

AYEH MAHJOUBI



Task Scheduling and Offloading in IoT-Edge-Cloud Systems

From Offline Optimization to Online Learning





Task Scheduling and Offloading in IoT–Edge– Cloud Systems

From Offline Optimization to Online Learning

Ayeh Mahjoubi

Faculty of Health, Science and Technology

Computer Science

DOCTORAL THESIS | Karlstad University Studies | 2026:9

Task Scheduling and Offloading in IoT–Edge–Cloud Systems

From Offline Optimization to Online Learning

Ayeh Mahjoubi

Task Scheduling and Offloading in IoT–Edge–Cloud Systems
- From Offline Optimization to Online Learning

Ayeh Mahjoubi

DOCTORAL THESIS

Karlstad University Studies | 2026:9

urn:nbn:se:kau:diva-108215

ISSN 1403-8099

ISBN 978-91-7867-662-0 (print)

ISBN 978-91-7867-663-7 (pdf)

<https://doi.org/10.59217/sbpt5891>

© The author

Distribution:
Karlstad University
Faculty of Health, Science and Technology
Department of Mathematics and Computer Science
SE-651 88 Karlstad, Sweden
+46 54 700 10 00

Print: Universitetstryckeriet, Karlstad 2026

Task Scheduling and Offloading in IoT-Edge-Cloud Systems: From Offline Optimization to Online Learning

AYEH MAHJoubi

Department of Mathematics and Computer Science

Abstract

The Internet of Things (IoT) devices are increasingly used in various settings, including factories, hospitals, homes, and vehicles. These IoT devices often have limited processing capabilities, which necessitate reliance on cloud servers to execute compute-intensive tasks. However, offloading tasks to the cloud presents several challenges, primarily due to the significant amount of data that must be transmitted between IoT devices and the cloud. Such high data volumes can saturate the available bandwidth, leading to network congestion, increased transmission delays, and ultimately higher end-to-end latency. These challenges are particularly pronounced in latency-sensitive and data-intensive IoT applications such as industrial monitoring and control, smart city video analytics, and continuous wearable health monitoring, where timely processing and dependable service delivery are critical. To mitigate these issues, the concept of edge computing was introduced, which brings computing resources closer to end users. Given the limited resources at edge nodes, it is essential to optimally schedule tasks among IoT devices, edge nodes, and cloud servers to fulfill the requirements of IoT applications.

This thesis presents a comprehensive framework for adaptive and delay-efficient task scheduling across the IoT-edge-cloud continuum, addressing both offline optimization and online learning perspectives. Specifically, we model the offloading and scheduling challenges within a three-tier device-edge-cloud architecture using a Mixed-Integer Linear Program (MILP) aimed at minimizing end-to-end service delay. While CPLEX provides strong benchmark solutions for the MILP, it becomes computationally intensive for larger instances. To create scalable solutions, we develop efficient heuristics along with two meta-heuristic approaches: a genetic algorithm (GA) and simulated annealing (SA). The GA typically improves solution quality compared to heuristics, but it requires more processing time. In contrast, SA achieves competitive accuracy, within 3–5% of the optimal MILP solution, while reducing runtime by more than an order of magnitude compared to the MILP solver, making it more suitable for large-scale offline planning.

Building on this groundwork, we introduce two online schedulers capable of functioning effectively under time-varying workloads and partial system knowledge. The first scheduler, SATS, extends simulated annealing for online hierarchical multi-access edge computing (MEC) by performing incremental neighborhood searches at decision points and utilizing predictions of service requests. The second scheduler employs a cooperative multi-agent reinforce-

ment learning (MARL) framework that addresses the entire device-edge-cloud stack and supports heterogeneous IoT services with interdependent tasks and varying deadlines. In this framework, IoT devices and the edge server act as agents that undergo centralized training and decentralized execution, using both Deep Q-Networks (DQN) and a variant called Double DQN to enhance stability.

Extensive simulations demonstrate that the proposed online methods reduce average latency by up to 35% and improve deadline satisfaction by over 20% compared to leading state-of-the-art baselines. Collectively, these contributions advance scalable and adaptive scheduling for emerging 5G/6G-enabled IoT systems, facilitating low-latency, resource-efficient, and resilient service delivery in domains such as smart manufacturing, intelligent transportation and smart cities, and real-time healthcare monitoring.

Keywords: task offloading, task scheduling, edge computing, optimization techniques, heuristics, meta-heuristics, reinforcement learning

Acknowledgements

I am deeply grateful to everyone who supported and accompanied me throughout this doctoral journey. The guidance, encouragement, and generosity I received made this work possible and sustained me through both its challenges and rewards.

I would like to begin by expressing my sincere gratitude to my supervisor, Karl-Johan Grinnemo. From the very beginning, you provided steady guidance, thoughtful mentorship, and unwavering support. You taught me how to navigate uncertainty, persist through difficulties, and maintain rigor when progress was slow. Your trust strengthened my confidence, your questions sharpened my thinking, and your consistency made it possible to bring this work to completion. I will carry the lessons I learned from your example, intellectual rigor, resilience, and integrity, well beyond this thesis.

I am also deeply thankful to my co-supervisor, Arunselvan Ramaswamy. When our collaboration began, I often felt uncertain and discouraged. Your insight, creativity in approaching complex problems, and calm encouragement helped me find direction and regain confidence. You transformed confusion into clarity and setbacks into structured paths forward.

I extend my sincere thanks to my examiner, Anna Brunstrom, for her careful reading, constructive feedback, and continuous support throughout this process.

I would also like to thank Stefan Alfredsson, Head of the Department, for always making time to discuss ideas and challenges; Johan Eklund for his openness, understanding, and willingness to help; Per Hurtig for reviewing my work and providing thoughtful and constructive feedback; Andreas Kassler for his support throughout this journey; and Javid Taheri for his guidance and help at the beginning of my doctoral path.

I would like to express my sincere gratitude to my opponent, Olov Schelen, for taking on this role and for the time and effort devoted to evaluating my work.

I am also grateful to the members of the examining committee, Mohammad Ashjaei, Carsten Griwodz, and Maria Kihl, for agreeing to serve and for the time and expertise they contribute to the evaluation of this thesis. I also thank Sebastian Herold, substitute member of the examining committee, for being willing to serve if needed. My sincere thanks also go to Leonardo Iwaya, chairman, for accepting the responsibility of chairing the defense and ensuring a fair and well-organized examination process.

To my colleagues and PhD fellows, Michele, Manal, Amal, Lejla, Tahir, Mohammadsadeq, Frey, Mohsen, Alexander, and Simon, thank you for the collegiality, encouragement, and everyday kindness that made the doctoral experience lighter and more rewarding.

I am especially grateful to my closest friends, Mozhgan and Sajjad, for always being there for me, with compassion, understanding, and steady encouragement, through every stage of this journey. I also thank my close friends, Farzaneh, Kyoomars, Najmeh, Jannet, Faezeh, Iliar, and Rozhan, for their

support, thoughtful conversations, and joyful moments that helped me stay grounded.

Finally, and most importantly, I thank my family, whose love and support formed the foundation of this achievement.

To my parents, my sister, Sahand and Elham, I am deeply grateful for your unconditional love and steadfast support throughout these years. Your encouragement has meant more to me than I can express.

To my son, Aryo, your joy, curiosity, and smile brought light and meaning to every day. You reminded me what truly matters and gave me strength beyond words.

To my husband, Omid, none of this would have been possible without you. Your patience, understanding, and quiet strength carried me through the most demanding moments. You were a constant source of reassurance, encouragement, and stability, and your belief in me never wavered, even when mine did. This thesis bears the imprint of your care and sacrifice on every page.

I also gratefully acknowledge the DRIVE research profile and the Knowledge Foundation for partially funding this work.

Karlstad, February 10, 2026

Ayeh Mahjoubi

*I dedicate this thesis to my husband, Omid,
with all my love and gratitude.*

List of Appended Papers

1. Ayeh Mahjoubi, Javid Taheri, Karl-Johan Grinnemo. Optimal Placement of Recurrent Service Chains on Distributed Edge-Cloud Infrastructures. IEEE 46th Conference on Local Computer Networks (LCN), 2021, pp. 495-502.
2. Ayeh Mahjoubi, Karl-Johan Grinnemo, Javid Taheri. EHGA: A Genetic Algorithm Based Approach for Scheduling Tasks on Distributed Edge-Cloud Infrastructures. Extended version of the paper published in 13th International Conference on Network of the Future (NoF), 2022, pp. 1-5.
3. Ayeh Mahjoubi, Karl-Johan Grinnemo, Javid Taheri. An Efficient Simulated Annealing-based Task Scheduling Technique for Task Offloading in a Mobile Edge Architecture. IEEE International Conference on Cloud Networking (CloudNet), Paris, France, November 2022.
4. Ayeh Mahjoubi, Arunselvan Ramaswamy, Karl-Johan Grinnemo. An Online Simulated Annealing-based Task Offloading Strategy for a Mobile Edge Architecture. IEEE Access, 2024, vol. 12, pp. 70707-70718.
5. Ayeh Mahjoubi, Arunselvan Ramaswamy, Karl-Johan Grinnemo. Deadline -Aware Service Scheduling via Multi-Head MARL in Device-Edge-Cloud Environments. Computer Networks, vol. 277, art. no. 112019, 2026.

Comments on my Participation

Paper I The concept for this paper originated with Javid Taheri. I am the principal author and was responsible for drafting the manuscript as well as designing, executing, evaluating, and analyzing the experiments. My advisors contributed to the planning process, reviewed the experimental results, and provided revisions to the manuscript.

Paper II The main idea for this paper originated from Javid Taheri, but I am the principal author. I wrote the majority of the manuscript, including the problem formulation, methodology, experimental design, and analysis sections. I also developed the genetic-based solution, designed the experiments, conducted the tests, and analyzed the results. Karl-Johan Grinnemo did an excellent job by providing technical feedback and revising and polishing the text.

Paper III I am the principal author of the paper. The central idea was developed in collaboration with my advisors. Karl-Johan Grinnemo provided valuable assistance with brainstorming and refining the work. I was responsible for writing the initial draft of the entire paper, including the introduction, methodology, experimental setup, results, and discussion, and for performing the subsequent revisions. My advisors contributed by reviewing the

manuscript, suggesting structural and content-related improvements, and providing editorial feedback.

Paper IV This paper builds on my earlier offline SA-based scheduler by adapting it to an online IoT–edge–cloud/MEC setting. I am the principal author and was responsible for organizing, executing, evaluating, and analyzing the experiments. I also drafted the manuscript and wrote the majority of the text. Arunselvan Ramaswamy and Karl-Johan Grinnemo contributed to the planning of the study, discussed the experimental design and interpretation of the results, and carefully reviewed the experimental results. They provided critical feedback on the content and structure of the paper and contributed to revising and polishing the text through comments and suggested edits.

Paper V This study builds on my earlier IoT–edge–cloud scheduling work. The proposed approach reformulates the scheduling objective in a learning framework to support adaptive decisions under dynamic workloads and partial system knowledge. I am the principal author and was responsible for designing the study, implementing the methods, conducting the experiments, and analyzing the results. I also wrote the initial draft of all sections of the manuscript. Arunselvan Ramaswamy and Karl-Johan Grinnemo contributed to the planning of the work, discussed and reviewed the experimental results, provided critical feedback on the structure and content of the paper, and helped improve the text through comments and revisions.

Other Publications

- Ayeh Mahjoubi, Javid Taheri, Karl-Johan Grinnemo. Optimal Placement of Recurrent Service Chains on Distributed Edge-Cloud Infrastructures, 17th Swedish National Computer Networking Workshop (SNCNW), June 16-17, 2022.

Contents

List of Appended Papers	ix
-------------------------	----

INTRODUCTORY SUMMARY 1

1 Introduction	3
2 Background and Related Work	4
2.1 Mathematical Optimization	5
2.1.1 Linear Programming	5
2.1.2 Heuristics and Metaheuristics	7
2.2 Reinforcement Learning	10
2.2.1 Markov Decision Processes	11
2.2.2 Model-Free Control (Tabular Q-Learning)	11
2.2.3 Q-Networks (Function Approximation for Q-Learning)	11
2.2.4 Deep Q-Networks	12
2.2.5 Double DQN	12
2.2.6 Multi-Agent Reinforcement Learning	13
2.3 IoT and Edge	13
2.4 Related Work	15
3 Research Questions	17
4 Research Methods	19
5 Contributions	21
6 Limitations	23
7 Summary of Appended Papers	24
8 Conclusions and Future Work	25

PAPER I: Optimal Placement of Recurrent Service Chains on Dis- tributed Edge-Cloud Infrastructures 35

1 Introduction	35
2 Related Work	37
3 Mathematical Modeling of Edge-Cloud Computing Platforms	38
3.1 Computing	39
3.2 Networking	39
3.3 Service Chains	40

4	Task Scheduling Problem Formulation	41
4.1	Calculation of Service Total Delay	41
4.2	Problem Formulation	42
4.3	Heuristics	43
5	Results and Discussions	45
5.1	Small Number of Services	45
5.2	Medium Number of Services	47
5.3	Large Number of Services	48
6	Conclusion	50

PAPER II:
EHGA: A Genetic Algorithm Based Approach for Scheduling Tasks on Distributed Edge-Cloud Infrastructures **57**

1	Introduction	57
2	Related Work	58
3	System Model and Formulation	59
4	Task Scheduling Problem Formulation	61
4.1	Calculation of Service Total Delay	61
5	Solution	62
5.1	Initialization	63
5.2	Fitness Function	65
5.3	Selection	65
5.4	Multi-point Crossover	65
5.5	Mutation	66
5.6	Healing	67
5.7	Develop an Extra Chromosome	67
6	Evaluation & Simulation	68
7	Results & Discussions	68
7.1	Different Number of IoT Devices	69
7.2	Different Number of Edge Servers	70
7.3	Different Number of Services	71
8	Conclusion	72

PAPER III:
An Efficient Simulated Annealing-based Task Scheduling

Technique for Task Offloading in a Mobile Edge Architecture	79
1 Introduction	79
2 System Model and Formulation	80
3 Simulated Annealing Approach	84
4 Evaluation and Discussion	85
4.1 Fixed Number of Edge Servers and Services	88
4.2 Fixed Number of Services and IoT Devices	89
4.3 Fixed Number of Edge Servers and IoT Devices	92
4.4 Takeaways	92
5 Related Works	92
6 Conclusion	93
PAPER IV:	
An Online Simulated Annealing-based Task Offloading Strategy for a Mobile Edge Architecture	99
1 Introduction	99
2 Background	101
2.1 Mobile Edge Computing	101
2.2 Optimization, Metaheuristics, and Simulated Annealing . . .	102
3 System Model and Representation	103
4 Problem Definition	104
5 Simulated Annealing Task Scheduling	108
6 Experiment Setup	108
7 Numerical Studies	111
7.1 Varying the number of CIoT devices	111
7.2 Varying the service frequency	114
8 Related Work	117
9 Conclusions	119
A Table of Notations	120

PAPER V:	
Deadline -Aware Service Scheduling via Multi-Head MARL in Device-Edge-Cloud Environments	127
1 Introduction	127
2 Related Work	130
3 System Model and Problem Formulation	131
3.1 Resource Model	131
3.2 Computation and Communication Model	134
3.3 Optimization Objective	135
4 Multi-Agent Reinforcement Learning Framework	136
4.1 State Space	136
4.2 Action Space	137
4.3 Reward Design	138
4.4 Multi-Head Q-Network Architecture	139
4.5 Training Algorithms: DQN and DDQN	139
5 Evaluation	140
5.1 Experimental Setup	140
5.2 Hyperparameter Configuration	141
5.3 Neural Network Architecture	142
5.4 Baselines	142
5.5 Metrics	143
5.6 Agent Loss Dynamics	144
5.7 Cumulative Reward Trends	146
5.8 Acceptance Ratio Analysis	147
5.9 Processing Time Comparison	148
5.10 Task Distribution by Location	150
6 Conclusion	151

Introductory Summary



1 Introduction

The proliferation of Internet of Things (IoT) devices over the past decade has enabled a wide range of applications across various industries, including healthcare, transportation, and smart homes, improving efficiency and quality of life. Representative examples include real-time patient monitoring and wearable-based alerting, smart traffic and connected-vehicle services, industrial condition monitoring and predictive maintenance, and video-enabled public safety and smart-building automation. However, most IoT devices possess limited processing and storage capabilities, which make them dependent on cloud servers for executing computationally intensive tasks. This dependency leads to the continuous transmission of vast volumes of data generated by sensors, actuators, and mobile devices to remote cloud data centers. Such transfers impose high latency and can lead to bandwidth constraints and excessive network congestion, rendering traditional cloud-based architectures inadequate for latency-sensitive and computation-intensive applications.

To alleviate these limitations, fifth-generation (5G) mobile networks, in conjunction with cloud infrastructures, improve cloud-system effectiveness by supporting low-latency data handling and near real-time analytical processing, thereby reducing end-to-end delay and energy consumption for time-critical services. Nonetheless, the increasing number of IoT devices and the growing demand for real-time services have shifted attention in both academia and industry toward *edge computing*—a paradigm that deploys computational resources closer to end users. By processing tasks near the data sources, edge computing mitigates congestion and latency, improves Quality of Service (QoS), and supports the stringent delay requirements of modern IoT applications. Consequently, IoT devices may offload computation to either edge or cloud servers. The central challenge, however, lies in determining *when*, *where*, and *how* such offloading should be performed.

When tasks are offloaded to edge or cloud resources, several challenges must be addressed, including dynamic network conditions, task dependencies, heterogeneous device capabilities, and constrained computational resources. In particular, IoT devices differ in terms of processing power, memory, and energy budgets, while tasks may exhibit varying latencies and resource requirements. Efficient offloading and scheduling must therefore account for these heterogeneities, ensure resource feasibility, and satisfy task deadlines. Furthermore, task-scheduling algorithms depend on how services are modeled and whether scheduling decisions are made *offline* (with full system knowledge) or *online* (under uncertainty). A review of prior work reveals that few studies address the complete three-layer IoT–edge–cloud architecture while considering node, link, and service characteristics in both offline and online modes. Moreover, existing approaches rarely account for the concurrent execution of multiple services and their interdependent tasks on individual devices.

This thesis first investigates **offline scheduling and offloading** strategies for concurrent IoT services across the device–edge–cloud continuum. We design a three-tier system model that captures the computational and networking char-

acteristics of all nodes and links. Based on this model, we formulate the task offloading and scheduling problem as a Mixed-Integer Linear Program (MILP) aimed at minimizing end-to-end service delay. To solve the formulated problem, several strategies are employed. First, an exact solution is obtained using the Simplex method within the IBM CPLEX optimizer [1], which serves as a performance benchmark despite its high computational complexity. Second, four tailored heuristic algorithms are proposed to generate near-optimal solutions efficiently. Finally, two metaheuristic strategies, the Genetic Algorithm (GA) and Simulated Annealing (SA) approaches, are developed to improve scalability and solution quality further.

Building upon the offline results, the thesis then extends the methodology to the **online regime**, where service requests arrive dynamically and decisions must be computed within strict time budgets. To ensure responsiveness, we design a time-windowed online simulated-annealing scheduler that warm-starts from previous solutions, applies lightweight neighborhood moves with incremental objective evaluation, and employs a time-aware cooling schedule to bound computation. Moreover, we cast online scheduling as a sequential decision-making problem and develop a cooperative multi-agent reinforcement learning (MARL) framework. In this framework, IoT devices and edge servers act as agents trained under a centralized training and decentralized execution approach, utilizing Deep Q-Networks (DQN) and Double DQN variants to determine whether each task should be executed locally, offloaded to the edge, or forwarded to the cloud. Once trained, the learned policies yield scheduling decisions with negligible inference delay.

The remainder of this introductory chapter is organized as follows. Section 2 provides background on mathematical optimization, IoT, edge computing, and reinforcement learning, and reviews related work. Section 3 formulates the research questions addressed in this thesis, while Section 4 outlines the research methodology. The main contributions are summarized in Section 5, and Section 7 provides an overview of the appended papers. Finally, Section 8 concludes this introduction by summarizing the thesis scope and discussing directions for future work.

2 Background and Related Work

This section offers a concise background on mathematical optimization and the foundational topics relevant to this thesis. Section 2.1 provides an overview of optimization, concentrating on linear programming, heuristics, and meta-heuristics key elements that are essential to our offline service placement formulations. Following this, Section 2.2 introduces reinforcement learning (RL) as a complementary data-driven method for sequential, online decision-making under uncertainty, such as variable task arrivals in IoT-edge-cloud systems. It also outlines the specific value-based techniques (DQN, DDQN) and multi-agent frameworks employed throughout this thesis. Section 2.3 delves into edge computing and the IoT devices, serving as an introduction to these criti-

cal areas. Finally, the most pertinent research related to our work is discussed in Section 2.4.

2.1 Mathematical Optimization

Mathematical optimization is a field of applied mathematics that focuses on finding the maximum or minimum values of a particular function, referred to as the objective function, while adhering to specific constraints. In the realm of computer science, mathematical optimization, often called programming, involves selecting the best solution or decision based on defined criteria. This section offers a brief overview of mathematical optimization and the methods relevant to solving task scheduling problems.

A succinct way to express a general optimization problem is given below. In the formulation, x denotes a vector of variables, and the objective is to find the maximal value of the function $f(x)$ given the constraint C [2].

$$\begin{aligned} \max \quad & f(x) \\ \text{s.t.} \quad & C \end{aligned}$$

There are numerous ways to formulate and solve an optimization problem; however, in the following, we only consider those most pertinent to this thesis work.

2.1.1 Linear Programming

Linear programming (LP) is a powerful mathematical optimization technique aimed at identifying the optimal solution to a problem characterized by a linear objective function and linear constraints expressed as equalities or inequalities:

$$\begin{aligned} \max \quad & \phi(x) = c^T x \\ \text{s.t.} \quad & Ax \leq b, \\ & x \geq 0, \end{aligned}$$

where x represents a vector of decision variables, c is a vector of objective coefficients, A is a matrix of coefficients that define the constraints, and b is a vector of constants that outline the constraints. The LP problem seeks to determine the values of the decision variables that will optimize the objective function. One of the most recognized methods for solving LP problems is the Simplex method [3], which involves several well-defined phases, as illustrated in Figure 1.

1. **Standard format of a linear problem:** The linear optimization problem must be presented in standard canonical form: the objective function should be framed as a maximization problem, all linear constraints must be expressed as less-than-or-equal-to inequalities, and all variables must be required to be non-negative.

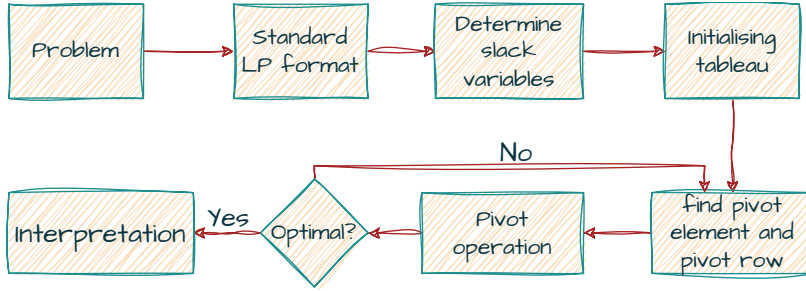


Figure 1: The steps required to solve an LP problem using the Simplex method.

2. **Determine slack variables:** To convert inequality constraints into equality constraints, slack variables can be introduced; these are additional variables that compensate for the difference in the inequalities.
3. **Initializing the tableau:** With the slack variables included, the tableau, the structured representation of the coefficients for the constraints and the objective function, can be set up for optimality testing. Row operations and optimality checks can be efficiently performed using the Simplex tableau.
4. **Find the pivot element:** Identify the pivot element's column by locating the objective function coefficient with the most significant negative value.
5. **Find the pivot row:** Each constraint's right-hand side is divided by its corresponding coefficient in the pivot column; the smallest positive value identifies the pivot row.
6. **Pivot operation:** Remove coefficients from the pivot column in all rows except for the row containing the pivot, and then normalize the pivot row by dividing it by the pivot element.
7. **Check for optimality:** If the objective function in the current tableau contains no negative coefficients, the solution is optimal. If improvement is still possible, repeat steps 4 to 6 until no further enhancement can be made.
8. **Interpretation:** Once the optimal solution has been determined, the next step is to interpret the values of the decision variables and the maximum achievable value of the objective function.

The IBM CPLEX optimizer harnesses the Simplex method to effectively solve LP problems [4].

Mixed Integer Linear Programming (MILP) problems represent a specialized category of LP problems, where certain decision variables are restricted to integer values.

$$\begin{aligned}
\min \quad & \phi(x) = c^T x \\
\text{s.t.} \quad & Ax \leq b, \\
& x_i \geq 0 \quad \forall i, \\
& x_i \in \mathbb{Z} \quad \forall i \in \mathcal{I}
\end{aligned}$$

In this formulation, x is the vector of decision variables, c is a vector of coefficients, A is a matrix of coefficients that define the constraints, b is a vector of constants, and the index set \mathcal{I} identifies those components of x that are restricted to integer values, while the remaining variables are continuous. One common approach for solving an MILP problem is the Branch and Bound technique [5]. This method utilizes a tree-based structure to derive lower bounds on the optimal objective function value by first partitioning the feasible region into smaller sub-regions, then solving LP relaxations of these sub-regions. These lower bounds are subsequently employed to eliminate potential sub-regions that cannot contain the optimal solution.

2.1.2 Heuristics and Metaheuristics

A heuristic is a problem-solving strategy designed to produce solutions quickly, though it does not guarantee an optimal outcome. Heuristics are particularly useful when finding an exact solution is computationally infeasible or impractical, especially within specific time constraints.

A metaheuristic is a higher-level, problem-independent algorithmic framework that provides guidelines or strategies for developing heuristic optimization algorithms. These techniques are adaptive and flexible, capable of solving various problems, including those lacking exact solutions. metaheuristics serve as principles for generating good approximations by integrating multiple heuristics, combining local and global search strategies [6].

A GA is a metaheuristic optimization technique that draws inspiration from the process of natural selection. The following steps outline how a GA may address a problem, as illustrated in Figure 2.

1. **Initialization:** The algorithm starts by randomly generating a population of potential solutions known as individuals, each represented by a unique set of parameters or values.
2. **Fitness Function:** The fitness function, which must be maximized or minimized, evaluates how well each solution performs. A fitness score is computed for every individual based on its efficiency in relation to the fitness function.
3. **Selection:** This procedure determines which individuals are selected to breed the next generation, relying on their fitness scores. A higher fitness score increases the likelihood of being chosen as a parent.

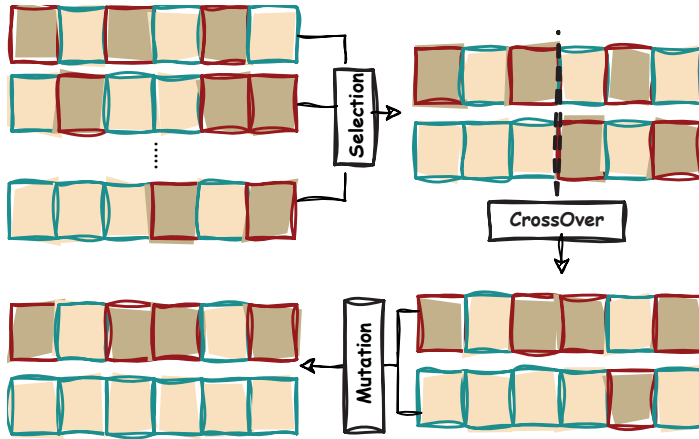


Figure 2: The process of selection, crossover, and mutation in a genetic algorithm.

4. **Crossover:** Crossover is the process by which new individuals are created by mixing characteristics of two parent solutions. This involves selecting two parents from the population and exchanging segments of their genetic information to produce an offspring.
5. **Mutation:** Mutation introduces random changes to an individual's genetic makeup. This step is crucial for maintaining diversity within the population, ensuring the algorithm does not converge too quickly on a suboptimal solution.
6. **Termination:** The algorithm concludes when it meets a stopping criterion, such as reaching a maximum number of iterations or achieving a specified fitness score threshold.

SA is a global search optimization algorithm inspired by the metallurgical process of annealing [7]. It can often find high-quality (near-optimal) solutions across a broad search space, particularly in scenarios where deterministic methods are too slow or impractical.

The algorithm initiates with a starting solution and a temperature parameter. Reflecting the cooling process in physical annealing, the temperature gradually decreases throughout the execution of the algorithm. At elevated temperatures, the algorithm may accept less optimal solutions with higher probabilities, allowing for extensive exploration of the search space. As the temperature declines, the algorithm becomes increasingly selective and typically converges on the best solution encountered during the process.

The simulated annealing algorithm can be summarized through the following steps, as depicted in Figure 3.

1. The algorithm begins with an initial solution, which may be generated randomly or obtained through a heuristic algorithm. An initial tem-

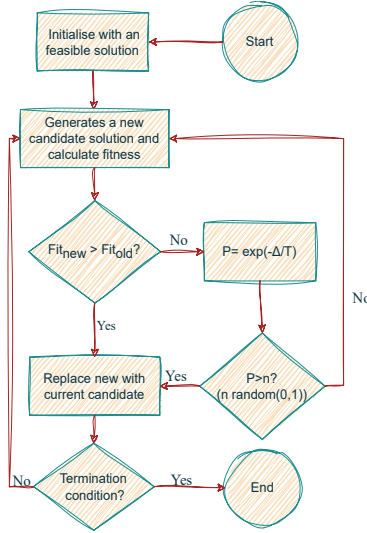


Figure 3: A flowchart of the simulated annealing algorithm.

perature and a cooling schedule are defined, dictating the temperature's reduction over time.

2. At each iteration, a new candidate solution is produced by randomly modifying the current solution.
3. The new solution is assessed using a fitness-measuring objective function, which can serve to either minimize or maximize a given outcome.
4. If the new solution is superior to the current one, it replaces the latter. If the new solution is suboptimal, it may still be accepted based on a probability that depends on the quality difference between solutions and the current temperature. This probability is calculated using the Boltzmann distribution:

$$P = \exp\left(\frac{-\Delta}{T}\right) \quad (1)$$

Here, Δ denotes the difference in quality between the new and current solutions, T is the current temperature, and \exp represents the exponential function. By implementing a probabilistic acceptance criterion, the simulated annealing algorithm broadens its search capabilities, minimizing the risk of getting trapped in local optima.

2.2 Reinforcement Learning

In service placement and task offloading problems, many decisions must be made online: requests arrive over time, link capacities fluctuate, and resource availability changes. Rather than precomputing a fixed schedule, RL aims to learn a *policy*, a rule for choosing actions, that performs well while interacting with a dynamic environment.

At a high level, an RL agent (e.g., a device or an edge scheduler) repeatedly: (i) observes the current *state* of the system (e.g., queue lengths, CPU/RAM/bandwidth utilization, deadlines); (ii) selects an *action* (e.g., execute locally, offload to the edge, or offload to the cloud); (iii) receives a scalar *reward* representing the design objective (e.g., minimizing latency, meeting deadlines, maximizing task acceptance); and (iv) observes the resulting transition to a new state. By trial and error over many iterations, the agent improves its policy to maximize the *long-run* reward.

Figure 4 illustrates the RL interaction loop. At each time step t , the agent observes the current state s_t , chooses an action a_t , obtains a reward r_{t+1} , and the environment transitions to a new state s_{t+1} . The goal is to learn a policy that prescribes which action to take in each state so as to maximize the expected long-term reward. RL methods learn directly from interaction data [8].

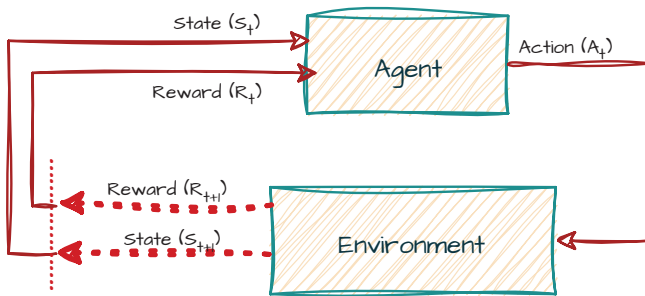


Figure 4: Reinforcement learning interaction loop.

This interaction can be formalized as a *Markov Decision Process* (MDP), but the underlying idea is simple: use feedback from the environment to learn improved sequential decision-making when models are incomplete or time-varying. In this thesis, we adopt value-based methods, Q-learning and its deep variants (DQN, DDQN), because the action space is discrete (e.g., where to place or offload tasks) and these methods perform well in large yet structured state spaces.

2.2.1 Markov Decision Processes

An MDP is defined by the tuple

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma),$$

where \mathcal{S} denotes the state space, \mathcal{A} the action space, $P(s' | s, a)$ the transition kernel, $R(s, a)$ the expected immediate reward, and $\gamma \in [0, 1)$ the discount factor. The Markov property implies that the future depends on the past only through the current state and action:

$$P(s_{t+1} | s_{0:t}, a_{0:t}) = P(s_{t+1} | s_t, a_t).$$

Given a policy $\pi(a | s)$, the objective is to maximize the expected discounted return

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}.$$

Two value functions summarize the expected future reward: the state-value function

$$V^\pi(s) = \mathbb{E}_\pi [G_t | s_t = s],$$

and the action-value function

$$Q^\pi(s, a) = \mathbb{E}_\pi [G_t | s_t = s, a_t = a].$$

An optimal policy π^* maximizes these values for all states and actions.

2.2.2 Model-Free Control (Tabular Q-Learning)

In model-free RL, the agent learns directly from sampled transitions without assuming prior knowledge of P or R . A canonical algorithm is *tabular Q-learning*, which iteratively updates the estimate of the optimal action-value function using temporal-difference learning:

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t) \right],$$

where $\alpha \in (0, 1]$ is the learning rate. Exploration is often handled using ε -greedy action selection. Tabular methods are feasible for small state-action spaces; for high-dimensional problems, Q is approximated by a function approximator such as a neural network (Figure 5).

2.2.3 Q-Networks (Function Approximation for Q-Learning)

When the state or action space is large, a *Q-network* parameterized by θ approximates $Q(s, a; \theta)$. For discrete actions, the network outputs one Q-value per action given input s . Parameters are optimized by minimizing the temporal-difference loss:

$$\mathcal{L}(\theta) = \mathbb{E}[(y_i - Q(s_i, a_i; \theta))^2], \quad y_i = r_i + \gamma \max_{a'} Q(s'_i, a'; \theta).$$

However, direct bootstrapping with a single network may be unstable due to correlations and off-policy updates, motivating the stabilizing techniques introduced in DQN.

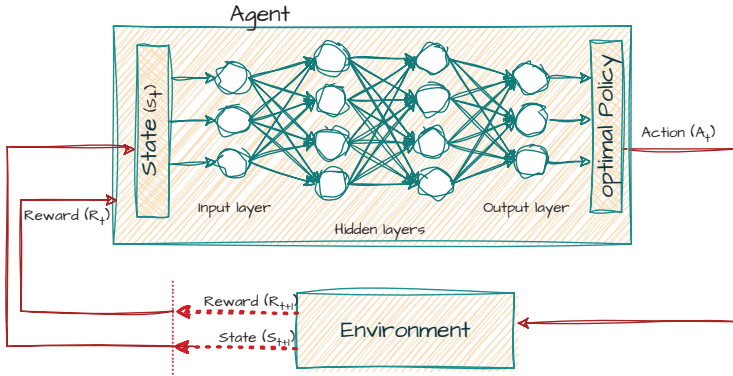


Figure 5: Deep Q-Network architecture.

2.2.4 Deep Q-Networks

DQN introduces two stabilization mechanisms:

- **Experience replay:** Transitions are stored in a replay buffer and sampled uniformly, reducing correlation among updates.
- **Target network:** A separate target network with parameters θ^- is maintained to compute target values. It is updated periodically or via a moving average from θ , improving stability.

The DQN target thus becomes

$$y_i = r_i + \gamma \max_{a'} Q(s'_i, a'; \theta^-),$$

and the loss is computed against the online network's prediction $Q(s_i, a_i; \theta)$. DQN has been shown to learn effective control policies directly from high-dimensional observations [9].

2.2.5 Double DQN

Since DQN uses the same network for action selection and evaluation, it can overestimate Q-values. Double DQN mitigates this bias by decoupling selection and evaluation:

$$y_i^{\text{DDQN}} = r_i + \gamma Q(s'_i, \arg \max_a Q(s'_i, a'; \theta), \theta^-).$$

Here, the online network (θ) selects the best action, while the target network (θ^-) evaluates it, reducing overestimation and improving convergence [10].

2.2.6 Multi-Agent Reinforcement Learning

In distributed IoT–edge–cloud systems, multiple agents (e.g., IoT devices and edge servers) act concurrently and influence one another through shared resources. From the viewpoint of any single agent, the environment appears non-stationary because others are also learning, motivating *multi-agent reinforcement learning* (MARL).

A stochastic (Markov) game is defined as

$$\mathcal{G} = (\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i \in \mathcal{N}}, P, \{R^i\}_{i \in \mathcal{N}}, \gamma),$$

where each agent $i \in \mathcal{N}$ selects actions $a^i \in \mathcal{A}^i$, the joint action $a = (a^1, \dots, a^n)$ induces transitions via $P(s'|s, a)$, and each receives reward $R^i(s, a)$. Agents may be cooperative (shared reward), competitive, or mixed. A widely used paradigm is *Centralized Training with Decentralized Execution* (CTDE), where agents use global information during training but act locally during inference for low-latency control.

Key challenges in MARL include: (i) *non-stationarity* (policy updates of one agent alter others' dynamics); (ii) *credit assignment* (disentangling individual contributions to team rewards); and (iii) *scalability and coordination* (the joint action space grows combinatorially). In this thesis, we adopt CTDE with value-based multi-agent extensions of DQN/DDQN and structured network heads that share representations while producing coupled offloading decisions.

2.3 IoT and Edge

The Internet of Things (IoT) denotes a network of physical devices equipped with embedded sensors, software, and communication capabilities that enable sensing, data collection, and connectivity. These devices autonomously acquire and transmit data over the Internet or other networks. The resulting IoT data can be processed and analyzed to provide valuable insights and support intelligent decision-making.

IoT has evolved through several generations. The first generation, which emerged in the 1970s, focused on developing networked devices for industrial and military applications. The second generation began in the 1990s with the advent of the Internet and the development of the TCP/IP protocol, paving the way for consumer applications of IoT. The third generation is defined by the proliferation of connected devices and the generation of vast amounts of data, which led to advancements in cloud computing and big data analytics. As we advance into the fourth industrial revolution, the integration of IoT devices into the global network has become increasingly prevalent. This integration generates a diverse array of data that can be analyzed and processed to deliver multiple services to end users. Cloud computing serves as a widely used platform to store and process this data, enabling the internet-based delivery of services such as virtual machines, storage, databases, software, and analytics. Users can readily access and release a shared pool of computing resources with minimal administrative effort and at high speed.

According to an IDC white paper sponsored by Seagate [11], global digital connectivity was projected to reach approximately six billion people, about 75% of the world's population, by 2025, with each connected individual generating a data interaction roughly every 18 seconds, largely driven by the rapid expansion of IoT devices, which are expected to produce over 90 zettabytes of data worldwide. These statistics underscore the substantial data output from IoT devices and emphasize the necessity for efficient storage and processing solutions to handle this influx of information. Expanding cloud computing capacity could address these challenges. However, potential issues may arise when numerous requests are directed to the cloud that require immediate processing and analysis. Specifically, when a substantial volume of requests is sent to the cloud, an overwhelming amount of data is transmitted across the network, potentially leading to congestion, delays, security concerns, high costs, and excessive bandwidth usage, particularly if the network is geographically dispersed and unstable.

To tackle these challenges, edge computing technology, including paradigms such as "fog computing," "cloudlets," "micro data centers," and "multi-access edge computing" [12–15], has emerged by relocating computational resources, such as CPU, RAM, storage, and data processing, closer to end users. An "edge device" can be any device capable of processing data and located in proximity to end users, such as at base stations or within local area networks. Edge computing further enhances the capabilities of IoT devices by enabling data processing at the edge, thereby diminishing the reliance on centralized data processing. This enhancement is particularly crucial in time-sensitive applications, such as gaming, augmented reality (AR), autonomous vehicles, and remote healthcare monitoring, where rapid response times are essential.

Below are the primary characteristics of edge computing [16, 17]:

- **Low-latency processing:** By executing computation close to data sources and end users, edge computing can significantly shorten response times compared with cloud-only approaches.
- **Support for near-real-time analytics:** Edge resources enable timely data processing that can improve end-to-end system responsiveness.
- **Reduced backhaul bandwidth:** Pre-processing, filtering, and aggregation at the edge can decrease the volume of data transmitted to central clouds.
- **Improved privacy and security potential:** Keeping sensitive data local can reduce exposure during transmission and limit reliance on centralized storage.
- **Enhanced service availability:** The distribution of edge resources can extend compute access and improve service continuity in geographically dispersed settings.
- **Lower operational costs:** Reducing cloud uploads can cut transmission-related costs and ease network resource demands.

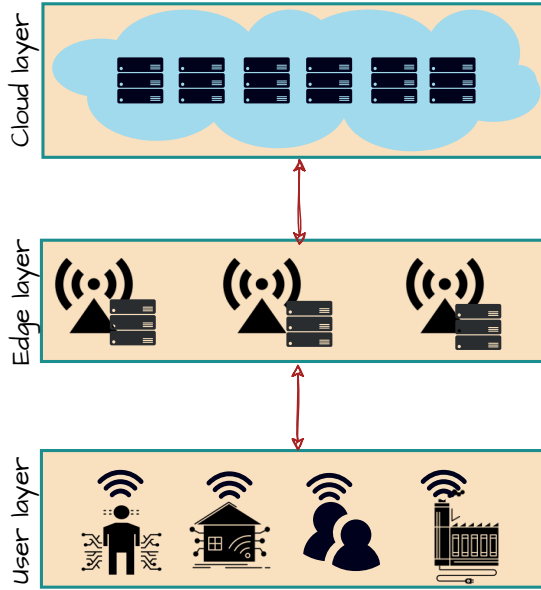


Figure 6: A typical three-layer edge-cloud network architecture.

- **Robustness to intermittent connectivity:** Local execution allows critical functions to continue even under unstable or limited network conditions.
- **Resource-capacity caveat:** For compute-intensive workloads, insufficient edge resources may lead to higher delays than cloud execution.

Figure 6 depicts a typical three-layer architecture of an edge-cloud network, spanning from the user layer to the edge layer and finally to the cloud layer. The massive data output from IoT devices necessitates an infrastructure capable of accommodating this influx. To meet these demands, a combination of 5G and edge computing can greatly benefit IoT devices and services. This synergy enhances communication networks and brings computational capabilities closer to users, yielding significant advantages for businesses. Together, they enhance application performance and are adept at handling large volumes of data, enabling businesses to rapidly respond to evolving demands.

2.4 Related Work

Prior research on task offloading and scheduling can be broadly categorized according to the underlying edge-IoT architecture assumed in the problem formulation. In particular, studies can be grouped into two classes: *two-layer architectures*, comprising a user layer and an edge/cloud layer, and *three-layer architectures*, which additionally include a dedicated edge layer between the user and cloud tiers.

Two-layer Architectures: Sharif *et al.* [18] developed a heuristic scheduling method for health monitoring applications that prioritizes tasks based on urgency, aiming to reduce bandwidth consumption and processing delay. Urgent tasks were executed locally at hospital workstations, while non-urgent ones were offloaded to the cloud. Zhu *et al.* [19] proposed a multi-objective scheduling strategy for Mobile Edge Computing (MEC) that jointly considers energy consumption and response time from the user perspective, and traffic load balancing from the edge provider's viewpoint. Ma *et al.* [20] proposed a heuristic scheduling approach that models inter-task relationships at the edge to determine optimal execution order and thereby minimize overall completion time.

Liao and Wu [21] designed a dynamic priority-based task scheduling algorithm within a hierarchical edge framework to minimize system latency, ensuring minimal delay for high-priority services. Fu *et al.* [22] studied a two-layer setup where resource-constrained IoT devices offload all processing to the edge; their mixed-integer nonlinear formulation aimed to minimize energy consumption. Liu *et al.* [23] proposed an edge-cooperative scheduling strategy that leverages idle device resources to reduce average task delay, whereas Zhang *et al.* [24] addressed multi-user, multi-task scheduling through energy-aware partitioning and dynamic voltage–frequency scaling, taking into account edge load and task deadlines.

Alghamdi *et al.* [25] and Deng *et al.* [26] introduced offloading schemes that minimize expected total delay and increase the likelihood of selecting the best-performing edge server. Tang *et al.* [27] developed a fully distributed, model-free deep reinforcement learning (DRL) scheme in which each device autonomously decides whether to compute locally or offload to an edge server using only local states and historical load traces, without global coordination. The policy minimizes a cost balancing delay and task-drop risk. Wang *et al.* [28] proposed an energy-efficient collaborative offloading method using DRL combined with graph neural networks; their algorithm (GCTO) exploits graph-based reinforcement learning and a penalty mechanism to enforce deadlines while jointly minimizing energy consumption and balancing load across devices. Similarly, Egwuche *et al.* [29] designed a distributed DRL framework for multi-domain IoT environments, employing convolutional neural networks and deep Q-learning to derive optimal offloading decisions from local information alone.

Our work differs from these studies in both architectural scope and modeling detail. In contrast to most prior work that assumes a two-layer architecture (e.g., device–cloud or edge–cloud), we consider a three-layer architecture comprising the device layer, the edge layer, and the cloud layer. We treat each service as a set of interdependent tasks that must be scheduled jointly, whereas most prior work considers independent tasks. Moreover, we model compute, memory, and bandwidth capacities explicitly at both node and service levels, which enables more realistic placement and improved performance for bandwidth-sensitive workloads.

Three-layer Architectures: Lou *et al.* [30] formulated a scheduling method

that minimizes task execution time across edge-cloud resources under startup latency and bandwidth constraints, considering multiple dependent tasks released by mobile devices. However, service frequency was not modeled. Bali *et al.* [31] proposed a priority-aware scheduling approach that classifies tasks by urgency to enhance Quality of Service (QoS), determining whether they should execute locally, at the edge, or in the cloud based on device capability. Ning *et al.* [32] applied a branch-and-bound heuristic at the edge to minimize execution time for single-device offloading and later extended it to resource sharing among multiple IoT devices. Lin *et al.* [33] developed a distributed, application-aware scheduling framework that adapts to job types to balance load and improve Quality of Experience (QoE). Wu *et al.* [34] introduced a deep-learning-driven offloading system capable of near-optimal decision-making for edge and cloud task allocation, while Zhao *et al.* [35] examined bounded-delay task execution across mobile, edge, and cloud layers to reduce energy usage. Kim *et al.* [36] exploited idle IoT resources for cooperative edge computing without affecting device performance, and Wang *et al.* [37] proposed a joint scheduling method to minimize processing time and power consumption. Hao *et al.* [38] presented a priority-aware MEC offloading framework that makes online task-level decisions to reduce waiting times and energy consumption via a deep reinforcement learning approach. Finally, Goudarzi *et al.* [39] proposed a distributed DRL framework for placing DAG-structured IoT application tasks across edge, fog, and cloud resources, optimizing end-to-end latency and device energy via a weighted cost model and pre-scheduling critical paths.

Compared to these efforts, our work provides a more comprehensive scheduling scope and richer system model. We jointly schedule services comprising multiple dependent tasks while accounting for detailed service-level and infrastructure-level parameters, including compute, memory, and bandwidth constraints. Furthermore, unlike many batch-based approaches, we operate in an *online* setting where services arrive continuously and are scheduled immediately upon arrival. This reduces decision latency and prevents queue buildup, thereby improving responsiveness and overall system efficiency.

3 Research Questions

The overarching goal of this thesis is to contribute to the design of resource-efficient IoT systems that can deliver low latency services with predictable performance in a three-tier device-edge-cloud infrastructure. More concretely, the thesis investigates both *offline* and *online* task scheduling and offloading of user-requested services within such an infrastructure, with the main objective of minimizing the *total system delay*, that is, the total time required to complete all tasks across all services requested by all users. To achieve this goal, the research has been structured around the following three research questions (RQs).

RQ1 *How can a system model for task scheduling in a three-layer IoT-edge-cloud*

architecture be designed and mathematically formulated?

When this research was initiated, most published studies on service and task offloading focused on two-layer edge-cloud architectures, while the relatively few that addressed three-layer architectures often neglected important characteristics of nodes, links, and task dependencies.

To address this gap, a comprehensive system model capturing the interaction between user devices, edge servers, and cloud servers was proposed in **Paper I**. This model incorporates critical information about network connectivity, available computing and memory resources, and workload specifications at each layer of the infrastructure. It provides a foundation for formulating the service offloading and scheduling problem as a MILP. The same model was extended and reused in **Paper II** and **Paper III**, and later adapted for online decision-making in **Paper IV** and **Paper V**, thereby ensuring methodological consistency between the offline and online formulations.

RQ2 *How can offline task scheduling problems in a three-layer IoT-edge-cloud architecture be solved both accurately and efficiently?*

Exact methods, such as the Simplex algorithm [3] combined with Branch-and-Bound [5], can deliver optimal solutions but are computationally demanding and impractical for large instances. To address scalability, several alternative approaches were investigated. First, the MILP problem was solved exactly using the Simplex-Branch-and-Cut method in IBM CPLEX to establish upper-bound benchmarks. Next, four tailor-made heuristics were developed in **Paper I** to yield near-optimal results with reduced computational cost. However, heuristics may struggle to satisfy all problem constraints in complex environments, motivating the exploration of more flexible *metaheuristic* methods. Two metaheuristic algorithms were therefore proposed: an evolutionary algorithm based on the genetic algorithm framework [42] (**Paper II**), and a simulated annealing approach [7] (**Paper III**), which achieves comparable accuracy while converging significantly faster.

Together, these methods provide a spectrum of trade-offs between optimality and scalability for offline scheduling in IoT-edge-cloud systems.

RQ3 *How can online task scheduling problems in a three-layer IoT-edge-cloud architecture be solved under dynamic and time-constrained conditions?*

In online settings, service requests arrive over time, and decisions must be made sequentially with incomplete knowledge of future arrivals and under strict per-decision time budgets. Given the computational cost of exact optimization, two complementary approaches were designed to generate high-quality schedules rapidly. First, **Paper IV** introduces an online simulated annealing scheme that incrementally refines the current schedule as new information becomes available. This method warm-starts from the previous solution, applies lightweight neighborhood moves with incremental objective evaluation, and uses a time-aware cooling

strategy to ensure convergence within each decision epoch. Second, **Paper V** formulates online scheduling as a sequential decision-making problem and applies value-based DRL. A cooperative multi-agent RL framework models IoT devices and edge schedulers as agents, each trained with DQNs and Double DQN architectures extended with multi-head Q-networks to manage service heterogeneity. This multi-head Double DQN (MH-DDQN) design improves training stability and convergence speed, resulting in robust performance under high and variable workloads. Collectively, these two methods demonstrate how adaptive online scheduling can be achieved in dynamic IoT–edge–cloud environments while maintaining near-optimal latency performance.

4 Research Methods

Approaches, techniques, and instructions for conducting research constitute parts of research methodologies. In other words, a research methodology entails solving problems by collecting and analyzing data according to a predetermined set of processes and principles [40]. Generally, the methods can be divided into two main categories: quantitative and qualitative. Quantitative research comprises various methodologies that employ statistical or numerical analysis. It generates predictions, examines correlations, and extends findings to larger groups. The qualitative research method aims to better understand reasons, opinions, and motivations. It enlightens us about the issue and allows us to formulate hypotheses or ideas for future quantitative studies. Participant observation, one-on-one interviews, and group discussions are all frequent techniques in qualitative research.

This thesis employs a scientific research methodology inspired by a quantitative approach described below and shown in Figure 7.

1. *Defining the research question:* The first stage in scientific research is to define the research issue or problem to be addressed, and may involve investigating a new technology or creating a solution to an existing problem.
2. *Conducting a literature review:* The researchers then perform a literature review to uncover pertinent studies and assess the field's current state of the art. A hypothesis or research topic is developed based on this information and then tested via experimentation or simulation.
3. *Formulating a hypothesis:* Based on the research question, researchers develop a hypothesis, a provisional explanation of the phenomenon under investigation.
4. *Designing and conducting experiments:* Typically, experimental research requires the design and execution of experiments to collect data that can be used to test hypotheses or validate models. Researchers may collect

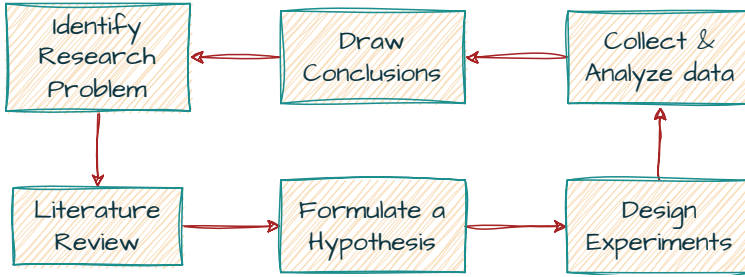


Figure 7: Research methods used in the thesis.

data through various means, including surveys, interviews, and observations. They may also utilize current data sets or conduct experiments to obtain new data.

5. *Collecting and analyzing data:* Once data is collected, researchers employ statistical approaches to examine obtained data in order to draw conclusions and reveal trends or correlations. This information is subsequently used to conclude the examined research subject or problem.
6. *Drawing conclusions:* Researchers determine the validity of a hypothesis or research topic based on the collected and processed data. These findings could result in new theories, technology, or solutions to existing problems.

In addition to the above-described scientific research method, hypothesis testing is a statistical technique used to evaluate the significance of a hypothesis about a population parameter based on a sample of data. In computer science, hypothesis testing is commonly used to assess the performance of algorithms, models, and systems and to make decisions based on data analysis. Simulation can be used for testing in scientific research. Simulation models are computer-based models that simulate the behavior of a system or process under different conditions. They can be used to test different scenarios and predict the outcomes of changes in the system.

This thesis identifies a gap in existing work on task scheduling and offloading in edge computing. It then conducts a focused literature review to identify relevant studies and assess the current state of the art in IoT–edge–cloud scheduling and offloading. Based on this analysis, the research problem is refined and the research questions **RQ1**, **RQ2**, and **RQ3** are formulated.

Building on these research questions, the thesis advances a set of overarching hypotheses concerning the effectiveness of optimization- and learning-based task scheduling approaches in reducing end-to-end latency and improving resource utilization under both offline and online settings. These hypotheses are investigated through a sequence of models, algorithms, and experiments presented across the included papers, which are structured as follows.

In **Paper I**, the task scheduling problem is formulated as a mixed-integer linear programming (MILP) problem with the objective of minimizing total latency, thereby establishing a formal optimization baseline. Analytical models are developed for the MILP formulation, and CPLEX solves MILP to proven optimality via branch-and-cut/branch-and-bound over LP relaxations. However, since exact methods become computationally prohibitive as problem size scales, **Paper I** further proposes heuristic solutions, while **Paper II** and **Paper III** introduce metaheuristic approaches.

As the methods presented in **Paper I**, **Paper II**, and **Paper III** address the offline setting, where all task requests are assumed to be known in advance, two additional approaches are introduced to handle dynamic arrivals in online environments. Specifically, **Paper IV** presents a time-windowed Online Simulated Annealing scheduler, while **Paper V** proposes a value-based deep reinforcement learning approach that explicitly accounts for per-epoch decision budgets.

For offline methods, this study uses a real-world data set of mobile users and base station locations [41]. In particular, the data set specifies the spatial distribution of users and the static locations of base stations. These inputs are used to instantiate the network layout and to derive feasible association and offloading choices. This data set was employed in all experiments in **Paper I**, **Paper II**, and **Paper III**, to guarantee that all methods used the same infrastructure for assessment. We leveraged the CPLEX as a MILP solver as a benchmark to validate the proposed methods, as it provides the exact solution. CPLEX is used as an exact MILP solver to obtain proven-optimal solutions, and it serves as a benchmark for quantifying how close the proposed methods are to the optimum.

For the online methods, **Paper IV** applies the method under three traffic-prediction models by assuming our proposed method uses a pre-trained ML-based predictor. Prediction errors are modeled as Gaussian noise added to the true service frequency, yielding neutral, conservative, and optimistic predictions, whereas **Paper V** benchmarks the reinforcement-learning approach against several heuristic baselines. We examine our problem in terms of accuracy and solving time and then analyze our proposed methods in these terms to see how they operate in different situations with various circumstances. We define three scenarios that cover all IoT devices, edge servers, and service scales, from small to large. Multiple tests can verify that our solution is applicable in different situations, which leads to a reliable and trustworthy solution.

All of our proposed methods were built using Python programming. Then we supplied results based on diagrams analyzed and verified.

5 Contributions

This thesis presents a framework for optimizing task scheduling and offloading in IoT-edge-cloud systems, combining mathematical optimization for offline planning with learning-based techniques for online adaptation. The contributions span the full problem spectrum, from system modeling and exact formu-

lations to scalable heuristic and learning-based solutions, and are structured to progressively address the research questions defined in Section 3.

In particular, this work first develops a system model and MILP formulation that capture the interplay between users, edge nodes, and cloud servers under heterogeneous resource constraints. It then proposes a set of exact, heuristic, and metaheuristic optimization methods that balance accuracy and computational efficiency for large-scale scheduling instances. Finally, it extends the framework to the online setting, introducing both metaheuristic and reinforcement learning-based schedulers capable of real-time decision-making under dynamic workloads. Collectively, these contributions advance the state of the art in scalable and adaptive service scheduling across the IoT-edge-cloud continuum. The main contributions are summarized below.

1. *A system model and mathematical formulation for minimizing total system delay in offline task scheduling within an IoT-edge-cloud infrastructure.* We design a three-layer architecture that integrates the *user*, *edge*, and *cloud* layers. The model captures both node- and link-level characteristics by incorporating CPU, RAM, and network capacities, as well as the bandwidth and latency between all nodes. Each task within a service is characterized by its specific CPU, RAM, and bandwidth requirements. The corresponding offline task-scheduling problem is formulated as a Mixed-Integer Linear Program (MILP) with the objective of minimizing the sum of end-to-end service delay across the entire system. This formulation and its analysis are presented in **Paper I**, addressing **RQ1**.
2. *Exact, heuristic, and metaheuristic solutions for an offline joint task placement and resource allocation problem in IoT-edge-cloud systems, where known interdependent service tasks are assigned under CPU, memory, and bandwidth constraints to minimize end-to-end delay.*

In **Paper I**, the MILP problem is solved to proven optimality using IBM CPLEX (via branch-and-cut/branch-and-bound over LP relaxations), providing optimal benchmark results. To provide scalable alternatives, we develop four problem-specific heuristic strategies that prioritize tasks based on criteria such as CPU demand and service frequency. These heuristics demonstrate that lightweight scheduling can achieve near-optimal performance while substantially reducing computational cost. To further improve scalability and accuracy, two metaheuristic algorithms are proposed. The first, introduced in **Paper II**, is the *Enhanced Healing Genetic Algorithm (EHGA)*, which leverages evolutionary operations to explore the solution space effectively. EHGA achieves high-quality solutions for small and medium-sized instances but exhibits slower convergence for large-scale problems compared to exact methods. To overcome this, **Paper III** presents a *Simulated Annealing (SA)*-based algorithm that attains near-optimal results within a fraction of the runtime of the exact solver, outperforming both heuristic and EHGA-based approaches. Together, these methods span an accuracy–runtime

spectrum: CPLEX is optimal but costly, heuristics are fastest but less reliable, EHGA improves robustness and solution quality over heuristics, and SA provides the best overall balance of accuracy and efficiency among the non-exact methods, contributing to **RQ2**.

3. *Online scheduling approaches for dynamic task offloading in IoT-edge-cloud environments.*

Extending the same architectural model to dynamic conditions, we propose two complementary online scheduling frameworks that operate under partial knowledge and time-varying workloads. The first, presented in **Paper IV**, introduces an *Online Simulated Annealing-based Task Scheduler (SATS)* that continuously refines task placement as new requests arrive. SATS reuses the previous scheduling state, applies incremental neighborhood searches with lightweight objective updates, and employs a time-aware cooling scheme to meet real-time decision constraints. The second, described in **Paper V**, formulates task scheduling as a sequential decision-making problem solved via *Deep Reinforcement Learning (DRL)*. Using a cooperative multi-agent design with Deep Q-Networks (DQN) and Double DQN, the framework enables decentralized agents, representing IoT devices and edge servers, to learn efficient policies that minimize end-to-end latency and improve deadline satisfaction. Stability mechanisms such as target networks and experience replay ensure robust convergence and fast inference during deployment. These contributions address **RQ3**.

6 Limitations

The contributions of this thesis are developed under a number of modeling and evaluation assumptions. The offline models assume that services and task dependencies are known in advance and that resource and network conditions are reasonably stable. These assumptions are useful for establishing clear baselines and enabling fair comparisons, but they do not fully capture environments with high uncertainty or rapid dynamics. In addition, for the online approaches, performance depends on the quality of learning and where applicable, and further investigation is required to assess generalization and coordination overhead at larger scales. Finally, as the results are obtained through simulation, complementary validation in real or high-fidelity emulated testbeds would be valuable to strengthen the practical evidence and to better account for platform-specific constraints.

7 Summary of Appended Papers

Paper I - Optimal Placement of Recurrent Service Chains on Distributed Edge-Cloud Infrastructures

In this work, a multi-level architecture is developed to address the issues of offline service placement and task scheduling in edge/cloud architectures. It models the CPU, RAM, and network capacity for all IoT devices, edge servers, and cloud server, together with link bandwidth and latency. We formalize the offline problem of service placement/task offloading as an MILP to reduce the system's total delay after completing all tasks for all services requested by all users. The Simplex method offers an exact solution, but at the cost of considerable computing time and capacity. As an alternative, four computationally inexpensive and effective heuristic approaches are developed.

Paper II - EHGA: A Genetic Algorithm Based Approach for Scheduling Tasks on Distributed Edge-Cloud Infrastructures

Paper I showed that the exact methods utilized for solving the MILP problem are computationally- and time-intensive, while the heuristics employed are dependent on the problem at hand. Therefore, to solve the MILP problem, we propose an enhanced healing method based on a genetic algorithm (EHGA). Using a healing characteristic, EHGA is able to recover from unsuccessful offsprings. EHGA adds an additional chromosome based on the total available population after healing, both to accelerate and to prevent early convergence. We use IBM's CPLEX optimizer to solve the MILP problem and to establish a benchmark. We finally evaluate EHGA against alternative methods. The evaluation demonstrates that EHGA provides more dependable and accurate solutions than the heuristics, with shorter runtimes than CPLEX for the MILP.

Paper III - An Efficient Simulated Annealing-based Task Scheduling Technique for Task Offloading in a Mobile Edge Architecture

The findings presented in Paper I highlight that exact methods are not cost-effective, while heuristic approaches tend to be problem-dependent. In contrast, the proposed meta-heuristic approach outlined in Paper II demonstrates a lower computational intensity compared to exact methods and is less tailored to specific problems than traditional heuristics. However, its speed remains inadequate. To address this issue, the current research introduces a simulated annealing-based scheduling technique aimed at optimizing the task offloading problem within a three-layer IoT-edge-cloud architecture. This study compares simulated annealing against several alternatives: the simplex method, two heuristic-based approaches, TCA (Task Continuation Affinity) and MPCPU (Most Powerful CPU), as well as a genetic algorithm, EHGA. Evaluation metrics include accuracy and computation time across three distinct test campaigns, each varying in the number of services, users, or edge servers while

maintaining consistent other characteristics. The results indicate that simulated annealing can achieve results comparable to those of the simplex method in a fraction of the time required by the CPLEX optimizer. Moreover, while EHGA exhibits accuracy levels nearly matching those of simulated annealing, it incurs a significantly longer computation time. TCA and MPCPU demonstrate the shortest computation times; however, their accuracy is heavily influenced by the system load and available resources.

Paper IV - An Online Simulated Annealing-based Task Offloading Strategy for a Mobile Edge Architecture

The paper addresses the challenge of online scheduling in a three-tier device-edge-cloud system, where services (task chains with deadlines) arrive over time. It enhances the Simulated Annealing (SA) algorithm for this context by operating within defined time windows. Each new time window is initialized using the best schedule from the previous window, leveraging lightweight neighborhood moves with incremental evaluation, and applying time-aware acceptance with two decaying temperatures. A straightforward demand predictor aids in the planning process. Among the neutral, optimistic, and conservative variants of the predictor, the conservative approach, characterized by intentional overestimation, demonstrates the most robust performance. In various scenarios with differing device counts and request frequencies, this method achieves high deadline satisfaction and short processing times, all while ensuring that computation within each time window remains bounded. This makes Simulated Annealing a practical choice for online scheduling in this environment.

Paper V - Deadline -Aware Service Scheduling via Multi-Head MARL in Device-Edge-Cloud Environments

This work presents a framework for deadline-aware scheduling across the device-edge-cloud architecture, formulated as a cooperative multi-agent reinforcement learning problem. In this framework, both devices and the edge server are treated as agents. Devices must decide whether to execute tasks locally or offload them to the edge server, while the edge server must choose between executing tasks itself or forwarding them to the cloud. A reward mechanism that considers deadlines encourages timely task completion. To enhance learning stability, we employ function approximation using multi-head Q-networks, with one head dedicated to each service type, and utilize a Double-DQN target. Replay buffers and target networks are also implemented to stabilize the learning process further. Experiments demonstrate that the reinforcement learning policies outperform static baseline methods in terms of meeting deadlines while maintaining competitive processing times.

8 Conclusions and Future Work

This thesis investigates the problem of task scheduling and offloading across the IoT-edge-cloud continuum, aiming to minimize overall system delay un-

der heterogeneous workloads and resource constraints. The research spans both offline optimization and online learning perspectives, providing a unified framework that integrates mathematical modeling, metaheuristic search, and reinforcement learning.

We began by formulating the offline scheduling problem as a MILP that accounts for the computational, memory, and bandwidth capacities of all nodes as well as link latency and inter-task dependencies. Solving this formulation to proven optimality using IBM CPLEX (via branch-and-cut/branch-and-bound over LP relaxations) provided an optimal benchmark for evaluating approximate methods. To overcome the computational intractability of exact optimization for large-scale instances, several heuristic and metaheuristic algorithms have been developed. The proposed context-aware heuristics provided rapid and feasible solutions, while the metaheuristic approaches, namely the evolutionary-based EHGA and the SA algorithm, achieved near-optimal delay performance with considerably lower computational cost. In particular, the SA-based method produced results within 3–5% of the MILP optimum while reducing runtime by more than an order of magnitude, making it a scalable choice for practical deployments.

Building on this foundation, two online schedulers were proposed to address time-varying workloads and partial system knowledge. The first, SATS, extended simulated annealing to an online context by performing incremental neighborhood searches at each decision epoch, guided by predictions of incoming service requests. The second, a cooperative MARL approach, modeled IoT devices and edge servers as decentralized agents trained through centralized training with decentralized execution. Employing DQN and Double DQN techniques, this method effectively captured the dynamic nature of the system, achieving high deadline compliance and low latency while maintaining adaptability under variable traffic conditions. Together, these contributions demonstrate that combining analytical optimization with adaptive learning mechanisms yields scalable, delay-efficient, and resilient scheduling strategies for modern IoT–edge–cloud systems.

Future research can extend this framework in several directions. One key challenge is to relax the assumption of static nodes and incorporate device mobility, which introduces factors such as variable backhaul quality, handover costs, and migration latency. Addressing these aspects will require mobility-aware scheduling and predictive control. On the algorithmic side, complementing the proposed off-policy learning with on-policy actor–critic methods such as Proximal Policy Optimization or Advantage Actor–Critic may provide deeper insights into stability–efficiency trade-offs in non-stationary and continuous action spaces. History- and context-aware mechanisms based on recurrent or attention-based neural architectures could further improve adaptability by exploiting temporal correlations in workloads and system dynamics.

Beyond delay minimization, future work should broaden the optimization objectives to jointly consider energy efficiency, fairness, and reliability under realistic constraints. Evaluating robustness under bursty arrivals, partial failures, and uncertainty in network state would also strengthen the prac-

tical relevance of the proposed methods. Finally, implementing and validating the algorithms in a real or emulated edge-cloud testbed with physical IoT devices would bridge the gap between simulation-based evaluation and deployment-ready solutions, contributing to the realization of intelligent and self-optimizing scheduling for next-generation 6G-enabled IoT systems.

References

- [1] IBM, “CPLEX Optimizer”.
<https://www.ibm.com/analytics/cplex-optimizer>. Accessed Jan. 2026.
- [2] Igor Griva, Stephen G. Nash, Ariela Sofer., ‘Linear and nonlinear optimization’, 2nd ed., Society for Industrial and Applied Mathematics(SIAM), 2009.
- [3] G. B. Dantzig and M. N. Thapa, ‘Linear Programming’. New York: Springer-Verlag, 1997. doi: 10.1007/b97672.
- [4] IBM ILOG CPLEX Optimization Studio Optimizers User’s Manual, Version 22.1.1, IBM Documentation.
<https://www.ibm.com/docs/en/icos/22.1.1?topic=optimizers-users-manual-cplex>. Accessed Jan. 2026.
- [5] H. A. Taha, Operations Research An Introduction, Tenth Edition. Boston: Pearson, 2017.
- [6] Blum, C. and Roli, A. (2003). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, 35(3), 268-308.
- [7] D. Bertsimas and J. N. Tsitsiklis, “Simulated annealing,” *Statistical Science*, vol. 8, no. 1, pp. 10–15, 1993.
- [8] Sutton, R. S., Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). The MIT Press.
- [9] V. Mnih et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.
- [10] H. van Hasselt, A. Guez, and D. Silver, “Deep Reinforcement Learning with Double Q-Learning,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, Mar. 2016, doi: 10.1609/aaai.v30i1.10295.
- [11] D. Reinsel, J. Gantz, and J. Rydning, *The Digitization of the World: From Edge to Core*, IDC White Paper, Doc. No. US44413318, sponsored by Seagate Technology, Nov. 2018, data refreshed May 2020.
- [12] Sabireen H., Neelanarayanan V., A Review on Fog Computing: Architecture, Fog with IoT, Algorithms and Research Challenges, *ICT Express*, Volume 7, Issue 2, 2021, Pages 162-176, ISSN 2405-9595, doi.org/10.1016/j.icte.2021.05.004.
- [13] M. Babar, M. S. Khan, F. Ali, M. Imran and M. Shoaib, “Cloudlet Computing: Recent Advances, Taxonomy, and Challenges,” in *IEEE Access*, vol. 9, pp. 29609-29622, 2021, Doi: 10.1109/ACCESS.2021.3059072.

- [14] F. Vhora and J. Gandhi, "A Comprehensive Survey on Mobile Edge Computing: Challenges, Tools, Applications," 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2020, pp. 49-55, doi: 10.1109/ICCMC48092.2020.ICCMC-0009.
- [15] K. Bilal, O. Khalid, A. Erbad, S. U. Khan, "Potentials, Trends, and Prospects in Edge Technologies: Fog, Cloudlet, Mobile Edge, and Micro Data Centers, *Computer Networks*", Volume 130, 2018, Pages 94-120, ISSN 1389-1286, doi.org/10.1016/j.comnet.2017.10.002.
- [16] F. Shirin Abkenar et al., "A Survey on Mobility of Edge Computing Networks in IoT: State-of-the-Art, Architectures, and Challenges," in *IEEE Communications Surveys and Tutorials*, vol. 24, no. 4, pp. 2329-2365, Fourthquarter 2022, doi: 10.1109/COMST.2022.3211462.
- [17] M. Mahbub and R. M. Shubair, "Contemporary advances in multi-access edge computing: A survey of fundamentals, architecture, technologies, deployment cases, security, challenges, and directions," *Journal of Network and Computer Applications*, vol. 219, p. 103726, Oct. 2023, doi: 10.1016/j.jnca.2023.103726.
- [18] Z. Sharif, L. Tang Jung, M. Ayaz, and M. Yahya, "Priority-based Task Scheduling and Resource Allocation in Edge Computing For Health Monitoring System," *J. King Saud Univ. - Comput. Inf. Sci.*, Jan. 2023, doi: 10.1016/j.jksuci.2023.01.001.
- [19] S. Zhu, J. Cai, and E. Sun, "Mobile Edge Computing Offloading Scheme Based on Improved Multi-objective Immune Cloning Algorithm," *Wirel. Netw.*, Jan. 2023, doi: 10.1007/s11276-022-03157-9.
- [20] T. Ma, M. Wang, and W. Zhao, "Task scheduling considering multiple constraints in mobile edge computing," in *2021 International Conference on Intelligent Computing, Automation and Systems (ICICAS)*, Dec. 2021, pp. 43-47. doi: 10.1109/ICICAS53977.2021.00016.
- [21] J. X. Liao and X. W. Wu, "Resource Allocation and Task Scheduling Scheme in Priority-Based Hierarchical Edge Computing System," in *2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, Oct. 2020, pp. 46-49. doi: 10.1109/DCABES50732.2020.00021.
- [22] Y. Fu et al., "Energy-efficient offloading and resource allocation for mobile edge computing enabled mission-critical internet-of-things systems," *EURASIP J. Wirel. Commun. Netw.*, vol. 2021, no. 1, p. 26, Feb. 2021, doi: 10.1186/s13638-021-01905-7.
- [23] Y. Liu, G. Xiong, F. Zhu, S. Chen, L. Zhang, and X. Liu, "Collaborative Scheduling of Computing Tasks for Edge Computing," in *2021*

- China Automation Congress (CAC), Oct. 2021, pp. 7824–7831. doi: 10.1109/CAC53003.2021.9727868.
- [24] J. Zhang, B. Jiang, H. Zhao, and Y. Xu, “Time-Sensitive Multi-User Oriented Mobile Edge Computing Task Scheduling Algorithm,” in 2020 2nd International Conference on Computer Communication and the Internet (ICCCI), Jun. 2020, pp. 145–149. doi: 10.1109/ICCCI49374.2020.9145970.
- [25] I. Alghamdi, C. Anagnostopoulos, and D. P. Pezaros, “Time-Optimized Task Offloading Decision Making in Mobile Edge Computing,” in 2019 Wireless Days (WD), Apr. 2019, pp. 1–8. doi: 10.1109/WD.2019.8734210.
- [26] S. Deng et al., “Optimal Application Deployment in Resource Constrained Distributed Edges,” *IEEE Trans. Mob. Comput.*, vol. 20, no. 5, pp. 1907–1923, May 2021, doi: 10.1109/TMC.2020.2970698.
- [27] M. Tang and V. W. S. Wong, “Deep Reinforcement Learning for Task Offloading in Mobile Edge Computing Systems,” in *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 1985–1997, 1 June 2022, doi: 10.1109/TMC.2020.3036871.
- [28] Shudong Wang, Shengzhe Zhao, Haiyuan Gui, Xiao He, Zhi Lu, Baoyun Chen, Zixuan Fan, Shanchen Pang, “Energy-efficient collaborative task offloading in multi-access edge computing based on deep reinforcement learning”, *Ad Hoc Networks*, Volume 169, 2025, 103743, ISSN 1570-8705, <https://doi.org/10.1016/j.adhoc.2024.103743>.
- [29] O. S. Egwuche, J. Greeff and A. El-Shamir Ezugwu, “Optimized Task Offloading in Multi-Domain IoT Networks Using Distributed Deep Reinforcement Learning in Edge Computing Environments,” in *IEEE Access*, vol. 13, pp. 26193–26207, 2025, doi: 10.1109/ACCESS.2025.3535870.
- [30] J. Lou, Z. Tang, W. Jia, W. Zhao, and J. Li, “Startup-aware Dependent Task Scheduling with Bandwidth Constraints in Edge Computing,” *IEEE Trans. Mob. Comput.*, pp. 1–15, 2023, doi: 10.1109/TMC.2023.3238868.
- [31] M. S. Bali, K. Gupta, D. Gupta, G. Srivastava, S. Juneja, and A. Nauman, “An effective technique to schedule priority aware tasks to offload data on edge and cloud servers,” *Meas. Sens.*, vol. 26, p. 100670, Apr. 2023, doi: 10.1016/j.measen.2023.100670.
- [32] Z. Ning, P. Dong, X. Kong, and F. Xia, “A Cooperative Partial Computation Offloading Scheme for Mobile Edge Computing Enabled Internet of Things,” *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019, doi: 10.1109/JIOT.2018.2868616.

- [33] L. Lin, P. Li, J. Xiong, and M. Lin, "Distributed and Application-Aware Task Scheduling in Edge-Clouds," in 2018 14th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN), Dec. 2018, pp. 165–170. doi: 10.1109/MSN.2018.000-1.
- [34] H. Wu, Z. Zhang, C. Guan, K. Wolter, and M. Xu, "Collaborate Edge and Cloud Computing With Distributed Deep Learning for Smart City Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8099–8110, Sep. 2020, doi: 10.1109/JIOT.2020.2996784.
- [35] T. Zhao, S. Zhou, L. Song, Z. Jiang, X. Guo, and Z. Niu, "Energy-optimal and delay-bounded computation offloading in mobile edge computing with heterogeneous clouds," *China Commun.*, vol. 17, no. 5, pp. 191–210, May 2020, doi: 10.23919/JCC.2020.05.015.
- [36] Y. Kim, C. Song, H. Han, H. Jung, and S. Kang, "Collaborative Task Scheduling for IoT-Assisted Edge Computing," *IEEE Access*, vol. 8, pp. 216593–216606, 2020, doi: 10.1109/ACCESS.2020.3041872.
- [37] Q. Wang, S. Guo, J. Liu, Y. Yang, "Energy-efficient computation offloading and resource allocation for delay-sensitive mobile edge computing", *Sustainable Computing: Informatics and Systems*, Volume 21, 2019, Pages 154-164, ISSN 2210-5379, <https://doi.org/10.1016/j.suscom.2019.01.007>.
- [38] H. Hao, C. Xu, W. Zhang, S. Yang and G. -M. Muntean, "Task-Driven Priority-Aware Computation Offloading Using Deep Reinforcement Learning," in *IEEE on Wireless Communications*, doi: 10.1109/TWC.2025.3564356.
- [39] M. Goudarzi, M. Palaniswami and R. Buyya, "A Distributed Deep Reinforcement Learning Technique for Application Placement in Edge and Fog Computing Environments," in *IEEE Transactions on Mobile Computing*, vol. 22, no. 5, pp. 2491-2505, 1 May 2023, doi: 10.1109/TMC.2021.3123165.
- [40] Murthy, S. N., Bhojanna, U. (2009). *Business Research Methods* (2nd ed.). New Delhi, India.
- [41] P. Lai et al., 'Optimal Edge IoT device Allocation in Edge Computing with Variable Sized Vector Bin Packing', in *Service-Oriented Computing*, Cham, pp. 230–245, 2018.
- [42] M. Mitchell, "An Introduction to Genetic Algorithm (1st Edition)", MIT Press, 1998. ISBN 978-0262631853.



Task Scheduling and Offloading in IoT– Edge–Cloud Systems

Internet of Things (IoT) devices are widely deployed in environments such as factories, hospitals, homes, and vehicles. Due to limited processing capabilities, these devices often offload tasks to the cloud, which can cause high communication overhead, network congestion, and increased end-to-end latency, particularly for latency-sensitive applications such as industrial control, smart city analytics, and healthcare monitoring. Edge computing alleviates these issues by bringing computation closer to end users, but its limited resources require efficient task scheduling across device, edge, and cloud tiers.

This thesis proposes an adaptive and delay-efficient scheduling framework for the device-edge-cloud continuum. The problem is first formulated as a Mixed-Integer Linear Program to minimize service delay, with CPLEX used to obtain benchmark solutions. To improve scalability, heuristic methods and two meta-heuristic approaches based on genetic algorithm (GA) and simulated annealing (SA), are developed, where SA method achieves near-optimal performance with significantly lower computational cost. Building on this, two online schedulers are introduced: one extending SA for online hierarchical edge computing, and another based on multi-agent reinforcement learning using Deep Q-Networks and Double DQN. Simulation results show notable latency reduction and improved deadline satisfaction compared to state-of-the-art baselines, demonstrating the effectiveness of the proposed approach.

ISBN 978-91-7867-662-0 (print)

ISBN 978-91-7867-663-7 (pdf)

ISSN 1403-8099

DOCTORAL THESIS | Karlstad University Studies | 2026:9

Internet of Things (IoT) devices are widely deployed in environments such as factories, hospitals, homes, and vehicles. Due to limited processing capabilities, these devices often offload tasks to the cloud, which can cause high communication overhead, network congestion, and increased end-to-end latency, particularly for latency-sensitive applications such as industrial control, smart city analytics, and healthcare monitoring. Edge computing alleviates these issues by bringing computation closer to end users, but its limited resources require efficient task scheduling across device, edge, and cloud tiers.

This thesis proposes an adaptive and delay-efficient scheduling framework for the device-edge-cloud continuum. The problem is first formulated as a Mixed-Integer Linear Program to minimize service delay, with CPLEX used to obtain benchmark solutions. To improve scalability, heuristic methods and two meta-heuristic approaches based on genetic algorithm (GA) and simulated annealing (SA), are developed, where SA method achieves near-optimal performance with significantly lower computational cost. Building on this, two online schedulers are introduced: one extending SA for online hierarchical edge computing, and another based on multi-agent reinforcement learning using Deep Q-Networks and Double DQN. Simulation results show notable latency reduction and improved deadline satisfaction compared to state-of-the-art baselines, demonstrating the effectiveness of the proposed approach.



DOCTORAL THESIS | Karlstad University Studies | 2026:9

ISBN 978-91-7867-662-0 (print) | ISBN 978-91-7867-663-7 (pdf)

