# An Enhanced Healthcare IoT Task Scheduling Using Hybrid Q-NEH and CS-GW Optimization in Fog Computing Environment

by

Zeinab SADEGHI CHEVINLI

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT OF A MASTER'S DEGREE
WITH THESIS IN TELECOMMUNICATION NETWORK ENGINEERING
M.A.Sc.

MONTREAL, 17 DECEMBER , 2024

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

**BOARD OF EXAMINERS**

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Michel Kadoch, Thesis supervisor
Département de génie électrique, École de technologie supérieure

Mr. Julien Clement, Chair, Board of Examiners
Département de génie des systèmes, École de technologie supérieure

Mr. Zbigniew Dziong, Member of the Jury
Département de génie électrique, École de technologie supérieure

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON 13 DECEMBER, 2024

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

## ACKNOWLEDGEMENTS

I would like to thank Professor Kadoch for supervising my thesis. Your help and advice made a big difference during my research. Thank you for giving me useful feedback and always being there to support me throughout this project.

I also want to thank my parents from the bottom of my heart. Your love and support mean everything to me while I'm studying far from home. When things get hard, your encouragement gives me the courage and strength to keep going. Thank you for understanding and supporting my goals.

A special thank you to my beloved brother **Ahmad**, who encouraged me to pursue this path and continue my studies. Your support has meant the world to me.

Finally, I want to express my deepest gratitude to my best friend **Hamidreza** for being an invaluable support during my research. Your constant availability and encouragement helped me navigate through challenging times. Thank you for always being there and motivating me throughout this journey. Your friendship has truly been one of the most meaningful parts of this academic experience

**Un ordonnancement amélioré des tâches IoT en santé utilisant une optimisation hybride Q-NEH et CS-GW dans un environnement de fog computing**

Zeinab SADEGHI CHEVINLI

## RÉSUMÉ

Cette recherche aborde le défi crucial de l'ordonnancement des tâches dans les environnements Internet des Objets (IoT) dédiés aux soins de santé, en utilisant une architecture de fog computing. Nous proposons un nouvel algorithme hybride qui combine l'algorithme Nawaz-Enscore-Ham amélioré par Q-learning (Q-NEH) pour la priorisation des tâches avec une approche d'optimisation Cuckoo Search-Gray Wolf (CS-GW). Le déploiement croissant des dispositifs de surveillance médicale génère d'énormes quantités de données sensibles au temps, nécessitant des stratégies de traitement efficaces capables de gérer plusieurs objectifs concurrents, notamment la minimisation du temps total d'exécution, la maximisation de l'utilisation des ressources et l'assurance de l'efficacité énergétique.

Notre solution hybride exploite les capacités d'apprentissage adaptatif de Q-NEH pour un ordonnancement intelligent des tâches, aux côtés des mécanismes équilibrés d'exploration-exploitation de CS-GW pour l'optimisation. L'évaluation expérimentale menée avec différentes charges de travail (100-600 tâches) sur diverses configurations de nœuds fog (10-60 nœuds) démontre que l'algorithme proposé obtient des performances supérieures par rapport aux approches traditionnelles, notamment l'optimisation par essaims particulaires (PSO), l'algorithme génétique (GA) et l'optimisation par colonies de fourmis (AC). Les résultats montrent que notre implémentation CS-GW a significativement réduit le temps total d'exécution par rapport aux méthodes conventionnelles, tandis que l'intégration de Q-NEH a encore amélioré l'efficacité de l'ordonnancement pour tous les algorithmes testés. La combinaison réussie de l'apprentissage par renforcement avec des techniques d'optimisation inspirées de la nature fournit un cadre robuste pour répondre aux exigences complexes d'ordonnancement des systèmes modernes de surveillance médicale.

**Mots-clés:** ordonnancement des tâches, fog computing, temps total d'exécution

# An Enhanced Healthcare IoT Task Scheduling Using Hybrid Q-NEH and CS-GW Optimization in Fog Computing Environment

Zeinab SADEGHI CHEVINLI

## ABSTRACT

This research addresses the critical challenge of task scheduling in healthcare Internet of Things (IoT) environments using fog computing architecture. We propose a novel hybrid algorithm that combines Q-learning enhanced Nawaz-Enscore-Ham (Q-NEH) for task prioritization with a Cuckoo Search-Gray Wolf (CS-GW) optimization approach. The increasing deployment of healthcare monitoring devices generates massive amounts of time-sensitive data, necessitating efficient processing strategies that can handle multiple competing objectives including minimizing makespan, maximizing resource utilization, and ensuring energy efficiency.

Our hybrid solution leverages Q-NEH's adaptive learning capabilities for intelligent task ordering alongside CS-GW's balanced exploration-exploitation mechanisms for optimization. Experimental evaluation conducted with varying workloads (100-600 tasks) across different fog node configurations (10-60 nodes) demonstrates that the proposed algorithm achieves superior performance compared to traditional approaches including Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and Ant Colony (AC) optimization. The results show that our CS-GW implementation significantly reduced makespan compared to conventional methods, while the integration of Q-NEH further enhanced scheduling efficiency across all tested algorithms. The successful combination of reinforcement learning with nature-inspired optimization techniques provides a robust framework for addressing the complex scheduling demands of modern healthcare monitoring systems.

**Keywords:** task scheduling, fog computing, makespan

## TABLE OF CONTENTS

Page

## LIST OF TABLES

Page

## LIST OF FIGURES

Page

## LIST OF ABBREVIATIONS

| | |
|---|---|
| ACO | Ánt Colony Optimization |
| AHP | Analytic Hierarchy Process |
| BOT | Bag-of-Tasks |
| BRW | Bias Random Walk |
| CNNs | Convolutional Neural Networks |
| COT | Cloud Of Things |
| CPM | Critical Path Method |
| CS | Cuckoo Search |
| CS-GW | Cuckoo Search-Gray Wolf |
| DMFO-DE | Discrete Moth-Flame Optimization with Differential Evolution |
| EDLB | Effective Dynamic Load Balancing |
| FCB | Fog-Cloud Blockchain |
| FCFS | First Come First Serve |
| FN | Fog Nodes |
| FRM | Fog Resource Monitors |
| HGA | Hybrid Genetic Algorithm |
| HMS | Healthcare Monitoring Systems |
| GA | Genetic Algorithms |
| GWO | Gray Wolf Optimizer |

| | |
|---|---|
| ILCO | Life Choice-based Optimization Algorithm |
| ILP | Integer Linear Programming |
| IoT | Internet of Things |
| LPT | Longest Processing Time |
| LRW | Lévy Random Walk |
| MOGWO | Multi-Objective Grey Wolf Optimizer |
| NBIOT | Narrow Band IoT |
| NEH | Nawaz-Enscore-Ham |
| NSGWO | Non-Dominated Sorting Grey Wolf Optimizer |
| PSG | Priority-aware Semi-Greedy |
| PSO | Particle Swarm Optimization |
| Q-NEH | Q-learning enhanced Nawaz-Enscore-Ham algorithm |
| QOS | Quality Of Service |
| SFF | Scheduling Failure Factor |
| SPT | Shortest Processing Time |
| SSA | Salp Swarm Algorithm |
| TPCS | Task Priority Calculating Server |
| VM | Virtual Machine |

# INTRODUCTION

- **Concepts:**

The integration of Internet of Things (IoT) technology in healthcare has created a paradigm shift in medical monitoring and patient care delivery. Modern Healthcare Monitoring Systems (HMS) continuously collect and process vast amounts of patient data through networked sensors and devices, enabling real-time health tracking and rapid response to medical emergencies. The scale of this transformation is evident in the projected growth of IoT devices, expected to reach 41 billion connections by 2030 IoT Analytics (2024). generating continuous streams of critical data from ECG readings and blood pressure measurements to fall detection alerts and medication monitoring.

- **Problem Statement:**

Traditional cloud computing approaches have proven inadequate for handling healthcare monitoring demands due to inherent latency between IoT devices and remote data centers, creating unacceptable delays for time-critical medical applications. While fog computing emerges as an architectural solution by bringing computational resources closer to the data source, the efficient scheduling of healthcare monitoring tasks across fog resources remains a critical and complex problem. This scheduling challenge is classified as an NP-hard problem, where traditional optimization approaches fail to provide adequate solutions for the dynamic healthcare monitoring requirements.

- **Objectives:**

The primary objective of this research is to minimize makespan in healthcare IoT task scheduling through fog computing environments. Makespan, defined as the total completion time from when the first task begins until the last task finishes, is crucial in healthcare monitoring applications

where processing delays can directly impact patient care quality. By optimizing makespan, we aim to ensure timely processing of critical health data while efficiently utilizing fog computing resources. This objective addresses the fundamental challenge of processing time-sensitive medical data efficiently while managing the constraints of fog computing environments.

- **Methodology:**

This thesis proposes a novel hybrid algorithm combining Q-learning enhanced Nawaz-Enscore-Ham algorithm (Q-NEH) with Cuckoo Search-Gray Wolf (CS-GW) optimization. The Q-NEH component provides intelligent task ordering by learning from scheduling outcomes and adapting its strategies accordingly, particularly valuable in healthcare environments where task patterns and priorities evolve over time. The CS-GW hybrid optimizer leverages:

- Cuckoo Search's Lévy flight patterns for balanced local-global search
- Gray Wolf optimization's structured exploitation through social hierarchy-based search

Our experimental evaluation assesses the algorithm's performance against established approaches including Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Genetic Algorithms (GA) across realistic healthcare monitoring scenarios.

- **Contribution and Innovation:**

The research advances the field through:

- Development of a novel hybrid scheduling approach specifically designed for healthcare IoT applications in fog environments
- Integration of reinforcement learning to enhance traditional scheduling heuristics
- Provides thorough performance analysis in healthcare scenarios
- Advanced understanding of hybrid optimization techniques in fog computing environments

- Our key innovation lies in combining three complementary approaches: Q-learning for adaptive decision-making, NEH for effective task ordering, and CS-GW for balanced optimization

- **Thesis Organization:**

The remainder of this thesis is organized as follows:

Chapter 2: Reviews current state of healthcare IoT systems, fog computing, and task scheduling approaches to establish the theoretical foundation

Chapter 3: Presents mathematical foundations and implementation details of the proposed hybrid algorithm

Chapter 4:Provides comprehensive analysis of experimental results and comparative performance evaluation

Chapter 5: Concludes with insights and future research directions

# CHAPTER 1

# BACKGROUND

This chapter covers the fundamentals of Fog computing, Task scheduling, Meta Heuristic Algorithms, Health Monitoring System, Cuckoo Search Algorithms and Order strategy. Following a broad discussion of IOT and task algorithms.

The Internet of Things (IoT) plays a vital role in emergency and survival situations by connecting various devices, applications, and systems. Through IoT technology, we can perform multiple functions and gather valuable, actionable data. IoT devices have the capability to independently access and process multimedia content for various purposes. The rapid growth of IoT has led to its adoption in many fields, with healthcare being a notable example through e-health applications.(Azaria, Ekblaw, Vieira & Lippman, 2016)

The rise of Healthcare Monitoring Systems (HMS) has become essential in today's medical care delivery, using advanced Internet of Things (IoT) capabilities to continuously track patient conditions and analyze their data in real-time"(Azaria *et al.*, 2016). With these systems collecting massive amounts of information from diverse monitoring devices, efficient data handling has become critical. To address challenges such as response time, limited bandwidth, and privacy concerns, healthcare providers utilize fog computing - an advanced form of cloud computing technology.(Bonomi, Milito, Zhu & Addepalli, 2012)

## 1.1 Healthcare Monitoring System

The Internet of Things (IoT) market is experiencing continued growth, with a rapid increase in diverse connected devices including sensors, smartphones, smartwatches, intelligent vehicles, connected home appliances, and smart industrial equipment.(Sharif, Jung, Razzak *et al.*, 2021) (Beg *et al.*, 2022)

Based on current trends, experts anticipate that IoT connections will surpass 41 billion devices by 2030, demonstrating the massive scale of IoT adoption Figure 1.1 taken from IoT Analytics (2024).

As wireless devices become more accessible and powerful, they've enabled the development of diverse IoT applications - from real-time health monitoring and tracking to video surveillance and self-driving vehicle systems(Hassan, Gillani, Ahmed *et al.*, 2018). Many IoT applications require rapid response times, while others produce massive amounts of data that can strain network capacity.(Gaouar & Lehsaini, 2021).

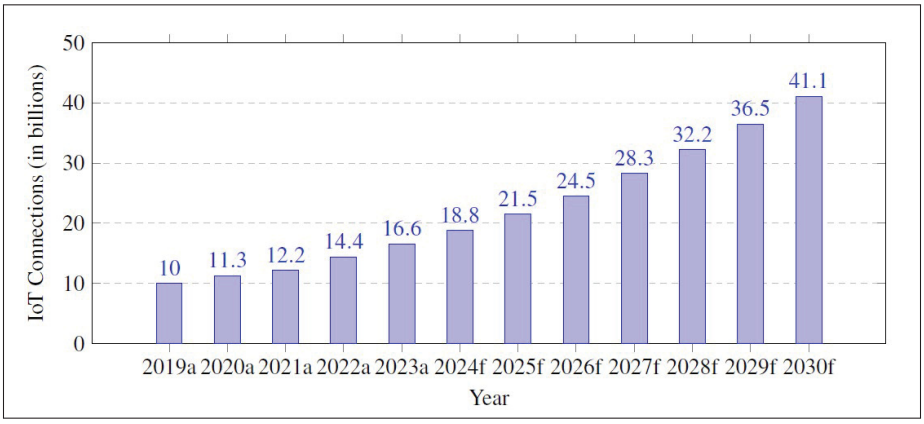


Figure 1.1 Number of global active IoT connections (installed base) in billions
Taken from IoT Analytics (2024)

The healthcare sector represents one of the most important areas for IoT expansion. IoT healthcare solutions help lower expenses while improving quality of life by enabling users to track daily activities - including diet, sleep patterns, and exercise - and receive personalized recommendations for maintaining better health (Cerina *et al.*, 2017). IoT has enhanced various aspects of healthcare, enabling real-time patient monitoring, efficient medical record management, emergency response systems, blood bank administration, and overall health monitoring (He & Zeadally, 2014). Healthcare IoT devices and sensors collect vast amounts of health data requiring processing and analysis. However, these sensor-equipped devices face constraints in power, memory, network capacity, and battery life, creating a need for efficient data computation, storage, access, and analysis systems (Khan & Salah, 2018).

Due to the time-sensitive nature of healthcare applications requiring quick responses, fog computing can provide real-time analysis of IoT device data with minimal processing delay (Verma & Sood, 2018). Healthcare monitoring has experienced continuous innovation through

new wearable IoT devices and services. However, these devices generally operate with limited resources, particularly regarding battery life and computational power.

## 1.2 Fog Computing

In modern life, the Internet plays a fundamental role by providing services and conveniences that simplify our daily routines. IoT devices enable these services, and their increasing adoption leads to a corresponding rise in data generation. The number of IoT devices worldwide is set to expand substantially, with IHS Markit projecting 41 billion devices by 2030 (Markit, 2017). To provide useful output, these devices must be capable of both executing and predicting data patterns (Yadav, Tripathi & Sharma, 2022).

Traditional cloud computing models won't be sufficient for future IoT devices' processing needs. The explosive growth in data production, combined with the physical distance to data centers, creates delays and reduces service quality. The current model of transmitting IoT data to distant centers risks overwhelming the cloud computing infrastructure. These challenges point to a need for bringing cloud capabilities closer to the source of data requests (Yadav *et al.*, 2022).

Cisco researchers introduced fog computing in 2010 (Bonomi *et al.*, 2012), a concept that enables users to connect with nearby processing devices. Instead of relying on distant cloud servers, fog computing processes tasks at the network's edge. By bringing cloud capabilities closer to the network's edge, applications can perform more reliably and efficiently (Memari, Mohammadi, Jolai & Tavakkoli-Moghaddam, 2022).

Fog computing integrates with cloud computing to resolve latency problems inherent in cloud technology, as illustrated in Figure 1.2 taken from (Beg *et al.*, 2022). This integration enables the creation of diverse IoT smart networks and devices. Fog Nodes (FNs) are positioned at the network's edge, bringing them closer to both smart devices and users. This strategic placement allows for data transmission, processing, and storage to occur with minimal delays since most operations happen near the smart devices at the network's edge.

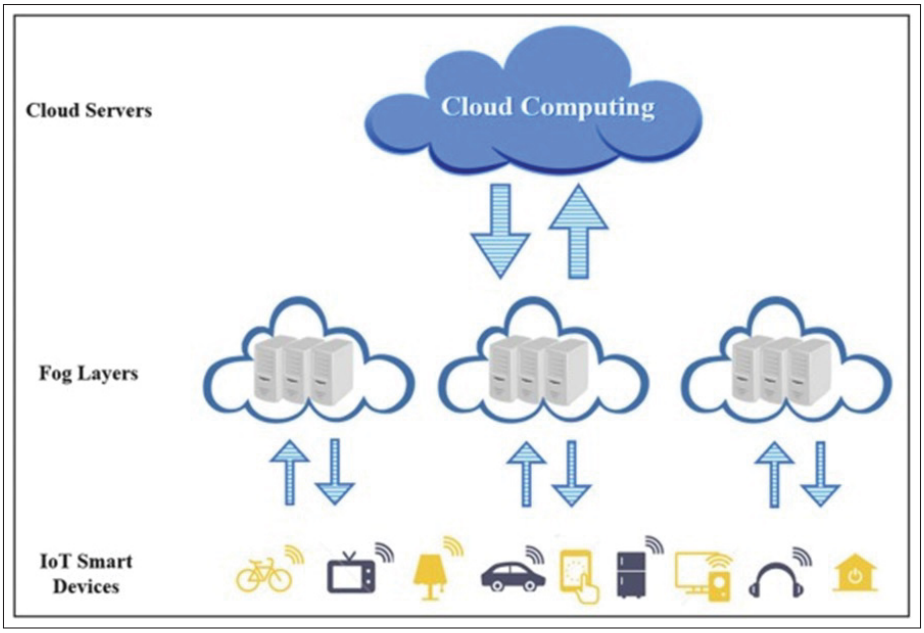


Figure 1.2 Fog Computing Architecture
Taken from Beg *et al.* (2022)

Fog computing layers help decrease IoT workloads on cloud data centers by providing virtual resources near IoT networks. Tasks from IoT devices can be transferred to fog computing nodes. Through task scheduling, these virtual fog computing resources can be managed efficiently(Najafizadeh, Salajegheh, Rahmani & Sahafi, 2022)

## 1.3 Task Scheduling

The implementation of fog computing layers near IoT networks provides virtual resources that decrease the load on cloud data centers. These fog nodes can handle tasks originally meant for IoT devices, and through strategic task scheduling, their virtual resources can be managed more effectively (Mohammad Hasani Zade & Mansouri, 2022).

The scheduling process assigns resources to tasks based on their priorities, aiming to reduce execution times and expenses while improving system performance. This process involves managing uncertainties in task flow, sequence, nature, and resource availability. Schedulers play a crucial role in selecting which process runs next from available options. The objectives

include workload optimization, cost reduction, energy efficiency, power usage maximization, reliability improvement, security risk minimization, and optimal virtual machine utilization (Hossain *et al.*, 2021). The organization and components of fog computing's task scheduling system are shown in Figure 3.1 taken from (Cvitić *et al.*, 2021)

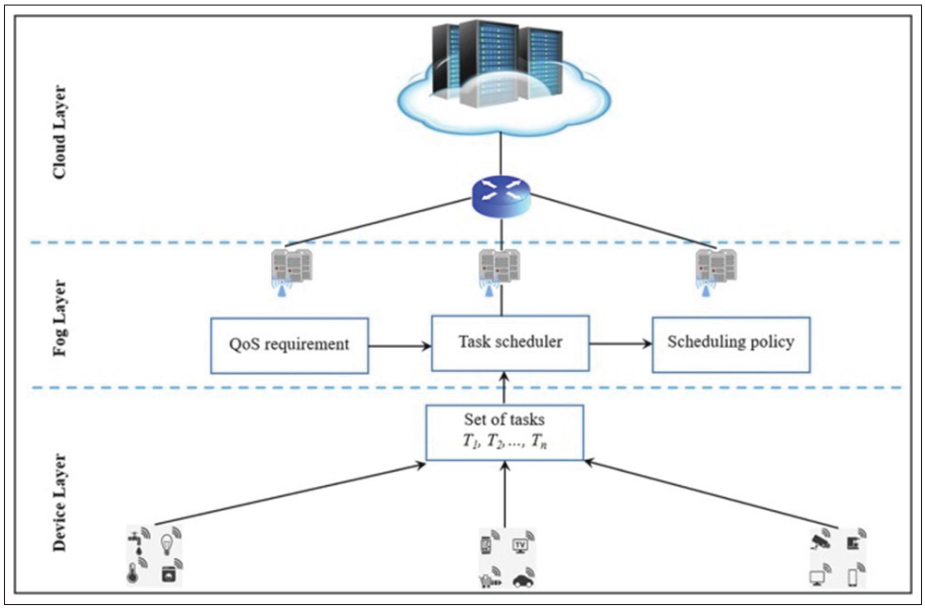


Figure 1.3 Fog Computing Architecture for task scheduling
Taken from Cvitić *et al.* (2021)

The process starts with users submitting job requests that contain system tasks. Scheduling these tasks requires considering multiple variables including deadlines, wait times, input processing, costs, and energy efficiency. A task scheduler works to optimize resource usage by matching tasks to available resources. This involves planning start and end times for each task while respecting system constraints, whether they're related to timing or resource availability. In fog and cloud computing environments, this scheduling can happen at various layers(Islam, Kumar & Hu, 2021).

Task scheduling efficiency can reduce power consumption in fog computing while enhancing throughput and decreasing costs. Though multiple heuristic techniques exist for scheduling tasks, the variety of resources and services involved makes it an NP-hard problem (Javaheri,

Gorgin, Lee & Masdari, 2022). Because standard heuristic algorithms prove inadequate for NP-hard problems, researchers utilize meta-heuristic approaches to quickly identify viable, if not perfect, solutions (Movahedi & Defude, 2021).

## 1.4 Meta Heuristic Algorithms

Engineering design typically involves balancing multiple objectives while dealing with complex nonlinear constraints. These objectives frequently conflict with one another, and perfect solutions are often impossible to achieve, necessitating compromises and approximate solutions (Cagnina, Esquivel & Coello, 2008). The complexity of design problems is further increased by numerous constraints, ranging from compliance with design codes and standards to limitations in materials, resources, and budgetary restrictions (Farina, Deb & Amota, 2004). Even single-objective global optimization problems prove challenging when dealing with highly nonlinear design functions. Many real-world problems are classified as NP-hard, meaning no efficient algorithmic solution exists. As a result, practitioners typically rely on heuristic algorithms combined with domain-specific knowledge to find solutions. Metaheuristic algorithms demonstrate significant effectiveness in handling such optimization challenges, with extensive coverage in both review literature and comprehensive textbooks (Blum & Roli, 2003).

Although challenging, multi objective optimization has developed numerous powerful algorithmic solutions that have proven successful in practical applications (Banks, Vincent & Anyakoha, 2008). Metaheuristic algorithms, often inspired by successful patterns found in biological systems and nature, have become increasingly important in multi objective global optimization (Adamatzky, Bull, De Lacy Costello, Stepney & Teuscher, 2007). Meta-heuristic algorithms effectively solve complex problems by finding optimal or near-optimal solutions quickly, even for large-scale issues. These algorithms work repeatedly searching around minimum or maximum functions using various operators (Houssein, Gad, Wazery & Suganthan, 2021). They achieve success by balancing detailed examination (exploitation) and broad searching (exploration). Metaheuristic optimization techniques have seen widespread adoption and research interest throughout the past two decades, demonstrating their versatility in solving complex problems

across numerous fields - from financial trading and economic modeling to engineering design, scheduling systems, image processing, and optimal control applications (Rodríguez-Esparza *et al.*, 2020). Meta-heuristics can tackle both single-objective and multi-objective optimization challenges, particularly when dealing with complex problem behaviors. When working with large datasets, traditional exact algorithms fall short due to their excessive computational demands and execution times. Meta-heuristic algorithms help solve NP-hard optimization problems by finding approximate solutions within practical timeframes (Dokeroglu, Sevinc, Kucukyilmaz & Cosar, 2019). Their popularity stems from their ability to handle random variables and black box problems. The removal of randomness allows meta-heuristics to flexibly switch between broad exploration and focused exploitation (Sharma & Tripathi, 2022).

### 1.4.1 Cuckoo Search algorithm

A metaheuristic search algorithm, called Cuckoo Search (CS), has been developed by Yang and Deb (2009) (Yang & Deb, 2009). The cuckoo family includes remarkable birds that combine beautiful vocalizations with sophisticated breeding strategies. Species like the ani and Guira cuckoos demonstrate this through their practice of communal nesting, though they often eliminate other birds' eggs to improve their own offspring's chances (Payne, Sorenson & Klitz, 2005). Various cuckoo species practice obligate brood parasitism, placing their eggs in other birds' nests, sometimes targeting different species. This parasitism appears in three forms: intraspecific, cooperative breeding, and nest takeover. Host birds may actively resist these intrusions; upon discovering foreign eggs, they either remove them or abandon their nests entirely. In response, some parasitic cuckoos, like the New World Tapera, have evolved sophisticated mimicry abilities, producing eggs that closely match their chosen host species' eggs in both color and pattern. By adopting these mimicry strategies, they improve their eggs' chances of survival and enhance their reproductive success.(Brown, Liebovitch & Glendon, 2007)

**Efficiency of Lévy flights:** Animals searching for food move in ways that can be described as random or near-random patterns. Each step in their foraging path represents a random walk, determined by their present location and the probability of moving in various directions

(Reynolds & Frye, 2007). These movement choices follow mathematical principles, and research shows that many animals and insects move in patterns that match the properties of Lévy flights (Barthelemy, Bertolotti & Wiersma, 2008). According to Reynolds and Frye, fruit flies (Drosophila melanogaster) move through space using a combination of straight flights and abrupt 90-degree turns, producing an intermittent scale-free search pattern that mirrors Lévy flight behavior. Light behavior also demonstrates properties that can be connected to Lévy flights (Pavlyukevich, 2007). This type of behavior has been adapted for optimization and search algorithms, with early results showing promising potential. In general terms, Lévy flights represent a type of random walk where the length of each step follows the Lévy distribution, typically described using a basic power-law formula (Yang, 2010)

Based on encouraging preliminary results that suggest better performance than PSO and other existing algorithms, we will expand CS to create a multi objective cuckoo search algorithm. Our approach involves first validating the algorithm against multi objective test functions, then implementing it for engineering design optimization applications. We'll briefly discuss the remarkable breeding strategies employed by some species of cuckoo birds. Based on cuckoo breeding behavior, the cuckoo search algorithm generates new solutions using two approaches: Lévy random walk (LRW) and bias random walk (BRW) (Yang & Deb, 2009). The LRW process follows a Lévy flight pattern as described in Equation:

$$X_i^{d+1} = X_i^d + \alpha \oplus \text{Lévy}(\eta) \tag{1.1}$$

In this equation, $X_i^d$ represents the $i$th solution in generation $t$, while $X_i^{d+1}$ indicates the $i$th new solution in the next generation $(d+1)$. The step size is denoted by $\alpha$, entry-wise multiplications are shown by $\oplus$, and $\text{Lévy}(\gamma)$ represents a random value drawn from the Lévy distribution. To simplify implementation $\text{Lévy}(\gamma)$ is calculated as:

$$\text{Lévy}(\eta) \sim \frac{\phi \times u}{|v|^{1/\eta}} \text{ and } \phi = \left( \frac{\Gamma(1+\eta) \times \sin \frac{\pi \times \eta}{2}}{\Gamma(\frac{1+\eta}{2}) \times \pi \times 2^{\frac{\eta-1}{2}}} \right)^{1/\eta} \tag{1.2}$$

In this equation, $u$ and $v$ represent random numbers drawn from a uniform distribution, and $\Gamma$ denotes a gamma function. When generating new solutions through LRW, they should be created near the current best solution, we can adapt Equation to become:

$$X_i^{d+1} = X_i^d + \alpha_0 \times \frac{\phi \times u}{|v|^{1/\eta}} \times (X_i^d - X_{best}) \tag{1.3}$$

With $\eta$ fixed at $\frac{3}{2}$ and scaling factor $\alpha_0$ at 0.01 (as recommended), and $X_{best}$ representing the current optimal solution, the equation is complete. The BRW component conducts local searches using biased random movements with varying step sizes, defined by Equation:

$$X_i^{d+1} = X_i^d + \alpha \oplus H(p_\alpha - \epsilon) \oplus (X_p^d - X_i^d) \tag{1.4}$$

The formula uses two randomly chosen population members ($p$ and $q$), along with random numbers $\alpha$ and $\epsilon$ that fall within $[0, 1]$. It also includes a fraction probability $\rho_\alpha$ and a step function $H$.

### 1.4.2 Grey Wolves Algorithm

The Grey Wolf Optimizer (GWO), initially created as an effective single-objective optimization algorithm, was later enhanced to handle multiple objectives as described in (Mirjalili, Mirjalili & Lewis, 2014). Due to GWO's advantage of not requiring hyper-parameters, researchers have developed multiple versions of multi-objective grey wolf optimizer (MOGWO). As demonstrated in (Mirjalili, Saremi, Mirjalili & Coelho, 2016) and (Liu, Yang & Li, 2020), these variations have been successfully applied across different fields, showcasing GWO's capability to handle complex engineering challenges. GWO was enhanced by incorporating non-dominated sorting (NS), a well-established method for handling multi-objective problems, creating NSGWO which was tested on engineering design challenges (Jangir & Jangir, 2018). However, despite the success of various MOGWO versions in certain applications, they share a

common limitation: the risk of getting stuck in local optima because GWO only uses current best solutions to update other solutions (Long *et al.*, 2019). Based on grey wolves' social hierarchy and hunting behavior, this meta-heuristic algorithm simulates how leader wolves direct the pack during hunts. The process progresses through searching and encircling phases before the final attack, with the prey representing the optimization's global optimal solution. The mathematical model of this hunting process is showing in Figure 1.4 taken from (Cvitić *et al.*, 2021).

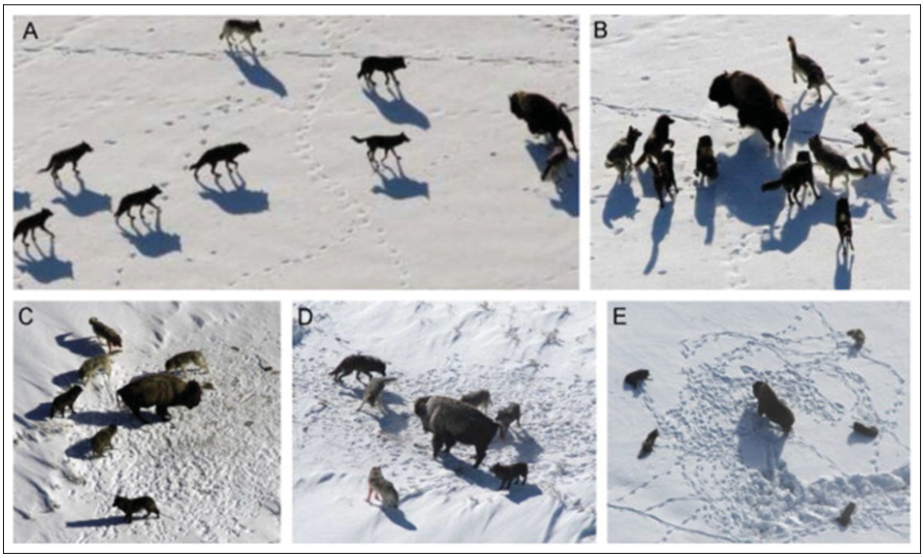


Figure 1.4 The hunting grey wolves' procedure: (A) Tracking and approaching prey (B–D) Pursuit, encircle and assault (E) The end of the hunt
Taken from Cvitić *et al.* (2021)

The algorithm initializes with N solutions (each representing a grey wolf) containing d design variables. In iterative steps, pack leaders estimate prey location, followed by position updates from other wolves based on prey distance. The process terminates upon reaching maximum iterations, with comprehensive details provided in Mirjalili *et al.* (2014)

## 1.5 Task Sequencing and Prioritization

The effectiveness of scheduling heuristics is strongly influenced by how jobs are differentiated and sequenced. Various successful scheduling approaches have emerged based on job sequencing, including SPT (Shortest Processing Time), FCFS (First Come First Serve), LPT (Longest

Processing Time), as well as algorithms by Page (1961), Palmer (1965), and the NEH heuristic by Nawaz et al. (1983). Despite these developments, determining job priorities remains challenging in task scheduling, particularly given today's customized mass production requirements.

### 1.5.1 NEH Heuristic

The NEH heuristic introduced by Nawaz et al. (Nawaz, Enscore Jr. & Ham, 1983) has long been considered the leading solution method for the classic permutation flow shop problem. The general problem involves scheduling $n$ jobs $(1, 2, \ldots, n)$ that are all available at start time, across $m$ sequential machines $(M_1, M_2, \ldots, M_m)$. Jobs must be processed in order through all machines, starting with $M_1$ and ending with $M_m$. The process has several constraints: machines can only handle one job at a time, jobs cannot be interrupted once started, processing times include all setup requirements, and there are no storage limitations between machines. This scheduling challenge, denoted as $F_m||C_{\max}$, aims to minimize the completion time of the final job on the last machine (makespan). A permutation schedule maintains identical job ordering across all machines, and finding the optimal sequence for this type is known as $F_m \mid \text{prmu} \mid C_{\max}$. While permutation schedules can provide optimal solutions for two or three machines, systems with more than three machines may require different job sequences on different machines for optimal results. Research by Potts et al. (Potts, Shmoys & Williamson, 1991) examines these variations, while Garey et al. (Garey, Johnson & Sethi, 1976) proved that the three-machine permutation flow shop problem ($F_3 \mid \text{prmu} \mid C_{\max}$) belongs to the strongly NP-hard class. In contrast, Johnson (Johnson, 1954) developed an algorithm that solves the two-machine case ($F_2||C_{\max}$) with $O(n \log n)$ time complexity.

The variable $p_{ij}$ defines processing time for job $j$ on machine $M_i$ $(i = 1, 2, \ldots, m; j = 1, 2, \ldots, n)$. When calculating the makespan $C_{\max}(\pi)$ for a job sequence $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$, we find the longest path in an acyclic network, as Pinedo (Pinedo, 2002) describes. For the basic sequence $\pi = (1, 2, \ldots, n)$, Figure 1.5 taken from (Kalczynski & Kamburowski, 2007). displays

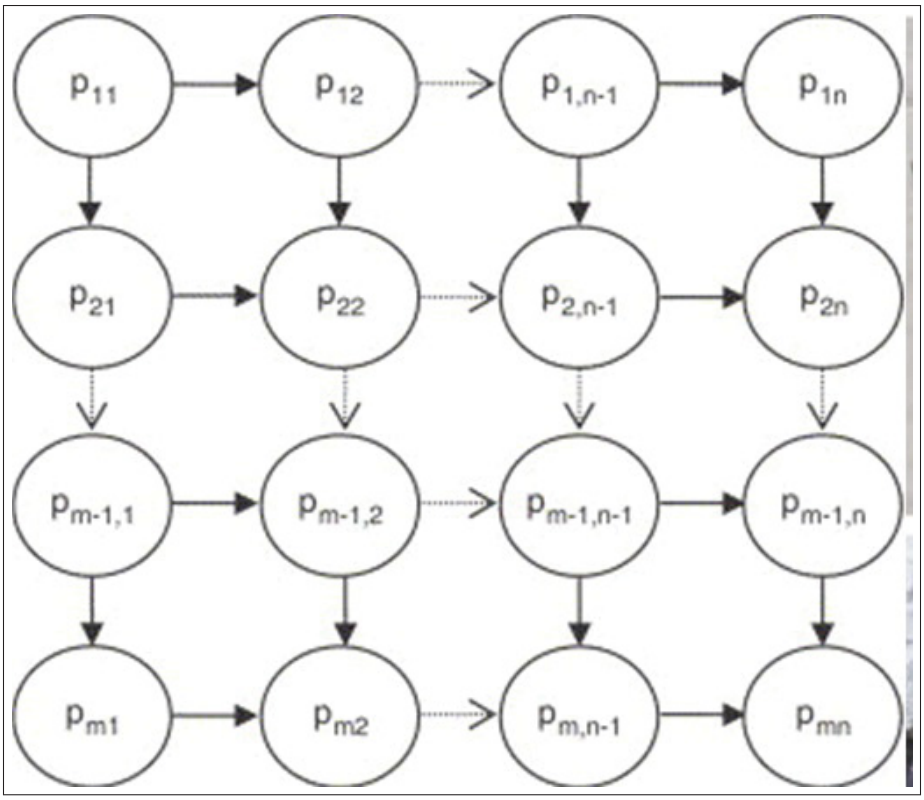


Figure 1.5 Network for computing the makespan
Taken from Kalczynski & Kamburowski (2007)

this network with nodes showing processing times. Makespan calculation takes $O(mn)$ time using the critical path method (CPM).

### 1.5.2 Brief Introduction of Q-Learning

While operations research has traditionally focused on exact, heuristic, and metaheuristic approaches, recent applications of reinforcement learning from the AI field show promise in addressing task scheduling challenges, as demonstrated by (Kayhan & Yildiz, 2021). In contrast to metaheuristic approaches that rely on complex mathematical optimization, reinforcement learning tackles task scheduling's sequential decisions through interactive learning with the environment to achieve specific goals, as described by (Sutton & Barto, 2018). The first application of reinforcement learning to flow-shop scheduling was conducted by Stefan (2003), who developed

a scheduler using Q-learning to approximate optimal solutions. (Stefan, 2003) Zhang et al. (2013) introduced an online TD ($\lambda$) algorithm for flowshop scheduling problems and concluded that reinforcement learning showed significant promise in this field, warranting additional research. (Zhang, Wang, Zhong & Hu, 2013) Recent research by (Brammer, Lutz & Neumann, 2022) applied reinforcement learning to PFSP involving multiple lines and demand plans. Their numerical evaluations demonstrated that this approach outperformed traditional constructive and iterative heuristics. (Karimi-Mamaghan, Mohammadi, Pasdeloup & Meyer, 2022) combined Q-learning with an iterated greedy algorithm to optimize perturbation operator selection in PFSP, proving its effectiveness through extensive comparisons with benchmark algorithms.

### 1.5.3 Q-NEH (Q-learning NEH)

Though Q-learning effectively solves small-scale task scheduling problems with makespan criteria without requiring a model, it has difficulty achieving stable or optimal solutions for larger instances with expansive state-action spaces. Daqiang et al. proposed enhancing Q-learning by incorporating NEH's established operations research knowledge, designed to improve learning efficiency and convergence rates for large-scale problems (Guo *et al.*). NEH was chosen because it's recognized as one of the most effective and straightforward approximation methods for makespan-based PFSP, as confirmed by (Ruiz & Maroto, 2005).

The proposed Q-NEH algorithm's fusion mechanism centers on incorporating NEH's proven domain expertise to enhance the agent's action selection process, using knowledge reinforcement to amplify effective actions while diminishing ineffective ones(Fernandez-Viagas, Ruiz & Framinan, 2017). The Q-NEH algorithm differs from standard Q-learning in its action implementation. While Q-learning simply takes action $a_t$ in state $s_t$, receiving reward $r_{t+1}$ and producing permutation $\tau_{s_t,a_t,s_{t+1}} = (\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_t)$, the Q-NEH algorithm incorporates NEH's domain knowledge through knowledge reinforcement. It does this by inserting action $a_t$ into $t$ possible positions within the previous permutation $\tau_{t-1} = (\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{t-1})$, generating $t$ potential permutations $\{\tau_t^1, \tau_t^2, \tau_t^3, \ldots, \tau_t^i, \ldots, \tau_t^t\}$. Here, $\tau_t^j = \tau_t$ and $\tau_t^i$ represents inserting action $a_t$ at

position $i$ in permutation $\tau_{t-1}$. The optimal permutation is selected using the criterion:

$$\tau_t^f = \arg \min_{C(\tau_i^t,m) \leqslant C(\tau_i^t,m)} \{\tau_i^t\} \tag{1.5}$$

The reward function is enhanced with an additional reinforcement reward when $\tau_t^f$ outperforms $\tau_t^i$ during the transition from state $s$ to $s'$ through action $a$ under knowledge reinforcement. This relationship is formulated as:

$$R^f(s_t, a_t, s_{t+1}) = R(s_t, a_t, s_{t+1}) + \frac{C(\tau_i^t, m) - C(\tau_t^f, m)}{C(\tau_i^t, m)} \tag{1.6}$$

The proposed Q-NEH algorithm follows this process while interacting with the task scheduling environment:

$$Q^f(s_t, a_t) \leftarrow Q^f(s_t, a_t) + \alpha \left[R^f(s_t, a_t, s_{t+1}) + \gamma \max Q^f(s_{t+1}, a_{t+1}) - Q^f(s_t, a_t)\right] \tag{1.7}$$

The Q-NEH algorithm refines its action selection through two mechanisms: knowledge reinforcement (which enhances good actions and diminishes bad ones) and an $\varepsilon$-greedy strategy for balancing exploration and exploitation, specifically:

$$a_t = \begin{cases} \arg\max_a \{Q^f(s_t, a)\} & \text{if } X > \varepsilon \\ a_{random} & \text{otherwise} \end{cases} \tag{1.8}$$

# CHAPTER 2

# LITERATURE REVIEW

This chapter presents a comprehensive review of the literature relevant to task scheduling in fog computing for Healthcare Monitoring Systems, with a specific focus on approaches aimed at optimizing makespan. The review encompasses several interconnected areas of research, providing a foundation for understanding the complex landscape in which the proposed hybrid algorithm is situated.

## 2.1 A Review of Task Scheduling Algorithms in Fog Computing Environments for Healthcare Systems

The literature review explores existing studies related to fog computing, task scheduling and healthcare monitoring, evaluating their contributions and contrasting their results with the proposed system.

(Paul *et al.*, 2018) proposed a three-tiered health monitoring architecture combining cloud and fog computing with biomedical sensors. The fog layer serves as an intermediary, collecting and analyzing edge device data before using a task scheduling algorithm to allocate work between fog and cloud resources. Through simulation, they demonstrated their system's advantages over traditional cloud-only solutions by examining metrics like network usage, time delays, and power consumption. The optimization of resource preparation is essential to achieve maximum utility and increased advantages for providers of fog and cloud services proposed in (Kabirzadeh, Rahbari & Nickray, 2017). High-priority tasks should be given advanced reservation status, allowing them to preempt best-effort tasks. When resources are fully occupied and new high-priority requests arrive, they must be queued on a waiting list. Task placement must consider priority values, and the QoS-aware system should focus on optimizing processing times for high-priority tasks rather than trying to maintain all tasks equally presented by (Mahmud *et al.*, 2019). Task criticality is determined by various parameters, such as priority, deadlines, and thresholds. (Fellir *et al.*, 2020) focused To address these factors alongside scheduling, balancing, interoperability, and resource availability challenges, organizations can implement multi-agent

systems - networks of interconnected agents that communicate with each other. Conventional scheduling methods for critical emergency healthcare tasks fail to efficiently distribute and manage resources, resulting in unpredictable and inequitable resource allocation presented by (Islam *et al.*, 2021). Medical devices face computational resource limitations that hinder their ability to effectively store and retrieve emergency healthcare data in real-time(Powell, Desiniotis & Dezfouli, 2020). The third challenge relates to concerns with task load balancing and prioritization when dealing with inadequate computing resources and telecommunication networks while obtaining the best scheduling of emergency healthcare tasks introduced by (Talaat *et al.*, 2020). Processing and analyzing emergency patient data demands substantial computational power and network resources. Previous research has examined various task scenarios - from single to multiple tasks, and both dependent and independent task placement - across cloud, fog, or integrated cloud-fog architectures. (De Maio & Kimovski, 2020) presented effective scheduling system and balancing mechanisms to ensure efficient coordination. (Maiti *et al.*, 2022) focused on critical challenge in multi-tier computation models that is developing better task scheduling approaches that balance computational workloads across nodes while optimizing resource utilization and system efficiency.

(Rezazadeh, Rezaei & Nickray, 2019) developed LAMP, an algorithm that considers latency when selecting optimal fog nodes by evaluating both resource availability and delay times in healthcare applications.(AlZailaa *et al.*, 2021) developed an e-health task scheduling algorithm that uses payload semantic analysis to establish task priorities. (Daraghmi *et al.*, 2022) proposed a three-tiered architectural system for remote health monitoring that utilizes Narrow Band IoT (NBIOT) technology. (Hajvali *et al.*, 2022) proposed an integrated healthcare architecture using IoT, fog, and cloud computing technologies, focusing on supporting user movement while meeting non-functional system needs. (Gupta & Chaurasiya, 2022) developed a health monitoring system that combines IoT and fog computing technologies with a Bayesian Belief Network. (Jasim & Al-Raweshidy, 2023) introduced HMAN, a hierarchical healthcare architecture that coordinates processing and offloading of vital signs across edge, fog, and cloud layers. However, their system doesn't address user and service priority management.

In(Ahmed *et al.*), the study introduced DMFO-DE, an optimization algorithm that combines

opposition-based discrete Moth-Flame Optimization with Differential Evolution to improve convergence and avoid local optima problems. Applied to fog computing workflow scheduling, the system uses DVFS and HEFT algorithms to manage task prioritization and resource allocation. The solution aims to optimize both scheduling efficiency and energy consumption, with experimental results showing DMFO-DE's superior performance in virtual machine management and energy usage. research by (Azizi *et al.*, 2022) presented two innovative scheduling algorithms for IoT tasks, specifically addressing timing constraints and fog node energy efficiency. By formulating the problem as a mixed integer nonlinear programming model, they targeted the dual optimization of energy consumption and deadline adherence. Their proposed algorithms - PSG-M and priority-aware semi-greedy (PSG) - implemented semi-greedy strategies with multi-start capabilities, achieving significantly better task completion rates compared to contemporary scheduling approaches. (Talaat *et al.*, 2020) developed a novel load balancing solution called EDLB, combining CNN technology with modified PSO algorithms. Their healthcare-focused fog computing framework consists of three essential components: FRMs for resource monitoring, CBCs for server classification, and ODS for process scheduling. The system excelled in multiple performance metrics, including resource efficiency, processing speed, and load distribution, while maintaining real-time healthcare information processing capabilities. This integrated approach demonstrated significant improvements over existing load balancing methods.

(Hussain & Begh, 2022) introduced HFSGA, a hybrid scheduling model integrating Genetic Algorithm and Flamingo Search Algorithm. This system aims to optimize three key factors: computation costs, communication costs, and deadline compliance. By addressing QoS requirements, the model effectively eliminates latency and congestion problems, resulting in improved makespan, better deadline satisfaction, and reduced operational costs.(Hosseinioun, Kheirabadi, Tabbakh & Ghaemi, 2020) proposed an energy-conscious system utilizing DVFS technology to optimize power consumption in fog computing. Their approach enables processors to reduce power usage during idle periods through voltage and frequency adjustments. By incorporating a hybrid evolutionary algorithm combining IWO-CA, they achieved optimal task sequencing, with experiments showing improved energy efficiency over conventional methods.

(Subbaraj & Thiyagarajan, 2021) introduced a QoS-focused task-resource mapping model for fog computing that considers multiple performance metrics including MIPS, RAM, storage, latency, bandwidth, trust, and cost. They implemented two decision-making approaches: one using only Analytic Hierarchy Process (AHP) for both priority weighting and fog device ranking, and another combining AHP for weight calculation with TOPSIS for device ranking. Their scheduling algorithm, which integrates performance, security, and cost considerations, assigns tasks based on device rankings to optimize fog computing performance. (Abd Elaziz, Abualigah & Attiya, 2021) developed AEOSSA, combining modified artificial ecosystem-based optimization with Salp Swarm Algorithm to schedule IoT requests in cloud-fog systems. They enhanced AEO's exploitation capabilities by incorporating modified SSA operators. The algorithm's effectiveness was tested using both synthetic and real-world datasets of varying sizes, with performance comparisons against established meta-heuristic methods showing AEOSSA's superiority in terms of makespan time and throughput.

A comparative analysis of fog computing scheduling schemes, presented in Table 2.1, reveals that most approaches utilize hybrid meta-heuristic algorithms. However, existing research lacks a comprehensive solution that simultaneously addresses makespan, cost, and energy considerations. This research gap suggests an opportunity to develop a new version of the AHA algorithm for task scheduling, combined with an AHP model for task prioritization in fog environments.

Table 2.1 Comparison of scheduling factors and approaches in related works

| References | Scheduling factors | Approach |
|---|---|---|
| Ahmed et al. | - Makespan<br>- Power Usage | Hybrid MFO-DE with Opposition-Based Learning |
| Azizi et al. | - Time Constraint Violations<br>- Power Usage | PSG Algorithm with Multi-Start Enhancement |
| Talaat et al. | - Data Storage Capacity<br>- Processing<br>- Memory Resources | Integrated CNN-PSO Framework |
| Hussain and Begh | - Resource Expenses | Combined Flamingo-Genetic Algorithm |
| Hosseinioun et al. | - Execution time<br>- Power Efficiency | Cultural-Invasive Weed Optimization Hybrid |
| Subbaraj and Thiyagarajan | - Memory and Storage<br>- Network Performance<br>- Bandwidth Efficiency<br>- Resource Costs<br>- System Reliability | Multi-Criteria Decision Making (AHP-TOPSIS) |
| Elaziz et al. | - Makespan | Enhanced Ecosystem Optimization |

# CHAPTER 3

# DESIGN AND IMPLEMENTATIONS

This section consists of two subsections that collectively define the system model. The first subsection introduces the general architecture of the IoT-fog-cloud system. The second subsection presents a mathematical formulation of the task scheduling problem within this model.

## 3.1 Architecture

The high-level system architecture of the IoT-fog-cloud environment is illustrated in Figure 3.1 taken from (Cvitić *et al.*, 2021). This architecture is composed of four distinct parts: IoT devices, gateways, the fog environment, and the cloud environment. The following sections will explain each of these components in detail.

- **IoT Devices**: The IoT healthcare devices, such as sensors, wearables, and monitoring equipment, are dispersed across various locations. These devices generate large amounts of time-sensitive data that needs to be processed rapidly. For instance, a healthcare monitoring system must collect and evaluate patient data in a timely manner, as any latency could be detrimental.
- **Smart Gateways**: The computation tasks offloaded from IoT devices are received by the smart gateways or access points located at the edge of the network. Based on the characteristics of the tasks (Adhikari, Mukherjee & Srirama, 2020), such as their priority and deadline, the gateways determine whether to submit the tasks for processing in the distributed fog nodes or the centralized cloud servers. This decision-making process (Guevara, Torres & da Fonseca, 2020)can be implemented using various methods, including machine learning techniques and data mining algorithms.
- **Task Manager**: The Task Manager receives various jobs, each with different lengths and memory needs. This is because the tasks are prioritized based on the number of associated devices and the information they contain. The Task Priority Calculating Server uses the Q-NEH algorithm to assign priority to the tasks.

- **Fog Environment**: The fog environment consists of a fog controller or broker, as well as the fog environment itself. These fog controllers are essential components of fog service providers, responsible for managing fog resources and scheduling tasks using appropriate algorithms. Within the fog network, devices are widely distributed across different geographical locations, including high-end servers, set-top boxes, routers, personal computers, and smart phones. These fog nodes share common computing, storage, and networking capabilities that enable them to perform IoT tasks (Muro, Escobedo, Spector & Coppinger, 2011). Fog gateways temporarily store the submitted tasks in a buffer, and the fog controller periodically runs a task scheduler algorithm based on the information reported by the fog nodes and the requirements of the tasks.
- **Cloud Environment**:The cloud environment is primarily composed of a set of virtual machines (VMs) with high computing power and storage capacity (Azizi, Zandsalimi & Li, 2020). The cloud VMs are generally more suitable for executing latency-tolerant and computationally intensive tasks compared to the fog nodes (FNs). This is because the cloud infrastructure has greater computing and storage resources at its disposal than the distributed fog nodes.(Calheiros, Ranjan & Buyya, 2011)

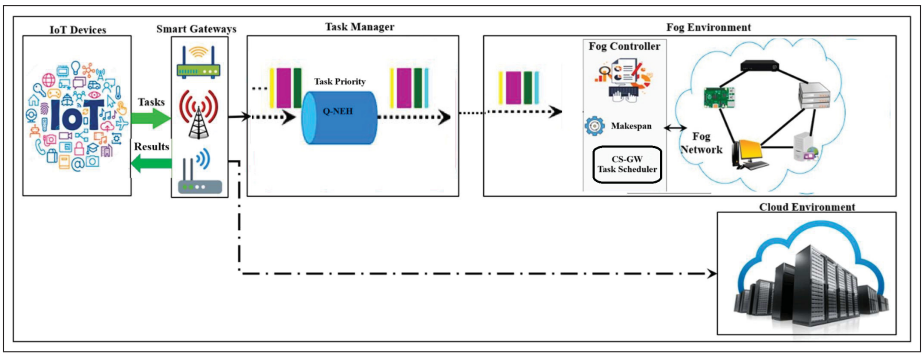


Figure 3.1 Fog Computing Architecture for task scheduling
Taken from Azizi *et al.* (2022)

## 3.2 Problem formulation

We present a structured approach to modeling task scheduling in heterogeneous fog computing environments. Our discussion covers the essential problem components, analyzes makespan as a

performance metric, and concludes with a comprehensive overview of the scheduling challenges addressed by our model.

### 3.2.1 Basic Elements

In our cloud-fog architecture, we define a task set T = T1, T2, ..., Tn containing n independent tasks, each characterized by length, memory, and CPU parameters.

The heterogeneous fog layer consists of m nodes, denoted as F = F1, F2, F3, ..., Fm. Each fog node Fj is defined by its operational attributes including processing capacity, resource usage costs (CPU and memory), and power consumption characteristics in both active and idle states

### 3.2.2 Makespan

One critical performance measure in computing environments is makespan - the total time span from when processing begins until all scheduled tasks are fully completed. This metric is particularly important because it reflects the overall efficiency of the system's task scheduling. By optimizing the schedule to minimize this total completion time, we can prevent bottlenecks and delays that often occur when long-running tasks are poorly positioned in the sequence. In our fog computing research, we specifically address how task scheduling algorithms can be improved to achieve the shortest possible makespan, which ultimately leads to better resource utilization and faster overall processing times. This optimization is crucial for maintaining system performance and meeting service level agreements in modern distributed computing environments. The calculation of makespan follows the formula given in Equation 3.1, as referenced in (Azizi *et al.*, 2022).

$$\mathrm{M}S = \max_{\forall j \in F}(ET_j) \tag{3.1}$$

As follows, $ET_i$ represents the expected time of task $T_i$'s execution on fog node $F_j$ (Azizi *et al.*, 2022):

$$\mathrm{ET}_i = \sum_{j \in F} (d_{ij} \times a_{ij}), \forall j \in F \tag{3.2}$$

The variables in the equation 3.2 are defined as:

- $d_{ij}$: execution time for task $T_i$ on node $F_j$
- $a_{ij}$: decision variable (binary assignment indicator)

The execution time $d_{ij}$ is calculated as 3.3:

$$\mathrm{d}_{ij} = \frac{T_i}{F_j}, \forall i \in T, \forall j \in F \tag{3.3}$$

- $i \in T$ (set of all tasks)
- $j \in F$ (set of all fog nodes)

Where:

- $T_i$ is task length (measured in MI - Million Instructions)
- $F_j$ is processing speed (measured in MIPS - Million Instructions Per Second)

The assignment matrix $A_{n \times m}$ uses binary values $a_{ij}$ defined as 3.4:

$$\mathrm{a}_{ij} = \begin{cases} 1 & \text{if task } T_i \text{ is assigned to FN } F_j \\ 0 & \text{if task } T_i \text{ is not assigned to FN } F_j \end{cases}, \forall i \in T, \forall j \in F \tag{3.4}$$

This applies for all tasks $i \in T$ and all fog nodes $j \in F$.

## 3.3 The proposed algorithm

In this section, we propose CS-GW (Cuckoo Search-Grey Wolf), a hybrid optimization algorithm designed to address task scheduling challenges in fog computing environments. The algorithm incorporates Q-NEH (Q-learning with NEH heuristic) to handle task prioritization during the

scheduling process. CS-GW aims to find optimal solutions for the scheduling optimization problem and is executed periodically by the Fog Controller (FC). A comprehensive description of our proposed algorithm follows.

### 3.3.1 Q-NEH (Q-learning NEH)

In line with the Q-learning algorithm framework, Figure 3.2 taken from Fang *et al.* (2024) illustrates the interaction between the agent and the fog computing environment. During each discrete time interval, the agent performs an action, obtains a reward, and transitions to a subsequent state.

Specifically, during each time step $t$, the agent observes the current state $S_t \in S$ within the fog environment, where $S$ represents all possible system states, comprising $2^n$ potential configurations of $n$ unprocessed tasks. Based on this observation, the agent chooses an action $a_t$ from the action set $A$, which contains all possible actions for the $n$ remaining unscheduled tasks. Following the action's execution, the system transitions to a new state $s_{t+1} \in S$, and the agent receives a reward $r_{t+1} \in \mathbb{R}$ according to the defined policy $\pi$.

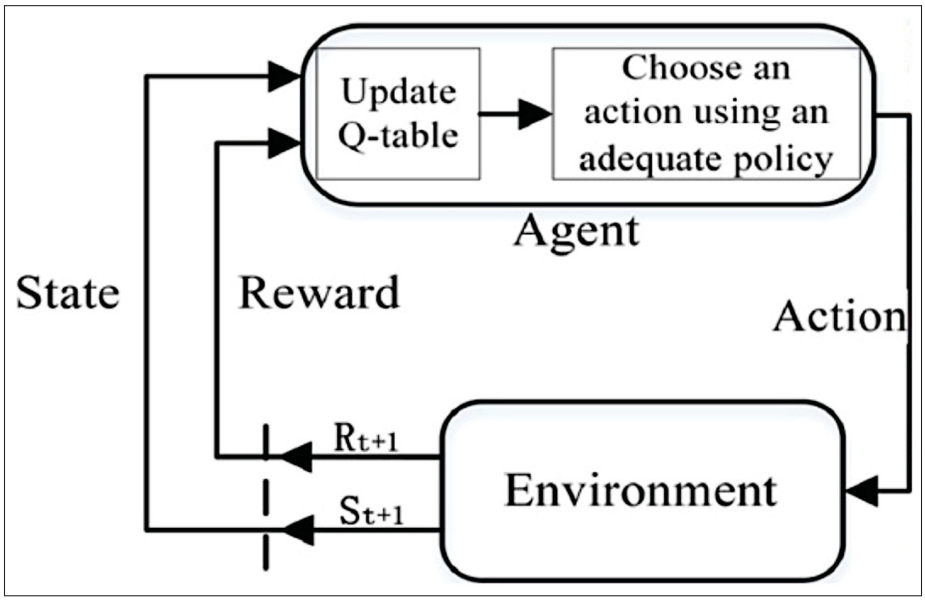


Figure 3.2 A rough framework of QL algorithm
Taken from Fang *et al.* (2024)

We present an innovative approach combining Q-learning with NEH's established domain expertise to tackle the task scheduling problem's makespan optimization. NEH was selected due to its recognized effectiveness and straightforward application in addressing task scheduling makespan challenges. The Q-NEH algorithm, shown in Figure 3.3 taken from Guo *et al.* (2024), combines Q-learning with NEH domain expertise to address TSP. The knowledge enhancement follows NEH's core principle: jobs requiring longer total processing time across all machines should be prioritized over those with shorter durations.

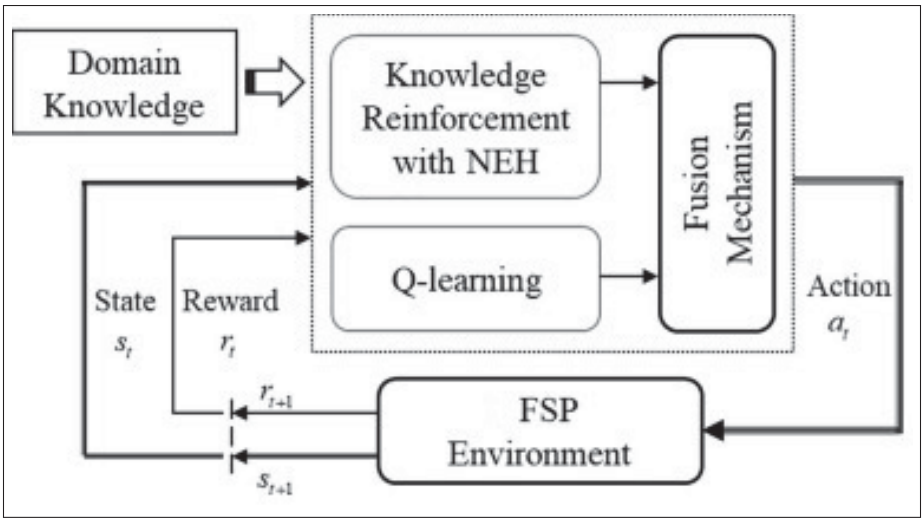


Figure 3.3 The fusion of Q-learning and domain knowledge of NEH for the FSP
Taken from Guo *et al.* (2024)

The Q-NEH algorithm's fusion mechanism enhances traditional Q-learning by incorporating NEH's domain expertise to optimize action selection, reinforcing effective actions while diminishing suboptimal ones. Unlike standard Q-learning, which directly executes action $a_t$ in state $s_t$ to receive reward $r_{t+1}$ and generate permutation $\tau_{s_t,a_t,s_{t+1}} = (a_1, a_2, a_3, \ldots, a_t)$, the Q-NEH algorithm implements knowledge reinforcement by testing action $a_t$ in all possible $t$ positions within the previous permutation $\tau_{t-1} = (a_1, a_2, a_3, \ldots, a_{t-1})$. This generates $t$ potential permutations $\tau_t^1, \tau_t^2, \tau_t^3, \ldots, \tau_t^i, \ldots, \tau_t^t$, where $\tau_t^t = \tau_t$ and $\tau_t^i$ represents inserting action $a_t$ at position $i$ in permutation $\tau_{t-1}$. The optimal permutation is then chosen based on specific criteria.

## 3.4 Hybrid Cuckoo Search and Grey Wolf Algorithm

The Cuckoo Search Algorithm is a nature-inspired metaheuristic optimization algorithm that mimics the brood parasitism behavior of cuckoo birds combined with Lévy flight patterns. In this algorithm, each cuckoo represents a potential solution to the optimization problem. The algorithm works by having cuckoos lay their eggs (solutions) in randomly chosen nests of other host birds, with each nest representing a candidate solution. The key feature that makes this algorithm particularly effective is its use of Lévy flights for generating new solutions. Lévy flight is a random walk pattern characterized by taking mostly small steps but occasionally making larger jumps, following a probability distribution that has a power-law tail (specifically, a heavy-tailed distribution). This movement pattern is observed in many natural phenomena and helps the algorithm maintain a balance between local exploitation (small steps for fine-tuning solutions) and global exploration (occasional long jumps for discovering new promising regions in the search space). The quality of the solutions is evaluated, and the worst nests (poorest solutions) have a probability of being discovered and abandoned by the host birds, leading to new nests being built at new locations. This combination of Lévy flights and selective abandonment of poor solutions helps the algorithm effectively navigate the search space and avoid getting trapped in local optima while searching for the global optimum solution.

The Grey Wolf Optimizer (GWO) is a population-based metaheuristic algorithm inspired by the social hierarchy and hunting behavior of grey wolf packs in nature. The algorithm mimics the strict social leadership hierarchy of wolves, where the pack is led by alphas ($\alpha$), followed by betas ($\beta$), deltas ($\delta$), and omegas ($\omega$) in descending order of dominance. In GWO, the optimization process is guided by the first three best solutions: alpha represents the best solution, beta the second-best, and delta the third-best solution, while the rest of the candidate solutions are considered as omega wolves.

The hunting strategy is modeled in three main phases: encircling prey (potential solutions encircle the prey by updating their positions), hunting (guided by the $\alpha$, $\beta$, and $\delta$ wolves), and attacking prey (exploitation) or searching for prey (exploration). During optimization, the wolves

update their positions based on their distance from the prey, with movement coefficients that gradually decrease from 2 to 0 to simulate the wolves approaching the prey. This decreasing coefficient helps balance between exploration (searching widely in the solution space) and exploitation (fine-tuning in promising areas). The algorithm's success lies in its ability to maintain diversity through the hierarchical leadership structure while effectively converging toward optimal solutions through the coordinated hunting behavior.

In this research, we present a novel hybrid optimization algorithm that combines Grey Wolf Optimization (GWO) and Cuckoo Search (CS) to address task scheduling challenges while mitigating the individual limitations of each algorithm. The GWO component, inspired by the social hierarchy and hunting strategies of grey wolves, contributes its exceptional multi-leader structure and robust local optima avoidance capabilities. Meanwhile, the CS algorithm, inspired by the brood parasitism of cuckoo birds, enhances the hybrid approach through its unique Lévy flight patterns and evolutionary strategy that combines local and global search capabilities. The host nest elimination mechanism in CS also ensures continuous population diversity and helps avoid stagnation in local optima. The integration of these two algorithms creates a synergistic optimization approach that leverages their complementary strengths. GWO's hierarchical structure (alpha, beta, and delta wolves) provides effective leadership in guiding the search process, while CS's intelligent search behavior through Lévy flights enables powerful exploration capabilities. The Lévy flight characteristic of CS is particularly valuable as it allows for both short-distance exploitation and long-distance exploration of the solution space, making it highly effective in discovering promising regions for task scheduling optimization. When applied to task scheduling problems, this hybrid approach demonstrates superior performance compared to traditional algorithms, exhibiting faster convergence rates and enhanced stability. The combination of CS's sophisticated search mechanism and GWO's structured approach creates an efficient optimization method that can effectively balance between exploration and exploitation phases while maintaining solution diversity.

Our proposed algorithm integrates reinforcement learning with meta-heuristic optimization through a two-branch approach Figura 3.4 . The first branch employs a Q-learning mechanism

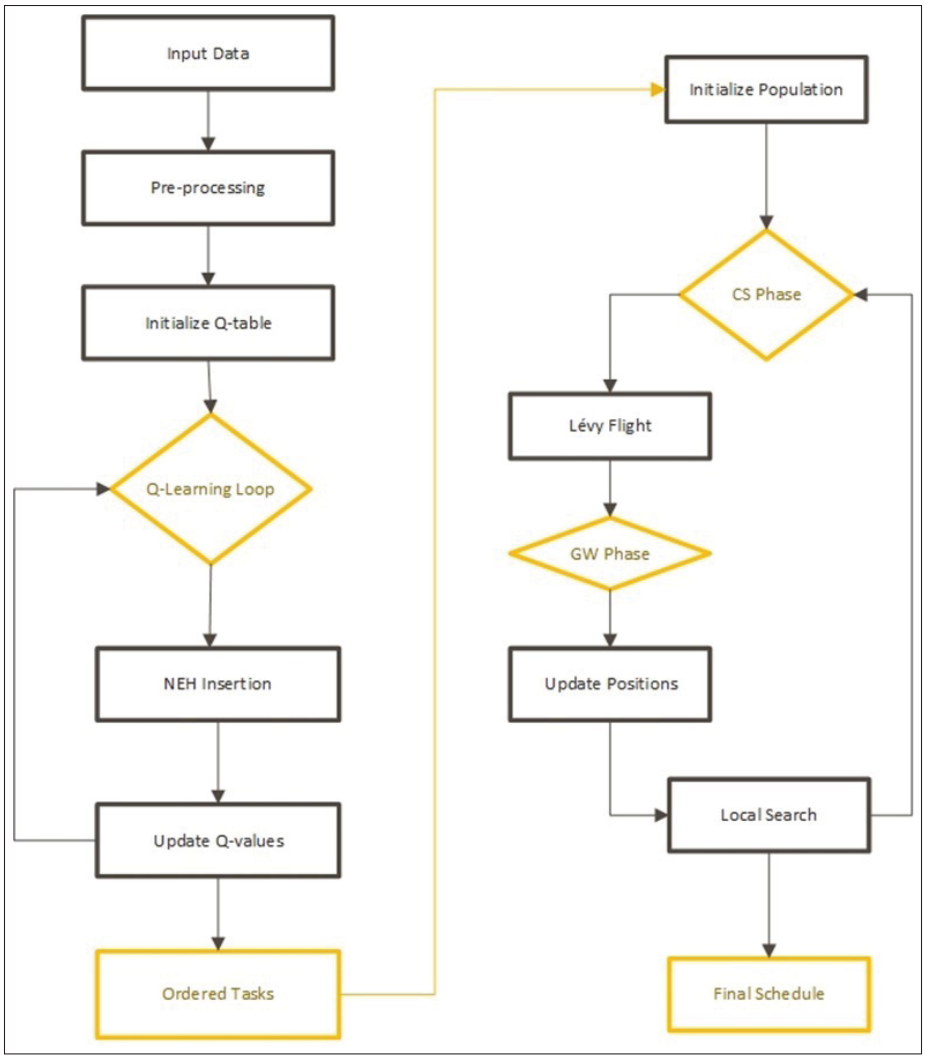


Figure 3.4 The fusion of Q-learning and domain knowledge of NEH for the FSP

enhanced with NEH heuristics, where the system learns to prioritize tasks through iterative interactions with the environment. The Q-table maintains and updates task priorities based on performance feedback, while the NEH insertion technique optimizes the sequence of tasks. This branch produces an intelligently ordered task list that serves as input for the second branch, which implements a hybrid Cuckoo Search-Grey Wolf (CS-GW) optimization strategy. The CS-GW phase begins with population initialization and uses Lévy flight patterns to explore the solution space, while the Grey Wolf optimization refines these solutions through position updates and local search procedures. This dual-branch architecture leverages both the adaptive

learning capabilities of Q-NEH and the robust optimization features of CS-GW to generate efficient task schedules in fog computing environments.

# CHAPTER 4

# RESULTS AND DISCUSSION

This chapter presents a comprehensive evaluation and analysis of our proposed hybrid CS-GW algorithm for task scheduling in fog computing environments. Our novel approach is compared with three widely-used meta-heuristic algorithms: Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and Ant Colony (AC). Furthermore, we enhance the analysis by examining each algorithm's performance under different task prioritization strategies, specifically implementing First-In-First-Out (FIFO) ordering and our proposed Q-NEH prioritization method, which combines Q-learning with the NEH heuristic for intelligent task sequencing. The experimental evaluation is structured to examine both the effectiveness of the base algorithms (PSO, GA, AC, and CS-GW) and the impact of task prioritization methods (FIFO vs. Q-NEH) on their performance. This dual-layer analysis allows us to assess not only which meta-heuristic approach performs best but also how different task ordering strategies affect their efficiency. Our comparative study focuses on crucial performance metrics including makespan minimization. Additionally, we examine the scalability and adaptability of each algorithm under different task sizes and fog computing configurations to validate their practical applicability in real-world implementations.

## 4.1 Compared algorithms

To show the effectiveness of our proposed, hybrid CS-GW algorithm with Q-NEH ,we compare them against the following baselines.

- **Particle Swarm Optimization (PSO)**(Kennedy & Eberhart, 1995) is a meta-heuristic algorithm inspired by the collective behavior of various animals such as insects, herds, birds, and schools of fish. The PSO algorithm simulates cooperative search patterns that emerge as these swarms learn from their own experiences as well as those of other members in the group to locate food sources.

- **Ant Colony Optimization (ACO)** (Blum, 2024) is a meta-heuristic algorithm inspired by the foraging behavior of ants. As ants search for food, they deposit pheromones on the paths they traverse. Other ants are able to detect these pheromone trails and are more likely to follow paths with stronger pheromone concentrations, as this indicates a promising route to a food source. Over time, the pheromone trails on the shortest and most efficient paths tend to become reinforced, while less optimal paths gradually fade away. The ACO algorithm simulates this emergent, collaborative problem-solving approach of ant colonies to find near-optimal solutions to complex optimization problems.
- **Genetic Algorithms** (Holland, 1992) are meta-heuristic optimization techniques inspired by the process of natural selection and evolution. GA operates on a population of candidate solutions, which are represented as "individuals" or "chromosomes." Just like in natural evolution, the fittest individuals in the population are more likely to "reproduce" and pass on their traits to the next generation of solutions. New candidate solutions are generated through operations like mutation and crossover, which introduce random variations and combine features of existing solutions. Over successive generations, the population gradually evolves towards higher quality, more optimal solutions, much like how species in nature adapt and improve their fitness over time. Genetic Algorithms are effective at exploring large, complex search spaces to find near-optimal solutions to challenging optimization problems.
- **Cuckoo Search** (Gandomi, Yang & Alavi, 2013) is a meta-heuristic optimization algorithm inspired by the brood parasitism behavior of cuckoos. Cuckoos are birds that lay their eggs in the nests of other host birds, often removing or destroying the host's own eggs. The host birds respond by either identifying and removing the cuckoo eggs or abandoning their nest altogether. The Cuckoo Search algorithm simulates this process, where candidate solutions (the cuckoo eggs) are laid in the nests of other solutions (host birds). High-quality solutions (cuckoo eggs) have a higher probability of surviving and producing even better solutions in the next generation. Meanwhile, low-quality solutions (host bird eggs) are likely to be discovered and replaced. Over iterations, the algorithm converges towards the optimal solution, much like how cuckoos exploit host nests to maximize the survival of their offspring

in nature. Cuckoo Search has been shown to be an efficient and effective meta-heuristic for solving complex optimization problems.

## 4.2 Simulation setting

For our simulation tests, we used a fog computing setup with multiple different types of interconnected fog nodes arranged in a random mesh pattern, handling various tasks sent from IoT devices. The experiments used between 10 to 60 fog nodes and tested with 100 to 600 IoT tasks. Each generated task is characterized by several key parameters:

- task type: including vital signs monitoring, ECG analysis, blood pressure checks, glucose monitoring, medication reminders, fall detection, sleep analysis, and oxygen saturation measurements,
- task priority: Tasks are classified into three priority levels (High, Medium, and Low). The scheduler considers these priorities when making task allocation decisions to ensure efficient resource utilization,
- arrival time: all tasks arrive simultaneously (time zero), providing a standardized baseline for makespan evaluation,
- execution time: varying from 5 to 600 seconds based on task complexity,
- CPU requirements: randomly assigned between 1-10 units,
- memory requirements: randomly assigned between 4096 to 64384 units,
- deadlines: calculated as three times the execution time from arrival.

To make the setup realistic, each fog node is defined by its unique specifications:

- CPU capacity: ranging from 4 to 64 units,
- memory capacity: varying from 4096 to 64384 units,
- power consumption rate: between 10-30 units.

This heterogeneous resource configuration reflects real-world scenarios where different computing nodes have varying capabilities and energy efficiency characteristics. The diversity in resource specifications allows us to evaluate how different scheduling algorithms perform

under various resource constraints and workload distributions, providing insights into their effectiveness in real-world healthcare monitoring systems.

All simulations are coded in Python programming language on PyCharm 2021.3.2 IDE. The experiments were carried out on a laptop running on Intel® Core i7 CPU, 2.7 GHz, 4 cores, 16 GB of RAM, and Windows 11 64-bit operating system. To provide results with high confidence, each experiment is repeated 30 times and reported an average of them.

## 4.3 Results

This section presents a comprehensive evaluation of the proposed task scheduling algorithms by comparing them with alternative approaches through multiple experiments focused on makespan performance. The experimental framework consists of three distinct configurations to thoroughly assess scheduling efficiency:

**Task Scalability Analysis:** The first experiment evaluates how the algorithms perform under varying workloads, testing with task counts ranging from 100 to 600 while maintaining a constant number of 30 Fog Nodes (FNs). This setup helps understand each algorithm's scalability and performance characteristics as the workload increases.

**Resource Scaling Assessment:** The second experiment focuses on resource utilization by fixing the workload at 200 tasks while varying the number of Fog Nodes from 10 to 60. This configuration helps determine the optimal resource allocation and its impact on processing efficiency.

**Comparative Performance Study:** The final experiment provides a standardized comparison between the proposed scheduling algorithm and existing alternatives, using a controlled environment of 200 tasks distributed across 30 Fog Nodes. This setup enables a direct performance comparison under identical conditions.

### 4.3.1 Makespan

The makespan comparison across different metaheuristic optimization approaches reveals significant variations in scheduling efficiency. Our proposed algorithm demonstrates superior performance by achieving the lowest makespan among all tested methods as illustrated in Figure 4.1 and Figure 4.2.

The bar chart 4.1 visualizes the makespan performance of four different algorithms (Proposed Algorithm, GA, PSO, and AC) across varying task loads ranging from 100 to 600 tasks. For smaller workloads around 100 tasks, all algorithms demonstrate relatively close performance with makespans between 9 to 10 minutes, showing their efficiency in handling lighter computational loads. As the number of tasks increases, the execution times naturally grow, but each algorithm shows different scaling patterns. For medium to large task loads (300-600 tasks), we observe a more pronounced difference in performance between the algorithms. At 400 tasks, the makespan ranges from approximately 22 to 25 minutes, while at 600 tasks, the execution times extend to between 30 and 34 minutes. The chart demonstrates that as the workload increases, the time difference between algorithms becomes more noticeable, indicating that algorithm selection becomes more critical for larger task sets. A consistent pattern emerges where our proposed algorithm maintains shorter makespans across all task sizes, with particularly noticeable efficiency at higher task loads.

Our proposed algorithm demonstrates significant performance advantages across all task categories, with improvement percentages ranging from 5.85% to 13.49% compared to the average performance of other algorithms (GA, PSO, and AC). The most notable improvement is observed at the lower end of the task spectrum (100 tasks) with a 13.49% reduction in makespan, while even at higher workloads of 600 tasks, proposed algorithm maintains an impressive 11.59% improvement. This indicates that proposed algorithm's optimization approach is particularly effective at both ends of the workload spectrum. Looking at the overall trend, proposed algorithm achieves an average improvement of 9.87% across all task sizes, with its lowest improvement of 5.85% occurring at 300 tasks. The algorithm's ability to maintain substantial improvements

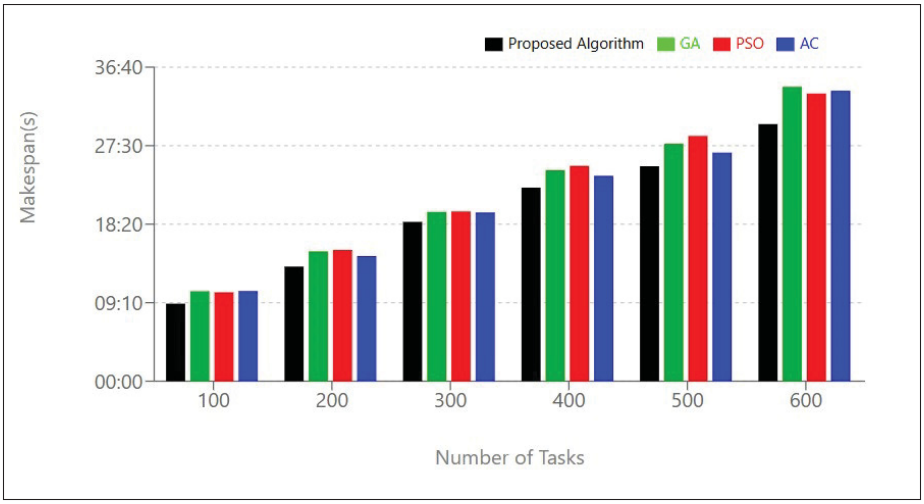


Figure 4.1 Makespan comparison - Number of Tasks

even as task numbers increase (9.31% at 500 tasks and 11.59% at 600 tasks) suggests excellent scalability and robust performance optimization capabilities. This consistent performance advantage across varying workload sizes indicates that proposed algorithm's scheduling strategy is well-suited for both small-scale and large-scale task scheduling scenarios.

The bar chart 4.2 illustrates the makespan (completion time) performance across different machine categories ranging from 10 to 60 machines. A clear descending trend is observed in the completion times as the number of machines increases, demonstrating a significant reduction in processing time. The most dramatic decrease occurs in the initial transition from 10 to 30 machines, where the completion time drops substantially from approximately 31 minutes to 14 minutes. This initial steep decline suggests that increasing the number of machines in this range has a substantial impact on improving processing efficiency.

After the 30-machine category, the rate of improvement becomes more gradual but continues to show consistent enhancements in performance. The system achieves its fastest completion time with 60 machines, completing the tasks in approximately 9 minutes. This pattern indicates that while adding more machines generally improves performance, the marginal benefits become less pronounced in the higher machine categories. The overall trend demonstrates a non-linear relationship between the number of machines and completion time, suggesting an optimization

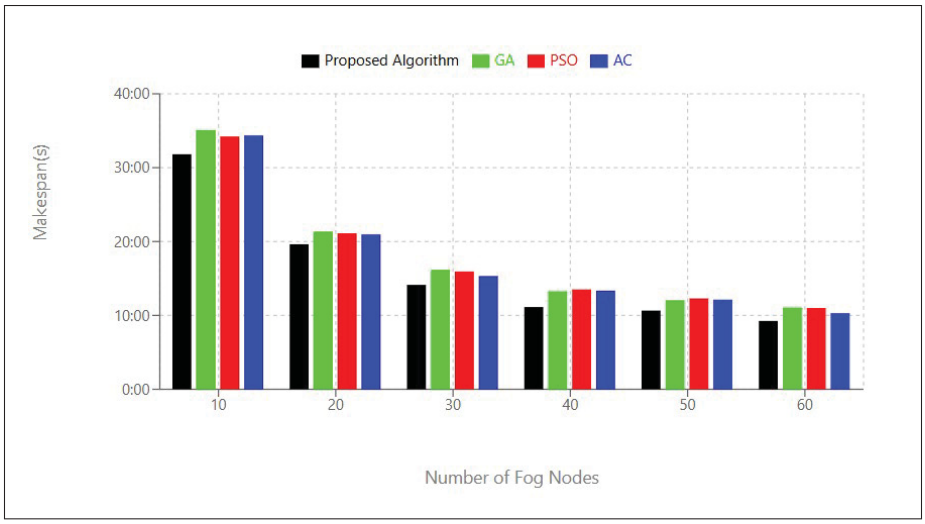


Figure 4.2 Makespan comparison - Number of Fog Nodes

point where adding more machines may yield diminishing returns in terms of performance improvement.

The proposed algorithm demonstrates consistent superior performance across all machine categories, with the most significant improvements observed in the 40-machine category, where it achieves remarkable optimization rates of 17.52% faster than PSO, 15.97% faster than GA, and 16.47% faster than AC. This pattern of improvement continues across different scales, with particularly strong performance in larger machine categories (30-60 machines), where the average improvement consistently exceeds 10% compared to other algorithms. The optimization capabilities of proposed algorithm become more pronounced as the problem complexity increases, suggesting its particular suitability for larger-scale scheduling scenarios. Looking at the overall optimization metrics, proposed algorithm achieves an impressive average improvement of 12.00% compared to PSO, 12.21% against GA, and 10.03% versus AC, resulting in an overall average improvement of 11.41% across all algorithms and machine categories. Even in the lower machine categories (10-20 machines), proposed algorithm maintains significant improvements, with optimization rates ranging from 6.22% to 9.36%. This consistent performance advantage across different scales demonstrates the robust and efficient nature of the proposed algorithm in

optimizing task scheduling and resource allocation, particularly in more complex scenarios with higher machine counts.

### 4.3.2 Q-NEH impacts

The bar chart 4.3 visualizes three different order scheduling strategies (FIFO, NEH, and Q-NEH) implemented with four metaheuristic algorithms (Proposed Algorithm, PSO, AC, and GA) for managing 200 tasks across 20 fog nodes. The chart 4.3 displays execution times in seconds, with Q-NEH (green bars) consistently showing shorter execution times compared to NEH (red bars) and FIFO (blue bars). The most efficient combination is CS-GW with Q-NEH at 13:27, while the slowest is PSO with FIFO at 15:58, demonstrating a performance gap of about 2.5 minutes between the best and worst scenarios.

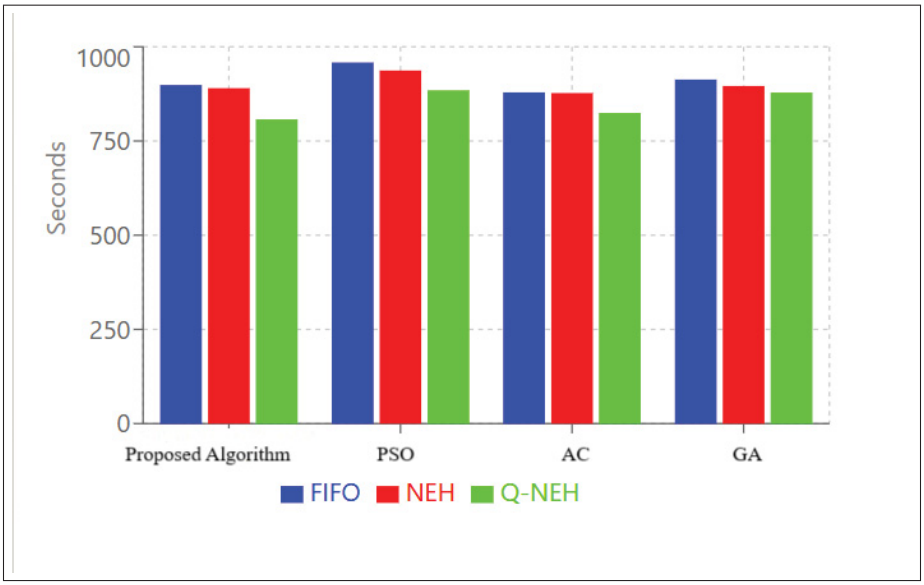


Figure 4.3 Comparison of Average Makespan Between Ordering

The bar chart 4.3 demonstrates the optimization performance of different scheduling strategies for handling 200 tasks across 20 fog nodes, where Q-NEH shows significant improvements over traditional FIFO and NEH approaches. When comparing execution times across different metaheuristic implementations (Proposed Algorithm, PSO, AC, and GA), Q-NEH consistently achieves better performance with improvements ranging from 10.1% to 15.8% compared to FIFO,

and 2.3% to 9.3% compared to NEH. The most notable improvement is observed in the proposed algorithm implementation, where Q-NEH reduces execution time by 91 seconds (15.8%) compared to FIFO and 83 seconds (9.3%) compared to NEH. This optimization is particularly evident in how Q-NEH enhances the scheduling efficiency through its improved task prioritization and resource allocation strategy. The strategy shows consistent performance improvements across all tested metaheuristic algorithms, with proposed algorithm implementation achieving the best results. In the PSO implementation, Q-NEH improves execution time by 74 seconds (7.7%) compared to FIFO and 53 seconds (5.7%) compared to NEH. Similarly, in AC and GA implementations, Q-NEH maintains its performance advantage with improvements of 55 seconds (6.3%) and 35 seconds (3.8%) respectively compared to FIFO, and 53 seconds (6.0%) and 18 seconds (2.0%) compared to NEH, demonstrating its superior scheduling capability for fog computing environments.

# CHAPTER 5

# CONCLUSION AND RECOMMENDATIONS

In this thesis, we explore task scheduling optimization for healthcare IoT applications in fog computing environments. Our research focuses on developing an efficient hybrid algorithm that combines Q-learning enhanced Nawaz-Enscore-Ham (Q-NEH) for task ordering with Cuckoo Search-Gray Wolf (CS-GW) optimization. Through comprehensive experimentation and evaluation, we have demonstrated the effectiveness of our proposed approach.

Our proposed hybrid Q-NEH-CS-GW algorithm, detailed in Chapter 3, provides a novel solution to healthcare task scheduling challenges in fog environments. The algorithm operates through two main components: First, Q-NEH employs reinforcement learning enhanced with NEH heuristics to intelligently order tasks based on their characteristics and priorities. Second, the CS-GW optimizer leverages Cuckoo Search's Lévy flight patterns for balanced exploration and Grey Wolf's structured exploitation through social hierarchy-based search. This dual mechanism enables effective navigation of the solution space while maintaining robust performance across varying workload conditions.

The experimental evaluation presented in Chapter 4 demonstrated the effectiveness of our approach through comprehensive testing. Using workloads ranging from 100-600 tasks across 10-60 fog nodes, we conducted comparative analyses against traditional methods including PSO, GA, and AC. Our results showed that the CS-GW component consistently achieved superior makespan optimization compared to these conventional approaches. Additionally, our examination of task prioritization strategies revealed that Q-NEH integration significantly enhanced scheduling efficiency compared to FIFO ordering across all tested algorithms, with the most substantial improvements observed when combined with CS-GW optimization. This success validates the synergistic benefits of combining intelligent task ordering with balanced exploration-exploitation in our hybrid approach.

## 5.1 Future Work

One area of future work is to extend our current algorithm to include multiple objectives beyond makespan optimization. By considering additional factors such as energy efficiency and resource utilization, we can develop a more comprehensive scheduling solution. Furthermore, we plan to enhance the Q-NEH component to include adaptive parameter tuning mechanisms that can better respond to varying healthcare task requirements.

Additionally, we aim to incorporate advanced security and privacy considerations into our scheduling framework, particularly important for healthcare applications. This includes developing secure task distribution mechanisms and privacy-preserving scheduling policies that can protect sensitive healthcare data while maintaining scheduling efficiency.

## BIBLIOGRAPHY


Abd Elaziz, M., Abualigah, L. & Attiya, I. (2021). Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments. *Future Generation Computer Systems*, 124, 142–154. doi: 10.1016/j.future.2021.05.026.

Adamatzky, A., Bull, L., De Lacy Costello, B., Stepney, S. & Teuscher, C. (2007). *Unconventional computing 2007*. UK: Luniver Press.

Adhikari, M., Mukherjee, M. & Srirama, S. N. (2020). DPTO: A deadline and priority-aware task offloading in fog computing framework leveraging multilevel feedback queueing. *IEEE Internet of Things Journal*, 7(7), 5773–5782.

Ahmed, O. H., Lu, J., Xu, Q., Ahmed, A. M., Rahmani, A. M. & Hosseinzadeh, M. Using differential evolution and Moth–Flame optimization for scientific workflow scheduling in fog computing. Publication details incomplete.

Alizadeh, M. R., Khajehvand, V., Rahmani, A. M. & Akbari, E. (2020). Task scheduling approaches in fog computing: A systematic review. *International Journal of Communication Systems*, 33, e4583.

AlZailaa, A. et al. (2021). Low-Latency Task Classification and Scheduling in Fog/Cloud Based Critical e-Health Applications. *ICC 2021 - IEEE International Conference on Communications*, pp. 1–6. doi: 10.1109/ICC42927.2021.9500985.

Azaria, A., Ekblaw, A., Vieira, T. & Lippman, A. (2016). MedRec: Using blockchain for medical data access and permission management. *2016 2nd International Conference on Open and Big Data (OBD)*, pp. 25–30.

Azizi, S., Zandsalimi, M. & Li, D. (2020). An energy-efficient algorithm for virtual machine placement optimization in cloud data centers. *Cluster Computing*, 23(4), 3421–3434.

Azizi, S., Shojafar, M., Abawajy, J. & Buyya, R. (2022). Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach. *Journal of Network and Computer Applications*, 201, 103333.

Banks, A., Vincent, J. & Anyakoha, C. (2008). A review of particle swarm optimization. Part II: hydridisation combinatorial, multicriteria and constrained optimization, and indicative applications. *Natural Computing*, 20, 109–124.

Barthelemy, P., Bertolotti, J. & Wiersma, D. S. (2008). A Lévy flight for light. *Nature*, 453, 495–498.

Beg, S., Handa, M., Shukla, R. et al. (2022). Wearable smart devices in cancer diagnosis and remote clinical trial monitoring: transforming the healthcare applications. *Drug Discovery Today*.

Blum, C. & Roli, A. (2003). Metaheuristics in combinatorial optimization: overview and conceptural comparison. *ACM Computing Surveys*, 35, 268–308.

Blum, C. (2024). Ant colony optimization: A bibliometric review. *Physics of Life Reviews*.

Bonomi, F., Milito, R., Zhu, J. & Addepalli, S. (2012). Fog computing and its role in the internet of things. *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16.

Brammer, J., Lutz, B. & Neumann, D. (2022). Permutation flow shop scheduling with multiple lines and demand plans using reinforcement learning. *European Journal of Operational Research*, 299(1), 75–86.

Brown, C., Liebovitch, L. S. & Glendon, R. (2007). Lévy flights in Dobe Ju/'hoansi foraging patterns. *Human Ecology*, 35, 129–138.

Cagnina, L. C., Esquivel, S. C. & Coello, C. A. (2008). Solving engineering optimization problems with the simple constrained particle swarm optimizer. *Informatica*, 32, 319–326.

Calheiros, R. N., Ranjan, R. & Buyya, R. (2011). Virtual machine provisioning based on analytical performance and QoS in cloud computing environments. *2011 International Conference on Parallel Processing*, pp. 295–304.

Cerina, L. et al. (2017). A fog-computing architecture for preventive healthcare and assisted living in smart ambients. *2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI)*.

Cvitić, I., Peraković, D., Periša, M. et al. (2021). Ensemble machine learning approach for classification of IoT devices in smart home. *International Journal of Machine Learning and Cybernetics*, 1–24.

Daraghmi, Y.-A. et al. (2022). Edge–Fog–Cloud Computing Hierarchy for Improving Performance and Security of NB-IoT-Based Health Monitoring Systems. *Sensors*, 22(22), 8646. doi: 10.3390/s22228646.

De Maio, V. & Kimovski, D. (2020). Multi-Objective Scheduling of Extreme Data Scientific Workflows in Fog. *Future Generation Computer Systems*, 106, 171–184. doi: 10.1016/j.future.2019.12.054.

Dokeroglu, T., Sevinc, E., Kucukyilmaz, T. & Cosar, A. (2019). A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*, 137, 106040.

Fang, W., Liao, Z. & Bai, Y. (2024). Improved ACO algorithm fused with improved Q-Learning algorithm for Bessel curve global path planning of search and rescue robots. *Robotics and Autonomous Systems*, 182, 104822.

Farina, M., Deb, K. & Amota, P. (2004). Dynamic multiobjective optimization problems: test cases, approximations, and applications. *IEEE Transactions on Evolutionary Computation*, 8, 425–442.

Fellir, F. et al. (2020). A Multi-Agent Based Model for Task Scheduling in Cloud-Fog Computing Platform. *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pp. 377–382. doi: 10.1109/ICIoT48696.2020.9089625.

Fernandez-Viagas, V., Ruiz, R. & Framinan, J. M. (2017). A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation. *European Journal of Operational Research*, 257(3), 707–721.

Gandomi, A. H., Yang, X. S. & Alavi, A. H. (2013). Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Engineering with Computers*, 29, 17–35.

Gaouar, N. & Lehsaini, M. (2021). Toward vehicular cloud/fog communication: A survey on data dissemination in vehicular ad hoc networks using vehicular cloud/fog computing. *International Journal of Communication Systems*, 34(13), e4906.

Garey, M. R. D., Johnson, D. S. & Sethi, R. (1976). The complexity of flow shop and job shop scheduling. *Mathematics of Operations Research*, 1, 117–129.

Ghobaei-Arani, M., Souri, A. & Rahmanian, A. A. (2020). Resource management approaches in fog computing: a comprehensive review. *Journal of Grid Computing*, 18, 1–42.

Guevara, J. C., Torres, R. d. S. & da Fonseca, N. L. (2020). On the classification of fog computing applications: A machine learning perspective. *Journal of Network and Computer Applications*, 159, 102596.

Guo, D., Liu, S., Ling, S., Li, M., Jiang, Y., Li, M. & Huang, G. Q. (2024). The marriage of operations research and reinforcement learning: Integration of NEH into Q-learning algorithm for the permutation flowshop scheduling problem. *Expert Systems with Applications*, 255, 124779.

Guo, D., Liu, S., Ling, S., Li, M., Jiang, Y., Li, M. & Huang, G. Q. Note: This appears to be an incomplete reference.

Gupta, A. & Chaurasiya, V. K. (2022). Efficient Task-Offloading in IoT-Fog Based Health Monitoring System. *2022 OITS International Conference on Information Technology (OCIT)*, pp. 495–500. doi: 10.1109/OCIT56763.2022.00098.

Hajvali, M. et al. (2022). Software Architecture for IoT-Based Health-Care Systems with Cloud/Fog Service Model. *Cluster Computing*, 25(1), 91–118. doi: 10.1007/s10586-021-03375-4.

Hassan, N., Gillani, S., Ahmed, E. et al. (2018). The role of edge computing in internet of things. *IEEE Communications Magazine*, 56(11), 110–115.

He, D. & Zeadally, S. (2014). An analysis of RFID authentication schemes for internet of things in healthcare environment using elliptic curve cryptography. *IEEE Internet of Things Journal*, 2(1), 72–83.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press.

Hossain, M. R., Whaiduzzaman, M., Barros, A., Tuly, S. R., Mahi, M. J. N., Roy, S., Fidge, C. & Buyya, R. (2021). A scheduling-based dynamic fog computing framework for augmenting resource utilization. *Simulation Modelling Practice and Theory*, 111, 102336.

Hosseinioun, P., Kheirabadi, M., Tabbakh, S. R. K. & Ghaemi, R. (2020). A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm. *Journal of Parallel and Distributed Computing*, 143, 88–96.

Houssein, E. H., Gad, A. G., Wazery, Y. M. & Suganthan, P. N. (2021). Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends. *Swarm and Evolutionary Computation*, 62, 100841. doi: 10.1016/j.swevo.2021.100841.

Hussain, S. M. & Begh, G. R. (2022). Hybrid heuristic algorithm for cost-efficient QoS aware task scheduling in fog–cloud environment. *Journal of Computer Science*, 64, 101828. doi: 10.1016/j.jocs.2022.101828.

IoT Analytics. [Accessed: 2024-11-14]. (2024). State of IoT 2024: Number of connected IoT devices growing 13Retrieved from: https://iot-analytics.com/number-connected-iot-devices/.

Islam, M. S. U., Kumar, A. & Hu, Y.-C. (2021). Context-aware scheduling in Fog computing: A survey, taxonomy, challenges and future directions. *Journal of Network and Computer Applications*, 180, 103008.

Jangir, P. & Jangir, N. (2018). A new Non-Dominated Sorting Grey Wolf Optimizer (NS-GWO) algorithm: Development and application to solve engineering designs and economic constrained emission dispatch problem with integration of wind power. *Engineering Applications of Artificial Intelligence*, 72, 449–467.

Jasim, A. M. & Al-Raweshidy, H. (2023). Towards a Cooperative Hierarchical Healthcare Architecture Using the HMAN Offloading Scenarios and SRT Calculation Algorithm. *IET Networks*, 12(1), 9–26. doi: 10.1049/ntw2.12064.

Javaheri, D., Gorgin, S., Lee, J.-A. & Masdari, M. (2022). An improved discrete harris hawk optimization algorithm for efficient workflow scheduling in multi-fog computing. *Sustainable Computing: Informatics and Systems*, 36, 100787.

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1, 61–68.

Kabirzadeh, S., Rahbari, D. & Nickray, M. (2017). A Hyper Heuristic Algorithm for Scheduling of Fog Networks. *2017 21st Conference of Open Innovations Association (FRUCT)*, pp. 148–155. doi: 10.23919/FRUCT.2017.8250177.

Kalczynski, P. & Kamburowski, J. (2007). On the NEH heuristic for minimizing the makespan in permutation flow shops. *OMEGA, The International Journal of Management Science*, 35(1), 53–60.

Karimi-Mamaghan, M., Mohammadi, M., Pasdeloup, B. & Meyer, P. (2022). Learning to select operators in meta-heuristics: An integration of Q-learning into the iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*.

Kayhan, B. M. & Yildiz, G. (2021). Reinforcement learning applications to machine scheduling problems: A comprehensive literature review. *Journal of Intelligent Manufacturing*, 1–25.

Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95-International Conference on Neural Networks*, pp. 1942–1948.

Khan, M. A. & Salah, K. (2018). IoT security: Review, blockchain solutions, and open challenges. *Future Generation Computer Systems*, 82, 395–411.

Liu, J., Yang, Z. & Li, D. (2020). A multiple search strategies based grey wolf optimizer for solving multi-objective optimization problems. *Expert Systems with Applications*, 145, 113134.

Long, W., Wu, T., Cai, S., Liang, X., Jiao, J. & Xu, M. (2019). A novel grey wolf optimizer algorithm with refraction learning. *IEEE Access*, 7, 57805–57819.

Mahmud, R. et al. (2019). Quality of Experience (QoE)-Aware Placement of Applications in Fog Computing Environments. *Journal of Parallel and Distributed Computing*, 132, 190–203. doi: 10.1016/j.jpdc.2018.03.004.

Maiti, P. et al. (2022). Internet of Things Applications Placement to Minimize Latency in Multi-Tier Fog Computing Framework. *ICT Express*, 8(2), 166–173. doi: 10.1016/j.icte.2021.06.004.

Marín-Tordera, E., Masip-Bruin, X., García-Almiñana, J., Jukan, A., Ren, G.-J. & Zhu, J. (2017). Do we all really know what a fog node is? Current trends towards an open definition. *Computer Communications*, 109, 117–130.

Markit, I. H. S. (2017). The Internet of Things: a movement, not a market. *IHS Mark.*, 1, 1.

Memari, P., Mohammadi, S. S., Jolai, F. & Tavakkoli-Moghaddam, R. (2022). A latency-aware task scheduling algorithm for allocating virtual machines in a cost-effective and time-sensitive fog-cloud architecture. *Journal of Supercomputing*, 78, 93–122.

Mirjalili, S., Mirjalili, S. M. & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, 69, 46–61.

Mirjalili, S., Saremi, S., Mirjalili, S. M. & Coelho, L. d. S. (2016). Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization. *Expert Systems with Applications*, 47, 106–119.

Mohammad Hasani Zade, B. & Mansouri, N. (2022). Improved red fox optimizer with fuzzy theory and game theory for task scheduling in cloud environment. *Journal of Computer Science*, 63, 101805. doi: 10.1016/j.jocs.2022.101805.

Movahedi, Z. & Defude, B. (2021). An efficient population-based multi-objective task scheduling approach in fog computing systems. *Journal of Cloud Computing*, 10, 1–31.

Muro, C., Escobedo, R., Spector, L. & Coppinger, R. (2011). Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations. *Behavioural Processes*, 88, 192–197.

Najafizadeh, A., Salajegheh, A., Rahmani, A. M. & Sahafi, A. (2022). Multi-objective Task Scheduling in cloud-fog computing using goal programming approach. *Cluster Computing*, 25, 141–165.

Nawaz, M., Enscore Jr., E. & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega, the International Journal of Management Science*, 11, 91–95.

Nayeri, Z. M., Ghafarian, T. & Javadi, B. (2021). Application Placement in Fog Computing with AI Approach: Taxonomy and a State of the Art Survey. *Journal of Network and Computer Applications*, 185, 103078. doi: 10.1016/j.jnca.2021.103078.

Paul, A. et al. (2018). Fog Computing-Based IoT for Health Monitoring System. *Journal of Sensors*, 1–7. doi: 10.1155/2018/1386470.

Pavlyukevich, I. (2007). Lévy flights, non-local search and simulated annealing. *Journal of Computational Physics*, 226, 1830–1844.

Payne, R. B., Sorenson, M. D. & Klitz, K. (2005). *The cuckoos*. Oxford University Press.

Pinedo, M. (2002). *Scheduling: Theory, Algorithms, and Systems*. Upper Saddle, NJ, Englewood Cliffs: Prentice-Hall.

Potts, C. N., Shmoys, D. B. & Williamson, D. P. (1991). Permutation vs. non-permutation flow shop schedules. *Operation Research Letters*, 10, 281–284.

Powell, C., Desiniotis, C. & Dezfouli, B. (2020). The Fog Development Kit: A Platform for the Development and Management of Fog Systems. *IEEE Internet of Things Journal*, 7(4), 3198–3213. doi: 10.1109/JIOT.2020.2966405.

Reynolds, A. M. & Frye, M. A. (2007). Free-flight odor tracking in Drosophila is consistent with an optimal intermittent scale-free search. *PLoS One*, 2, e354.

Rezazadeh, Z., Rezaei, M. & Nickray, M. (2019). LAMP: A Hybrid Fog-Cloud Latency-Aware Module Placement Algorithm for IoT Applications. *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, pp. 845–850. doi: 10.1109/K-BEI.2019.8734958.

Rodríguez-Esparza, E., Zanella-Calzada, L. A., Oliva, D., Heidari, A. A., Zaldivar, D., Pérez-Cisneros, M. & Foong, L. K. (2020). An efficient Harris hawks-inspired image segmentation method. *Expert Systems with Applications*, 155, 113428.

Ruiz, R. & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165(2), 479–494.

Sharif, Z., Jung, L. T., Razzak, I. et al. (2021). Adaptive and Priority-based Resource Allocation for Efficient Resources Utilization in Mobile Edge Computing. *IEEE Internet of Things Journal*.

Sharma, V. & Tripathi, A. K. (2022). A systematic review of meta-heuristic algorithms in IoT based application. *Array*, 14, 100164. doi: 10.1016/j.array.2022.100164.

Shome, S. & Bera, R. (2020). Narrowband-IoT base station development for green communication. In *Advances in greener energy technologies* (pp. 475–487). Springer.

Stefan, P. E. T. E. R. (2003). Flow-shop scheduling based on reinforcement learning algorithm. *Production Systems and Information Engineering*, 1(1), 83–90.

Subbaraj, S. & Thiyagarajan, R. (2021). Performance oriented task-resource mapping and scheduling in fog computing environment. *Cognitive Systems Research*, 70, 40–50. doi: 10.1016/j.cogsys.2021.07.004.

Sun, Z. et al. (2019). Intelligent Sensor-Cloud in Fog Computer: A Novel Hierarchical Data Job Scheduling Strategy. *Sensors*, 19(23), 5083. doi: 10.3390/s19235083.

Sutton, R. S. & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Talaat, F. M., Ali, H. A., Saraya, M. S. & Saleh, A. I. (2022). Effective scheduling algorithm for load balancing in fog environment using CNN and MPSO. *Knowledge and Information Systems*, 64, 773–797. doi: 10.1007/s10115-021-01649-2.

Talaat, F. M. et al. (2020). A Load Balancing and Optimization Strategy (LBOS) Using Reinforcement Learning in Fog Computing Environment. *Journal of Ambient Intelligence and Humanized Computing*, 11(11), 4951–4966. doi: 10.1007/s12652-020-01768-8.

Verma, P. & Sood, S. K. (2018). Fog assisted-IoT enabled patient health monitoring in smart homes. *IEEE Internet of Things Journal*, 5(3), 1789–1796.

Yadav, A. M., Tripathi, K. N. & Sharma, S. C. (2022). A bi-objective task scheduling approach in fog computing using hybrid fireworks algorithm. *Journal of Supercomputing*, 78, 4236–4260.

Yang, X. S. (2010). *Engineering optimisation: an introduction with metaheuristic applications*. John Wiley and Sons.

Yang, X. S. & Deb, S. (2009). Cuckoo search via Lévy flights. *Proceedings of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009 India)*, pp. 210–214.

Zhang, Z., Wang, W., Zhong, S. & Hu, K. (2013). Flow shop scheduling with reinforcement learning. *Asia-Pacific Journal of Operational Research*, 30(05), 1350014.