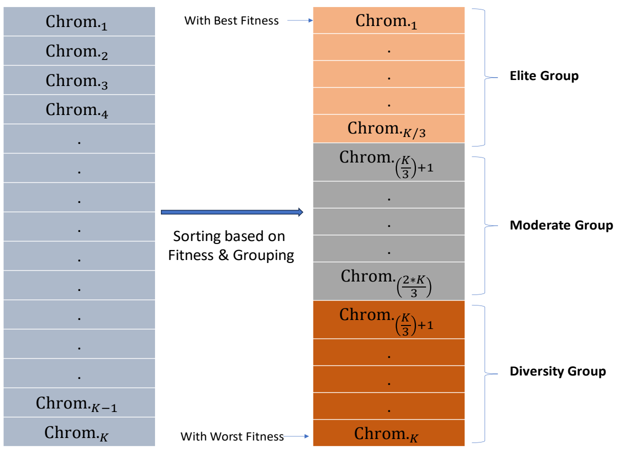


Fig. 4. Population sorting and grouping.

5) *Selection*

The selection of parents in a genetic algorithm is a critical step that governs the genetic diversity and evolutionary progress of the genetic algorithm. This process involves choosing a subset of individuals from the population to serve as parents for the subsequent genetic operators. During the selection phase of the genetic algorithm, individuals are selected based on their fitness values. The total fitness function F_{total} in Eq. (15) is used to evaluate the individuals in our approach. The selection operator will be performed separately for each group to select two parents, the first parent will be selected as the individual with the highest fitness in his group, while the second parent will be selected randomly from the same group.

6) *Crossover operator*

Crossover is the process where two or more parent solutions are selected from the population to create one or more offspring solutions. These offspring solutions inherit genetic material from their parents in a way that mimics biological reproduction. The goal of crossover is to explore the unseen regions of the search space and produce offspring that inherit the best characteristics of their parents, ideally leading to improved solutions over time [14].

The proposed adaptive crossover implements the crossover operator within each group. For the elite group, a high-exploitation crossover technique and a lower crossover rate are chosen to preserve their valuable traits. the elite group will use a one-point crossover with a low crossover rate. Single-point crossover involves selecting a single crossover point (c_p) along the chromosome, and genes to the right of that point in one parent are swapped with the genes to the right of the same point in the other parent. This creates two new offspring, each containing a combination of genes from both parents.

The intermediate group will use a two-point crossover with a moderate crossover rate to explore a broader solution space. Two-point crossover is similar to single-point crossover, but it involves selecting two crossover points (c_{P1}, and c_{P2}). Genes between the two points in one parent are swapped with the corresponding genes between the two points in the other parent. This results in two offspring with genes exchanged in the segments between the two points.

To encourage exploration within the diversity group, we will apply a uniform crossover with a higher crossover rate to promote diversity and potentially discover a good solution. Uniform crossover is a more flexible crossover method. Instead of specifying fixed crossover points, it randomly selects genes from both parents with equal probability (P_{uni}) to decide whether it comes from the first or second parent. This method can introduce greater diversity into the offspring by allowing for a more random mixing of genes. Fig. 5 explains the three types of crossover operators used in the proposed approach.

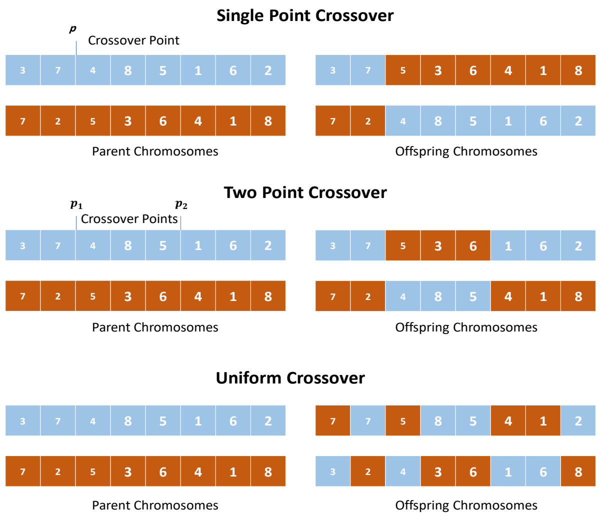


Fig. 5. Crossover operator types.

The pseudo code of the proposed adaptive crossover strategy is described in Algorithm 2.

Algorithm 2: the proposed Adaptive Crossover algorithm

```
Input:
        Two Parents for each group (Elite, Moderate, and Diversity)
Output:
        Two offspring for each group
```

```
1.   P1_ elite =the first individual in elite group
2.   P2_ elite = randomly selected from elite group
3.   r= generated randomly [0–1]
4.   If (r <$P_{c1}$) do
5.       // generate crossover point ($c_P$)
         $c_P$= generate random number between [1–M]
6.       Single point crossover (P1_ elite, P2_ elite, $c_P$ )
7.   End if
8.   P1_ moderate =the first individual in moderate group
9.   P2_ moderate = randomly selected from moderate group
10.  If (r < $P_{c2}$) do
11.      // generate crossover points ($c_{P1}$ and $c_{P2}$)
         $c_{P1}$= generate randomly as: $1 \leq c_{P1} < M$
12.      $c_{P2}$= generate randomly as: $c_{P1} < c_{P2} \leq M$
13.      two-point crossover (P1_ moderate, P2_ moderate, $c_{P1}$, $c_{P2}$)
14.  End if
15.  P1_ diversity =the first individual in diversity group
16.  P2_ diversity= randomly selected from diversity group
17.  If (r < $P_{c3}$) do
18.      $P_{\text{uni}} = 0.5$  // probability of exchanging positions
19.      uniform crossover (P1_ diversity, P2_ diversity, $P_{\text{uni}}$)
20.  End if
```

7) *Mutation operator*

Mutation in GAs is considered the key operator that increases the population diversity and enables the exploration of promising areas in the search space. This operator is applied to the offspring of the crossover operator with a probability called the mutation (P_m). In the proposed work, a multi-point flip mutation is performed on each offspring. The multi-point flip mutation is similar to the bit-flip mutation except that it works on integer numbers. Each gene will be mutated if the generated random number is less than P_m, in this way, the tasks chosen are assigned to be processed on another node. Fig. 6 explains the multi-point flip mutation.

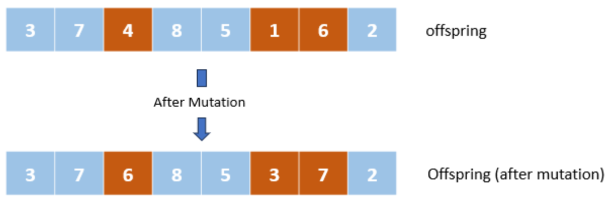


Fig. 6. Multi-point flip mutation.

8) *Overall model of the proposed GTGA*

The flowchart of the proposed GTGA approach is explained in Fig. 7. As shown in the flowchart, first the GTGA initializes the first population using the diversity-guided initialization method. Subsequently, the fitness value of each individual within the population is assessed, these individuals are then categorized into three different groups based on their fitness levels. After that, GTGA evolves the population by performing the proposed adaptive crossover and the multi-point flip mutation. Following this evolution, the newly generated offspring will be evaluated and replace their parents if they exhibit superior fitness values. This iterative process continues until a specified number of iterations ($I_{\max}$) is reached. In the final iteration, the algorithm selects the best individual from the population as the solution to the task-scheduling game.

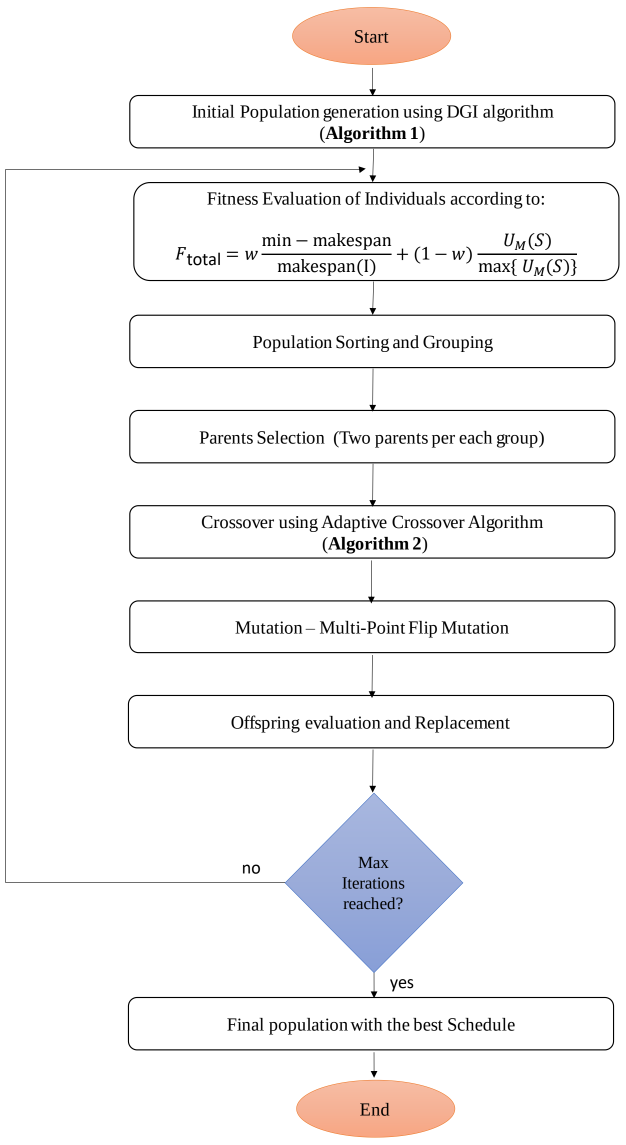

Start
Initial Population generation using DGI algorithm (Algorithm 1)
Fitness Evaluation of Individuals according to:
$F_{\text{total}} = w\frac{\min-\text{makespan}}{\text{makespan}(I)} + (1-w)\frac{U_M(S)}{\max\{U_M(S)\}}$
Population Sorting and Grouping
Parents Selection (Two parents per each group)
Crossover using Adaptive Crossover Algorithm (Algorithm 2)
Mutation – Multi-Point Flip Mutation
Offspring evaluation and Replacement
no
Max Iterations reached?
yes
Final population with the best Schedule
End

Fig. 7. The proposed GTGA approach flowchart.

The pseudo-code of the proposed GTGA approach is outlined in Algorithm 3.

Algorithm 3: the proposed GTGA algorithm

```
Input:
        T: set of MCS tasks
        V: set of fog-cloud nodes
        Imax : Maximum number of Iteration
Output:
        Best mapping of T on V
1       elite-list =new list, moderate-list =new list
        diversity-list = new list, k=1
2       pop= DGI algorithm ($P_{\text{size}}$, $H_{\text{threshold}}$, $N$, $M$),
3       Evaluate the fitness of each individual according to (15)
4       $F_{\text{total}} = w\frac{\min-\text{makespan}}{\text{makespan}(I)} + (1-w)\frac{U_M(S)}{\max\{U_M(S)\}}$
5       While (k ≤ Imax) do
6           //Sort individuals ascendingly according to their fitness
            pop. sort ()
7           For i=1 to ($P_{\text{size}}$ /3) − 1 do
8               elite-list. Add (Ind$_i$ )
```

```
9           End for
10          For i=($P_{\text{size}}$ /3) to (2$P_{\text{size}}$ /3) − 1 do
11              moderate-list. Add ($\text{Ind}_i$ )
12          End for
13          For i= (2$P_{\text{size}}$ /3) to $P_{\text{size}}$ do
14              diversity-list. Add ($\text{Ind}_i$ )
15          End for
16          Parents selections (two parents per each group)
17          Adaptive crossover using algorithm 2
18          Multi-point flip mutation (offspring)
19          Evaluate offspring finesse according to (15)
20          If (offspring fitness > parents' fitness) do
21          Replace parents with offspring
22          End if k=k+1
23      End while
24      Final population (with individual that represent best schedule
        of T on V)
```

D. *Evaluation Metrics Modeling*

To evaluate the proposed approach, five evaluation metrics are employed. They are makespan, total system cost, throughput, energy consumption, and degree of imbalance.

The term "makespan" represents the maximum duration consumed to complete a given set of tasks. Many factors can affect the makespan such as task size, length, resource availability, system load, and the algorithms applied for task scheduling. Reducing the makespan is essential for enhancing resource usage and improving the performance of the fog-cloud system. Mathematically, makespan [28, 29] is formulated as

$$\text{makespan} = \max(\text{ET}v_i) \ \forall\, i \in 1,2,3, \dots n \quad (16)$$

In this context, ETv_i denotes the total execution time taken for all tasks allocated to the *i*th node and is calculated as

$$\text{ETv}_\text{i} = \textstyle\sum_{j=1}^{m} \text{ET}_{j,i} \times A_{i,j} \quad (17)$$

where $A_{i,j}$ is a Boolean variable, is set to be 1 when a task T_j is scheduled on the node V_i.

The second evaluation parameter utilized for our study is throughput, denoting the total tasks completed within a specified makespan, and mathematically formulated as [30]

$$\text{Throughput} = \frac{\sum A_{i,j}}{\text{makespan}} \quad (18)$$

The next evaluation metric is the total system cost which represents the summation of the costs of all tasks. Where, the cost of a given task compromises the processing cost, bandwidth usage cost, and memory usage cost. The total system cost is calculated as [23]

$$\text{Total system Cost} = \textstyle\sum_{j=1}^{M} \text{Cost}_j \quad (19)$$

Another evaluation metric used in the experiments is energy consumption. the energy consumption of each node depends on two main factors: its state mode (idle or active) and the time consumed in each state. The power consumption of node V_i in an active state P_i^{active} is calculated as

$$P_i^{\text{active}} = \Upsilon \times \text{MIPS}_i^2 \quad (20)$$

where $\Upsilon = 10^{-8}$, the power consumption of the node V_i in an idle state P_i^{idle} is calculated as

$$P_i^{\text{idle}} = 0.6 \times P_i^{\text{active}} \quad (21)$$

The total power consumed by the node V_i is calculated as [25]

$$\mathcal{E}_i = [\text{ETv}_i \times P_i^{\text{active}} + (\text{makespan} - \text{ETv}_\text{i}) \times P_i^{\text{idle}}] \times \text{MIPS}_i \quad (22)$$

The total energy consumption of the system (all nodes) of executing the set M of tasks is obtained as

$$\mathcal{E}^{\text{tot}} = \textstyle\sum_{i=1}^{N} \mathcal{E}_i \quad (23)$$

The last evaluation metric is the degree of imbalance. Evaluating the balance of workloads among nodes can effectively demonstrate the efficiency of the proposed algorithm. Therefore, we use the Degree of Imbalance metric to evaluate the load balancing, Degree of Imbalance (DI) can be calculated as [25]

$$\text{DI} = \frac{\text{makespan} - \min_{i \in N}(\text{ETv}_i)}{\sum_{i=1}^{|N|} \text{ETv}_i / |N|} \quad (24)$$

IV. Performance Evaluation

This section presents details about the experimental environment setup, workload characteristics, results, and discussions.

A. *Experimental Setup*

The proposed approach is implemented using the CloudSim simulator. The CloudSim is a Java-based simulation toolkit that can model and simulate cloud computing and data center environments. We extended the CloudSim simulator by adding some classes for the proposed genetic algorithm and modifying some of the existing classes to meet the requirement for simulating the integrated fog-cloud environment. The experimental setup comprises an Intel Core i7 2.7 GHz CPU, 16.00 GB of RAM, and a 512 GB hard drive. The experiments were conducted using the Eclipse IDE 2021 R in conjunction with CloudSim. The specific configuration properties of the simulation experimental environment are detailed in Table II.

Table II: Simulation Environment Parameters

Parameter	**Value**
Processor	Intel Core i7 2.7 GHz
RAM	16.00 GB Memory
Operating system	Windows 10
Simulation Environment	CloudSim simulator
IDE Tool	Eclipse IDE

In our implementation, we examined two simulation scenarios, each one with different simulation parameters. This approach allowed us to evaluate the performance of the proposed model using a variety of evaluation metrics.

1) *Scenario 1*

In this experimental scenario, three fog nodes with ten cloud nodes are included. Cloud and Fog nodes exhibit different processing power as well as resource usage costs. It was assumed that each node is characterized by its individual processing capacity, measured in MIPS (Million Instructions Per Second), along with associated costs related to CPU, memory, and bandwidth usage. Grid Dollars (G$), a simulation currency, is used in the simulation environments as a cost unit. The fog-cloud system characteristics for this scenario are listed in Table III.

TABLE III: FOG-CLOUD SYSTEM ATTRIBUTES FOR SCENARIO 1

Parameter	Fog	Cloud	Unit
No. of Nodes	10	3	-
CPU Rate	[500–1500]	[3000–5000]	MIPS
Memory cost	[0.01–0.03]	[0.02–0.05]	Grid $ per MB
Bandwidth cost	[0.01–0.02]	[0.05–0.1]	Grid $ per MB
CPU usage cost	[0.1–0.4]	[0.7–1]	Grid $ per MB

The main responsibility of the fog-cloud system is executing all MCS tasks. Each task possesses specific attributes, including the number of instructions, memory requirements, input file size, and output file size. A varying number of tasks ranging from 40 to 500 tasks were incorporated in this experiment scenario. The task attributes are generated randomly in this experiment to ensure a diverse range of task sizes and workloads for comprehensive analysis. Table IV shows the task attributes for Scenario 1.

TABLE IV: TASK ATTRIBUTES FOR SCENARIO 1

Parameter	Value	Unit
Task Length	[1–00]	10^9 MI
Memory Required	[50–200]	MB
Input file size	[10–100]	MB
Output file size	[10–100]	MB

In Scenario 1, the proposed GTGA algorithm is evaluated against discrete non-dominated sorting genetic algorithm II (DNSGA-II) [23], time-cost aware scheduling (TCaS) [20], and BLA [31]. The superiority of the proposed methods over other methods is presented in the next subsection. Table V shows the parameter settings of three evolutionary algorithms for Scenario 1.

TABLE V: PARAMETERS SETTING FOR SCENARIO 1

Parameter	GTGA	TCaS	DNSGA-II	BLA
Running times	30	30	30	30
Population Size	100	100	100	100
Crossover rates	P_{c1}=0.4, P_{c2}=0.7, P_{c3}=0.9	0.9	1	0.9
Mutation Rates	*0.01*	0.01	0.01	0.01
Number of Generations	500	500	500	500

2) Scenario 2

In this experimental scenario, the fog-cloud broker receives multiple job requests, each job comprising a variable number of independent tasks that can be processed on different fog and/or cloud nodes. The number of tasks within each job is randomly chosen, ranging from 1 to 10, and the length of each independent task is chosen randomly within the 500 to 5000 million instructions (MI) range. Table VI shows the task parameters for Scenario 2.

TABLE VI: TASK ATTRIBUTES FOR SCENARIO 2

Parameter	Value	Unit
Number of tasks per job	[1–10]	-
Task length	[500–5000]	MI
Input file size	[0.5–5]	MB
Output file size	[0.1–1]	MB
Job deadline	[1–10]	Second

The number of jobs is selected to be 5, 10, 15, 20 and 25. It's important to note that the number of available fog and cloud nodes remains constant throughout this experimental scenario, with 30 fog nodes and 10 cloud nodes. This experiment investigates the impact of different numbers of jobs and tasks on the system's performance. Table VII shows the fog and cloud node attributes for this experiment scenario.

TABLE VII: FOG-CLOUD SYSTEM ATTRIBUTES FOR SCENARIO 2

Parameter	Fog	Cloud	Unit
No. of nodes	30	10	-
CPU rate	[1000–4000]	[5000–20000]	MIPS

The proposed GTGA algorithm is compared against the Grasshopper Optimization Algorithm (GOA) [25], Grey Wolf Optimization (GWO) [25], MFO [32], and GA [22] The simulation is conducted iteratively, repeated 10 times, with 500 iterations for each run. All algorithms employ a population size of 200. In the case of the Genetic Algorithm (GA), the crossover rate is set to 0.9, and the mutation rate is configured at 0.01. In our GTGA algorithm, the crossover rates are set to be $P_{c1} = 0.4$, $P_{c2} = 0.4$, $P_{c3} = 0.4$, and the mutation rate $P_m = 0.01$.

B. Simulation Results

This section introduces the experimental results of the proposed GTGA algorithm for the two scenarios. GTGA has been evaluated and compared using various inertial parameters. The main evaluation metrics conducted in the simulation are Makespan, throughput, total cost, power consumption, and degree of imbalance. The performance of the GTGA task scheduling approach is measured using Eqs. (16), (18), (19), (23), and (24), respectively.

1) Experimental results for Scenario 1

The proposed method will be evaluated in terms of makespan, total cost, and throughput in this scenario. Fig. 8 clearly shows that our method performs better than DNSGA-II, TCaS, and BLA. It can be illustrated that, on average, the GTGA approach achieves a reduction in makespan of 8.1%, 27.3%, and 46.6% compared to DNSGA-II, TCaS, and BLA methods, respectively, when scheduling 40 to 500 tasks. This signifies the effectiveness of our algorithm in seeking the optimal schedule by integrating the concept of non-cooperative game theory with the genetic algorithm.

The results in Fig. 9 demonstrate that GTGA has the lowest system cost in all cases of Scenario 1, while the highest cost is for the tasks executed with the TCaS. GTGA can save a total system cost of 4.91%, 41.73%, and 40% as compared to DNSGA-II, TCaS, and BLA, respectively. The proposed method integrates the utility function of MCS tasks into the overall fitness function of the system. This ensures that the execution cost of tasks is minimized by utilizing nodes that offer the lowest cost.

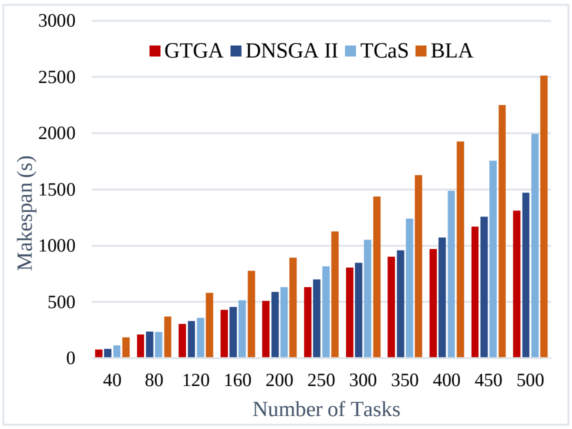


Fig. 8. Comparison of makespan for Scenario 1.

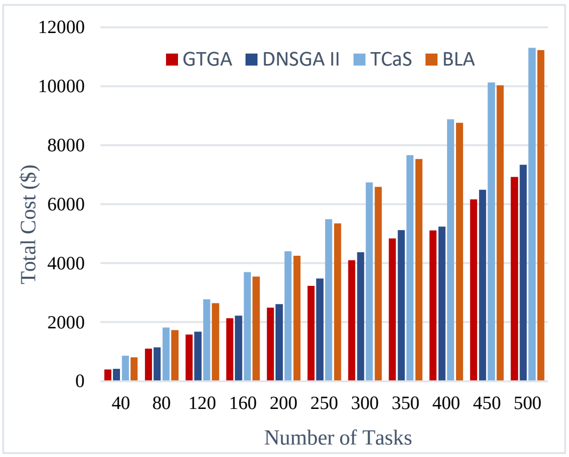


Fig. 9. Total cost comparison of Scenario 1.

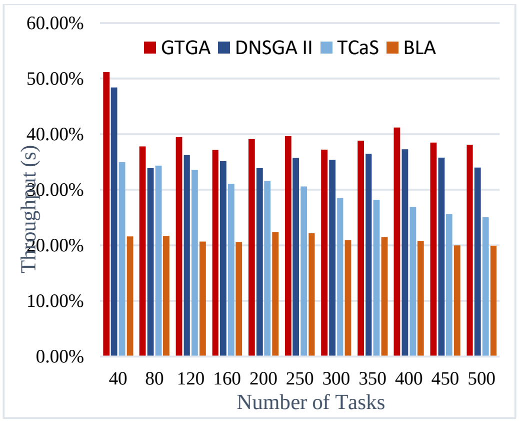


Fig. 10. Throughput comparison of Scenario 1.

Fig. 10 illustrates the throughput performance for Scenario 1. It can be observed that the average throughput is enhanced by 9%, 33.2%, and 87.4 as compared with DNSGA-II, TCaS, and BLA methods respectively.

2) *Experimental results for Scenario 2*

This section introduces the experimental results of implementing Scenario 2. For this scenario, we evaluate the performance of the proposed GTGA approach in terms of makespan, energy Consumption, and degree of imbalance. The results are compared with GOA, GWO, MFO, and GA.

Fig. 11 shows the makespan comparison results for Scenario 2. The results indicate that the GTGA method performed better than GOA, GWO, MFO, and GA with a reduction in makespan of 9.1%, 27.06%, 61.2%, and 36.87% respectively.

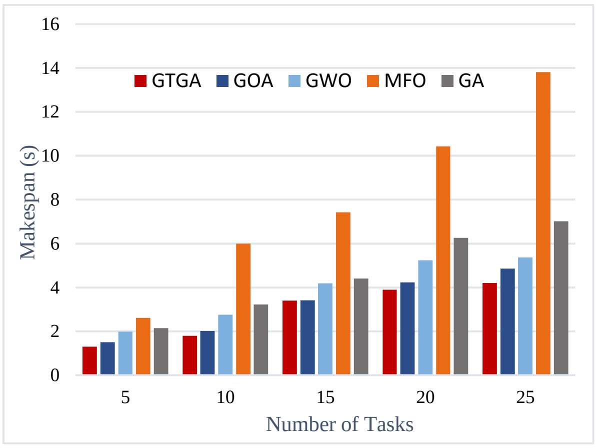


Fig. 11. Comparison of makespan for Scenario 2.

The results of energy consumption for Scenario 2 are evaluated and compared with the other algorithms. In this context, the proposed GTGA algorithm exhibits superior performance when compared with GOA, GWO, MFO, and GA by reducing the total energy consumption by 11.24%, 24.56%, 58.20%, and 36.75% respectively. Fig. 12 Shows the power consumption results for Scenario 2.

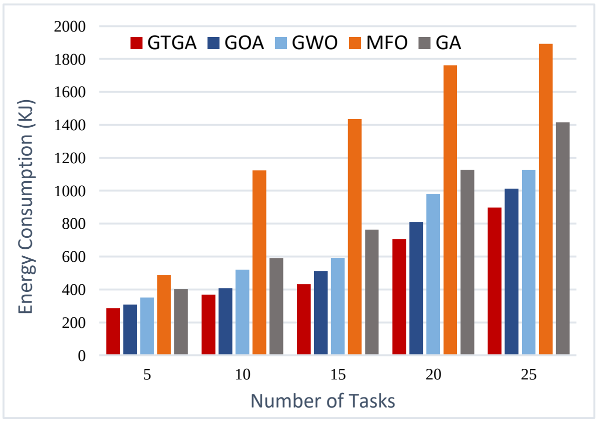


Fig. 12. Energy consumption comparison for Scenario 2.

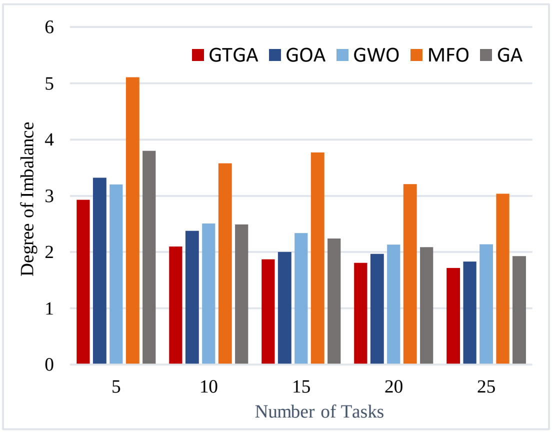


Fig. 13. Degree of imbalance comparison for Scenario 2.

Comparing the results of the degree of imbalance in a computational system is essential for assessing the effectiveness of various algorithms in achieving load distribution and resource utilization. In this context, the proposed GTGA algorithm demonstrates its superiority when compared with GOA, GWO, MFO, and GA by enhancing the DI by 8.83%, 15.90%, 44.29%, and 15.87% respectively as shown in Fig. 13.

V. Conclusion

This paper proposed a game theoretical model for scheduling MCS tasks in the fog-cloud system. The proposed GTGA approach modeled the interaction between the self-interested MCS participants as a non-cooperative game aiming to mitigate their selfishness. A modified genetic algorithm has been utilized to solve the proposed scheduling game model. In the proposed approach, three main enhancements for GA are introduced to improve the exploitation and exploration abilities of the GA and to maintain a good balance between them. First, the initial population is generated in a way that helps to explore the whole search space to avoid falling in the local optima during the first iterations. Then, the population is partitioned into three groups, namely, elite, moderate, and diversity, each group exhibits different characteristics regarding exploration and exploitation. Finally, an adaptive crossover operator is introduced in which different crossover types with different crossover rates are introduced to meet the specific requirements of each group. The proposed scheduling model benefits both the fog-cloud nodes, and the MCS participants, and maximizes the overall performance of the system. The simulation results have shown (i) Increased throughput and decreased makespan values (ii) reduced total system cost for processing the MCS tasks (iii) decreased total energy consumption on the fog-cloud nodes. The proposed approach is observed to have better performance as compared to the state of art algorithms.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Furkan Rabee and Ahmed R. Kadim conducted the research; Furkan Rabee, Ahmed R. Kadim analyzed the data; Ahmed R. Kadim designed and implemented the algorithms; Furkan Rabee and Ahmed R. Kadim wrote the paper; Furkan Rabee and Ahmed R. Kadim had approved the final version.

References

[1] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, and P. Bouvry, "A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2419–2465, 2019.

[2] Y. Liu, H. Li, X. Guan, K. Yuan, G. Zhao, and J. Duan, "Review of incentive mechanism for mobile crowd sensing," *J Chongqing Univ Posts Telecommun (Natural Science Edition)*, vol. 30, no. 2, pp. 147–158, 2018.

[3] C. K. Ng and N. A. Jumadi, "IoT-based instrumentation development for reaction time, kick impact force, and flexibility index measurement," *International Journal of Electrical and Electronic Engineering & Telecommunications*, vol. 11, no. 1, pp. 82–87, 2022.

[4] S. H. Supangkat, R. Ragajaya, and A. B. Setyadji, "Implementation of digital geotwin-based mobile crowdsensing to support monitoring system in smart city," *Sustainability*, vol. 15, no. 5, #3942, 2023.

[5] C. Yang, Q. Huang, Z. Li, K. Liu, and F. Hu, "Big data and cloud computing: innovation opportunities and challenges," *International Journal of Digital Earth*, vol. 10, no. 1, pp. 13–53, 2017.

[6] A. M. A. Al-muqarm and N. A. Hussien, "Dynamic cost-optimized resources management and task scheduling with deadline constraint for mobile crowd sensing environment," *Int. J. Intell. Eng. Syst.*, vol. 16, no. 3, 2023, doi: 10.22266/ijies2023.0630.16.

[7] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[8] R. O. Aburukba, M. AliKarrar, T. Landolsi, and K. El-Fakih, "Scheduling internet of things requests to minimize latency in hybrid fog--cloud computing," *Future Generation Computer Systems*, vol. 111, pp. 539–551, Oct. 2020.

[9] J. Jiang, Z. Li, Y. Tian, and N. Al-Nabhan, "A review of techniques and methods for IoT applications in collaborative cloud-fog environment," *Security and Communication Networks*, vol. 2020, #8849181, 2020.

[10] G. Manogaran and B. S. Rawal, "An efficient resource allocation scheme with optimal node placement in IoT-fog-cloud architecture," *IEEE Sensors Journal*, vol. 21, no. 22, pp. 25106–25113, 2021.

[11] R. O. Aburukba, T. Landolsi, and D. Omer, "A heuristic scheduling approach for fog-cloud computing environment with stationary IoT devices," *Journal of Network and Computer Applications*, vol. 180, #102994, 2021.

[12] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–73, 1992.

[13] F. A. Omara and M. M. Arafa, "Genetic algorithms for task scheduling problem," *Journal of Parallel and Distributed Computing*, vol. 70, no. 1, pp. 13–22, 2010.

[14] F. Rabee and Z. M. Hussain, "Oriented crossover in genetic algorithms for computer networks optimization," *Information*, vol. 14, no. 5, #276, 2023.

[15] M. J. Osborne, *An Introduction to Game Theory*, Oxford New York: University Press 2004.

[16] T. Roughgarden, "Algorithmic game theory," *Communications of the ACM*, vol. 53, no. 7, pp. 78–86, 2010.

[17] F. Hoseiny, S. Azizi, M. Shojafar, F. Ahmadiazar, and R. Tafazolli, "PGA: a priority-aware genetic algorithm for task scheduling in heterogeneous fog-cloud computing," in *Proc. of IEEE Conference on Computer Communications Workshops* , 2021, pp. 1–6.

[18] M. A. Elaziz, L. Abualigah, and I. Attiya, "Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments," *Future Generation Computer Systems*, vol. 124, pp. 142–154, Nov. 2021.

[19] S. Wang, T. Zhao, and S. Pang, "Task scheduling algorithm based on improved firework algorithm in fog computing," *IEEE Access*, vol. 8, pp. 32385–32394, 2020.

[20] B. M. Nguyen, H. Thi Thanh Binh, T. The Anh, and D. Bao Son, "Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud--fog computing environment," *Applied Sciences*, vol. 9, no. 9, #1730, 2019.

[21] N. A. Alsamarai, O. N. Uçan, and O. F. Khalaf, "Bandwidth-deadline IoT task scheduling in fog--cloud computing environment based on the task bandwidth," *Wireless Personal Communications*, pp. 1–17, 2023, doi: 10.1007/s11277-023-10567-1.

[22] R. O. Aburukba, M. AliKarrar, T. Landolsi, and K. El-Fakih, "Scheduling internet of things requests to minimize latency in hybrid fog–cloud computing," *Future Generation Computer Systems*, vol. 111, pp. 539–551, Oct. 2020.

[23] 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. on Cloud Computing*, vol. 10, no. 4, pp. 2294–2308, 2020.

[24] F. Hoseiny, S. Azizi, M. Shojafar, and R. Tafazolli, "Joint QoS-aware and cost-efficient task scheduling for fog-cloud resources in a volunteer computing system," *ACM Trans. on Internet Technology*, vol. 21, no. 4, pp. 1–21, 2021.

[25] S. Dabiri, S. Azizi, and A. Abdollahpouri, "Optimizing deadline violation time and energy consumption of IoT jobs in fog--cloud computing," *Neural Computing and Applications*, vol. 34, no. 23, pp. 21157–21173, 2022.

[26] C. A. Holt and A. E. Roth, "The Nash equilibrium: A perspective," *Proceedings of the National Academy of Sciences*, vol. 101, no. 12, pp. 3999–4002, 2004.

[27] A. Bookstein, V. A. Kulyukin, and T. Raita, "Generalized hamming distance," *Information Retrieval*, vol. 5, pp. 353–375, 2002.

[28] D. Alsadie, "TSMGWO: Optimizing task schedule using multi-objectives grey wolf optimizer for cloud data centers," *IEEE Access*, vol. 9, pp. 37707–37725, 2021.

[29] A. Dhari and K. I. Arif, "An efficient load balancing scheme for cloud computing," *Indian Journal of Science and Technology*, vol. 10, no. 11, pp. 1–8, 2017.

[30] A. R. Kadhim and F. Rabee, "Deadline and cost aware dynamic task scheduling in cloud computing based on stackelberg game," *International Journal of Intelligent Engineering & Systems*, vol. 16, no. 3, 2023, doi: 10.22266/ijies2023.0630.14.

[31] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," *Enterprise Information Systems*, vol. 12, no. 4, pp. 373–397, 2018.

[32] M. Ghobaei-Arani, A. Souri, F. Safara, and M. Norouzi, "An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing," *Trans. on Emerging Telecommunications Technologies*, vol. 31, no. 2, #e3770, 2020.

Ahmed R. Kadhim is Lecturer at Faculty of Computer Science and Mathematics, University of Kufa, Najaf, Iraq. He received his bachelor degree in Computer Engineering from University of Basra in 2010. Later, he earned his master degree in Network Engineering from K. N. T University, Iran in 2017. His research interests are computer network, internet of things, scheduling algorithms, mobile crowdsensing, fog computing and cloud computing. He can be contacted at email: ahmedr.alkhafajee@uokufa.edu.iq.

Furkan Rabee is a staff member in the Computer Science Department, Faculty of Computer Science and Mathematics, at the University of Kufa, Iraq. He got BSc and MSc in Computer Engineering from AL- Nahrian University in 2000 and 2008, Bagdad, Iraq. He got Ph.D. in Computer Science and IT from the School of Computer Science and Engineering, UESTC, Chengdu, China 2015. The research interests include real-time scheduling algorithms, real-time locking protocols, operating systems, parallel processing, distributed systems, computer network, iot, rfid design, mobile computing, cloud computing, and smart cities.