POZNAN UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTING AND TELECOMMUNICATIONS

INSTITUTE OF RADIOCOMMUNICATIONS

Optimization of Energy Efficiency in Fog Computing with Latency Constraints

DOCTORAL DISSERTATION

AUTHOR: BARTOSZ KOPRAS
SUPERVISOR: HANNA BOGUCKA
AUXILIARY SUPERVISOR: FILIP IDZIKOWSKI

September 30, 2024,
Poznań, Poland

Abstract

Motivated by increased mobile communication traffic, a vast amount of data processing and associated energy consumption, as well as strict latency requirements, the author of this thesis presents his research on energy consumption minimization in fog computing networks that distribute communication and computation services along the cloud-to-end-devices continuum. Task execution latency constraints are also considered.

The thesis of this dissertation is the following: *There exist optimal solutions to computational task offloading problems in fog networks, minimizing energy consumption while maintaining required latency levels.* The main goal of the thesis is to propose such solutions.

After the state-of-the-art research on energy-aware fog computing networks is analyzed, the author's original contributions to solving the problem of communication and computing task allocation in fog networks are presented. First, the author focuses on modeling the delay and energy consumption within the fog and cloud tiers of the network. Models are parameterized using values representing real-world equipment for communication and computing resources and diverse user requests. Results presenting the impact of different core network parameters on energy consumption and delay in fog computing networks are shown for various parameter setups.

Then, the author formulates an optimization problem to find an assignment of offloaded tasks to nodes in the fog and cloud tiers that minimize energy consumption while keeping their delay requirements. The objective function includes energy costs related to transmission and computing, while the optimization space includes choice of Fog Nodes (FNs) and Cloud Nodes (CNs) executing the tasks as well as their Central Processing Unit (CPU) frequencies. Two solutions, called Energy-EFFicient Resource Allocation (EEFFRA) and Low Complexity (LC)-EEFFRA, to this non-convex optimization problem are proposed. The simulation results for various input parameters are provided and compared against the benchmark algorithms.

Next, the fog-network model is expanded by including the wireless transmission between Mobile Devices (MDs) and FNs. It has an impact on the objective function and constraints, and adds a new set of decision variables. Despite this, an analytical solution to the optimization problem is found. The results are examined for various input parameters and compared against those achieved by the baseline solutions.

Finally, the author explores the offloading of tasks modeled as sequential graphs. These tasks consist of smaller subtasks, each of which can be processed at a different node. The optimization problem is still about minimizing energy consumption and maintaining the required delay while the proposed solution involves clustering similar nodes to significantly reduce the size of the search space.

The major conclusion of this dissertation is that the author's original solutions can significantly reduce energy consumption in the fog network with latency constraints compared with standard cloud-delegation practices. Key parameters, such as arithmetic intensity, are identified and their impact on the efficiency of offloading solutions is shown through the results of multiple simulations.

Streszczenie

Motywowany zwiększonym ruchem w komunikacji mobilnej, rosnącą skalą zbierania i analizy danych oraz związanym z nią zużyciem energii, a także wymaganym gwarantowanym opóźnieniem zadań komunikacyjnych i obliczeniowych, autor niniejszej pracy przedstawia swoje badania nad minimalizacją zużycia energii w architekturach sieciowych typu *mgła obliczeniowa*, które dystrybuują komunikację i usługi obliczeniowe w kontinuum pomiędzy chmurą obliczeniową a urządzeniami końcowymi. Uwzględnia także wymagania związane z zadanymi ograniczeniami opóźnień w wykonywaniu zadań.

Teza tej rozprawy jest następująca: *Istnieją optymalne rozwiązania problemów związanych z odciążaniem urządzeń końcowych i przekazywaniem zadań obliczeniowych do sieci mgłowych, minimalizujące zużycie energii przy jednoczesnym zachowaniu wymaganych opóźnień.* Głównym celem pracy jest zaproponowanie takich rozwiązań.

Po omówieniu aktualnego stanu wiedzy na temat energetycznie oszczędnych sieci typu mgła autor przedstawia swój oryginalny wkład w rozwiązanie problemów przydziału zadań obliczeniowych w tej sieci. W pierwszej kolejności autor koncentruje się na modelowaniu opóźnień i zużycia energii w warstwach mgły i chmury. Modele są parametryzowane przy użyciu wartości reprezentujących rzeczywisty sprzęt sieciowy i komputerowy oraz przy założeniu wystepowania różnorodnych rodzajów zadań obliczeniowych. Przedstawiono wyniki wpływu różnych konfiguracji tych parametrów na zużycie energii i opóźnienia.

Autor formułuje problem optymalizacyjny w celu znalezienia takiego przypisania zadań do węzłów w warstwach mgły i chmury, które minimalizuje zużycie energii przy jednoczesnym zachowaniu wymagań dotyczących opóźnień. Funkcja celu obejmuje zużycie energii na transmisję i obliczenia, natomiast przestrzeń optymalizacyjna obejmuje węzły realizujące zadania oraz częstotliwości ich jednostek centralnych (CPU). Zaproponowano dwa rozwiązania tego niewypukłego problemu optymalizacji, zwane Energy-EFFicient Resource Allocation (EEFFRA) i Low Complexity (LC)-EEFFRA. Przedstawiono wyniki symulacji dla różnych parametrów wejściowych i porównano je z wynikami wzorcowymi.

Następnie model sieci mgłowej jest rozwijany poprzez uwzględnienie w rozważaniach optymalizacyjnych transmisji bezprzewodowej pomiędzy urządzeniami końcowymi a węzłami mgły. Wpływa to na funkcję celu (przez dodanie energii zużytej na transmisje) i ograniczenia (przez dodanie opóźniej). Ponadto przestrzeń rozwiązań jest powiększona o dodatkowy zbiór zmiennych. Zaproponowano analityczne rozwiązanie problemu optymalizacyjnego, a uzyskane wyniki porównywano z rozwiązaniami bazowymi.

Na koniec autor bada modelowanie przenoszenia zadań obliczeniowych do mgły za pomocą skierowanych grafów acyklicznych. Zadania te składają się z podzadań, z których każde może być przetwarzane w innym węźle mgły. Problem optymalizacyjny to nadal minimalizacja zużycia energii przy utrzymaniu wymaganego opóźnienia. Proponowane rozwiązanie polega na grupowaniu podobnych węzłów w celu znacznego ograniczenia przestrzeni poszukiwania optimum.

Głównym wnioskiem z przedstawionych badań jest to, że oryginalne rozwiązania autora mogą

znacznie zmniejszyć zużycie energii w sieci mgłowej przy ograniczeniach opóźnieniowych zadań w porównaniu ze standardowymi praktykami delegowania zadań do chmury. Parametry, w tym intensywność arytmetyczna (stosunek liczby operacji do rozmiaru zadania), mające kluczowy wpływ na wydajność przenoszenia obliczeń są zidentfikowane. Wnioski wynikają z wielu symulacji komputerowych przeprowadzonych dla różnorodnych scenariuszy.

Table of Contents

List of Figures

List of Tables

List of Acronyms

A2C	Advantage Actor-Critic
ABC	Artificial Bee Colony
ADMM	Alternating Direction Method of Multipliers
AI	Artificial Intelligence
ALR	Adaptive Line Rate
AMO	Ant Mating Optimization
AP	Access Point
AR	Augmented Reality
BB	Branch and Bound
BS	Base Station
CDF	Cumulative Distribution Function
CDN	Content Delivery Network
CMOS	Complementary Metal-Oxide Semiconductor
CN	Cloud Node
CNE	Clustered Network Embedding
CS	Contract-based under Symmetric information
CPU	Central Processing Unit
D2D	Device-to-Device
DAG	Directed Acyclic Graph
DC	Data Center
DDS	Deep Dynamic Scheduling
DDPG	Deep Deterministic Policy Gradient
DL	DownLink
DVFS	Dynamic Voltage and Frequency Scaling
EEFFRA	Energy-EFFicient Resource Allocation
EH	Energy Harvesting
EON	Elastic Optical Network
EPON	Ethernet Passive Optical Network
FFBD	Feasibility Finding Benders Decomposition
FI	Fog Instance
FLOP	Floating Point Operation
FN	Fog Node
GA	Genetic Algorithm
GBD	Generalized Benders Decomposition
GenAI	Generative Artificial Intelligence
GFLOPS	Giga Floating Point Operations per Second
GHG	Greenhouse Gas
HGSA	Hybrid Genetic Simulated Annealing

ICT Information and Communication Technology
IoT Internet of Things
IP Internet Protocol
KKT Karush–Kuhn–Tucker
LAN Local Area Network
LA-VNE Latency-Aware Virtual Network Embedding
LC Low Complexity
LNA Low Noise Amplifier
LO Local Oscillator
LP Linear Programming
LTE Long Term Evolution
MCC Mobile Cloud Computing
MD Mobile Device
MDP Markov Decision Process
MEC Mobile/Multi-Access Edge Computing
MILP Mixed-Integer Linear Programming
MIMO Multiple-Input and Multiple-Output
MINLP Mixed-Integer Non-Linear Programming
ML Machine Learning
MPA Marine Predators Algorithm
N/A Not Applicable
NEP Nash Equilibrium Problem
NIC Network Interface Card
NSGAII Non-dominated Sorting Genetic Algorithm II
OFDMA Orthogonal Frequency-Division Multiple Access
PA Power Amplifier
PC Personal Computer
PDS Post-Decision State
PE Preliminary Embedding
PGN Portable Game Notation
PON Passive Optical Network
PSO Particle Swarm Optimization
QoS Quality of Service
RF Radio Frequency
RL Reinforcement Learning
RRH Remote Radio Head
RTT Round-Trip Time
SCA Successive Convex Approximation
SCS Splitting Conic Solver
SDR SemiDefinite Relaxation
SID Solution IDentifier
SL Substrate Link
SN Substrate Node
SubP Sub-Problem

TDMA	Time Division Multiple Access
UCB	Upper Confidence Bound
UL	UpLink
VC	Virtual Cluster
VL	Virtual Link
VM	Virtual Machine
VN	Virtual Node
VNE	Virtual Network Embedding
VNR	Virtual Network Request
WAN	Wide Area Network
WDM	Wavelength Division Multiplexing

1 Introduction

1.1 Motivation

The number of devices connected to the Internet is growing exponentially. Billions of smartphones, cameras, and sensors generate an unprecedented volume of data waiting to be shared with other devices in the omnipresent Internet of Things (IoT). However, the small size of the things often correlates with low computational power and limited battery capacity, which in turn limits the scope of processing and applications that can be run by them. However, relying solely on distant abundant resources provided by cloud computing can be inefficient for many latency-sensitive applications. With an ever-growing number of devices connected to the IoT and increasing quality of multimedia services, satisfying Quality of Service (QoS) with sparsely located Data Centers (DCs) becomes unsustainable.

Moreover, as the global Information and Communication Technology (ICT) sector is estimated to cause 2.1-3.9% of Greenhouse Gas (GHG) emissions worldwide [1], movements towards "greening" ICT are gaining popularity in both industry and academia. One of the challenges is an increase in energy consumption associated with the growth of data traffic and its processing worldwide. The energy consumption in the cloud is projected to continue growing [1, 2]. Recent years have brought an even greater surge in ICT energy consumption caused by unprecedented demand from Artificial Intelligence (AI) and Generative Artificial Intelligence (GenAI) algorithms [3] with image generation being particularly energy-intensive [4].

A more decentralized *fog* computing paradigm has been proposed to augment the cloud and provide computational, networking, and storage capabilities at the edge of the network – close to the end users [5]. The fog is essentially a hierarchical network organization where communication and computing tasks can be performed flexibly using diverse resources available in a network. Fog is an architecture that distributes communication and computation services along the cloud-to-things continuum. It includes information processing, storage, control, and networking to serve many growing applications. The architecture of a fog network (shown in Figure 2.1 and discussed in Chapter 2 in detail) consists of three tiers: things, fog, and cloud. Smartphones, sensors, and other linked IoT devices are present in the things tier. Powerful data servers are deployed in the cloud layer. Connected computing devices (personal computers, servers, computing clusters, etc.) that can process, communicate, and store data are called Fog Nodes (FNs). Multiple hierarchical levels of FNs may exist in the fog tier. Cooperation, including both vertical and horizontal communication, is possible between geographically distributed FNs. Utilizing these nodes allows for more flexible and effective offloading and content distribution schemes. Consequently, this has the potential to reduce energy consumption as has been shown in numerous published works.

Motivated by the increased mobile communication traffic, required high data rates and associated energy consumption, as well as the guaranteed latency levels, the author of this thesis presents his research leading to new approaches to optimizing energy consumption in fog networks with

latency constraints. Multiple solutions with varying levels of complexity are proposed for different fog-network scenarios.

1.2 Dissertation thesis and main goals

This dissertation presents the following thesis:

There exist optimal solutions to computational task offloading problems in fog networks minimizing energy consumption while maintaining required levels of latency.

The main goal of the thesis is to propose such solutions and in particular:

- To analyze the impact of offloading computational tasks to fog and cloud on energy consumption and latency. This goal is addressed in Chapter 3).
- To formulate and solve the optimization problem of allocating offloaded tasks within the fog and cloud tiers of the network, and dynamically scaling computing frequency. The objective function includes the energy spent on wired transmission and task processing and the corresponding delay is one of the constraints. This goal is addressed in Chapter 4.
- To formulate and solve the optimization problem of allocating offloaded tasks within the things, fog, and cloud tiers of the network, and dynamically scaling computing frequency. The objective function includes the energy spent on wireless transmission, wired transmission, and task processing and the corresponding delay is one of the constraints. This goal is addressed in Chapter 5.
- To formulate and solve the optimization problem of allocating tasks consisting of subtasks within the fog and cloud tiers of the network. Each subtask can be sequentially computed at a different node in a network. The objective function includes the energy spent on wired transmission and task processing and the corresponding delay is one of the constraints. This goal is addressed in Chapter 6.
- To develop effective, low-complex versions of the aforementioned algorithms while maintaining similar results. This goal is addressed in Chapters 4 – 6.

Moreover, this thesis includes a comprehensive survey of energy consumption in contemporary research works. It includes, in particular:

- Review of models used for describing fog networks including nodes, links, and traffic. The focus is on energy consumption and delay caused by both communication and computing. Various aspects of modeling are shown in Section 2.1.
- Review of parameterization of these models. Chosen scenarios and particular parameter values have a critical impact on examined network operations and achieved results. This often-neglected area is examined in Section 2.2.

- Review of optimization problems and solutions. Examined problems are grouped, compared, and contrasted according to chosen objective functions and methods used in solving them. Much impact is put on the achieved results and chosen baseline solutions. This analysis is given in Section 2.3.

1.3 Dissertation Outline

The state-of-the-art research on energy-aware fog computing networks is presented in Chapter 2. The focus is on various aspects of energy consumption especially related to communication and computing. It shows in Section 2.1 how fog networks are modeled and what types of applications they are used for. Then, it examines how these models are parameterized, and what differences and similarities there are between scenarios chosen in different works in Section 2.2. Finally, various types of optimization problems and methods used to solve these problems are examined in Section 2.3.

Chapters 3 through 6 describe the author's original contributions in the field of energy-aware (energy-consumption minimizing) fog networks. Each of these chapters examines a problem of computation offloading (or task allocation in the fog nodes) in fog networks and each of them focuses on different aspects of modeling and optimizing the problem of energy consumption.

In Chapter 3, the author of this thesis focuses on modeling the delay and energy consumption within the fog and cloud tiers of the network. Models are parameterized using values representing real-world equipment for communication and computing resources and diverse user requests. Results presenting the impact of different network parameters on energy consumption and delay are shown for various scenarios.

Chapter 4 is devoted to the optimization of energy consumption in the fog network. It builds on the model from Chapter 3. First, energy consumption and delays related to transmission, queuing, and computing of tasks in the fog and cloud tiers are modeled. Then, the author of this thesis proposes an optimization problem to find an assignment of offloaded tasks to nodes in the fog and cloud tiers that minimize energy consumption while keeping their delay requirements. The decision variables correspond to the allocation of tasks to nodes and adjusting their Central Processing Unit (CPU) frequencies. Two solutions to this non-convex optimization problem are proposed, both applying Successive Convex Approximation (SCA). Finally, the simulation results for various input parameters are provided and compared against those achieved by the baseline solutions from the literature.

In Chapter 5, the model from Chapter 4 is expanded by including the wireless transmission between Mobile Devices (MDs) and FNs in the optimization considerations. When compared to Chapter 4, it changes the objective function (by adding energy cost for things-fog transmission) and constraints (by adding the wireless transmission-associated delay), and includes a new set of decision variables (extending the optimization solution search space). Moreover, an analytical solution to the optimization problem is proposed. The results are examined for various input parameters and compared against those achieved by the baseline solutions.

Chapter 6 explores the offloading of tasks modeled as sequential graphs. These tasks consist of smaller subtasks and each of these subtasks can be processed at a different node. Similarly to problems in Chapters 4-5, the optimization problem involves minimizing energy consumption and maintaining the required delay. Unlike them, the search space increases with the number

of subtasks and cannot be transformed into a version of the assignment problem. The proposed solution involves clustering of similar nodes to significantly reduce this search space.

Finally, this dissertation is concluded in Chapter 7, in which the main findings regarding energy-efficient communications and computing task allocation in fog networks with latency constraints are summarized.

1.4 Author's published contributions

The above-described major contributions of this dissertation have been published in a number of papers listed below in reverse chronological order.

Papers published in international journals and magazines

- [6] **B. Kopras**, F. Idzikowski, and H. Bogucka, "A Survey on Reduction of Energy Consumption in Fog Networks – Communications and Computations," *Sensors*, vol. 24, no. 18, 2024.
- [7] H. Bogucka, **B. Kopras**, F. Idzikowski, B. Bossy, and P. Kryszkiewicz, "Green time-critical fog communication and computing," *IEEE Communications Magazine*, , vol. 61, no. 12, pp. 40–45, 2023.
- [8] H. Bogucka and **B. Kopras**, "Uberization of telecom networks for cost-efficient communication and computing," *IEEE Communications Magazine*, vol. 61, no. 7, pp. 74–80, July 2023.
- [9] **B. Kopras**, F. Idzikowski, B. Bossy, P. Kryszkiewicz, and H. Bogucka, "Communication and computing task allocation for energy-efficient fog networks," *Sensors*, vol. 23, no. 2, 2023.
- [10] **B. Kopras**, B. Bossy, F. Idzikowski, P. Kryszkiewicz, and H. Bogucka, "Task allocation for energy optimization in fog computing networks with latency constraints," *IEEE Transactions on Communications*, vol. 70, no. 12, pp. 8229–8243, 2022.

Papers published in the proceedings of international conferences

- [11] **B. Kopras**, F. Idzikowski, W.-C. Chen, T.-J. Wang, C.-T. Chou, and H. Bogucka, "Latency-aware virtual network embedding using clusters for green fog computing," in *2020 IEEE Globecom Workshops*, 2020.
- [12] P. Kryszkiewicz, F. Idzikowski, B. Bossy, **B. Kopras**, and H. Bogucka, "Energy savings by task offloading to a fog considering radio front-end characteristics," in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2019.
- [13] **B. Kopras**, F. Idzikowski, and P. Kryszkiewicz, "Power consumption and delay in wired parts of fog computing networks," in *2019 IEEE Sustainability through ICT Summit (StICT)*, Montreal, Canada, June 2019.

Papers published in national journals and magazines

- [14] H. Bogucka, F. Idzikowski, P. Kryszkiewicz, B. Bossy, and **B. Kopras**, "Mgła – nowa architektura sieci dla zrównowazonego rozwoju Internetu Rzeczy," *Przegląd Telekomunikacyjny – Wiadomości Telekomunikacyjne*, no. 7, pp. 505–511, 2019.
- [15] **B. Kopras** and F. Idzikowski, "Porównanie efektywnośći energetycznej mgły i chmury obliczeniowej - przegląd," *Przegląd Telekomunikacyjny – Wiadomości Telekomunikacyjne*, no. 6, pp. 307–310, 2019.

2 Energy Consumption and Efficiency in Fog Computing Networks – State-of-the-art

In this chapter, the author of this thesis overviews the state-of-the-art in the area of energy consumption reduction in fog computing networks. First, the author presents the commonly understood notion and architecture of such networks. Moreover, the modeling of this architecture and related energy consumption, as well as fog use cases and scenarios considered in the literature are reviewed and discussed. Then, the parameterization of these models is summarized in Section 2.2. Finally, the various optimization problems and solutions are analyzed in Section 2.3. They are grouped by the type of utilized optimization methods and by their objective functions.

Analysis of works surveyed in this chapter has been published in [6]. Throughout this chapter, works written and co-written by the author of this thesis are shown in **bold**.

2.1 Modeling the Fog

The author surveys approaches related to (i) modeling of the fog network, (ii) type of application served, and (iii) tasks and incurred traffic in the first three subsections. The last two subsections are devoted to models of energy consumed for communication and for computation. Tab. 2.1 contains the main content of this section. Works that do not provide models for energy consumption such as [16–18] are explicitly neglected.

2.1.1 Network

First, let us compile the general view (in terms of terminology) of networks examined by works cited throughout our survey. The key aspect of fog is to provide resources (usually computational, but also networking and storage) close to the end-users – close to the edge of the network. Throughout this dissertation, the author refers to elements providing these resources as Fog Nodes (FNs) – the term used in many works on fog including the reference architecture from [19]. Many surveyed works use other terms for nodes performing the same role, such as Mobile/Multi-Access Edge Computing (MEC) server, edge node, or mobile cloud. Still, throughout this work, they are described as FNs and call the tier (by some researchers called layer [20] or strata [21]) of the network which consists of these FNs as a fog tier. It is the middle tier of the network out of three tiers as shown in Fig. 2.1 and explained in Section 2.1.3. The tier of the network that contains end-users (such as sensors, MDs, smart vehicles, etc.) is called the things tier as in the IoT. Throughout this work, the term MDs is consistently used to refer to all these *things*. MDs can benefit from utilizing resources provided by the FNs. On the other side of the network, there is the cloud tier consisting

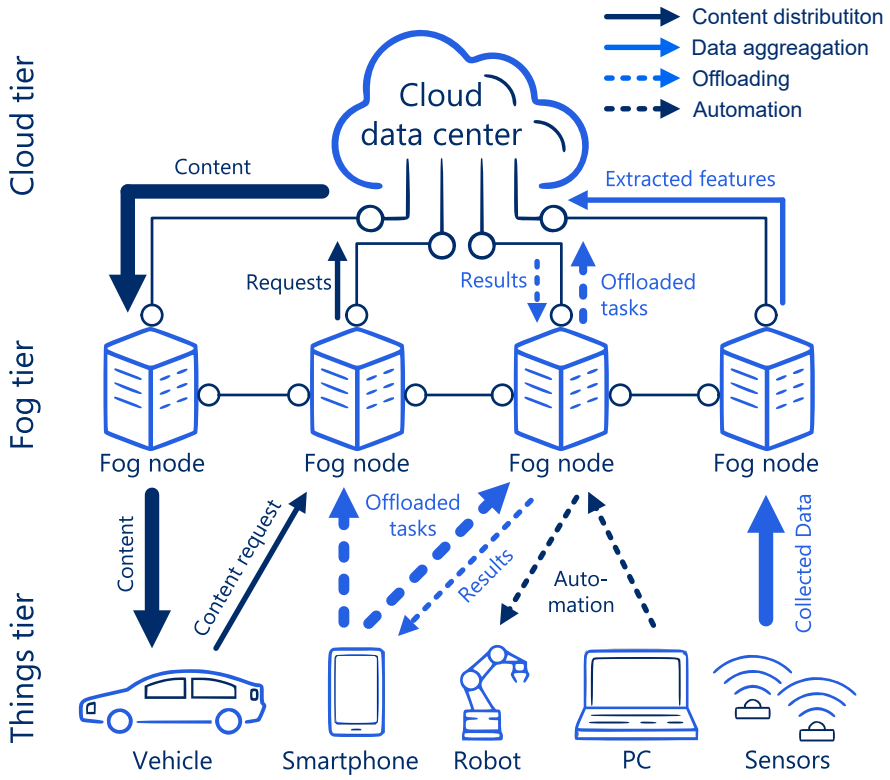


Figure 2.1: Fog network architecture and example tasks performed.

of resources from one or more cloud DCs. The cloud can also provide resources to users in the things tier.

Tab. 2.1 shows modeled networks through the lenses of costs (energy and delay) occurring in the network. The listed works (the first column of Tab. 2.1) are described with their authors and publication year. The *Application Type* column shows what kind of tasks are served by the network. The application types are explained in Section 2.1.2.

The next three columns show whether transmission of tasks occurs between nodes in given tiers and what costs related to this transmission are modeled. Depending on the work "Intra-Fog" can include transmission between different FNs, between FNs and other nodes such as Remote Radio Heads (RRHs), and between servers/VMs within a given FN. Similarly, the next three columns show whether processing (computation) of tasks occurs in these nodes. The term "Not Applicable (N/A)" means that in a modeled network, there is no possibility of (i) transmission between given nodes or (ii) computation by given nodes. "–", on the other hand, means that there is transmission/computation, but the related costs are ignored or assumed to be negligible. "E" stands for energy consumption and "D" stands for delay caused by transmission or computation. An asterisk "(*)" is added to the cells where the mathematical description of the model does not correlate with the description in the text. The *Discrete Tasks/Continuous Workload* column describes how the tasks performed by the network are modeled. The term "Discrete Tasks" means that there are individual tasks characterized with various parameters. The term "Continuous Workload" means that the tasks are modeled as part of the workload. The differences between these approaches are explained in Section 2.1.3. Letters in brackets (a, b, ...) in 2.1 show different types of modeled tasks in terms of divisibility as later shown in Fig. 2.2. The "default" examined network archi-

Table 2.1: Comparison of surveyed network models – where (E)nergy is spent and (D)elay incurred.

Work	Application type	Communication			Computation			Discrete tasks/ Cont. workload	Comments
		Things – Fog	Intra-Fog	Fog – Cloud	Things	Fog	Cloud		
1. Energy spent only by things – mobile devices									
Huang et al. [22] (2012)	Offloading	E & D	N/A	D	E & D	N/A	D	Tasks (e, f)	Single MD, no computing nodes in the fog tier.
Sardellitti et al. [23] (2015)	Offloading	E & D	N/A	N/A	N/A (*)	D	N/A	Tasks (a)	Inter-FN interference; local execution is mentioned multiple times yet no equations nor costs are provided.
Muñoz et al. [24] (2015)	Offloading	E & D	N/A	N/A	E & D	D	N/A	Tasks (c)	Single MD, separate modeling of Things-Fog UL and DL, single FN.
Mao et al. [25] (2016)	Offloading	E & D	N/A	N/A	E & D	–	N/A	Tasks (a)	Single FN, single MD, DVFS for MD.
Dinh et al. [26] (2017)	Offloading	E & D	N/A	D	E & D	D	D	Tasks (a)	Single MD; DVFS for MD; cloud is modeled not in conjunction with FNs, but in a separate scenario.
You et al. [27] (2017)	Offloading	E & D	N/A	N/A	E & D	D	N/A	Tasks (c)	Single FN.
Liu et al. [28] (2018)	Offloading	E & D	–	D	E & D	D	D	Tasks & Workload (c)	Social ties between MDs, single FN with multiple servers.
Feng et al. [29] (2018)	Offloading	E & D	N/A	N/A	E	–	N/A	Tasks (c)	Single FN.
Cui et al. [30] (2019)	Offloading	E & D	D	N/A	E & D	D	N/A	Tasks & Workload (a)	Single FN, multiple small cell BSs acting as relay nodes.
Kryszkiewicz et al. [12] (2019)	Offloading	E	N/A	N/A	E	–	N/A	Workload (a)	Single MD, single FN.
He et al. [31] (2020)	Offloading	E & D	N/A	N/A	E & D	D	N/A	Tasks (a)	
Shahidinejad and Ghobaei-Arani [32] (2020)	Offloading	E & D	D	D	E & D	D	D	Tasks (a)	Resource provisioning, edge gateways acting as relay nodes.
Nath and Wu [33] (2020)	Offloading and task caching	E & D	–	–	E & D	D	N/A	Tasks (a)	One or more FNs, FNs can cache tasks, FNs can fetch tasks from other FNs or cloud, fetching incurs costs but neither E nor D.
Bai and Qian [34] (2021)	Offloading	E & D	N/A	D	E & D	D	–	Tasks (a)	
Vu et al. [35] (2021)	Offloading	E & D	N/A	E & D	E & D	D	D	Tasks (a)	Includes direct transmission things-cloud.
Bian et al. [36, 37] (2022)	Data aggr. – distributed training	E & D (*)	N/A	N/A	E & D (*)	N/A	N/A	–	Delay and energy spent are random in time, their models are not given; a single FN coordinating distributed learning; [37] is the longer version
2. Energy spent only by fog nodes									
Ouesis et al. [38] (2015)	Offloading	–	E & D	N/A	N/A	D	N/A	Tasks (a)	
Xu et al. [39] (2017)	Offloading	D	–	E & D	N/A	E & D	D	Workload (c)	Transmission to cloud and processing in cloud are modeled jointly; single FN with multiple servers; resource provisioning.
Chen et al. [40] (2018)	Offloading	E & D	D	N/A	N/A	E & D	N/A	Tasks (a)	Things-Fog transmission costs spent by the FN.
Murtaza et al. [41] (2020)	Offloading	D	N/A	D	N/A	E & D	D	Tasks (a)	Service provisioning.
Gao et al. [42] (2020)	Offloading	N/A	E	–	N/A	E & D	–	Workload (c)	2 tiers of FNs, DVFS for FNs.

Table 2.1: Comparison of models – where (E)nergy is spent and (D)elay incurred (continued).

Work	Application type	Communication			Computation			Discrete tasks/ Cont. workload	Comments
		Things – Fog	Intra-Fog	Fog – Cloud	Things	Fog	Cloud		
2. Energy spent only by fog nodes (continued)									
Vakilian et al. [43] (2020)	Offloading	N/A	D	D	N/A	E & D	D	Workload (c)	
Vakilian et al. [44] (2021)	Offloading	N/A	D	D	N/A	E & D	D	Workload (c)	
Vakilian et al. [45] (2021)	Offloading	N/A	D	– (*)	N/A	E & D	–	Workload (c)	Transmission to a cloud is mentioned throughout the model, but the mathematical description is missing.
Abdel-Basset et al. [46] (2021)	Offloading	N/A	–	N/A	N/A	E & D	N/A	Tasks (a)	Multiple Virtual Machines (VMs), each VM can be thought of as a separate FN.
Sun and Chen [47] (2023)	Offloading & Service caching	D	N/A	– (*)	– (*)	E & D	N/A	Tasks (a)	Includes non-energy costs paid by FNs for transmission and storage. Local processing by MDs and transmission to a cloud are mentioned in the model, but their mathematical descriptions are missing.
3. Energy spent only by the cloud									
Do et al. [48] (2015)	Streaming	N/A	N/A	–	N/A	N/A	E	Workload (c)	Energy spent for computing by cloud refers to video processing, energy expressed in terms of carbon footprint.
4. Energy spent by nodes in multiple tiers of the network									
Deng et al. [49] (2016)	Offloading	–	–	D	N/A	E & D	E & D	Tasks & Workload (a, b)	Multiple clouds, DVFS for FNs and clouds.
Sarkar and Misra [50] (2016)	Offloading	E & D	–	E & D	N/A	E & D	E & D	Tasks & Workload (a)	
Zhang et al. [51] (2017)	Offloading	N/A	E & D	E & D	N/A	E & D	E & D	Workload (c)	Incomplete/ambiguous model descriptions due to magazine style.
Sarkar et al. [52] (2018)	Offloading & data aggr.	E & D	–	E & D	N/A	E & D	E & D	Tasks & Workload (a)	Additional E cost due to infinite processing in cloud. Multiple VMs per FN, intra-FN, and inter-FN resource management.
Wang et al. [53] (2019)	Offloading	E & D	N/A	N/A	E & D	E & D	N/A	Tasks (a)	Includes transmission between MDs.
Sun et al. [54] (2019)	Content caching	E & D	N/A	E	N/A	N/A	E	–	Energy spent for computing by cloud refers to signal processing.
Kopras et al. [13] (2019)	Offloading	E & D	–	E & D	N/A	E & D	E & D	Tasks & Workload (a)	
Djemai et al. [55] (2019)	Offloading	E & D	E & D	E & D	E & D	E & D	E & D	Tasks (e, f)	
Abbasi et al. [56] (2020)	Offloading	–	–	D	N/A	E & D	E & D	Tasks & Workload (a, b)	Model taken directly from [49].
Roy et al. [57] (2020)	Offloading	E & D	–	–	N/A	E & D	E & D	Tasks (b)	2 tiers of FNs: dew and edge; each FN and cloud has multiple VMs; no modeling of intra-fog and fog-cloud transmission despite being shown as part of the system; includes failure & repair times of nodes

Table 2.1: Comparison of models – where (E)nergy is spent and (D)elay incurred (continued).

Work	Application type	Communication			Computation			Discrete tasks/ Cont. workload	Comments
		Things – Fog	Intra-Fog	Fog – Cloud	Things	Fog	Cloud		
4. Energy spent by nodes in multiple tiers of the network (continued)									
Wang and Chen [58] (2020)	Offloading	E & D	N/A	N/A	E & D	E & D	N/A	Tasks (a)	Single FN, DVFS for MDs and for an FN.
Kopras et al. [11] (2020)	Offloading	N/A	E & D	E & D	N/A	E & D	E & D	Tasks (d)	
Khumalo et al. [59] (2020)	Offloading	E & D	N/A	D	N/A	E & D	D	Task (a)	
Gazori et al. [60] (2020)	Offloading	D	D	D	N/A	E & D	E & D	Tasks (a)	2 tiers of FNs, multiple VMs per FN and cloud.
Zhang et al. [61] (2020)	Offloading	E & D	N/A	N/A	E & D	E & D	N/A	Workload (c)	Single FN, multiple RRHs.
Ghanavati et al. [62] (2022)	Offloading	E & D	–	N/A (*)	N/A	E & D	N/A (*)	Tasks (b)	Transmission between the MDs and broker (gateway or AP) rather than MDs and FNs; broker-FNs transmission has no costs; transmission to a cloud is mentioned throughout the model, but the mathematical description is missing.
Kopras et al. [10] (2022)	Offloading	D	E & D	E & D	N/A	E & D	E & D	Tasks (a)	DVFS for FNs.
Kopras et al. [9] (2023)	Offloading	E & D	E & D	E & D	N/A	E & D	E & D	Tasks (a)	DVFS for FNs.

tecture includes multiple MDs, multiple FNs, and a single cloud. Deviations from this setup are included in the *Comments* column.

The existence of the things tier and the fog tier is either directly stated or implied in all surveyed works. N/A in the *Computation-Things* column does not mean that the things tier does not exist. It means that a given work does not consider computation of requests by the things. On the other hand, multiple works do not consider the cloud tier at all (e.g., [61, 62]). All but three of the works [22, 36, 48] consider one or more FNs with computing capabilities. Do et al. [48] and Bian et al. [36] examine other applications than computation offloading. Meanwhile, Huang et al. [22] is the earliest of the works examined in Section 2.1 and it predates the named concept of fog computing [5]. While it is an example of Mobile Cloud Computing (MCC), the author chooses to leave it in Tab. 2.1 as it provides a valuable reference for comparison with works on fog computing.

Let us divide the works based on which devices spend energy in their models: 1. only those in the things tier, 2. only in the fog, 3. only in the cloud, as well as 4. devices in multiple tiers. All of the works in the group focused on the things tier consider energy spent on *Things–Fog Communication*. Only one of them ([12]) does not include the corresponding delay. On the other hand, only 5 out of 15 works in the same group consider communication delay between the fog and the cloud tiers. 4 works out of those 5 additionally consider the delay caused by *Computation in the Cloud Tier*.

Works in the second group (*Energy Spent Only by Fog Nodes*) in Tab. 2.1 are not much more diverse in terms of coverage of energy and delay. Understandably, all but [38] cover energy and delay induced by *Computation in the Fog Tier*. *Computation in the Cloud Tier* is either not considered (4 works), considered but only influencing delay (4 works), or considered with its costs ignored (2 works). Less focus is given to *Communication* with only 1 work each covering energy consumption and delay incurred by *Things–Fog Communication* ([40]), *Intra-Fog Communication* ([38]), and *Fog–Cloud Communication* ([39]).

Do et al. [48] is the only work that belongs to the category 3. It considers energy spent on *Computation in the Cloud Tier*. Delay is neglected.

The final group encompasses works that model energy consumption within multiple network tiers. Consequently, it is the most diverse group. All but one work ([54]) considers energy and delay induced by *Computation in the Fog Tier*. 4 out of 18 works consider costs (delay and energy) related to *Computation in the Things Tier* and 14 out of 18 works consider those in the cloud tier (delay and energy in 12 cases, only energy in [54], and only delay in [59]). Eventually, it is significant to point out that only one work ([55]) covers energy consumption and delay incurred at all tiers and incurred by both communication and computation. Another work ([9]) misses only energy and delay costs related to *Computation in the Things Tier*.

2.1.2 Application type

Three broad types of services that a fog network can provide include offloading, content distribution, and data aggregation as shown in Fig. 2.1. These concepts are detailed in the second column of Tab. 2.1:

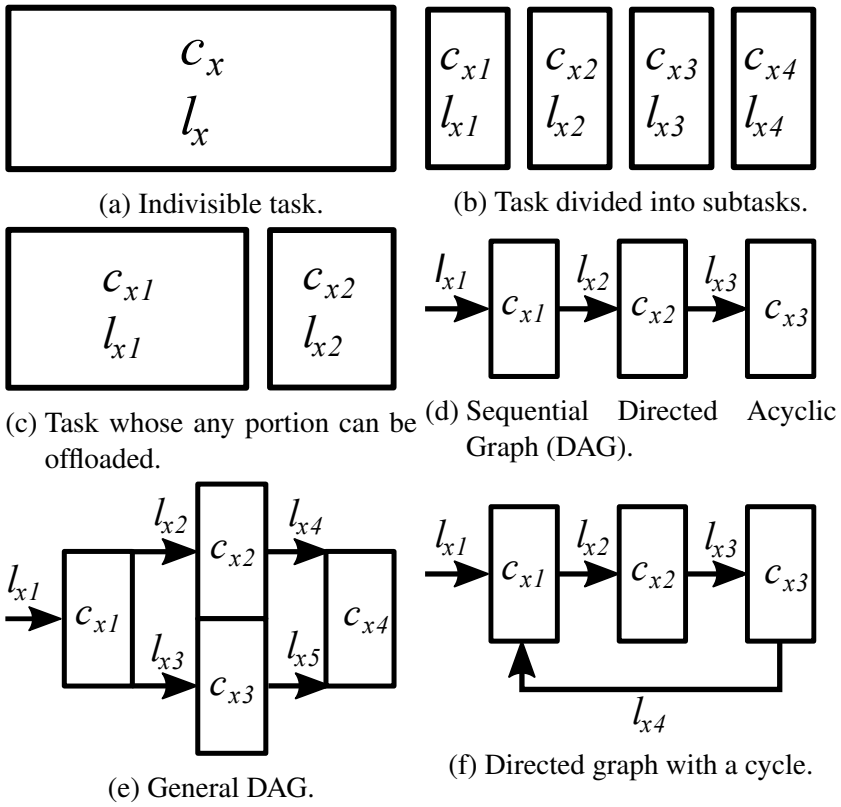
2.1.2.1 Computation offloading

Task offloading means transferring the task to another device (or another processing unit within the device) for processing. The idea to offload tasks from a device with limited capabilities to a more powerful one is not new. Experimental results from 1998 show how to increase the battery life of a laptop by putting some of its computational load to a nearby Base Station (BS) [63] or another laptop [64]. In the context of fog, offloading typically involves devices from the things tier sending computational requests to one or more nearby FNs. Upon receiving such a request, FN can process it itself or, depending on implementation, forward it to another FN or cloud. Finally, a processed task (result) can be transmitted back to its sender. As can be seen in Tab. 2.1, computation offloading is by far the most commonly studied use-case for fog computing. Terms *resource provisioning* and *service provisioning* are sometimes used and describe the process of managing the network to assure the performance of offloading.

2.1.2.2 Data processing and aggregation

In the case of offloading, the MDs offloading tasks can typically be seen as users who are interested in the results of processing of offloaded tasks. This section is devoted to a group of applications called *data processing and aggregation*. The MDs – typically sensors – are only the source of data in this group. The data can be processed, aggregated, and stored in nodes in higher tiers of the network.

Works covering the topic of data processing and aggregation tend to focus on delay and do not take energy consumption into consideration. Such works are not included in Tab. 2.1. Examples include [65] and [66] which examine the transmission of health data to the cloud after being processed in a FN. Both works show that such an approach generates considerably less traffic and is faster than sending raw data to the cloud. Out of all works from Tab. 2.1 data aggregation is used only in [36, 37, 52].



(a) Indivisible task.

(b) Task divided into subtasks.

(c) Task whose any portion can be offloaded.

(d) Sequential Directed Acyclic Graph (DAG).

(e) General DAG.

(f) Directed graph with a cycle.

Figure 2.2: Comparison of different types of computational tasks.

2.1.2.3 Content distribution

Another broad group of applications features the users (MDs) accessing content from the network. For the last two decades Content Delivery Networks (CDNs) have been used to quickly provide requested content using spatially distributed proxy servers [67]. Fog networks can act in a similar manner by caching content in FNs close to the end-users. A particularly popular type of content distribution is streaming, i.e., continuous transmission of multimedia files. Looking at the works from Tab. 2.1, task caching is performed in [20], service and content caching are performed in [47], while streaming is performed in [48].

2.1.3 Tasks and traffic

Traffic modeling is crucial for modeling a network. While the topologies of examined networks and types of applications served can differ significantly between works from Tab. 2.1, there are many similarities in how the tasks performed by the network are modeled. Below, the parameters describing the traffic are described. The works based are divided on whether they model individual computation tasks or traffic as a continuous flow of data.

2.1.3.1 Discrete computational tasks

Vast majority of works are concerned with the offloading of computations (Tab. 2.1). First, let us examine tasks modeled discretely. Such computational tasks are usually parameterized by their

size and their computational burden. Let l_x be the size of task x, and c_x be the total number of computations required to process x. l_x is given in bits or bytes. c_x can be given in a number of Floating Point Operations (FLOPs) or other instructions as well as a number of clock cycles. c_x can either be given directly or calculated by multiplying l_x by the arithmetic intensity i_x of x:

$$c_x = l_x i_x \tag{2.1}$$

l_x is the main parameter influencing costs related to the transmission of x (discussed in Section 2.1.4), while c_x influences costs related to computations (Section 2.1.5). Finally, another parameter referring to the size of the task x is the size r_x of the results transmitted back to the user. Many authors assume that $r_x << l_x$ and ignore the costs related to the transmission of results altogether.

Another aspect of a computational task is its divisibility showcased in Fig. 2.2. In many works (with one-task-one-node constraint as indicated in the in Tab. 2.4), tasks are indivisible. Each task has to be fully computed by a given node (Fig. 2.2a). On the other hand, there are works in which each task can be divided into subtasks as in [57], and each of these subtasks can be transmitted to and computed at a different node (Fig. 2.2b). Also, in some works, the tasks are infinitely divisible as in [24, 27, 29, 42], and in such cases, any portion of these tasks can be processed locally or offloaded (Fig. 2.2c). This type of divisibility is often used in works that do not model individual tasks but rather continuous workload like [43–45]. Finally, there are tasks that are divisible in a significantly more complex way. In [22], [55], and [11] the tasks are multi-step computations with computational steps connected by inputs and outputs of varying sizes. These tasks are modeled as directed graphs with each node representing a computational subtask and each vertex representing the size of data. Each subtask can be computed by a different node but only in an order specified by the graph. Directed Acyclic Graphs (DAGs) are shown in Figs. 2.2d (simple, sequential) and 2.2e (more complex). Interestingly, while [55] describe their tasks as having DAG structure, the example graphs used in [55] have cycles as in Fig. 2.2f.

Delay tolerance $d_{\max,x}$ is sometimes used to describe the maximum delay tolerated for processing task x. This threshold can be hard, as in [9, 10, 27, 31, 33, 49, 58], which means that r has to be processed within this time. Alternatively, $d_{\max,r}$ can be soft, as in [53, 55], which means that exceeding tolerated delay incurs a penalty.

o_x denotes the origin of task x. In works that include MDs in their models, o_x describes an MD which offloads (or processes locally) x. In works that do not model individual MDs, o_x describes either the FN to which the x is originally offloaded or some intermediate node such as RRH.

2.1.3.2 Continuous Computational Workload

Now, let us discuss computational requests modeled not as discrete, individual tasks, but rather as continuous workload. Computational workload refers to the amount of processing power or resources required to perform computational tasks. It is typically measured using various metrics, including: CPU usage (in time, clock cycles or percentage), memory usage, input/output operations performed by a hard disk, data volume to be processed or benchmarking metrics of the workload against some defined computing performance capacity. These metrics characterize the workload requested from a fog network. In general, they may take continuous values. Naturally, these values translate to the energy consumption of the network. There are some works in Tab. 2.1

labeled as "Tasks & Workload". In these works, there are parameters that characterize individual tasks (size, arithmetic intensity), but the delay and energy costs are calculated for the whole workload, ignoring individual tasks.

In [39] the arriving (requested) workload is considered as a whole (not arriving from discrete sources or end-devices), although a discrete-time model is considered with a defined intensity and duration. The total arriving workload is counted in *units*. Likewise, in [43–45], the workload arriving at a fog node with a certain rate is abstracted from the particular metrics describing it. It is treated as some continuous aggregated value. The decision on offloading is made by a central controller on the basis of known processing power available at FNs and the arrival workload information.

In [50] and [52], the workload is considered as data volume generated by the so-called Virtual Clusters (VCs) consisting of end-devices and transported through the edge gateways towards the fog computing tier. Fog devices are also grouped to offer higher storage and computing capabilities. A group of fog devices is called a Fog Instance (FI). The workload is defined as two values: the amount of data in bytes, generated by a VC in a time slot and demanded to be served (processed) and stored. Additionally, service requests are classified as not requiring the cloud intervention (e.g., real-time services to be processed and served by FI) and requiring the intervention of the cloud computing layer (e.g., for analysis based on historical data sets and for long-term storage). Thus, the workload is simply the amount of aggregated data volume to be stored and processed with fog/cloud preference.

2.1.3.3 Other types of tasks

It is worth noting that there are some examples of task request modeling that do not fit into the computational offloading models discussed in the previous sections. Bian et al. [36, 37] examine a distributed learning scheme with a single FN and multiple MDs. There are no tasks to be modeled. The FN chooses MDs for training at each learning round. The chosen MDs spend some predetermined amount of energy on training if they are chosen. Do et al. [48] consider video streams from a cloud to FNs. This is represented by the amount of video streaming (like l_x) and a universal conversion factor from size to computations (similar to i_x).

2.1.4 Model of energy spent on communication

Many models presented in this section and in Section 2.1.5 show power consumption rather than energy. From these results, energy consumption can be calculated by multiplying power consumption by the time necessary to transmit/compute a given/average task. Some authors may avoid this step altogether and instead examine/optimize/show average power consumption.

2.1.4.1 Wireless transmission

Estimation of the power consumption in a wireless network is a crucial aspect of designing energy-efficient wireless communication systems. The power consumption model P presented in [68] consists of the power required for radio emission P_{T} and the power consumed by the radio transceiver circuits P_{C} which can be divided into power consumed by the baseband signal processing P_{BB} (both at the transmitter $P_{\mathrm{BB-TX}}$ and at the receiver $P_{\mathrm{BB-RX}}$) and by the intermediate-frequency

and Radio Frequency (RF) signal processing P_{RF} (again both at the transmitter $P_{\mathrm{RF-TX}}$ and at the receiver $P_{\mathrm{RF-RX}}$):

$$P = P_{\mathrm{T}} + \underbrace{\underbrace{P_{\mathrm{BB-TX}} + P_{\mathrm{BB-RX}}}_{P_{\mathrm{BB}}} + \underbrace{P_{\mathrm{RF-TX}} + P_{\mathrm{RF-RX}}}_{P_{\mathrm{RF}}}}_{P_{\mathrm{C}}}. \tag{2.2}$$

The power of transmission (radio emission) P_{T} depends on the required coverage and channel conditions. The power consumed by the transmitter and receiver circuits P_{C} depends on the transmission and reception techniques, applied technologies and standards, algorithms implementations, etc. Three approaches to power consumption modeling are distinguished in the literature:

- high-level power consumption model,
- measurement-based estimation,
- transmitter and receiver components power estimation.

The high-level models can estimate the power consumption of many used technologies universally but at the cost of low estimation accuracy. [69, 70] contain the most basic total power consumption model. The transmit power (dependent on the channel condition) and the constant circuit power (independent of the channel quality) comprise these studies' overall power consumption model. A static term plus a dynamic term are included to model the power dissipation in a chip [71–73]. The latter is dependent on the supply voltage, clock frequency, and circuit capacitance, among other factors. The dynamic term, which is dependent on the clock frequency, is thought to scale with the data rate. As a result, the attained data rate is linearly represented as the circuit power.

Measurements provide the basis of the second method for estimating wireless device power usage. High power estimation accuracy is guaranteed by this method, however, implementation, vendors, and equipment/link/network setup play a major role. This method measures the overall power utilized, including the transmission power. As a result, the relationship between the transmit power and the circuit power cannot be ascertained a posteriori (after measurement), which implies that the transmission power allocation algorithm cannot be used with such models. Detailed power consumption data of a variety of commercially accessible devices is available in [74–78].

The most precise, but intricate, method is to assess the power usage of every transmitter and receiver component independently. Considering that the transceiver is integrated into a single chip, measuring any of its parts is barely possible. As a result, the architecture of each transceiver component is typically used to estimate its power consumption in the literature. The circuit-by-circuit power consumption model PC in this method is provided in [68]:

$$\begin{aligned} P_{\mathrm{C}} = & \\ & \underbrace{P_{\mathrm{ENC}} + P_{\mathrm{MOD}} + P_{\mathrm{IFFT}} + P_{\mathrm{DAC}}}_{P_{\mathrm{BB-TX}}} \\ & + \underbrace{P_{\mathrm{LPF}} + P_{\mathrm{ADC}} + P_{\mathrm{FFT}} + P_{\mathrm{DEMOD}} + P_{\mathrm{DEC}}}_{P_{\mathrm{BB-RX}}} \\ & + \underbrace{P_{\mathrm{PA}} + P_{\mathrm{MIX}} + P_{\mathrm{LO}}}_{P_{\mathrm{RF-TX}}} + \underbrace{P_{\mathrm{RFF}} + P_{\mathrm{LNA}} + P_{\mathrm{MIX}} + P_{\mathrm{LO}}}_{P_{\mathrm{RF-RX}}}, \end{aligned} \tag{2.3}$$

where P_{PA}, P_{LNA}, P_{LO}, P_{RFF}, and P_{MIX} are the power consumption of the Power Amplifier (PA), Low Noise Amplifier (LNA), Local Oscillators (LOs), RF filter, and a mixer, respectively. The baseband processing power includes the power consumption of the analog-to-digital converter P_{ADC}, the digital-to-analog converter P_{DAC}, modulation P_{MOD} and demodulation P_{DEMOD}, encoding P_{ENC} and decoding P_{DEC}, low-pass filter P_{LPF}, and when applicable, inverse fast Fourier transform P_{IFFT} and fast Fourier transform P_{FFT}.

It is evident that a transceiver's power consumption model might vary based on its construction. However, a few components are shared by the majority of digital transmission systems. These components that have the greatest share in the power consumption are presented and modeled in [79–82]. [83] contains additional system-level energy models for the RF front-end components of a wireless transceiver with exemplary power consumption numbers from the majority of frequently cited works.

Further illustrative power consumption statistics are provided in the context of Long Term Evolution (LTE) technology in [84]. In [85, 86], the power consumption models from the previously cited studies have been modified for the multi-user massive Multiple-Input and Multiple-Output (MIMO) situation. The model has been expanded to include components unique to the scenario being described.

The majority of the aforementioned articles concentrate on the power usage of the RF front-end and channel coding, ignoring the power used by other baseband signal processing techniques, which account for a sizable portion of power consumption when it comes to short links. This aspect is given additional attention in [87–91]. For example, the number of operations required to encode or decode the information bit for the channel coding techniques is available in [87, 88]. The total power used by channel coding can then be calculated using the energy expenditure per operation.

Lastly, there is a trade-off between the power consumption models' accuracy and their definitional difficulties. It is evident that a low representation of the actual system is found when the power consumption model is simple to construct. However, if the power consumption model's accuracy is high, it can be very challenging to establish, for example, if all of the transmitter and receiver components are integrated into a single chip. As a result, the power consumption determined by the measurements and enhanced by stochastic modeling or interpolation appears to be a good solution.

2.1.4.2 Wired transmission

Power consumption spent on wired transmission is more predictable than power consumption spent on wireless transmission due to the predictable (stable) communication channel. Wired transmission is realized at several parts of the fog network (Fig. 2.1), i.e., rarely in the things tier, usually in the edge/fog tier, and essentially always in the cloud tier. Different transmission technologies are used throughout these tiers, e.g., GigabitEthernet in the things tier, Passive Optical Network (PON) in the edge/fog tier, and Wavelength Division Multiplexing (WDM) or Elastic Optical Network (EON) in the cloud tier. Power consumption P_{wd} of a wired device depends on load as follows

$$P_{wd} = P_{idle} + \alpha \cdot P_{load}, \tag{2.4}$$

where P_{idle} is the power consumption of an idle device, α denotes load in the range $< 0; 1 >$, and P_{load} denotes power dependent on load [92]. P_{idle} is relatively high with respect to the maximum

power consumption of devices equal to $P_{idle} + P_{load}$. Power is consumed by devices located at network nodes and at network links:

1. Network nodes cannot be switched off or completely deactivated for connectivity reasons unless they are MDs and the end-user decides so. However, parts of them related to links (wired transmission) can be adapted or even deactivated if the device has a modular structure. This refers to Adaptive Line Rate (ALR) in Ethernet networks [93] and datarate adaptation in EONs [94]. The modules need to be related to links (Network Interface Cards (NICs) in a router) in order to be dynamically switched between active and sleep states. Attempts are made to make P_{idle} as small as possible as well as to change the profile of power dependency on load from a linear profile to a cubic one [95]. Dynamic Voltage and Frequency Scaling (DVFS) is the technology used behind these attempts.

2. Network links can be completely deactivated if there are alternative paths between the pair of nodes located at their ends. This provides higher power saving than link adaptation, however, comes at higher costs in terms of calculation of alternative paths, traffic rerouting, as well as time needed to activate and deactivate devices installed along the links (including NICs). Eventually, the reconfiguration process may lead to jitter and packets arriving out of order. Therefore the reconfiguration has to be planned in advance with rather conservative traffic predictions and fair margins on link utilization [96].

2.1.5 Model of energy spent on computation

Let us look at the energy consumption of device (node) n processing task x. The common way to model energy consumption is to take the number of computations $c_{x,n}$ performed by device n processing task x and multiply it by the unit cost e_n of an instruction or a cycle.

$$E_n(x) = c_{x,n} e_n \tag{2.5}$$

Similarly, the inverse $\frac{1}{e_n}$ (usually given in Giga Floating Point Operations per Second per Watt) can be used instead. What differentiates works is how the e_n (or $\frac{1}{e_n}$) cost is parameterized. For some works, it is simply a value assigned to nodes in a network. Such examples include: MDs in [29, 32, 35], FNs in [40, 50, 52], cloud in [10, 50, 52]. A full list of works and used parameter values can be found in Tab. 2.3 in Section 2.2.

Similarly, the energy cost $E_n(x)$ of node n processing task x can be achieved by multiplying power consumption P_n of node n by the time of execution of a given task $t_{x,n}$:

$$E_n(x) = t_{x,n} P_n \tag{2.6}$$

as in [22] for MDs and [59] for FNs. [30, 43–45] use the average time, including both execution and queuing. Meanwhile, the authors of [62] write that their parameter denotes energy consumption per cycle, however, they multiply it by the time of task execution.

For some works, there is no need to multiply power consumption by time to get energy as they only examine the levels of power consumption. Eq. (2.7) shows an example of a model which substitutes the number of computations $c_{x,n}$ from Eq. (2.5) with computational load (number of computations per second) $l_{x,n}$:

$$P_n(x) = l_{x,n} e_n. \tag{2.7}$$

It is used for MDs in [12].

For all types of nodes, works often use a variation of Eq. (2.8) to describe power consumption $P_n(u_n)$ of node n with relative processing usage/workload u_n ($0 \leq u_n \leq 1$):

$$P_n(u_n) = (P_{n,\text{peak}} - P_{n,\text{idle}}) \cdot u_n + P_{n,\text{idle}}, \tag{2.8}$$

where $P_{n,\text{peak}}$ and $P_{n,\text{idle}}$ denote maximum power and idle power of the node, respectively. It corresponds to Eq. (2.4) used for wired transmission. Eq. (2.8) or similar is used to describe: P_n of cloud in DC [48], P_n of FNs in [41], P_n of all nodes in [55, 57]. Results from Eq. (2.8) can be then multiplied by a certain time (e.g., time of execution of a given task) to get the energy cost.

Power models used in the surveyed works depend on clock frequency and/or number of servers as outlined below.

2.1.5.1 Function of clock frequency – nonlinear models

Equations such as Eq. (2.5) and Eq. (2.6) are general and rely heavily on parameters e_n and P_n respectively. They offer no insight into what influences their values. Below, is a brief summary of models which consider the clock frequency as the main factor.

The power consumption of processors (Complementary Metal-Oxide Semiconductor (CMOS) powered) has three major components: *switching*, *short-circuit current*, and *leakage* [97] with *switching* being the main one. In a simplified form the switching power follows the formula:

$$P_{sw} = CV^2 f \tag{2.9}$$

with C representing the total effective capacitance being switched per clock cycle, V representing voltage, and f representing clock frequency. Higher voltage allows the circuit to operate at a higher frequency by decreasing the delay [97]. Therefore, many works (all of the surveyed ones) look at power consumption as a function of frequency. DVFS is the technology that enables adjusting frequency and allows saving energy by operating at lower frequencies [98]. Assuming a linear relation between voltage and frequency we end up with power being proportional to a cube of frequency ($P_{sw} \propto f^3$). However, higher operating frequency means that a given task can be computed faster and the energy cost (the power multiplied by the time spent computing) is proportional to the square of frequency ($E_{sw} \propto \frac{P_{sw}}{f} \propto f^2$). One of these relations ($P_{sw} \propto f^3$ or equivalent $E_{sw} \propto f^2$) is used in most surveyed works using nonlinear energy models. They can be found in Tab. 2.2 middle column in a row "Monomial". "Monomial" means that power/energy is modeled as a monomial function of frequency and therefore directly proportional to the frequency raised to a given power. On the other hand, works in the row "Polynomial" use polynomial functions of a given degree, i.e.,:

$$P_{sw}(f) = \sum_{k=0}^{n} a_k f^k, \tag{2.10}$$

where a_k are coefficients and n is the degree of the polynomial. As seen in Tab. 2.2, degrees other than 3 for power and 2 for energy are rarely used. Even then, in [10], while the model supports nth degree polynomial for modeling power of FNs, the scenarios used in the simulations assume 3rd degree. The cloud model used in [49] (and then cited in [56]) is not a typical polynomial, but in practice, it corresponds to a polynomial of 3rd degree with $a_3, a_0 > 0$, $a_2, a_1 = 0$.

Table 2.2: Comparison of nonlinear models of power (P) and energy (E) consumption

Type \ Order	2nd degree P	3rd degree P/ 2nd degree E	Other
Monomial	–	MDs: [25, 26, 31, 33, 34, 53, 58, 61] FNs: [42, 47, 54, 58, 61]	Cloud: [54] (3rd degree E)
Polynomial	FNs: [49, 56]	FNs: [9, 13, 53], Cloud: [49, 56]	FNs: [10] (nth degree P)

In some works, e.g., [34, 38, 47], the terms like processing power or computational capability are used instead of frequency. Still, these parameters behave just like the clock frequency and are often symbolized by the letter f.

2.1.5.2 Function of number of servers

In some of the surveyed works [28, 39, 46, 49, 54, 60], a computing node (FN or cloud) is modeled as a set of multiple servers (or processors/machines/VMs) that can operate independently. In some of these works, the power consumption depends on the number of active servers/processors: of FN in [39], of cloud in [49, 54, 56]. In [39], the authors only state that the power consumption of an FN increases with the number of active servers and with the size of the processed workload, but the function formula is not given. Later in the work, it is only specified that it is jointly convex in these variables.

2.2 Fog Scenarios and Parameterization

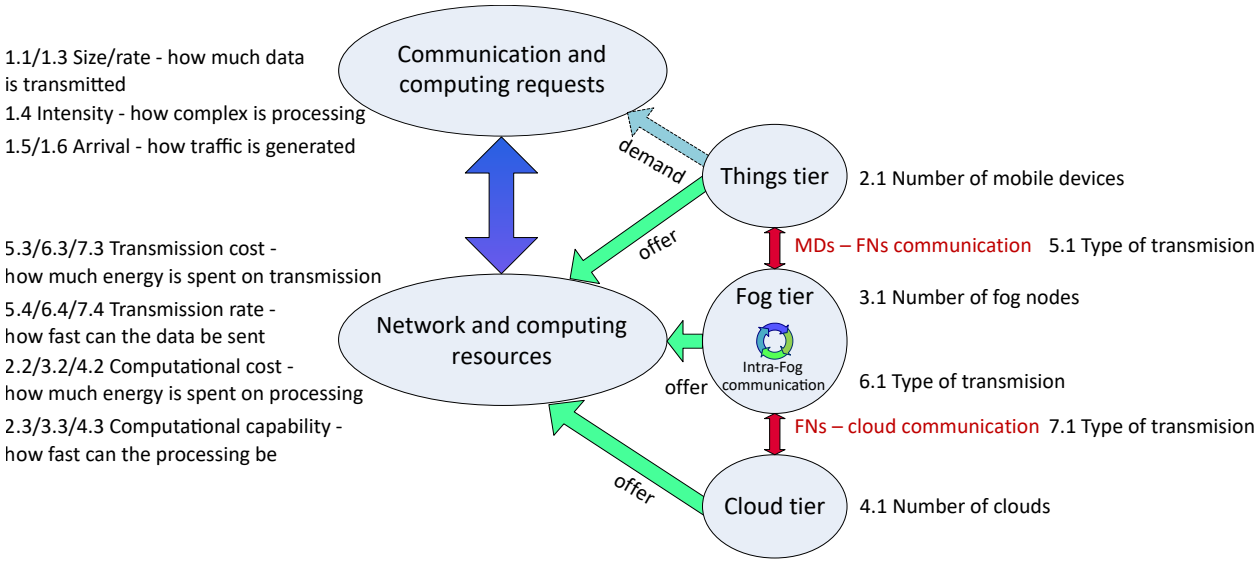


Figure 2.3: Common elements and parameters of the examined scenarios.

2.2.1 What constitutes a scenario?

Surveyed works propose models that describe the mathematical foundations of how the fog network operates. There are many similarities between chosen approaches as shown in Section 2.1. What really differentiates works from one another are the values that authors choose to parameterize these models. The network performance (shown with metrics such as delay, energy costs, and others) always depends on the parameterization of nodes, connections between nodes, and traffic. Such a full set of parameter values defines a scenario under which a network is examined. Fig. 2.3 shows certain key parameters defining a scenario and their relations. Meanwhile, Tab. 2.3 provides a comprehensive view of parameterization used by the surveyed works.

Practically, every paper considered in this chapter has a different set of testing parameters. However, the fog network scenarios are always parameterized with respect to two major fields: the communication and computing demand (generated tasks) and offer (network communications and computing resources). Our aim is to highlight the parameters that impact the performance (energy consumption or energy efficiency, which are the focus of this paper).

Communication and computing requests (tasks) stem from the things tier and are characterized by size and rate (how much data is to be transmitted), intensity (how complex it is to process the request), and the arrival process (Fig. 2.3). The three network tiers (characterized by the number of nodes in each tier) offer network and computing resources to serve requests. The networking resources depend on the type of transmission, transmission cost, and transmission rate, while computing resources depend on computational cost and computational capability. Parameters not shown in Fig. 2.3 include demand for particular content and offering of that content as well as demands and offers related to storing (rather than processing) data. The author is also aware that one cannot fully simulate real-world communication using only the transmission rate, especially if it is assumed to be constant. There are many physical phenomena (interference, noise, shadowing) and traffic-related issues (jitter, packet loss) impacting it.

2.2.2 Scenario parameterization – comparison of surveyed works

Tab. 2.3 contains a concise comparison of scenarios examined in the surveyed works. It shows key parameters characterizing the networks and traffic. The majority of them are visualized in Fig. 2.3. Tab. 2.3 is structured as follows. Its first part (1.) highlights the parameter values characterizing traffic in modeled networks. As discussed in Sections 2.1.2 and 2.1.3, these are most often the tasks/workloads that can be offloaded to other nodes for processing. The next three parts refer to nodes divided into three tiers (part 2 MDs, part 3 FNs, and part 4 cloud). The key parameters here represent the number of nodes and their computational capabilities and costs. The final three parts refer to communication between the nodes: part 5 between MDs and FNs, part 6 within the fog tier, and part 7 fog-to-cloud. They summarize key parameter values that impact this transmission with a focus on energy costs.

2.2.2.1 Classification rules for works in Tab. 2.3

Different works use different units. They are converted to the ones presented in Tab. 2.3 when applicable. This process often includes the multiplication/division of multiple parameters, e.g., division of FLOP/cycle by Giga Floating Point Operations per Second (GFLOPS)/W to get nJ/cy-

cle. Sometimes the author makes assumptions as the surveyed work does not clearly state certain values. For instance, both data size and "computational load" are given in bits in [38], while arithmetic intensity or total required number of instructions/operations or cycles are never mentioned in [38]. This "computational load" (in bits) is divided by "computational rate" (in instructions per second) to get the delay. Therefore, we can deduce that 1 bit corresponds to one instruction which, in turn, corresponds to one clock cycle. The arithmetic intensity is therefore 8 cycles/B. However, some works use units which cannot be converted in this way. This includes [39] where most parameters are defined in terms of abstract "units", e.g., "units of power" instead of Watts. For this reason, apart from [39], the works [43–45, 49, 56] should also be omitted from Tab. 2.3. The author also omits [46, 51] due to a lack of explanation of many of the parameters used as well as [59] for presenting a model without assigning parameter values to it. Finally, the works [36, 48, 54] do not fit into Tab. 2.3. Their modeled applications and examined scenarios differ significantly from the ones used in the rest of the surveyed works. Therefore, to avoid adding multiple works in the table that "fit" mostly into "N/A" and "unspecified" rows, let us omit them entirely from the table. The author also makes a conscious choice to not add other sets of rows to Tab. 2.3, each referring to works [36, 48, 54]. All other works from Tab. 2.1 are present in Tab. 2.3

Works presented in the "None" row when referring to the number of nodes (i.e., MDs, FNs, and clouds reflected in rows 2.1, 3.1, and 4.1 in Tab. 2.3) are omitted from the rest of the corresponding table sections 2.2–2.3, 3.2–3.3, and 4.2–4.3. Similarly, works present in "N/A" row when referring to the type of transmission (rows 5.1, 6.1, and 7.1) are also omitted from the rest of the corresponding table sections 5.2–5.6, 6.2–6.4, and 7.2–7.4. This is done to avoid clogging Tab. 2.3 with multiple additional "N/A" rows.

"Unspecified" is used in Tab. 2.3 when a parameter is missing in the surveyed work. In contrast, "Directly stated, no values given" is used when the work uses a particular parameter but does not state what value(s) is (are) assigned to it. Examples can be seen in [32] where the arithmetic intensity of task i is denoted by μ_i, but no value is assigned to μ_i. Similarly, the size of the task $L^{\text{in},i}$ has a uniform distribution in the range <a, b>, but no values are assigned to a and b. Meanwhile "indirectly stated" means that a parameter value depends on values assigned to other parameters. A common example includes transmission rate being a function of transmission power (and other parameters like noise, interference, and channel bandwidth).

Parameter values describing the number of nodes used in surveyed works are grouped in corresponding rows. For other rows with numerical values, the numbers are written just before the citation of the work. Multiple values are often written next to a given work. The notation {a, b, c} means that each of discrete values a, b, and c is used in the corresponding work. Meanwhile, <a, b> means that the values used belong to the closed interval from a to b, similarly (a, b) and (a, b> mean open and semi-open intervals albeit their use is significantly less common. There are many different reasons why certain works use multiple different values for a given parameter. They include: (i) different models of equipment used in a simulated scenario, e.g., two types of MDs in [55] having different computational capability {2.8, 4.5} Gcycle/s and cost {1.11, 1.14} nJ/cycle; (ii) different scenarios having different parameter setups, e.g., most plots in [13] show results of fog network scenarios with an arithmetic intensity of 80 FLOP/B and some plots show results for scenarios with "easier" (8 FLOP/B) and "harder" (800 FLOP/B) tasks converted to 20, 2, and 200 cycles/B using 4 FLOP/cycle value characterizing FNs in [13]; (iii) certain parameters being randomly generated at the beginning of the simulation, e.g., the computational capability of each FN is drawn from a uniform distribution <10, 15> Mcycles per second in [38]; (iv) certain

Table 2.3: Parameterization of examined scenarios.

Parameters	Description	Works and values	#
		1. Tasks/Workload	
1.1 Task/sub-task data size [kB]	Constant	0.125 [25, 31]; 65.5 [50]; {100, 1000} [13]; {100, 200, ..., 1000} [26]; 1000 [28]; 5000 [24]; {0, 625, ..., 6250} [53]	7
	Random: uniform <min, max>	<0.034, 66.5> [52]; <12.5, 62.5>[58]; <10, 100> [60]; <62.5, 125> [30]; <100, 500> [27]; <250, 1250> [38]; <1000, 5000> [9, 34]; <200, 6000>[29]; <1000, 10000> [10, 35]; <1250, 12500> [11]; <70000, 80000> [33]	14
	Random: exponential (mean)	25 [40]	1
	Directly stated, no values given	[23, 32]	2
	Unspecified	[12, 22, 47, 55, 62]	5
1.2 Sub-tasks	Number per task	{2, 3, 4, 5, 6} [11]; 3 [55]; 8 [22]; 5 [62]	4
1.3 Workload data rate [Mbit/s]	Constant	{1.5, 2.5} [61]	1
	Real-world based (mean)	7 [42]	1
1.4 Arithmetic intensity [cycles/B]	Constant	0.125 [50]; 8 [38]; 92.2 [61]; {2, 20, 200} [13]; {62.5, 75, 87.5} [28]; 297 [42]; 330 [24, 26]; <550, 1100> [23]; 1000 [60]; 5600 [31]; 5900 [25]	12
	Random: uniform <min, max>	(0, 10> [47]; <0.5, 50>, <0.5, 250> [10]; <62.5, 187.5> [27, 29]; <0.5, 250> [9]; <100, 1000>, <200, 1100>, ..., <1000, 1900> [35]	6
	Random: quotient of two uniform distributions <min, max>	<5.63, 7.86> [33]; <0.5, 50> [11]; <10, 300> [34]; <160, 2400> [58]; <800, 8000> [30]	5
	Random: other (mean)	1600 [40]	1
	Directly stated, no values given	[32, 62]	2
	Unspecified	[12, 22, 52, 53, 55]	5
1.5 Arrival of tasks	One at a time	[11, 13, 25, 30, 40, 47, 50, 52]	8
	In groups	[9, 10, 26, 27, 29, 32–35, 38, 55, 58, 62]	13
	Unspecified	[12, 22–24, 28, 31, 53, 60]	8
1.6 Arrival distribution	Constant	[25, 27, 33]	3
	Random: exponential (Poisson process)	[9–11, 13, 28, 30, 40, 52]	8
	Real-world based	[32, 42]	2
	Unspecified	[12, 22–24, 26, 27, 29, 31, 34, 35, 38, 47, 50, 53, 55, 58, 60, 62]	18

parameters being randomly generated during the simulation, e.g., size and arithmetic intensity of computational tasks in [35]. The sizes of tasks are drawn from a uniform distribution <1, 10> MB while the arithmetic intensities are drawn from one of ten uniform distributions starting from <100, 1000> cycles/B and ending with <1000, 1900> cycles/B with a step of 100 cycles/B. Results from simulations of scenarios with different intensity distributions are then compared with each other. Eventually, all parameter values are rounded to 3 significant digits in Tab. 2.3.

2.2.2.2 Differences and similarities in chosen scenarios

Let us take a closer look at the content of Tab. 2.3. We start with part 1, which describes the demand parts of the scenarios. The vast majority of surveyed works examine tasks with either constant size or ones with sizes drawn from a uniform distribution (row 1.1). These sizes vary greatly between the works, from less than a kB to tens of MB. All 4 of the examined works in which tasks are divided into predetermined subtasks (row 1.2) choose a single-digit number of subtasks. Only 2 works that model the demand as a continuous flow of workload rather than discrete tasks fit into Tab. 2.3 (row 1.3). Arithmetic intensity (row 1.4), just like task size, is mostly chosen to be constant or random following uniform distribution. Still, 5 works draw the sizes of tasks and the total number of computations per task from uniform distributions. This corresponds to arithmetic intensity being drawn from a quotient of two uniform distributions. Interestingly, the size and intensity of tasks is unspecified in 5 works, while the model is provided in 2 works, but failed to be specified with numerical values within the works. In [12], the arithmetic intensity is a simulation result rather than an input parameter (how high should it be for offloading to be energy efficient for

Table 2.3: Parameterization of examined scenarios (continued).

Parameters	Description	Works and values	#
		2. Mobile Devices	
2.1 Number of MDs	1	[12, 22, 24–26, 31]	6
	Few (2-19)	[9, 23, 28–30, 33–35, 38, 47, 55, 58, 60, 61]	14
	Many (20-99)	[10, 27, 30, 32, 34, 38, 58, 61]	8
	Hundreds (100-999)	[10, 34, 40, 50]	4
	Thousands (1000+)	[50, 52]	2
	Unspecified	[53, 62]	2
	None	[11, 13, 42]	3
2.2 Computational cost [nJ/cycle]	Constant	0.1 [34]; 0.2 [12]; 1.0 [33]; {1.11, 1.14} [55]; 1.37 [35]; 2.08 [24]; {4.38, 6.78} [32]; 1e8 [28]; (*) [30]	9
	Random: uniform <min, max>	(0, 0.2) [27, 29]	2
	Depending on frequency	[25, 26, 31, 53, 58, 61]	6
	Unspecified	[22, 23, 47]	3
	N/A – no computation by MDs	[9, 10, 38, 40, 50, 52, 60, 62]	8
2.3 Computational capability [Gcycles/s]	Constant	{0.0016, 0.0018, 0.002} [28]; 0.4 [24]; 0.5 [35]; {0.1, 0.2, ..., 1} [27]; {0.2, 0.4, 0.6, 0.8, 1} [30]; 1.0 [33, 34]; {1.7, 2.7} [32]; {2.8, 4.5} [55]	9
	Random: uniform <min, max>	<0.5, 1.0> [30]	1
	Depending on frequency	<0.15, 0.6> [58]; <0.2, 0.8> [26]; <0, 1.5> [25]; [31]; [53, 61]	6
	Unspecified	[12, 22, 23, 29, 47]	5
	N/A – no computation by MDs	[9, 10, 38, 40, 50, 52, 60, 62]	8
2.4 Mobility	Yes	[9, 22, 33, 53, 55]	5
	No	[10, 12, 23–32, 34, 35, 38, 40, 47, 50, 52, 58, 60–62]	23
		3. Fog Nodes	
3.1 Number of FNs	1	[12, 24–31, 33, 50, 58, 61]	13
	Few (2-9)	[13, 23, 26, 32, 34, 35, 47, 50, 55, 60]	10
	Many (10-99)	[9–11, 13, 34, 38, 50, 62]	10
	Hundreds (100-999)	[40, 42, 52]	3
	Unspecified	[53]	1
	None	[22]	1
3.2 Computational cost [nJ/cycle]	Constant	{1.65, 3.7, 5.2} [55]; 8.2 [40]; 80 [50]	3
	Random: uniform <min, max>	<0.8, 1.0> [11]	1
	Depending on frequency	[9, 10, 13, 42, 47, 53, 58, 61]	8
	Directly stated, no values given	[52, 62]	2
	Unspecified	[60]	1
	No cost	[12, 23–35, 38]	15
3.3 Computational capability [Gcycles/s]	Constant	0.004 per task, 0.04 total [28]; 0.8 [24]; 1.26 [50]; {2, 2.2} [26]; 5 [32, 33]; {1, 3, 5, 7, 10, 15, 20} [34]; {4, 8} [42]; 6 [27]; 10 [23, 35]; {6.75, 13.5, 60} [55]; {10, 20, 30, 40, 50} [30]	12
	Random: uniform <min, max>	<0.01, 0.015> [38]; <10, 50> [11]	2
	Depending on frequency	{2, 3} [13]; <1.6, 4.2> [9, 10]; <0, 15> [58]; [47, 53, 61]	7
	Directly stated, no values given	[62]	1
	Infinite	[25, 27, 29]	3
	Unspecified	[12, 31, 40, 52, 60]	5
		4. Cloud	
4.1 Number of clouds	1	[9–11, 13, 22, 26, 28, 32, 34, 35, 42, 47, 50, 55, 60]	15
	Few (2-9)	[52]	1
	None	[12, 23–25, 27, 29–31, 33, 38, 40, 53, 58, 61, 62]	15
4.2 Computational cost [nJ/cycle]	Constant	0.2 [11]; <0.33, 1.25> [9]; {0.1, 1, 2} [13]; <0.2, 2> [10]; 1.44 [55]; 80 [50]	6
	Unspecified	[52, 60]	2
	No cost	[22, 26, 32, 34, 35, 42, 47, 99]	8
4.3 Computational capability [Gcycles/s]	Constant	0.01 per task [28]; 1.5 per task [9, 10, 13]; {4, 10} [26]; 10 per task, 40 total [35]; 50 [32]; 120 [55]; 125 [50]; 200 [11];	10
	Infinite	[34, 42]; total [9, 10, 13, 28]	6
	Unspecified	[22, 47, 52, 60]	4

Table 2.3: Parameterization of examined scenarios (continued).

Parameters	Description	Works and values	#
5. MD-FN Communication			
5.1 Type of transmision	Wi-Fi	[9, 10, 22, 35, 60]	5
	Cellular – TDMA	[27]	1
	Cellular – OFDMA	[12, 27, 29, 30, 61]	5
	Cellular – general	[22–24, 26, 28, 33, 47, 53, 55, 58]	10
	Unspecified	[25, 31, 32, 34, 40, 50, 52, 62]	8
	N/A	[11, 13, 38, 42]	4
5.2 Power consumption [W]	Directly stated	{0.01, 0.03, 0.04, 0.15} [22]; {0.1, 0.2, 0.3, 0.4, 0.5} [30]; 0.9 [58]; 1.22 [62]; {1.18, 1.26} [26]; 2.51 [23]	6
	Indirectly stated	[9, 12, 24, 25, 27–29, 31, 33, 34, 40]	11
	Not included	[10, 32, 35, 47, 50, 52, 53, 55, 60, 61]	10
5.3 Transmission cost [nJ/bit]	Directly stated	2.5 [50]; 142 [32, 35]; 407 [62]	4
	Directly stated, no values given	[53, 55]	2
	Indirectly stated	[9, 12, 22–31, 33, 34, 40, 52, 58, 61]	18
	No cost	[10, 47, 60]	3
5.4 Transmission rate [Mbit/s]	Directly stated	2 [53]; {0.8, 1, 4, 6} [22]; 3.01 [62]; 30 [58]; {0.25, 1, 1000} [55]; {0, 6.5, 13, 18.5, 26, 39, 52, 58.5, 65} [9]; 72 [32]; 600 [60]; 1000 [52]	9
	Indirectly stated	[10, 12, 23–31, 33–35, 40, 47, 50, 61]	18
5.5 Interference consideration	Intra-cell/Intra-FN	[10, 12, 28, 33, 47]	5
	Inter-cell/Inter-FN	[23, 30, 61]	3
	No	[9, 22, 24–27, 29, 31, 32, 34, 35, 40, 50, 52, 53, 55, 58, 60, 62]	19
5.6 Choice of FN	MD always transmits to the same/closest FN	[10, 12, 23–25, 27–33, 40, 47, 50, 52, 53, 55, 58, 60–62]	23
	MD can choose an FN	[9, 22, 26, 34, 35]	5
6. Inter-FN Communication			
6.1 Type of transmission	Wired	[9–11, 13, 28, 32, 40, 55, 60]	9
	Wireless	[30, 38, 42]	3
	Unspecified	[33, 50, 52, 62]	4
	N/A	[12, 22–27, 29, 31, 34, 35, 47, 53, 58, 61]	15
6.2 Power consumption [W]	Directly stated	3.3 [13]; 82 [9]	2
	Indirectly stated	[38, 42]	2
	Not included	[10, 11, 28, 30, 32, 33, 40, 50, 52, 55, 60, 62]	12
6.3 Transmission cost [nJ/bit]	Directly stated	0.2 [11]; 0.3 [10, 13]; {4, 6} [9]	4
	Directly stated, no values given	[55]	1
	Indirectly stated	[38, 42]	1
	No cost	[28, 30, 32, 33, 40, 50, 52, 60, 62]	9
6.4 Transmission rate [Mbit/s]	Directly stated	<5,10> [11]; 72 [32]; 100 [40]; 1000 [9, 10, 13]; 1500 [60]; 10000 [55]	8
	Indirectly stated	[30, 38]	2
	Infinite rate/negligible delay	[28, 33, 42, 50, 52, 62]	6
7. FN-Cloud Communication			
7.1 Type of transmission	Wired	[9–11, 13, 26, 34, 35, 42, 47, 50, 52, 55, 60]	13
	Wireless	[22]; MD-Cloud [35]	2
	Unspecified	[28, 32, 33]	3
	N/A	[12, 23–25, 27, 29–31, 38, 40, 53, 58, 61, 62]	15
7.2 Power consumption [W]	Directly stated	{5.5, 6570} [13]	1
	Not included	[9–11, 22, 26, 28, 32–35, 42, 47, 50, 52, 55, 60]	16
7.3 Transmission cost [nJ/bit]	Directly stated	2.5 [50]; 8 [11]; 10 [10]; {6.38, 12.6, 18.7} [13]; 12.7 [9]; {278, 658} MD-Cloud [35]	6
	Directly stated, no values given	[55]	1
	Indirectly stated	[52]	1
	No cost	[22, 26, 28, 32–35, 42, 47, 60]	10
7.4 Transmission rate [Mbit/s]	Directly stated	{1, 2, ..., 10} [35]; 72 [32]; 1000 [10, 26]; 10000 [9, 13, 52, 55]; 40000 [60]	9
	Indirectly stated	[11, 28, 34, 50]; MD-Cloud [35]	5
	Infinite rate/negligible delay	[22, 33, 42, 47]	4

given scenarios). About half (14) of the works assume that multiple tasks arrive in the network simultaneously (row 1.5). This allows them to develop algorithms optimizing the assignment of multiple tasks to nodes. 8 works assume that each task arrives one at a time and 8 works do not specify this. Meanwhile, when it comes to parameterizing the arrival distribution of tasks (row 1.6), a majority of the works do not provide the important information. Out of those that do, the majority choose the exponential distribution, while two works [32, 42] use real distributions.

Part 2 of Tab. 2.3 examines devices that constitute the lowest tier of a fog network – the MDs. While few (6) works choose to focus on a single MD, the vast majority examine scenarios with multiple devices from several to over a thousand (row 2.1). Interestingly, 3 works do not model a single MD. They only look at the distribution of tasks within fog and cloud tiers. When it comes to computations performed by the MDs (row 2.2), constant value describing the energy cost of processing is used in 8 works and models where the value depends on the clock frequency of the device (see Section 2.1.5.1) are used in 7 works. An asterisk (*) is added when describing the computational cost in [30] as the units used there are inconvertible. 8 works do not consider the possibility of MDs to process tasks, i.e., they are only the source of the demand and not of the offer. The computation capability of MDs (row 2.3) shows the same picture as their computational cost (row 2.2). The spread in values characterizing MDs (ignoring [28]) is between 0.1 and 7 nJ/cycle and between 0.1 and 5 Gcycles/s – less than 2 degrees of magnitude. It is significantly smaller than when it comes to task size/intensity that spread over 4 and 5 degrees of magnitude respectively. Meanwhile, [28] is an exception – it parameterizes MDs as having only 1.6-2.0 Mcycles/s capability – significantly less than other works. Their values for fog and cloud computational capabilities are also by far the lowest of the surveyed works. Moreover, the contrast in chosen computing cost 0.1 J/cycle = 1e8 nJ/cycle is even greater. This 0.1 J/cycle value may be a result of a mistake as an MD running at 2 Mcycles/s with such cost leads to 200 kW power consumption. 5 out of 28 works include mobility of the MDs as part of their models and scenarios (row 2.4).

The parameters of FNs (part 3 of Tab. 2.3) also significantly vary between the works. Unlike when it comes to MDs, more works (13) have a single FN, and only [22] considers no computational nodes in the fog tier (row 3.1). For the rest of the works, the numbers vary between several and multiple hundred. As for the processing of tasks (row 3.2), about half of the works (15 out of 30) do not consider energy costs carried out by the FNs. Of those that do, most (8) assume that the costs vary depending on clock frequency, while 3 works use constant values. Random values (from a relatively narrow range) are used in [11] and no values are provided in [52, 62] despite the parameters being directly stated. For computational capability (row 3.3), many works consider multiple different values for different scenarios and within scenarios. Few (3) assume that FNs have infinite capacity which results in negligible (zero) computational delay.

Part 4 of Tab. 2.3 concerns the cloud. Half of the works do not consider a cloud at all (row 4.1). A single work considers 8 cloud DCs [52] and the other works consider one. The computational cost per bit (row 4.2) is either ignored or assumed to be constant. Unlike in FNs and MDs, no works examine the cost as a function of frequency. The computational capability (row 4.3) is typically higher than in FNs and MDs, and assumed infinite in 5 works. For some works, there is a distinction between how fast a cloud can process a single task (per task capability) and the total capability. E.g., for works [9, 10, 13] the cloud can process any number of tasks simultaneously, but each at the rate of 1.5 Gcycles/s.

Separate sets of parameters are used when describing the things–fog–cloud communication in Tab. 2.3 parts 5 through 7. Communication between MDs and FNs is considered wireless (Wi-

Fi/cellular, with varying levels of specificity) or not specified at all (row 5.1). Only a few (4) works do not consider any form of MD-FN transmission. Conversely, the majority of works (15) do not consider inter-fog transmission, and out of those that do, the majority model the connections as wired (row 6.1). As for fog-cloud transmission, about half of the works consider the connection as wired, and half of the works do not consider this transmission at all (row 7.1).

All but 3 works considering MD-FN transmission consider the energy costs related to it (row 5.3). The transmission power is either not included in the model (10 works), included as a function of other changing parameters (11), or directly stated (6), with values ranging from 10 mW to 2.51 W (row 5.2). For most (19) works, transmission cost per bit is indirectly stated – it depends on other parameter values such as channel gain and transmission rate (row 5.3). For two-thirds of works, the transmission rate also depends on other parameters. In the remaining one-third, the rate is directly stated and ranges from 0.25 Mbit/s to 1 Gbit/s (row 5.4). The vast majority of works do not include interference in their calculations (row 5.5) and assume that a given MD cannot choose FN to which it transmits data (row 5.6).

For the majority of works that include inter-FN communication, the energy costs are not considered at all (row 6.2). They are directly stated in [9, 13] and indirectly stated in [38, 42]. Transmission cost (row 6.3) is directly stated in more (4) works than indirectly stated, but it is usually assumed to have no cost (9 works). Transmission rates are typically higher than the MD-FN ones, ranging from 5 Mbit/s to 10 Gbit/s, and 6 works assume an infinite rate (row 6.4).

Assumptions on FN-cloud transmission rate (row 7.3) are made in 7 works and range from 2.5 nJ/bit to 658 nJ/bit. No transmission cost is assumed in the majority of works (10). The stated transmission rates are even higher, reaching 40 Gbit/s (row 7.4). As far as energy is concerned, it is worth looking at the values that stand out. [13] is the only work where the power consumption of networking equipment is directly included (row 7.2). The two values used in [13] differ significantly because they describe different types of equipment: 5.5 W is the active power consumption of an Ethernet gateway and 6572 W describes a core network router (row 7.2). Meanwhile, scenarios examined in [35] allow for both FN-cloud and direct MD-cloud transmission (see rows 7.3 and 7.4). This direct transmission is wireless and characterized by significantly higher energy-per-bit cost than in any fog-cloud transmission. It is comparable, albeit still higher, to the costs seen in wireless things-fog transmission in works [32, 35, 62]. In [33], like in most other works, the inter-FN and FN-cloud transmission incurs no delay nor energy costs. There are, however, other undefined costs related to these transmissions that are included in their models and scenarios.

2.2.3 Parameterization summary

Parameterization of a network model plays a crucial role in determining the network operation and calculated energy-efficiency. Data compiled in this section shows that the chosen values vary greatly between surveyed works, spanning multiple orders of magnitude: size from 0.125 kB to 80 MB, arithmetic intensity from 0.125 cycle/B to 8000 cycle/B. It does not mean that some of these values are wrong as different applications utilize offloading of different kinds of tasks, e.g., a Portable Game Notation (PGN) file representing a chess game [100] is tiny (< 1 kB) compared to a high-resolution image. However, analysis of works in this section shows that no standardized benchmarks are commonly used by the researchers despite the fact that several benchmarks have been proposed in recent years [101, 102].

Most works assign simple, discretionary values to characterize the traffic, e.g., constant or uni-

formly distributed sizes of tasks, and the Poisson process for the arrival of tasks. Only a few works [32, 42] use real-world data for this characterization. Unsurprisingly, clouds are typically parameterized with higher computational capabilities than FNs, which in turn tend to have higher capabilities than FNs. This picture becomes less clear when it comes to the computational cost, particularly due to frequent ignoring of energy consumed by FNs and clouds for performing computations.

2.3 Energy-saving in the Fog

First, let us summarize fundamental terms related to optimization problem formulation (Section 2.3.1) and to optimization methods that can be used to reduce energy consumption in fog networks (Section 2.3.2). A detailed comparison of surveyed works through the lenses of optimization is provided in Tab. 2.4 and Section 2.3.3. Section A.1.1 provides the discussion on the results achieved with optimization methods and applied technologies. This detailed analysis spans 9 pages of solid text and therefore has been moved to the Appendix A.

2.3.1 Optimization problem formulation

The columns of Tab. 2.4 correspond to the design steps of an optimization problem [103], i.e., definition of its objective function, decision variables, constraints, and scenario (parameterization). The first column states from which work a problem originates, details of the notation used are discussed in Section 2.3.3. The last column is related to the methods used to solve the optimization problem.

Scenario broadly shows what kind of application the network runs and what kind of a problem (usually assignment of tasks to nodes) there is.

The *Objective function* is shown in the third column of Tab. 2.4. Examples of objectives include minimizing energy consumption or delay, or maximizing some sort of utility or user experience. In general, optimization problems can be single-objective or multi-objective. However, in all "multi-objective" surveyed works except for [30], the authors choose to use a single-objective function that combines multiple objectives, usually a weighted sum. Throughout Sections 2.3.1 and 2.3.2 it is assumed that the objective function is to be minimized, e.g., there is a discussion on local/global minima and whether a function is convex. Maximization problems with objective function $f(x)$ can easily be turned to minimization problems as follows: $\max(f(x)) = -\min(-f(x))$.

Constraints columns lists the constraints that restrict the optimization. Formally, they are in the form of inequalities or equalities that certain parameters have to meet. While it is possible to formulate an unconstrained optimization problem, all the surveyed works propose constrained problems.

Decision variables column lists the variables over which the optimization is performed. The goal is to find the values of decision variables that optimize the objective function. They can be discrete or continuous. The set of all possible decision variables is the search space of the problem.

The optimization problem is determined by the following characteristics [103]: smoothness, linearity, modality, convexity, and stochasticity. The author does not provide an in-depth discuss the characteristics of optimization problems proposed in surveyed works and do not label their classes, but sometimes refer to the aforementioned characteristics throughout Section 2.3. Still, the

vast majority of optimization problems from Tab. 2.4 belong to either Mixed-Integer Non-Linear Programming (MINLP) or, more rarely, Mixed-Integer Linear Programming (MILP). Decision variables related to task allocation are usually discrete (including binary) and others (e.g., transmit power, operating frequency) are usually continuous.

2.3.2 Optimization method classification

Methods used to solve optimization problems are shown in column *Optim. methods* of Tab. 2.4.

2.3.2.1 Optimization method attributes

The following attributes of optimization algorithms are listed in [103]:

Order of information shows how much information is given in the model. *Zeroth-order* means that there are only values of the objective function and constraints. *First-order* algorithms include information about gradients – first-order derivatives – of these functions with respect to the decision variables. Similarly, *second-* and *higher-order* algorithms include higher-order derivatives. None of the surveyed works specifically state that they provide gradients in their optimization problems. Still, multiple algorithms shown in Tab. 2.4 are gradient-based e.g., [10, 29]. Typically, utilizing higher-order information makes an algorithm converge faster to the minimum. For example, in [23] the solution utilizing the second-order information is shown to be the fastest.

Search – the ways to search the design space can be broadly classified as either *local* or *global*. A global search spans the entire space to try to find the global minimum. In contrast, a local search starts at a certain point and from it, through a number of steps, tries to converge to a local minimum. If and only if a function is unimodal, then finding a local minimum is synonymous with finding a global one.

Algorithm – Mathematical vs. Heuristic – this division depends on whether the algorithm relies on provable mathematical principles or follows a practical method (heuristic) that is shown (but not proven) to be sufficient for a given problem. The author finds that this is the key difference between solutions to optimization problems proposed in the surveyed works. Therefore, Section 2.3.2.2 provides a summary of various families of mathematical and heuristic methods used in the optimization of fog networks.

Function evaluation can be *direct* – obtained by solving the provided models. In contrast, one can also generate *surrogate* models and optimize based on their evaluation. Surrogate models can be a result of interpolation or projection. The goal of using surrogate models is to build a model that fits the original one and is easier (faster) to optimize.

Stochasticity of the algorithm depends on whether it always (given the same initial conditions) evaluates functions at the same points and therefore converges to the same results. If that is the case, then the algorithm is *deterministic*. On the other hand, *stochastic* algorithms can evaluate different sets of points each time it is run according to some sort of random number generation. Surveyed works use many kinds of stochastic and deterministic algorithms e.g., all used meta-heuristics are stochastic.

Time dependence – finally, the optimization problems can be divided into *static* and *dynamic* problems. Static problems involve solving the complete model at each optimization iteration. In contrast, dynamic problems involve solving a sequence of different problems as time progresses based on information that becomes available during this time. One can integrate over time to solve

the problem for the entire time history and reduce a series of problems to a single one. Such an approach is commonly used as can be seen in Tab. 2.4 – many authors use limits of expected values of objective functions and constraints as the time interval goes to infinity.

2.3.2.2 Optimization method families

Tab. 2.4 shows that there is a multitude of different optimization methods used in the surveyed works. Similarly to [104], they are grouped into 6 broad families and discussed in the next subsection.

Convex optimization: As mentioned in Section 2.3.1 convex problems can typically be easily solved by well-known methods such as interior-point methods or Newton's method. Most of the convex optimization algorithms are mathematical (non-heuristic) and deterministic. Method of Lagrange multipliers and Karush–Kuhn–Tucker (KKT) conditions are used to directly include constraints in the objective functions. For networks modeled stochastically, Lyapunov optimization is used to stabilize the queuing times in the network while optimizing an objective such as minimizing energy [31, 42] or delay [36, 40]. While Lyapunov optimization is the most commonly used one with convex optimization methods, it can also be used with other methods e.g., heuristics in [22].

In some works, the proposed optimization problem is originally non-convex but then gets transformed (e.g., in [38]) or approximated into a convex one. A particular example of this approach is SCA – a technique that iteratively approximates non-convex functions by convex ones around given values as used in [10].

Heuristics: Heuristics are practical methods that are shown to provide good results (e.g., through trial and error) but are not mathematically proven to be optimal [103]. Many heuristics involve lowering the search space of possible solutions on the basis that the optimum is unlikely to be found in cut-off regions. They can also involve using some "clever" approximations or adjusting the order in which the problem is solved. An example whose variations come up in multiple works is the greedy algorithm which relies on separating the problem into multiple stages and sequentially making the optimal choice at each stage (e.g., allocating a node sequentially for each task as in [10] or allocating time slots sequentially to each task in [27]).

Metaheuristics: "A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions" [105]. All metaheuristic algorithms shown in Tab. 2.4 are based on concepts of natural science, e.g., evolution – Genetic Algorithm (GA) as in [30] or animal behavior – Marine Predators Algorithm (MPA) as in [46]. They are typically model-agnostic and can be adapted to solve many different optimization problems. They belong to the heuristic category in a broader sense – their effectiveness is not proven mathematically. They are also predominantly stochastic, randomly adjusting parameters throughout the optimization.

Machine learning: Machine learning methods form another family of optimization techniques that can be distinguished. Surveyed articles mostly use variations of Reinforcement Learning (RL) – agents learn to take actions maximizing the reward and deep learning – using neural networks with multiple layers. These works also belong to the broader heuristic category.

Game theory: In most studied works there is a common objective for the entire network. For optimization problems where each user/agent/node has its own goal, a solution based on game

theory can be proposed as in [61, 106]. These methods are not heuristic, they follow well-proven mathematical principles.

Mixed approach: Many optimization problems shown in Tab. 2.4 are split into subproblems, each analyzed in a separate row. The solutions to these subproblems often utilize multiple algorithms from different families. These are put together in a separate part of Tab. 2.4.

Non-convex optimization: While it is generally easier to solve convex problems, there are also algorithms that solve various kinds of non-convex problems that do not fit into any aforementioned families. No optimization problem shown in Tab. 2.4 exclusively fits to this family. Therefore the family is not included in Tab. 2.4. The authors of [49] and [61] use such methods for some of their subproblems.

2.3.3 Comparison of works on fog optimization

In this thesis, the focus is on energy/power. Therefore the author leaves out works on fog optimization that do not have energy/power as part of their objective function or constraints like [60]. Works such as [12, 13, 32, 41, 50, 52] do not state any optimization problems as defined in Section 2.3.1. These works show the results related to the operation of fog networks and perform parameter sweeps but do not perform the optimization. These works show the results related to the operation of fog networks and perform parameter sweeps but do not perform the optimization. They are therefore excluded. [51] is also excluded as it is written in a magazine style and it does not include essential information (mathematical formulas) such as constraints and decision variables for energy optimization. Similarly, Khumalo et al. [59] present a simplified version of an optimization problem, but do not provide their solution results leaving it for future work. Optimization results are provided in the next work of Khumalo et al. [107]. However, the authors modify the model and optimization problem so that they no longer include energy consumption. Also, works [108–116] are only mentioned in this section as the origins of solutions chosen by the authors of other works for comparison. They neither model nor optimize energy consumption.

The summary of fog optimization problems and proposed solutions can be found in Tab. 2.4. It is structured in the following way. First, there is the name of the work from which the optimization problem originates and the description of the scenario under which the network operates. The names are marked with Solution IDentifiers (SIDs). The next columns follow with the objective function, chosen constraints, and decision variables for the optimization problem. Eventually, the proposed solutions are named.

There are multiple entries in Tab. 2.4 for some works. In some cases, a surveyed work proposes multiple different optimization problems. Let us then use multiple SIDs for problems within the same work. This includes works: [27] – C12, C13 and C14, H2 and H3 have three different optimization problems, [26] – C11 adds another decision variable to optimization problem compared with C9 and C10, [33] – ML4 and ML5 have also different problems. In others, there are multiple solutions (often with varied levels of complexity) to the same problem. This includes works [9–11, 26, 35, 40, 46]. Finally, some optimization problems are split into various subproblems, each solved with a different method. This includes works [9, 10, 24, 25, 35–37, 42, 47, 49, 54, 61]. These subproblems are assigned a SID with a sub-ID in Tab. 2.4. E.g., two distinct solutions (MA8 and MA9) are proposed in [10]. Each of them is split into subproblems (MA8.1, MA8.2, and MA8.3, MA9.1, MA9.2, and MA9.3) listed in Tab. 2.4.

In the *Constraints* the author chooses to not to list "obvious" constraints on the decision variables

listed next to it. For example, when a work has MD transmission power as a decision variable, there is no need to put maximum MD transmission power as a constraint nor the fact that it cannot be lower than zero. Similarly, "obvious" constraints on allocation variables are also omitted, e.g., the index of a node to which a task is sent must be an integer from a given set, a portion of offloaded workload must be between 0 and 1. These choices are made to not clog Tab. 2.4 with superfluous data. The term "over time" is used to describe the expected value of a parameter over a time period as its length approaches infinity. Such an approach is often used in surveyed optimizations (C16, C18, C19, H1, ML1, ML4, ML5, G1, MA2, MA5) for their objective functions. The authors do it to make their optimization problem deterministic and static. Decision variables "allocation of tasks/workload" take many forms in surveyed works. They can be binary (whether to offload) or continuous (what fraction to offload). They also decide which nodes tasks are offloaded to in works with multiple nodes that can process them. The abbreviations max., min., and opt. refer to maximum/maximal, minimum/minimal, and optimum/optimal, respectively.

Tab. 2.4 does not include baseline solutions used by the authors only for comparison with their proposed algorithms. Instead, they are mentioned while describing works in Section A.1.1. Example baseline solutions include trivial algorithms such as random assignment or sending all tasks to the same node, or, on the other end of the spectrum, performing an exhaustive search.

Table 2.4: Approaches to optimization in the fog with respect to energy consumption.

SID. Work	Scenario	Objective function	Constraints	Decision variables	Optim. methods
1. Convex optimization					
C1. Do et al. [48] (2015)	One cloud streaming video to multiple FNs	Maximize the utility (amount of video streaming) minus cost (cloud energy – carbon footprint)	Computational capacity of cloud	Amount of streaming to each FN	Proximal algorithm, ADMM
C2. Ouesis et al. [38] (2015)	Multiple FNs receiving tasks from MDs, FNs processing tasks or sending to other FNs	Minimize energy spent on the transmission of tasks by the FNs	Max. delay, FN computing rates	FN transmission power, allocation of computational resources to tasks	Reformulation into a convex problem, Lagrange method
C3. Sardellitti et al. [23] (2015)	Single MD offloading tasks to a single FN through a BS	Minimize energy consumption spent by the MD on transmission	Max. delay, max. MD transmission power	MD transmit covariance matrix, FN computing resources used	Reformulation into a convex problem, water-filling algorithm
C4. Sardellitti et al. [23] (2015)	Multiple MDs offloading tasks to a single FN through multiple BSs, multiple MDs transmitting without offloading	Weighted sum of MD transmission energy consumption	Max. delay for offloading MDs, min. rate for non-offloading MDs, max. MD transmission power	MD transmit covariance matrices, allocation of FN computing resources to tasks	SCA
C5. Sardellitti et al. [23] (2015)	Multiple MDs offloading tasks to a single FN through multiple BSs, multiple MDs transmitting without offloading (decentralized for each BS)	Weighted sum of MD transmission energy consumption	Max. delay for offloading MDs, min. rate for non-offloading MDs, max. MD transmission power	MD transmit covariance matrices, allocation of FN computing resources to tasks	SCA, separation of delay constraint in covariance matrices, dual decomposition
C6. Sardellitti et al. [23] (2015)	As in C5	As in C5	As in C5	As in C5	Decomposition using slack variables, SCA with or without second-order information
C7.1. Muñoz et al. [24]	One MD transmitting (UL) a task to a single FN	Minimize energy consumption spent on transmission by the MD	Min. transmission	MD transmission covariance matrix	Water-filling algorithm
C7.2	One MD receiving (DL) offloaded task from the FN	Maximize DL transmission rate	Max. FN transmission power	FN transmission covariance matrix	As in C7.1
C7.3	One MD offloading a task to a single FN, includes C7.1 and C7.2	Minimize total energy consumption spent by the MD	Max. delay, max. DL transmission power	Portions of tasks offloaded and processed locally, UL and DL transmission times	Problem reformulation in terms of portion of tasks offloaded and UL transmission rate, analytically finding opt. rate, finding opt. offloaded portion through gradient descent
C8. Muñoz et al. [24]	As in C7.3	Minimize total energy consumption spent by the MD	Max. DL transmission power	Portions of tasks offloaded and processed locally, UL and DL transmission times	Mathematically proving that partial offloading can never be opt. and finding the threshold above which local processing is opt.
C9. Dinh et al. [26] (2017)	One MD offloading tasks to multiple FNs or processing them locally	Minimize the weighted sum of latency and MD energy consumption	One-task-one-node	Allocation of tasks	LP relaxation
C10. Dinh et al. [26] (2017)	As in C9	As in C9	As in C9	As in C9	SDR

Table 2.4: Approaches to optimization in the fog with respect to energy consumption (continued).

SID. Work	Scenario	Objective function	Constraints	Decision variables	Optim. methods
1. Convex optimization (continued)					
C11. Dinh et al. [26] (2017)	As in C10	As in C10	As in C10	Allocation of tasks, MD operating frequency	As in C10
C12. You et al. [27] (2017)	Multiple MDs offloading tasks (also partially) to a single FN with *infinite computing capacity* (TDMA) or processing them locally	Minimize the sum of energy consumption of all MDs weighted by unspecified fairness	Max. delay, one task per time slot	Size of offloaded portions of tasks, time slot allocation	Lagrange method, bisection search, this problem is used as a subproblem for other optimizations by You et al.
C13. You et al. [27] (2017)	Multiple MDs offloading tasks (also partially) to a single FN (TDMA) or processing them locally	As in C12	As in C12	As in C12	Lagrange method, utilizing solution C12, 2D search for Lagrange multipliers (time slots, offloaded portions)
C14. You et al. [27] (2017)	Multiple MDs offloading tasks (also partially) to a single FN (OFDMA) or processing them locally	As in C13	Max. delay	Size of offloaded portions of tasks, subchannel allocation	Relaxation-and-rounding
C15. Feng et al. [29] (2018)	Multiple MDs offloading tasks (also partially) to a single FN or processing them locally	Minimize energy consumption of an MD with the highest consumption	Min. transmit rate, assignment of an IoT device to one and only one subcarrier	Subcarrier allocation, size of offloaded portions of tasks	Lagrange method, relaxation, subgradient projection
C16. Chen et al. [40] (2018)	Multiple MDs offloading load to FNs, FNs sharing load between each other – decentralized decision-making	Minimize delay over time	Max. energy cost over time, max. energy cost per time slot, max. delay per time slot	Task allocation to nodes (from one FN to another)	Lyapunov optimization
C17. Vakilian and Fanian [43] (2020)	Multiple FNs receiving offloaded workloads and processing them or sending them to other FNs or cloud for processing	Minimize weighted and normalized sum of energy consumption of FNs and delay	Processing rate of FNs	Allocation of workload to nodes	SCS
C18. He et al. [31] (2020)	A single MD offloading tasks to multiple FNs with potential non-colluding adversaries sensing its presence	Minimize energy consumption of an MD over time	Max. delay, max. task drop rate, max. likelihood of detection by adversaries, one-task-one-node	Task allocation to nodes, task dropping, MD transmission power	Lyapunov optimization
C19. He et al. [31] (2020)	A single MD offloading tasks to multiple FNs with potential colluding adversaries sensing its presence	As in C18	As in C18	As in C18	As in C18
C20.1 Gao et al. [42] (2020)	Multiple FNs receiving offloaded workload and processing it or sending it (or portion of it) to higher-tier FNs or cloud for processing	Minimize average total power consumption of all FNs	Stability of queues, max. size of queues	Allocation of workload to nodes, FN transmit power, FN operating frequency	Lyapunov optimization, decomposition into C20.2, C20.3, and C20.4, workload prediction
C20.2	FNs processing workload	Minimize drift-plus-penalty for FN frequency decision	–	FN operating frequency	Polynomial (3rd degree) minimization

Table 2.4: Approaches to optimization in the fog with respect to energy consumption (continued).

SID. Work	Scenario	Objective function	Constraints	Decision variables	Optim. methods
1. Convex optimization (continued)					
C20.3	Lower-tier FN wirelessly transmitting workload to higher-tier FN	Minimize drift-plus-penalty for transmit power decision	–	FN transmit power	Water-filling algorithm
C20.4	Lower-tier FN processing workload or sending it to higher-tier FN, higher-tier FN processing workload or sending it to cloud	Minimize drift-plus-penalty for offloading decision	–	offloading decision	Polynomial (1st degree) minimization
C21. Vu et al. [35] (2021)	Multiple MDs offloading tasks to multiple FNs or a cloud, or processing them locally	Minimize the sum of energy consumption of all MDs	One-task-one-node, max. delay	Allocation of tasks to nodes, uplink, downlink, and computing rates of FNs, uplink, downlink, and computing rates of cloud	Relaxation-and-rounding
C22. Vu et al. [35] (2021)	As in C21	As in C21	As in C21	As in C21	Relaxation, improved BB algorithm
C23.1 Vu et al. [35] (2021)	As in C21	As in C21	As in C21	As in C21	FFBD into the following problems
C23.2	Multiple MDs offloading tasks to multiple FNs or a cloud, or processing them locally	Minimize the sum of energy consumption of all MDs	–	Allocation of tasks to nodes	Integer programming
C23.3	Checking if a solution to C23.2 is feasible	–	One-task-one-node, max. delay	Uplink, downlink, and computing rates of FNs, uplink, downlink, and computing rates of cloud	None (verification of C23.2)

Table 2.4: Approaches to optimization in the fog with respect to energy consumption (continued).

SID. Work	Scenario	Objective function	Constraints	Decision variables	Optim. methods
			2. Heuristics		
H1. Huang et al. [22] (2012)	Single MDs offloading parts of tasks (directed graphs) to the cloud for offloading or processing them locally	Minimize energy consumption of the MD over time	Max. percentage of tasks exceeding delay constraint, stability of the system	Sub-task allocation to nodes	Lyapunov optimization, 1-opt local search algorithm
H2. You et al. [27] (2017)	Multiple MDs offloading tasks (also partially) to a single FN (TDMA) or processing them locally	Minimize the sum of energy consumption of all MDs weighted by unspecified fairness	Max. delay, one task per time slot	Size of offloaded portions of tasks, time slot allocation	Lagrange method, utilizing solutions to C12, greedy time slot allocation based on priority based on channel gain and intensity, 1D search for offloading Lagrange multiplier
H3. You et al. [27] (2017)	Multiple MDs offloading tasks (also partially) to a single FN (OFDMA) or processing them locally	Minimize the sum of energy consumption of all MDs weighted by unspecified fairness	Max. delay	Size of offloaded portions of tasks, subchannel allocation	Sequentially performing the following: assigning one subchannel to each MD according to priority, determining the total subchannel number and offloaded data size for each MD, assigning specific subchannels to MDs according to priority, finding the final offloaded data size for each MD
H4. **Kopras et al.** [11] (2020)	Single MD offloading tasks (directed graphs) to multiple FNs or cloud	Minimize the sum of energy consumption of all devices	Scheduling in nodes and links (one task per time slot), max. delay	Allocation of tasks to nodes	Clustering of nodes, exhaustive search over clusters
H5. Roy et al. [57] (2020)	Multiple MDs offloading tasks consisting of sub-tasks to two tiers of FNs or a cloud	Maximize fitness (combination of availability, delay, and power consumption)	Constraints included in the objective function	Allocation of sub-tasks to nodes	Adaptive PSO, GA

Table 2.4: Approaches to optimization in the fog with respect to energy consumption (continued).

SID. Work	Scenario	Objective function	Constraints	Decision variables	Optim. methods
3. Metaheuristics					
MH1. Djemai et al. [55] (2019)	Multiple MDs offloading tasks (directed graphs) to multiple FNs or cloud nodes, or processing them locally	Minimize the sum of energy consumption of all devices plus penalties for delay violations	Min. memory, min. computing capability	Allocation of tasks to nodes	Discrete PSO
MH2. Cui et al. [30] (2019)	Multiple MDs offloading tasks to an FN through relay nodes or processing them locally	Minimize energy consumption of MDs and delay (multi-objective)	One-task-one-node, processing rate of MDs and FN	Task allocation (binary – local or to the FN)	Modified NSGAII with task allocation as chromosomes and MDs as genes
MH3. Wang and Chen [58] (2020)	Multiple MDs offloading tasks to a single FN or processing them locally	Minimize total delay	One-task-one-node, max. task delay, max. task energy consumption	Task allocation to nodes, computation capability (operating frequency) of MDs and an FN	HGSA
MH4. Abbasi et al. [56] (2020)	Offloaded workload being shared between multiple FNs and clouds	Minimize fitness (a function of energy consumption and delay – not clearly stated)	Max. computing capacity of FNs	Allocation of workload to nodes, FN-cloud traffic rates, cloud on/off state, cloud operating frequency, number of turned on machines at cloud	NSGAII with assignment of workload to nodes as genes
MH5. Vakilian et al. [44] (2021)	Multiple FNs receiving offloaded workloads and processing them or sending them to other FNs or cloud for processing	Minimize the weighted sum of normalized energy consumption of FNs and normalized delay	Processing rate of FNs	Allocation of workload to nodes	ABC
MH6. Vakilian et al. [45] (2021)	Multiple FNs receiving offloaded workloads and processing them or sending them to other FNs or cloud for processing	Minimize the weighted sum of energy consumption of FNs and delay adjusted by fairness (processing rate of a given FN divided by the total processing rate of all FNs)	Processing rate of FNs	Allocation of workload to nodes	Cuckoo algorithm
MH7. Abdel-Basset et al. [46] (2021)	Single FN allocating offloaded tasks to multiple VMs	Minimize the weighted sum of total energy consumption and the highest delay over VMs	One-task-one-node	Allocation of tasks to VMs	MPA
MH8. Abdel-Basset et al. [46] (2021)	As in MH7	As in MH7	As in MH7	As in MH7	Modified MPA
MH9. Abdel-Basset et al. [46] (2021)	As in MH7	As in MH7	As in MH7	As in MH7	Improved modified MPA
MH10. Ghanavati et al. [62] (2022)	Multiple MDs offloading sets of tasks to multiple FNs or a cloud	Minimize the weighted sum of energy spent by FNs and total delays for each set of tasks	One-task-one-node	Allocation of tasks to nodes	AMO

Table 2.4: Approaches to optimization in the fog with respect to energy consumption (continued).

SID. Work	Scenario	Objective function	Constraints	Decision variables	Optim. methods
4. Machine Learning					
ML1. Xu et al. [39] (2017)	Single FN with multiple servers processing offloaded workload or sending it to a cloud for processing	Expected cost (delay, battery depreciation, backup power usage) over time with discount factor giving less weight to cost later in the future	Not clearly defined	Number of active servers in an FN, fraction of workload offloaded to the cloud	Utilizing RL to solve PDS-based Bellman equations
ML2. Wang et al. [53] (2019)	Multiple MDs offloading tasks to other MDs or an FN, or processing them locally	Minimize the weighted sum of energy consumption and delay, and exceeded delay penalties	MD battery level	Task allocation (wait, process locally, offload to an FN, offload to other MDs)	Deep RL scheduling
ML3. Wang et al. [53] (2019)	As in ML2	As in ML2	As in ML2	As in ML2	DDS, deep Q-learning
ML4. Nath and Wu [33] (2020)	Multiple MDs offloading tasks to one FN or processing them locally, FN caching previous tasks, FN fetching (downloading to cache) tasks from the cloud	Minimize the weighted sum of energy consumption, delay, and fetching cost over time	One-task-one-node, max. cached data size, max. delay	Allocation of tasks to nodes, caching decisions, fetching from cloud decisions, MD operating frequency, MD transmission power	Deep RL, DDPG
ML5. Nath and Wu [33] (2020)	Multiple MDs offloading tasks to one of multiple FNs (each MD can only offload to one FN) or processing them locally, FNs caching previous tasks, FNs fetching tasks from the cloud or other FNs	As in ML4	As in ML4	Allocation of tasks to nodes, caching decisions, fetching from cloud decisions, fetching from FNs decisions, MD operating frequency, MD transmission power	As in ML4
ML6. Bai and Qian [34] (2021)	Multiple MDs offloading tasks to multiple FNs or a cloud	Minimize the weighted sum of energy spent by MDs and delay	One-task-one-node	Allocation of tasks to nodes, allocation of channel resources to tasks, allocation of FN computing resources to tasks	A2C algorithm
5. Game Theory					
G1. Chen et al. [40] (2018)	Multiple MDs offloading loads to FNs, FNs sharing load between each other – decentralized decision-making	Minimize delay over time	Max. energy cost over time, max. energy cost per time slot, max. delay per time slot	Task allocation to nodes (from one FN to another)	Best-response algorithm

Table 2.4: Approaches to optimization in the fog with respect to energy consumption (continued).

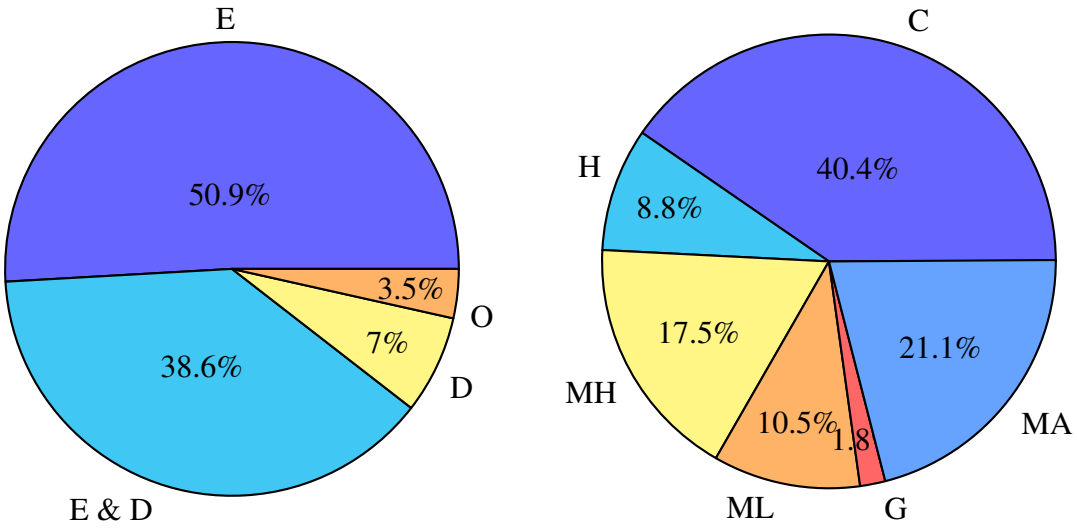
SID. Work	Scenario	Objective function	Constraints	Decision variables	Optim. methods
6. Mixed Approach					
MA1.1 Deng et al. [49] (2016)	Offloaded workload being shared between multiple FNs and clouds	Minimize the sum of energy consumption of FNs and clouds	Max. delay, max. computing capacity of FNs, max. FN-cloud traffic rate	Allocation of workload to nodes, FN-cloud traffic rates, cloud on/off state, cloud operating frequency, number of turned on machines at cloud	Approximation – decomposition into MA1.2, MA1.3, and MA1.4
MA1.2	Workload being shared between multiple FNs	Minimize the weighted sum of delay and energy consumption of FNs	Max. computing capacity of FNs	Allocation of workload to nodes	Convex optimization – interior-point method
MA1.3	Workload being shared between multiple clouds	Minimize the sum of energy consumption of clouds	Max. delay, max. computing capacity of clouds	Allocation of workload to nodes, FN-cloud traffic rates, cloud on/off state, cloud operating frequency, number of turned on machines at cloud	Non-convex optimization - GBD
MA1.4	Workload being transmitted between multiple FNs and clouds	Minimize total transmission delay	–	FN-cloud traffic rates	Hungarian algorithm
MA2.1 Mao et al. [25] (2016)	Energy harvesting MD offloading tasks to a single FN or processing it locally	Minimize weighted sum of delay and dropped task penalty over time	One-task-one-node, MD battery level	Allocation of tasks to nodes, MD transmission power, MD operating frequency, MD harvested energy	Proving opt. frequency should be constant for a task, using Lyapunov optimization to transform MA2.1 to a per-slot MA2.2 deterministic problem
MA2.2	As in MA2.1	Minimize weighted sum of virtual energy queue length, delay, and dropped task penalty over time	As in MA2.1	As in MA2.1	Finding opt. harvested energy with LP leading into MA2.3
MA2.3	As in MA2.2	As in MA2.2	As in MA2.2	Allocation of tasks to nodes, MD transmission power, MD operating frequency	Finding opt. values for frequency and trans. power, separately calculating costs for local execution, offloading, and dropping the task and choosing the lowest one
MA3. Liu et al. [28] (2018)	Energy harvesting MDs offloading workload (or portions of it) to a single FN (consisting of multiple servers) or cloud, or processing it locally	For each MDs minimize execution cost (delay + dropped task penalty) increased by weighted execution costs of other MDs in its "social group"	Stability of queues, battery level of MDs	Allocation of workload to nodes	Game theory – partial penalization, convex optimization – formulation of KKT conditions, convex optimization – semi-smooth Newton method with Armijo line search
MA4.1 Sun et al. [54] (2019)	Multiple MDs receiving content from a cloud through RRHs or D2D transmitters	Minimize the sum of energy consumption of all devices	–	On-off states of cloud processors, communication modes of MDs	Machine Learning – Deep RL
MA4.2	Multiple MDs receiving content from a cloud through RRHs or Device-to-Devices (D2Ds) transmitters	Minimize the sum of network-wide precoding vectors	Min. transmission rates, max. transmission power, computing resources of cloud	MDs data rates	Convex optimization

Table 2.4: Approaches to optimization in the fog with respect to energy consumption (continued).

SID. Work	Scenario	Objective function	Constraints	Decision variables	Optim. methods
6. Mixed Approach (continued)					
MA5.1 Zhang et al. [61] (2020)	Multiple MDs offloading load to one FN through RRHs or processing it locally	Minimize energy consumption divided by the size of processed tasks over time	Stability of queues	FN and MD operating frequency, a fraction of offload, FN transmission power, subchannel allocation	Lyapunov optimization, decomposition into MA5.2, MA5.3, MA5.4, MA5.5
MA5.2	Single MD offloading load to one FN through RRHs or processing it locally	Minimize Lyapunov drift-plus-penalty for the offloading decision	–	A fraction of offloaded load	Polynomial minimization
MA5.3	Multiple MDs offloading load to one FN through RRHs	Minimize Lyapunov drift-plus-penalty for the subchannel and transmit power allocation	–	FN transmission power, subchannel allocation	Game theory – two-side swap matching game, non-convex optimization – geometric programming
MA5.4	One FN processing offloaded load	Minimize Lyapunov drift-plus-penalty for FN computing resource allocation	–	FN operating frequency for computation and transmission	Convex optimization – decomposition
MA5.5	One MD processing tasks locally	Minimize Lyapunov drift-plus-penalty for MD computing resource allocation	–	MD operating frequency for computation	As in MA5.4
MA6. **Kopras et al.** [11] (2020)	Single edge node offloading tasks (directed graphs) to multiple FNs or cloud	Minimize the sum of energy consumption of all devices	Scheduling in nodes and links (one task per time slot), max. delay	Allocation of tasks to nodes	Heuristic – clustering of nodes, metaheuristic – discrete PSO
MA7.1 Bian et al. [36, 37] (2022)	Single FN distributing parameters to and from multiple MDs in distributed training scheme	Minimize regret (a function of delay)	Wireless channel capacity, max. expected energy consumption of MDs, fairness – min. expected MD selection rates	Selection of MDs for training	Convex optimization – Lyapunov optimization, transformation into MA7.2
MA7.2	Single FN distributing parameters to and from multiple MDs in distributed training scheme	Minimize weighted sum of virtual queues (for energy consumption and fairness) and UCB term (estimated mean regret plus exploration cost)	Max. number of MDs the FN can communicate with in each round	Selection of MDs for training	UCB-based bandit algorithm, transformation into MA7.3
MA7.3	Single FN distributing parameters to and from multiple MDs in distributed training scheme	Minimize weighted sum of virtual queues	Selection of MD with lowest UCB term, max. number of MDs the FN can communicate with during each round	Selection of MD for training	Heuristic – greedy approach
MA8.1 **Kopras et al.** [10] (2022)	Multiple FNs receiving tasks offloaded from MDs, processing them or sending them to other FNs or cloud for processing	Minimize the sum of energy consumption of all devices	One-task-one-node, max. delay	Allocation of tasks to nodes, FN operating frequency	Decomposition into MA8.2 and MA8.3
MA8.2	An FN processing offloaded task	Minimize the energy consumption of the FN	Max. delay	FN operating frequency	Convex optimization – SCA, Lagrange method
MA8.3	FNs processing offloaded tasks or sending them to other FNs or cloud	Minimize the sum of energy consumption of all devices	One-task-one-node	Allocation of tasks to nodes	Hungarian algorithm

Table 2.4: Approaches to optimization in the fog with respect to energy consumption (continued).

SID. Work	Scenario	Objective function	Constraints	Decision variables	Optim. methods
6. Mixed Approach (continued)					
MA9.1. **Kopras et al.** [10] (2022)	Multiple FNs processing offloaded tasks or sending them to other FNs or cloud for processing	Minimize the sum of energy consumption of all devices	One-task-one-node, max. delay	Allocation of tasks to nodes, FN operating frequency	Decomposition into 2 subproblems MA9.2 and MA9.3
MA9.2	An FN processing offloaded task	Minimize the energy consumption of the FN	Max. delay	FN operating frequency	Convex optimization – SCA, Lagrange method
MA9.3	Multiple FNs processing offloaded tasks or sending them to other FNs or cloud for processing	Minimize the sum of energy consumption of all devices	One-task-one-node	Allocation of tasks to nodes	Heuristic – greedy algorithm
MA10.1 **Kopras et al.** [9] (2023)	Multiple MDs offloading tasks to multiple FNs or a cloud	Minimize the sum of energy consumption of all devices	One-task-one-node, max. delay	Allocation of MD-FN transmission, allocation of tasks to nodes, FN operating frequency	Decomposition into 3 subproblems MA10.2, MA10.3, and MA10.4
MA10.2	An FN processing offloaded task	Minimize the energy consumption of the FN	Max. delay	FN operating frequency	Rational (3rd degree) function minimization
MA10.3	An MD offloading a task to one of multiple FNs	Minimize the sum of energy consumption of all devices	–	Allocation of MD-FN transmission for all possible computation allocations	Exhaustive search
MA10.4	Multiple FNs processing offloaded tasks or sending them to other FNs or cloud for processing	As in MA10.3	One-task-one-node	Allocation of tasks to nodes	Hungarian algorithm
MA11.1 **Kopras et al.** [9] (2023)	Multiple MDs offloading tasks to multiple FNs or a cloud	As in MA10.4	One-task-one-node, max. delay	Allocation of MD-FN transmission, allocation of tasks to nodes, FN operating frequency	Decomposition into 3 subproblems MA11.2, MA11.3, and MA11.4
MA11.2	An FN processing offloaded task	Minimize the energy consumption of the FN	Max. delay	FN operating frequency	Rational (3rd degree) function minimization
MA11.3	An MD offloading a task to one of multiple FNs	Minimize the sum of energy consumption of all devices	–	Allocation of MD-FN transmission	Heuristic – always transmitting with the lowest MD-FN cost
MA11.4	Multiple FNs processing offloaded tasks or sending them to other FNs or cloud for processing	Minimize the sum of energy consumption of all devices	One-task-one-node	Allocation of tasks to nodes	Hungarian algorithm
MA12.1 Sun and Chen [47] (2023)	MDs offloading tasks to FNs or processing them locally, FNs caching services	Maximize utility (price paid by MDs minus costs including energy) for the network provider	One-task-one-node, Max. FN power consumption, max. FN storage	MD transmission power, FN operating frequency, offloading cost, cache location	Reducing incentive constraints, decomposition into MA12.2 and MA12.3
MA12.2	FNs processing offloaded tasks	Maximize utility	Max. FN power consumption	FN operating frequency	Convex optimization
MA12.3	MDs offloading tasks to FNs (MD connected to a single FN) or processing them locally, FNs caching services	As in MA12.3	Max. FN storage	MD transmission power, cache location	Exhaustive generation of all possible cache locations, greedy algorithm to find opt. cache locations, convex optimization to find MD transmission power



(a) Distribution of objective functions.

(b) Distribution of optimization methods.

Figure 2.4: Charts summarizing solutions examined in Section 2.3. Objective function components: (E)nergy, (D)elay, (O)ther. Optimization methos families: (C)onvex, (H)euristic, (M)eta(H)euristic, (M)achine (L)earning, (G)ame theory, (M)ixed (A)pproach.

2.3.4 Optimization summary

Many works have been published on optimization of energy-efficient fog networks which still is a timely topic. The examined works often provide multiple optimization problems and solutions to these problems. Tab. 2.4 provides a concise summary of key elements of stated optimization problems. Moreover, Section A.1.1 (moved to the Appendix) provides an analysis of chosen optimization methods and results achieved by them as well as the baseline solutions used for comparison.

In these problems, energy consumption is used as either an optimization objective, a constraint, or both. Over half (29 out of 57) solutions use energy consumption as the objective function as shown in Fig. 2.4. The objective function often includes summing the energy costs over multiple devices, sometimes with unequal weights as described in detail in the *Objective function* column of Tab. 2.4.

We find that most proposed solutions (23, 40.4% of total) belong to the family of convex optimization as presented in Fig. 2.4b. There is also a significant share of algorithms (10, 17.5/%) based on metaheuristics, all of which are nature-inspired. Another large portion (12, 21.1%) use approaches from different optimization families to solve corresponding problems. Still, solutions from within the same family can vary greatly.

There is no clear trend as to which optimization algorithms are becoming more popular among the researchers. There is also no easy answer to the question of which algorithms produce the best results. Still, based on the survey conducted, the author can provide general recommendations below:

1. Use the well-tested convex optimization methods if the problem you are examining can be solved with them.

2. Otherwise, test the performance of the existing metaheuristics or Machine Learning (ML) algorithms on your problem. Adjustment of the method to better suit your particular problem is advisable.

3. As shown in Tab. 2.4 it can be beneficial to split the problem into smaller subproblems where each subproblem can be solved with a different method.

Some authors choose algorithms proposed in other research works as baseline solutions. In a vast majority of cases (except for when [29] compares itself with [24], and to a lesser extent when [58] compares itself with [111]) the proposed solutions show improvement over the cited ones. Still, most works use trivial baseline solutions such as no offloading or random assignment.

3 The Impact of the Fog and Cloud Tiers Parameters on Latency and Energy Consumption in the Network

This chapter provides the model, parameterization, and calculations of energy consumption and delay in fog computing networks. Here, the author of this thesis considers the wired segment of a fog network. This is because the aim is to examine scenarios, in which it is more beneficial (from the energy-efficiency point of view) to execute computational tasks in the fog tier or in the cloud. Considering the wireless segment of the fog does not change the conclusions of this chapter.

The estimation of the energy consumption and latency discussed below provides the basis for energy optimization considerations provided in Chapters 4, 5, and 6. The considerations and results presented below have been published by the author of this thesis in [13].

3.1 Network Model

This section defines models used for estimating delay and power consumption related to the offloading of computations in fog computing networks. Computational requests are modeled in Section 3.1.2. The power consumption of the network is modeled in Section 3.1.3, and delay is modeled in Section 3.1.4.

3.1.1 Network Description

In the bottom tier of the network, there are end devices (e.g., smartphones, sensors) that may require offloading computational tasks. These tasks can be processed in either the fog tier (consisting of a set $\mathbb{F}$ of FNs) or the cloud tier (set $\mathbb{C}$ of DCs). Data is sent to the FNs through the RRHs as shown in Fig. 3.1. Then the results are transmitted back to the MD.

The FNs are capable of sharing the computational load between themselves. To simplify the model and calculations that follow, they do so by inducing neither additional delays nor power consumption. These costs have also been left out by other researchers [49, 50, 52].

3.1.2 Offloaded Tasks – Computational Requests

Let there be a total of N requests offloaded during analyzed time period T. $\mathcal{R}_k^F$ and $\mathcal{R}_k^C$ are the sets of requests offloaded to the fog tier and cloud tier respectively. Let L^{R_i} be the size (in bits) of the *i-th* request R_i. Total amount of offloaded data L, amount of data offloaded to the fog tier X

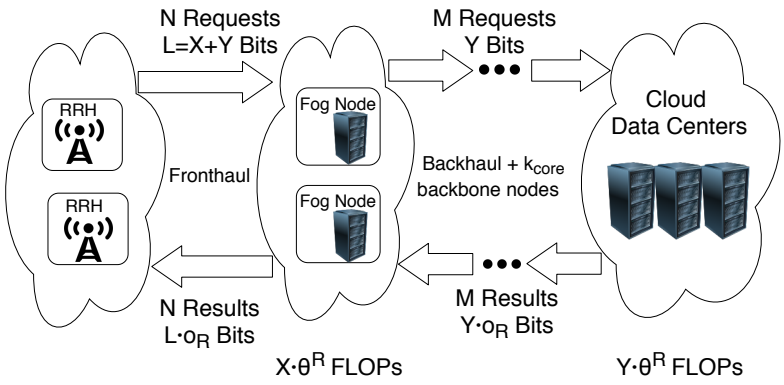


Figure 3.1: Diagram explaining the flow of data through the considered fog computing network.

and cloud tier Y are therefore:

$$L = X + Y,\ X = \sum_{R_i \in \mathbb{R}^F} L^{R_i},\ Y = \sum_{R_i \in \mathbb{R}^C} L^{R_i} \tag{3.1}$$

Concepts discussed in this section are illustrated in Fig. 3.1. It is important to note that, in this work, the requests are examined as they enter the fog tier of the network. The cost of transmitting a single request from MDs to RRH/FN is not considered.

The volume of offloaded data is not the only thing that matters when it comes to processing data. For certain applications, the number of required computations can be disproportionate to the size of an input, e.g., a PGN file representing a chess game [100] is minuscule (< 1 kB) compared to the amount of computation used to analyze the game. Let θ^{R_i} be the arithmetic intensity of task R_i defined as a ratio of single precision FLOPs required to process this task to its size in bits.

When the data has been successfully processed, the results are transmitted back to the MD. Let o^R be the average ratio of size of the result to the size of the offloaded task. If $o^R = 0$, then there is no transmission of the results.

3.1.3 Power Consumption

The power consumption model is divided into two parts: communication (transmission and reception of the data) and computation (processing of data). In this work, the power consumption of end devices is not considered.

3.1.3.1 Communication

For the power consumption of networking equipment, the linear model from [117] is used which includes idle power P_{idle} and active power that scales with load C (in bits/second) by parameter γ_b (in Joules/bit):

$$P(C) = P_{\text{idle}} + C\frac{P_{\text{max}} - P_{\text{idle}}}{C_{max}} = P_{\text{idle}} + C\gamma_b \tag{3.2}$$

where P_{max} is maximum power consumption and C_{max} is maximum load. Energy-per-bit cost of transmitting data through the core, backhaul, and fronthaul networks is equal to the number

of devices through which data flows (k_{core}, k_{bck}, k_{fr} respectively) multiplied by the average γ_b parameter, i.e., $\overline{\gamma_b}$, of devices in this part of network:

$$\gamma_{b-core} = k_{core}\overline{\gamma_b},\ \gamma_{b-bck} = k_{bck}\overline{\gamma_b},\ \gamma_{b-fr} = k_{fr}\overline{\gamma_b} \tag{3.3}$$

The number k_{core} can vary from 1 if the Cloud Node (CN) is directly connected to the Fog tier of the network to 5-20 in a more realistic scenario of cloud being few hundred/thousand kilometers away [118]. For backhaul transmission it is assumed that there is only a single hop/node from the FNs to the core network. k_{fr} is equal to 2 (RRH and FN).

Total energy E_{act}^{comm} used for "active" transmission is equal:

$$E_{act}^{comm} = \gamma_{b-fr} L\left(1 + o_R\right) + \left(\gamma_{b-core} + \gamma_{b-bck}\right) Y\left(1 + o_R\right). \tag{3.4}$$

Eq. (3.4) clearly shows that L bits go through the fronthaul to the fog and, from these L, Y are transmitted further (via backhaul and core) to the cloud.

Idle power consumption of devices in the fog tier of the network should also be considered. For fronthaul communication it is assumed that there is one networking device in each of F FNs and H RRHs. Idle power consumption P_{idle}^{front} of devices in fronthaul is defined as:

$$P_{idle}^{front} = \left(F + H\right) P_{idle}^{f} \tag{3.5}$$

If there is a fixed number of B networking devices in the backhaul (e.g., one per FN), then the backhaul idle power consumption P_{idle}^{back} is given:

$$P_{idle}^{bck} = B P_{idle}^{b} \tag{3.6}$$

P_{idle}^{f} and P_{idle}^{b} denote idle power of a single networking device in fronthaul and backhaul respectively.

Total power P_{tot}^{comm} spent on transmission:

$$P_{comm} = \frac{E_{act}^{comm}}{T} + P_{idle}^{front} + P_{idle}^{back} \tag{3.7}$$

Respective values γ_b, P_{idle}, and C_{max} are taken from [119] and [120]. For IP routers we calculate $\gamma_b = \frac{P_{\text{max}} - P_{\text{idle}}}{C_{max}}$ assuming that P_{idle} is equal to 90% of P_{max} [120].

3.1.3.2 Computation

Different power consumption models for cloud DCs and FNs should be used as their scale of operation is much different. FNs are utilized solely by the tasks from within examined fog computing network while clouds are full of computational requests from various devices across the Web. For FNs let us assume that they consume P_{act} watts of power when performing computations and P_{idle} when in idle-state. Values of P_{act} and P_{idle} depend on type and model of the device as well as parameters such as clock frequency.

$\mathbb{R}^{F_i}$ (subset of $\mathbb{R}^F$) is the set of requests processed in the FN i. Assuming the clock frequency f^{F_i} of FNs is fixed while performing calculations, then FN i is actively computing for:

$$t_{act}^{F_i} = \frac{\sum_{R_i \in \mathbb{R}^{F_i}} L^{R_i} \theta^{R_i}}{f^{F_i} s^{F_i}} \tag{3.8}$$

$t_{act}^{F_i}$ and $t_{idle}^{F_i}$ are the time-lengths in which FN i is in active and idle state respectively, $T = t_{act}^{F_i} + t_{idle}^{F_i}$. Therefore total power consumption $P_{\text{cp}}^{\text{fog}}$ spent on computations in fog equals:

$$P_{\text{cp}}^{\text{fog}} = \frac{\sum_{F_i \in \mathbb{F}} \left(t_{act}^{F_i} P_{act}^{F_i} + t_{idle}^{F_i} P_{idle}^{F_i} \right)}{T} \tag{3.9}$$

For CN, it is not feasible to estimate direct impact of offloaded workload on the clock frequency (and number of running servers). Instead, a simpler measure called GFLOPS per watt is used, this is a benchmark that is often used in comparing performance of supercomputers [121]. Let β^{C_j} be the GFLOPS per watt value characterizing CN j. $\mathbb{R}^{C_j}$ (subset of $\mathbb{R}^C$) is the set of requests processed in the cloud DC j. Power $P_{cp}^{C_j}$ consumed in CN j is calculated as follows:

$$P_{cp}^{C_j} = \frac{\sum_{R_i \in \mathbb{R}^{C_j}} L^{R_i} \theta^{R_i}}{\beta^{C_j}} \tag{3.10}$$

Power $P_{\text{cp}}^{\text{cld}}$ consumed in whole cloud tier is equal to the sum of power consumption of each CN:

$$P_{\text{cp}}^{\text{cld}} = \sum_{C_j \in \mathbb{C}} P_{cp}^{C_j} \tag{3.11}$$

Total power P consumed in the network is the sum of power consumption spent on transmission of data (P_{comm}) and on performing computations in fog – $P_{\text{cp}}^{\text{fog}}$ and in cloud – $P_{\text{cp}}^{\text{cld}}$.

$$P_{tot} = P_{comm} + P_{\text{cp}}^{\text{fog}} + P_{\text{cp}}^{\text{cld}} \tag{3.12}$$

3.1.4 Latency

The delay model is also divided into two parts: communication and computation. The wireless communication channel between end devices and FNs/RRHs is not considered. Wireless transmission from MDs is pivotal to other research works [26, 28].

3.1.4.1 Communication

For delays associated with communication the Round-Trip Time (RTT) [122] metric will be used. 7.5μs/km is chosen as a numerical value assigned to RTT based on [118]. It can be seen that for data computed in the fog tier the RTT value is negligibly low as FNs are meant to be located close to the end users. On the other hand, when it comes to offloading computation to the cloud tier, the RTT value is significant as the cloud can be a few hundred (thousand) kilometers away from the source of the request.

The average-per-request transmission delay in the fronthaul D_{comm}^{front} is defined as:

$$D_{comm}^{front} = \frac{L\,(1 + o_R)}{N r_{b,front}} \tag{3.13}$$

where L/N is the average size (in bits) of a computational request sent to be processed in either the fog tier or the Cloud tier and $r_{b,front}$ is the bitrate of fronthaul link between an RRH and an FN. RTT estimated by the length of the fronthaul link is assumed to be negligible.

Requests sent to cloud traverse multiple links and hops in the network. Of all of these links it is assumed that the fronthaul link is the slowest in terms of bitrate. Only the fronthaul delay caused by packet size will be considered as it is seen as the "bottleneck" in this network scenario. However, backbone/backhaul network can introduce significant "distance" delay – RTT. Let $\overline{d}$ be the average distance between CN and FNs which forward requests to this CN. The average-per-request transmission delay in the backhaul and the backbone equals

$$D_{comm}^{bck} = \overline{d} \cdot 7.5\mu s/km \tag{3.14}$$

Total (average-per-request) communication delay D_{comm} is calculated based on the premise that all N requests go through the fronthaul while M requests are transmitted through the backhaul and backbone network:

$$D_{comm} = \frac{ND_{comm}^{front} + MD_{comm}^{back}}{N} \tag{3.15}$$

3.1.4.2 Computation

For estimating computational delays in fog tier of the network the author let us assume that there is a queueing system. As an example, in [49] each FN is modeled as an M/M/1 queue. However, as in this work the FNs can balance the load between themselves, the queueing for each FN should not be independent – as long as there is an unutilized node the request should not "wait in queue" of another node. Therefore, the entire fog tier is modeled as an M/M/n queue. $\lambda^{fog} = \frac{(N-M)}{T}$ is the average request arrival rate. $n = F$ is equal to the number of FNs. The service rate μ^{fog} is calculated as the ratio of GFLOPS performance of an FN[1] and an average number of FLOPs that are needed to process a request sent to the fog tier:

$$\mu^{fog} = \frac{f^{F_i} s^{F_i}}{\frac{\sum_{R_i \in \mathbb{R}^F} L^{R_i} \theta^{R_i}}{N-M}} \tag{3.16}$$

Then, the average delay D_{cp}^{fog} of a request caused by computing (and queueing) in the fog tier equals:

$$D_{cp}^{fog} = \frac{C(n, \lambda^{fog}/\mu^{fog})}{n\mu^{fog} - \lambda^{fog}} + \frac{1}{\mu^{fog}} \tag{3.17}$$

where $C(n, \lambda^{fog}/\mu^{fog})$ is the Erlang C formula.

For computations in CN, it is assumed that the computational resources are vast and there is never a need for queueing tasks. It is modeled as an M/M/∞ queue. So the average latency D_{cp}^{cld} depends only on the computation and is equal to $\frac{1}{\mu^{cld}}$, where:

$$\mu^{cld} = \frac{f^{cld} s^{cld}}{\frac{\sum_{R_i \in \mathbb{R}^C} L^{R_i} \theta^{R_i}}{M}} \tag{3.18}$$

is analogous to Eq. (3.16). It is worth noting that by assuming infinite computational resources at each CN the number of DCs in the network does not impact the performance. Total (average-per-request) computational delay D_{cp} is equal:

$$D_{cp} = \frac{(N-M)D_{cp}^{fog} + MD_{cp}^{cld}}{N} \tag{3.19}$$

[1] All FNs are modeled to have the same frequency and FLOP/cycle measure to allow the use of the M/M/n model.

Total (average-per-request) delay D_{tot} is calculated as a sum of computational and communicative delays:

$$D_{tot} = D_{cp} + D_{comm} \tag{3.20}$$

3.2 Simulation Results

This section covers results showing power consumption and delay in various scenarios for "fog without cloud" and for full fog computing. Evaluation has been performed using GNU Octave [123].

3.2.1 Fog Computing Network without the Cloud

First, let us consider a network with $H = 20$ RRHs, $F = 10$ FNs and no possibility of offloading to cloud DCs. Each request has size 1 MB, and values $\theta^R = 10$ and $o^R = 0.1$. The length of examined time period T is set to 60 s.

3.2.1.1 Computations

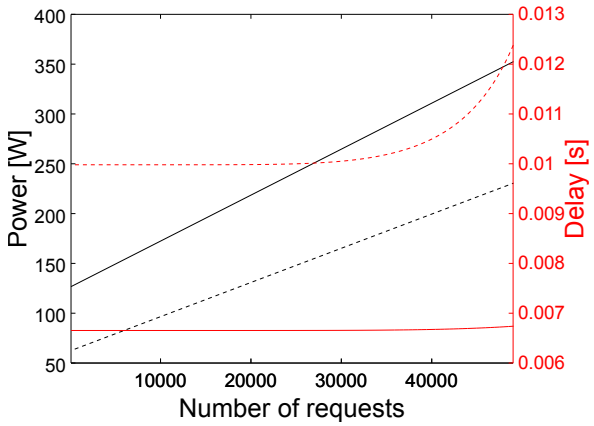
Let us examine the computational delay and power consumption in the FNs. Each FN is equipped with *Intel Core2 Duo E6850* as its CPU. $s^{F_i} = 4$ as that is the maximum number of double precision FLOPs that can be calculated in a single cycle of this processor [124]. P_{act} and P_{idle} are parametrized using the power consumption model from [98]:

$$P_{cpu} = 8.4503\nu_{proc}V_{cpu}^2 f_{cpu} + 36.3851V_{cpu} - 33.9503 \tag{3.21}$$

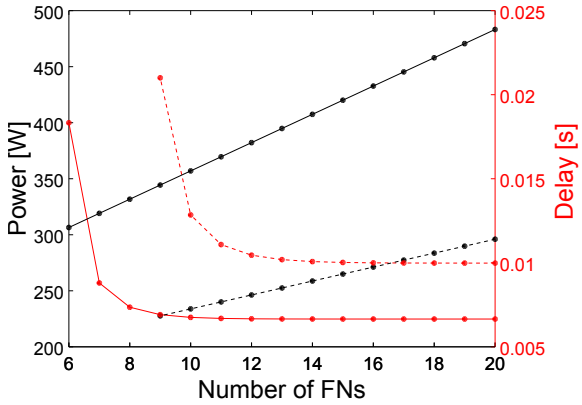
where ν_{proc} is the total utilization of the processor and the units of P_{cpu}, V_{cpu}, and f_{cpu} are W, V, and GHz respectively. It is assumed that for the idle state $\nu_{proc} = 0$ and for the active state $\nu_{proc} = 1$. Let us look at power consumption of an FN at two different frequency-voltage levels measured in [98]: $f_{cpu} = 3.006$ GHz, $V_{cpu} = 1.28$ V later called as '3 GHz' and $f_{cpu} = 2.004$ GHz, $V_{cpu} = 1.104$ V later called as '2 GHz'. At 3 GHz $P_{act}^{F_i} = 54.241$ W and $P_{idle}^{F_i} = 12.6226$ W. At 2 GHz $P_{act}^{F_i} = 26.859$ W and $P_{idle}^{F_i} = 6.2189$ W. These values are then inserted into Eq. (3.9) to calculate the computational power consumption of FNs. Delay and power consumption of FNs are plotted against the number of computational requests in Fig. 3.2a. It shows that decreasing the CPU frequency lowers power consumption, but increases delay. Results in Fig. 3.2a are calculated for the number of FNs $F = 10$. Fig. 3.2b plots power consumption and delay against the number of FNs for number of requests $N = 50000$. It can be seen that higher number of FNs decreases delay. There are diminishing returns as the delay caused by queueing quickly disappears and processing delay does not change with number of FNs. Moreover, utilizing more FNs increases power consumption.

3.2.1.2 Communications

The communications delay in the fronthaul depends on the bit rate of fronthaul links and the size of the requests (and responses) as in Eq. (3.13). The term *home gateway* is used in [119] to describe an access interface composed of several components, namely a processor plus memory, a



(a) Dependency on the number of requests; number of FNs $F = 10$.



(b) Dependency on the number of FNs; number of requests $N = 50000$.

Figure 3.2: Power consumption and delay related to computation in the fog tier (clock frequency: dashed line – 2 GHz, solid line – 3 GHz).

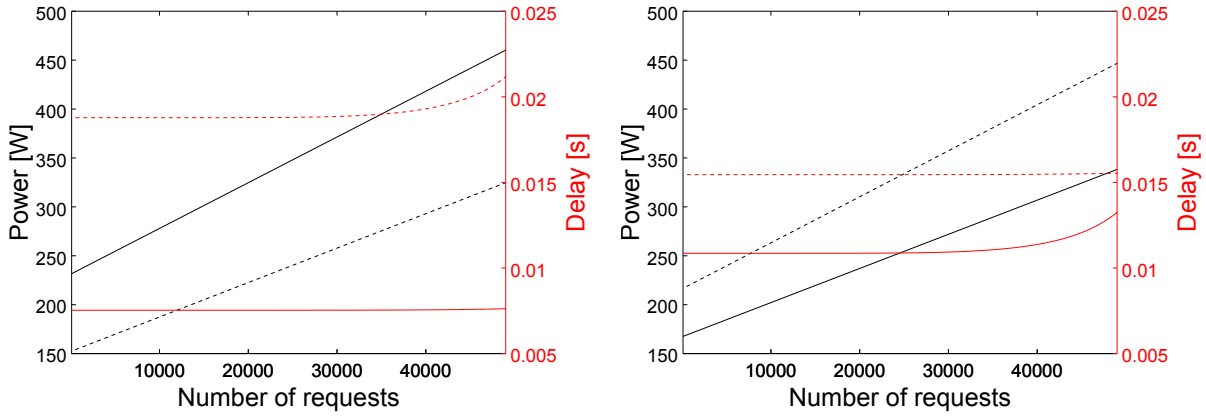
Table 3.1: Power consumption of networking equipment.

Equipment	**Capacity**	P_{act}	P_{idle}	γ_b
Gateways [119]				
1G EPON	1 Gb/s	3.3 W	3.0 W	0.3 nJ/bit
10/10G EPON	10 Gb/s	5.5 W	3.5 W	0.2 nJ/bit
Core routers [120]				
Juniper T1600	640 Gb/s	6572 W	5915 W	1.03 nJ/bit

Wide Area Network (WAN) interface, and several Local Area Network (LAN) interfaces. In this work, it is used to model interfaces of RRHs and FNs. Parameters of Ethernet Passive Optical Network (EPON) home gateways are detailed in Tab. 3.1.

3.2.1.3 Overall performance

The values calculated in Sections 3.2.1.1 and 3.2.1.2 are added and presented in Fig. 3.3. Fig. 3.3 shows total (stemming from computations and transmission) power consumption and delay in a „fog without cloud" network. The following conclusions can be drawn. The average delay is low – hovering around 10-20 ms. There is also a visible trade-off between power consumption and delay – higher clock frequency and interface bandwidth results in shorter delay and greater power consumption. With that in mind, a combination of FNs running at 2 GHz frequency and 10 Gbit fronthaul outperforms FNs running at 3 GHz and 1 Gbit fronthaul in both estimated metrics at given traffic conditions.



(a) Dashed line: f=2 GHz, r=1 Gbps; solid line: f=3 GHz, r=10 Gbps.

(b) Dashed line: f=3 GHz, r=1 Gbps; solid line: f=2 GHz, r=10 Gbps.

Figure 3.3: Total power consumption and delay in "fog without cloud" network. f – FN clock frequency, r – the fronthaul bit rate.

3.2.2 Fog Computing Network Including the Cloud

Let us consider a network with $H = 20$ RRHs, $F = 10$ FNs and one CN. RRHs and FNs are connected through 1G EPON. FNs are connected through the 10G EPON backhaul to the core network. For estimating delays and power consumption occurring in the core network the following three distance scenarios are chosen. In *near* scenario the CN is located 100 km and 6 nodes away from the fog tier of the network. The other scenarios are *medium* – 2000 km, 12 nodes and *far* – 8000 km, 18 nodes. There is a Juniper T1600 router (Tab. 3.1) installed at each node. Cloud computational power efficiency is assumed to be 1 GFLOPS/W (Giga Floating Point Operations per Second (GFLOPS) per Watt), its processors are running at 1.5 GHz frequency, and can calculate up to 32 FLOPs in a single cycle ($s^{cld} = 32$). FNs are assumed to work at 2 GHz with $s^{F_i} = 4$ FLOPs per cycle as in the previous subsection. Each request has size 1 MB, and values $\theta^R = 10$ and $o^R = 0.1$. The length of time period remains $T = 60$ s.

Let us vary the number of requests M that are sent to CN while the total number of offloaded requests stays the same ($N = 50000$). Power consumption and average delay are plotted in Fig. 3.4 against the fraction of requests $\frac{M}{N}$ sent to the cloud. There are a few interesting observations. First, the distance (physical and logical) to the CN plays a key role in determining whether offloading data all the way to the CN is beneficial. In the *near* scenario utilizing cloud reduces both latency and power consumption. More interesting results can be seen in the *medium* scenario: performing computations in the cloud decreases power consumption while performing them in fog results in lower average latency.

The power efficiency of cloud DCs contributes significantly to total power consumption. As shown in Fig. 3.5a, if power efficiency is decreased (from 1 to 0.5 GFLOPS/W), then sending requests to the cloud is more expensive than computing them in fog for all distance scenarios. On the other hand, at 10 GFLOPS/W (world-class efficiency [121]), sending requests to the cloud consumes less power (Fig. 3.5b). Delay does not depend on power efficiency.

The impact of arithmetic intensity of requests θ^R on network performance is also studied. The results for $\theta^R = 1$ and $\theta^R = 100$ are shown in Figs. 3.5c and 3.5d. The corresponding share of

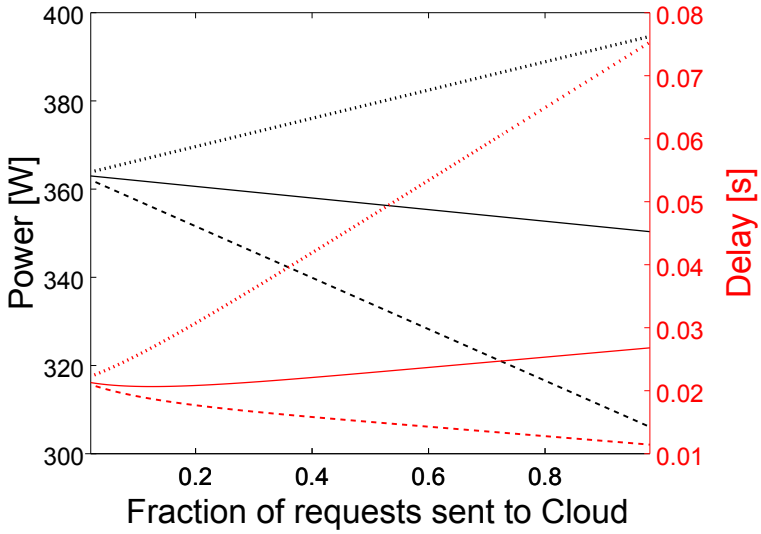


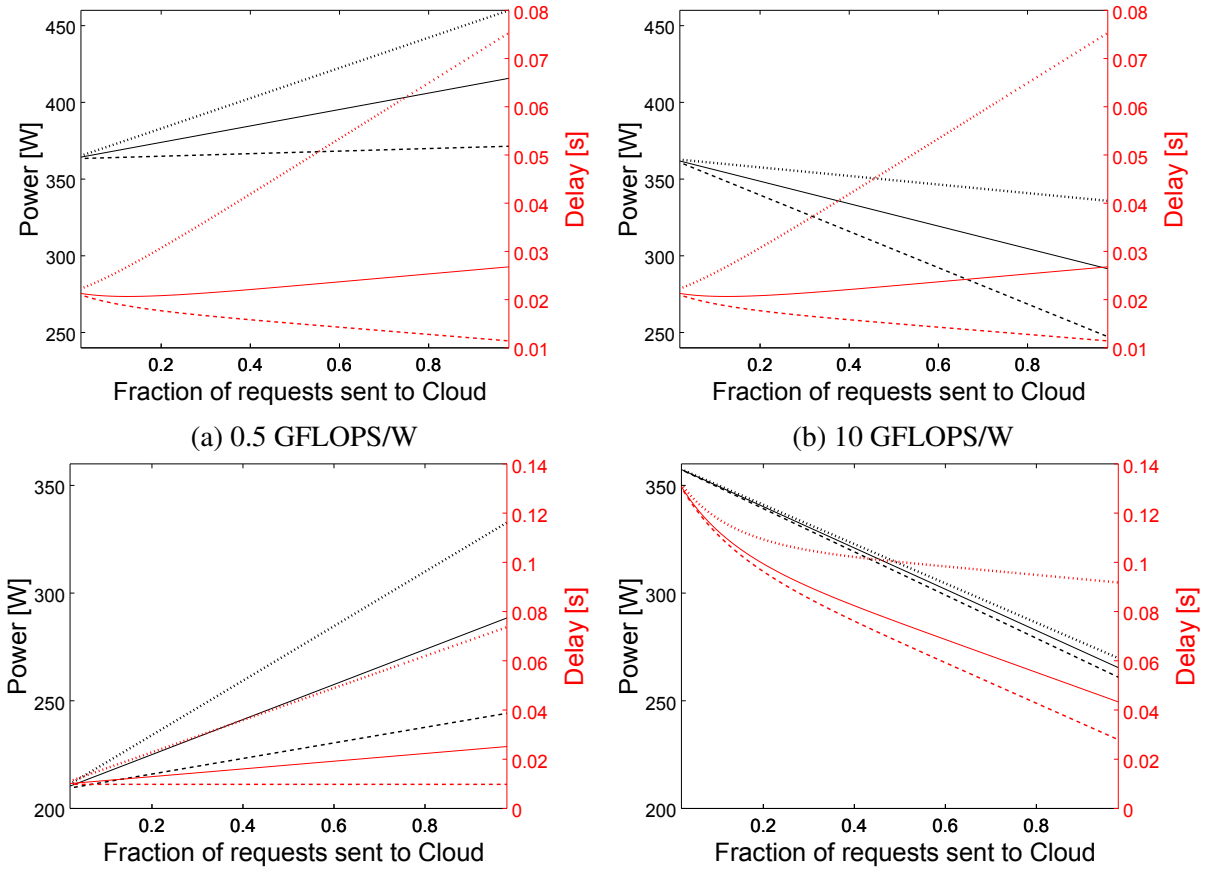
Figure 3.4: Total power consumption and delay in fog computing network including the cloud vs. the fraction of requests sent to the cloud for *near*, *medium*, *far* scenarios (dashed, solid, and dotted lines respectively).

total delay and power consumption spent on computation and communication is plotted in Fig. 3.6. For $\theta^R = 1$ computing requests in the fog tier is faster and consumes less power than processing them in the cloud tier. The delay and power consumption caused by data sent through the core network outweighs the fact that cloud DCs are more efficient at computing. On the contrary, when the offloaded requests require heavy computations, it is beneficial to send these requests all the way to the cloud as high θ^R means lower communication cost compared to computation cost.

3.3 Chapter Summary

In this chapter, the impact of different parameters on power consumption and delay in fog computing has been examined. The computational tasks have been analyzed with respect to their complexity (how computationally intensive they are to process) and the size of messages representing the computing requests and the results. The communication and computational resources have been characterized by the computing machines' power efficiency (in GFLOPS/W), the clock frequency, and the distance from the request source. It has been observed that there is usually a trade-off between task execution delay and power (or energy) consumption. For example, Fog Nodes (FNs) working at a higher frequency consume more power but provide lower latency. The high arithmetic intensity of offloaded requests favors (as expected) processing in the cloud. In such cases, the high processing speed and computational power efficiency can offset the delay and power consumption caused by transmitting data through the core network. Conversely, requests that require relatively few operations to process are best served by nearby FNs.

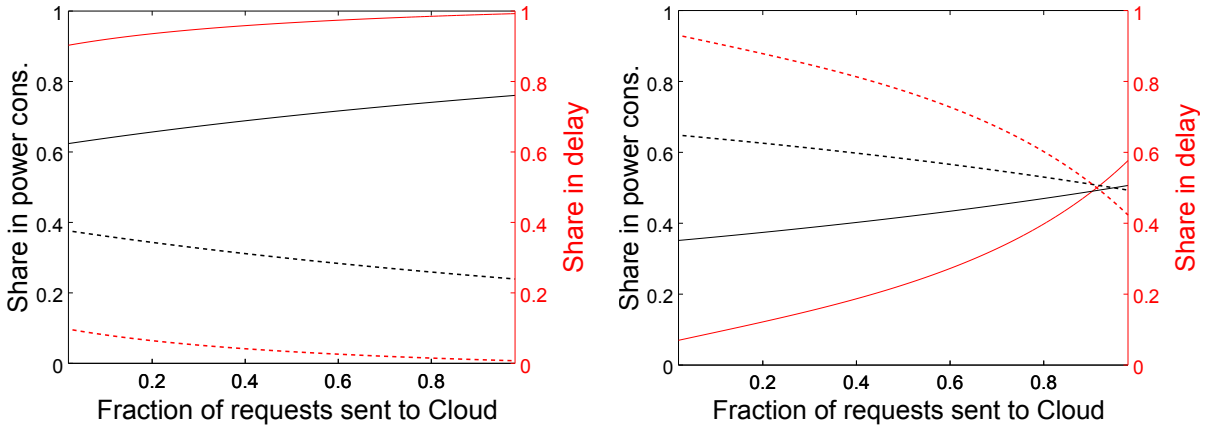
In the scenarios where tasks are characterized by average arithmetic intensity, and the cloud offers average (cloud-typical) computing power and efficiency, the power consumption and latency depend on the fraction of tasks delegated to the cloud. Interestingly, in the scenario, when the cloud server is far from the network edge (here above defined as a fiberline distance of 8000 km) it is



(a) 0.5 GFLOPS/W (b) 10 GFLOPS/W

(c) $\theta^R = 1$ (d) $\theta^R = 100$, $N = 5000$

Figure 3.5: Total power consumption and delay in full fog computing network vs. the fraction of requests sent to the cloud, cloud power efficiency (GFLOPS/W), and arithmetic intensity of requests θ^R. Dashed line – *near* scenario, solid line – *medium*, dotted line – *far*.



(a) $\theta^R = 1$ (b) $\theta^R = 100$, $N = 5000$

Figure 3.6: Share of computation (dashed line) and communication (solid line) in power consumption and delay in a full fog computing network, *medium* scenario.

not energy-efficient to delegate the tasks to the cloud server. The average latency is also increased due to the communication delay over the core network. Consequently, in such a scenario, the fog tier computing resources are more recommended to be used for the energy- and latency-aware applications.

4 Optimization of Energy Consumption in the Fog and Cloud Tiers

The work presented in this chapter considers task distribution between many FNs and a single Cloud Nodes (CNs). The objective is to minimize network energy consumption while meeting delay constraints specific to each offloaded task. As shown in Figure 4.1, the assumption made is that an offloaded task can be processed in the node to which it is originally sent (solid blue arrows), in another FN (solid green arrows), or in the cloud (hollow red arrows).

First, a realistic network model is proposed encompassing the energy consumption as well as the delay, related to both the necessary computations and communication. The network model includes network parameters reflecting the characteristics of real-world equipment in the wired segment of the fog network (as provided in Chapter 3). Based on this model, the author of this thesis formulates an optimization problem to minimize the total (task transmission- and computation-related) energy consumption while fulfilling individual delay constraints. The optimization considers not only the assignment of tasks to the nodes but also the Central Processing Unit (CPU) frequency at each utilized node.

The problem is a non-convex Mixed-Integer Non-Linear Programming (MINLP) problem, so the Successive Convex Approximation (SCA) method is applied, which transforms it into a series of convex MINLP problems and provides the optimal solution by using the primal and dual decomposition techniques as well as the Hungarian algorithm. A sub-optimal, lower-complexity solution is also proposed.

The work presented in this chapter has been published by the author of this thesis in [10].

4.1 Network model

The network model is introduced in this section. The notation used in this chapter is presented in Tab. 4.1. Letters in superscript are used throughout this chapter as upper indices, not exponents, e.g., L^r does **not** denote L to the power of r.

In the bottom tier of the network, there are end devices (e.g., smartphones, sensors) with some specific computational tasks. Throughout this work, they are referred to as Mobile Devices (MDs) Serving these tasks requires offloading them, i.e., they either cannot be processed in the MD or the MD chooses to offload them rather than process them locally. Then, they can be processed either in the fog tier, consisting of set $\mathcal{F}$ of FNs, or in the cloud tier (set $\mathcal{C}$ of CNs). A set of all computing nodes in the network is denoted as $\mathcal{N} = \mathcal{F} \cup \mathcal{C}$. An example of a network with 2 FNs and 1 CNs is shown in Fig. 4.1.

Unlike works that focus on MD (such as [26–28]), this thesis examines energy consumption from the point of view of the fog network. Modeling and optimizing wireless transmission is a key part of these works, e.g., allocating sub-channels to mobile devices in [27] or interference

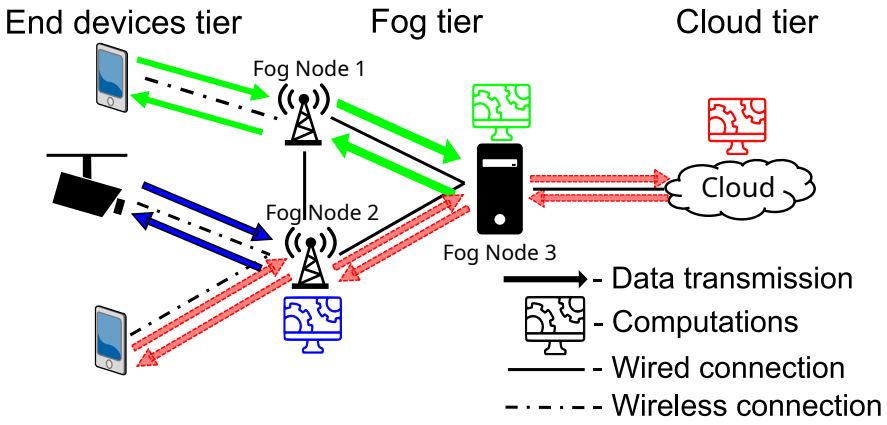


Figure 4.1: Simplified fog network with offloaded tasks. Each color represents a different task.

affecting transmission rates in [28]. The model presented in this chapter is agnostic to the wireless technology and model/condition of the channel used for transmission between MDs and FNs. We examine requests as they appear in the fog tier of the network. Instead, we focus on the efficient distribution of offloaded tasks between fog and cloud nodes.

4.1.1 Computational Requests

Let $\mathcal{T}$ be a numbered set $\{T_1, T_2, ..., T_{|\mathcal{T}|}\}$ of all time instances at which computational requests arrive in the network, and have to be allocated computing resources. Let $\mathcal{R}_k$ be a set of all requests arriving in the network at time T_k. Each computational request $r \in \mathcal{R}_k$ is described by the following parameters: (i) size L^r in bits, (ii) arithmetic intensity θ^r in FLOP/bit (used in FLOP/byte in [125, 126]), (iii) ratio o^r of the size of the result to the size of the offloaded task (most related works do not consider the transmission of the results [50, 52] or assume that its contribution is negligible [49, 127]; o^r equal to 1 implies the output has the same size as the input; in case of electrocardiography signals $o^r \simeq 0.07$ [65]), (iv) FN $g^r \in \mathcal{F}$ to which the request is originally sent (before allocation) (v) maximum tolerated delay $D^r_{\max}$. Let us define a binary variable a^r_n that shows where the request is computed, i.e., a^r_n equals 1 if $r \in \mathcal{R}_k$ is computed at node $n \in \mathcal{N}$, and 0 otherwise.

Let us use the example of offloading shown in Fig. 4.1 to illustrate the meaning of g^r and a^r_n. Assume green arrows represent request 1, blue represent request 2, and red represent request 3. For nodes, assume FNs are numbered as shown in Fig. 4.1 and a single CN is numbered 0. $g^1 = 1$ as the 1st request is originally sent to FN 1. Meanwhile, $g^2 = 2$ and $g^3 = 2$ as both remaining requests arrive at FN 2. Values $a^1_3 = 1, a^2_2 = 1, a^3_0 = 1$ mean that requests 1, 2, and 3 are processed at nodes 3, 2, and 0 respectively. In this example, a^r_n is equal to zero for all other sets of values (r,n), $r \in \mathcal{R}_k, n \in \mathcal{N}$.

Table 4.1: The notation used for modeling the network and defining the optimization problem in Chapter 4.

	Symbol	Description
Parameters	b_{back}	link bitrate in the backhaul and backbone network
	b_r^n	link bitrate between FNs $n \in \mathcal{F}$ and $g^r \in \mathcal{F}$
	$\mathcal{C}$	set of all Cloud Nodes
	χ	a parameter characterizing delay depending on distance
	d^n	fiberline distance to CN
	D^r_{max}	maximum tolerated delay requirement for request $r \in \mathcal{R}_k$
	$\mathcal{F}$	set of all Fog Nodes
	$f_{\text{max},n}$	maximum clock frequency of node $n \in \mathcal{N}$
	$f_{\text{min},n}$	minimum clock frequency of node $n \in \mathcal{N}$
	g^r	FN to which the request $r \in \mathcal{R}_k$ is originally sent
	γ_n^l	energy-per-bit cost of transmitting data of request $r \in \mathcal{R}_k$ between nodes $g^r \in \mathcal{F}$ and $n \in \mathcal{N}$
	L^r	size of request $r \in \mathcal{R}_k$
	$\mathcal{N}$	set of all nodes
	o^r	output-to-input data size ratio of request $r \in \mathcal{R}_k$
	$p_{n,q}$	q-th coefficient of polynomial modeling power consumption of CPU installed in node $n \in \mathcal{N}$
	Q	degree of polynomial P_n
	$\mathcal{R}_k$	set of all computational requests offloaded at T_k
	$\mathcal{R}'_k$	set of rejected computational requests offloaded at T_k
	s_n	number of FLOPs performed per single clock cycle at node $n \in \mathcal{N}$
	$\mathcal{T}$	set of all considered time instances, when one or more computational requests arrive
	θ^r	arithmetic intensity of request $r \in \mathcal{R}_k$
	T_k	time at which request $r \in \mathcal{R}_k$ arrives in the network, $k \in \{1, ..., \lvert\mathcal{T}\rvert\}$
	t_n	time at which node $n \in \mathcal{N}$ finishes computing its last task
Variables and metrics	a_n^r	whether request $r \in \mathcal{R}_k$ is computed at node $n \in \mathcal{N}$, $a_n^r \in \{0, 1\}$
	β_n	energy efficiency (GFLOPS per Watt) characterizing node $n \in \mathcal{N}$
	$D^r_{\text{comm},n}$	delay caused by transmitting request $r \in \mathcal{R}_k$ between nodes $g^r \in \mathcal{F}$ and $n \in \mathcal{N}$
	$D^r_{\text{dl},n}$	delay caused by transmitting request $r \in \mathcal{R}_k$ between nodes $g^r \in \mathcal{F}$ and $n \in \mathcal{N}$ – downlink
	$D^r_{\text{ul},n}$	delay caused by transmitting request $r \in \mathcal{R}_k$ between nodes $g^r \in \mathcal{F}$ and $n \in \mathcal{N}$ – uplink
	D^r_{cp}	computational delay caused by processing request $r \in \mathcal{R}_k$ in the network
	$D^r_{\text{cp},n}$	computational delay caused by processing request $r \in \mathcal{R}_k$ at node $n \in \mathcal{N}$
	D^r_{max}	maximum tolerated delay requirement for request $r \in \mathcal{R}_k$
	$D^{r,l}_{\text{queue},n}$	queuing delay of request $r \in \mathcal{R}_k$ at node $n \in \mathcal{N}$
	D^r_{tot}	total delay of request $r \in \mathcal{R}_k$
	$D^r_{\text{tot},n}$	total delay of processing request $r \in \mathcal{R}_k$ at node $n \in \mathcal{N}$
	E^r_{comm}	energy spent on transmission of request $r \in \mathcal{R}_k$
	$E^r_{\text{comm},n}$	energy cost for transmission of request $r \in \mathcal{R}_k$ between nodes $g^r \in \mathcal{F}$ and $n \in \mathcal{N}$
	E^r_{cp}	energy spent in the network on processing request $r \in \mathcal{R}_k$
	$E^r_{\text{cp},n}$	energy cost of processing request $r \in \mathcal{R}_k$ at node $n \in \mathcal{N}$
	E^r_{tot}	energy spent on transmission and processing of request $r \in \mathcal{R}_k$
	$E^r_{\text{tot},n}$	energy cost of offloading request $r \in \mathcal{R}_k$ when computing it at node $n \in \mathcal{N}$
	f_n	clock frequency of node $n \in \mathcal{N}$
	P_n	power consumption related to computations at node $n \in \mathcal{N}$

4.1.2 Energy Consumption

The energy consumption model consists of two parts: communication (transmission of data) and computation (processing of data). Energy E^r_{cp} spent on processing request $r \in \mathcal{R}_k$ equals:

$$E^r_{\text{cp}} = \sum_{n \in \mathcal{N}} a^r_n E^r_{\text{cp},n} = \sum_{n \in \mathcal{N}} a^r_n \frac{L^r \theta^r}{\beta_n}, \tag{4.1}$$

where $E^r_{\text{cp},n}$ is the energy spent on processing of request $r \in \mathcal{R}_k$ at node $n \in \mathcal{N}$. β_n characterizes computational efficiency of node $n \in \mathcal{N}$ given in Floating Point Operations (FLOPs) per second per watt [128]. For CNs, let us assume constant CPU clock frequency f_n and efficiency β_n. For FNs, β_n depends on CPU clock frequency f_n of node $n \in \mathcal{F}$, its power consumption P_n, and number s_n of FLOPs performed within a single clock cycle of this node [124]:

$$\beta_n = \frac{f_n s_n}{P_n} = \frac{f_n s_n}{\sum_{q=0}^{Q} p_{n,q} f_n^q} \tag{4.2}$$

P_n is modeled as a Q-th degree polynomial of f_n using parameters $p_{n,q}$ based on [98]. This allows the model to cover various models of CPUs. Moreover, clock frequency f_n must be within the range of minimum and maximum frequencies of the CPU installed in node $n \in \mathcal{F}$, i.e., $f_{\text{min},n} \leq f_n \leq f_{\text{max},n}$.

The energy spent on the transmission of request $r \in \mathcal{R}_k$ equals:

$$E^r_{\text{comm}} = \sum_{n \in \mathcal{N}} a^r_n E^r_{\text{comm},n} = \sum_{n \in \mathcal{N}} a^r_n L^r (1 + o^r) \gamma^l_n, \tag{4.3}$$

where $E^r_{\text{comm},n}$ is the energy required to transmit (communicate) request $r \in \mathcal{R}_k$ between FN g^r and node $n \in \mathcal{N}$ while γ^l_n is the energy-per-bit cost of transmitting data request $r \in \mathcal{R}_k$ between n and g^r. $L^r o^r$ is the size (in bits) of results transmitted back to g^r. Thus, the total energy spent on offloading request $r \in \mathcal{R}_k$ is given by:

$$E^r_{\text{tot}} = \sum_{n \in \mathcal{N}} a^r_n E^r_{\text{tot},n} = \sum_{n \in \mathcal{N}} a^r_n \left(E^r_{\text{cp},n} + E^r_{\text{comm},n} \right), \tag{4.4}$$

where $E^r_{\text{tot},n}$ is the energy cost of offloading request $r \in \mathcal{R}_k$ when it is computed at node $n \in \mathcal{N}$. Energy spent on wireless transmission between MDs and FNs is not included in $E^r_{\text{tot},n}$ as we examine requests already sent by the MDs (as they appear in the FNs).

4.1.3 Delay

The delay model is divided into three parts: communication, computation, and queuing. The delay D^r_{cp} caused by computing request $r \in \mathcal{R}_k$ equals:

$$D^r_{\text{cp}} = \sum_{n \in \mathcal{N}} a^r_n D^r_{\text{cp},n} = \sum_{n \in \mathcal{N}} a^r_n \frac{L^r \theta^r}{f_n s_n}, \tag{4.5}$$

where $D^r_{\text{cp},n}$ is the time required to compute request $r \in \mathcal{R}_k$ at node $n \in \mathcal{N}$. Moreover, there are significant differences between models of communication delay for requests processed in the fog

tier and the cloud tier of the network. It stems from the fact that clouds are assumed to have huge (practically infinite) computational resources with parallel-computing capabilities, and there is no need for queuing multiple requests served by the CN $n \in \mathcal{C}$. They can be processed simultaneously. Meanwhile, if multiple requests are sent to the same FN $n \in \mathcal{F}$ for processing in a short time span, additional delays may occur due to congestion of computational requests (an arriving request cannot be processed until processing of all the previous requests has been completed). On the other hand, it is assumed that CNs are located far away from the rest of the network (hundreds or even thousands of kilometers away) which introduces additional, transmission-related delay. Delay caused by transmitting request $r \in \mathcal{R}_k$ between (to and from) FN $g^r \in \mathcal{F}$ and cloud node $n \in \mathcal{C}$ is equal:

$$D^r_{\text{comm},n} = \frac{L^r(1+o^r)}{b_{\text{back}}} + d^n \cdot \chi, \tag{4.6}$$

where b_{back} is the link bitrate in the backhaul and backbone network, while d^n is the fiberline distance to CN $n \in \mathcal{C}$. The parameter χ indicates the rate at which delay increases with distance d^n [118].

For describing delays related to transmission between FNs let us split it into the uplink (sending a request to be processed) and downlink (sending calculated results back to the origin of said request) parts denoted $D^r_{\text{ul},n}$ and $D^r_{\text{dl},n}$ respectively. For transmission between FNs, we assume delay caused by the distance between them ($d^n \cdot \chi$ in Eq. (4.6)) to be negligible – well below 1 ms as we use the value of 7.5μs/km for parameter χ [118] – and therefore we ignore it. The total delay caused by communication between FNs $g^r \in \mathcal{F}$ and $n \in \mathcal{F}$ for request $r \in \mathcal{R}_k$ equals:

$$D^r_{\text{comm},n} = D^r_{\text{ul},n} + D^r_{\text{dl},n} = \frac{L^r}{b^n_r} + \frac{L^r o^r}{b^n_r}, \tag{4.7}$$

where b^n_r is the link bitrate between g^r and n. As discussed earlier, when a request is sent to FN $n \in \mathcal{F}$ and there is another request being processed at this node, the request is put in a queue and waits to be processed. Let us define a scheduling variable $t_n \in \mathbb{R}^+$ which indicates when the processing of the last request scheduled at FN $n \in \mathcal{F}$ is completed. Queuing delay of the request $r \in \mathcal{R}_k$ at node $n \in \mathcal{F}$ is calculated as follows:

$$D^{r,l}_{\text{queue},n} = \max(0, t_n - T_k - D^r_{\text{ul},n}). \tag{4.8}$$

$D^{r,l}_{\text{queue},n}$ has positive values when $t_n > T_k + D^r_{\text{ul},n}$, i.e., when request r arrives at node n at time $T_k + D^r_{\text{ul},n}$ and it is queued until processing of other request(s) is completed at time t_n. For each node $n \in \mathcal{C}$ (CNs), $D^{r,l}_{\text{queue},n}$ is always equal to zero, i.e., each request arriving at the cloud can immediately be processed regardless of the number of requests already being processed due to parallel processing. Thus, the total delay of processing request $r \in \mathcal{R}_k$ is the sum of delays related to transmission, queueing, and computation:

$$D^r_{\text{tot}} = \sum_{n \in \mathcal{N}} a^r_n D^r_{\text{tot},n} = \sum_{n \in \mathcal{N}} a^r_n \left(D^r_{\text{comm},n} + D^{r,l}_{\text{queue},n} + D^r_{\text{cp},n} \right). \tag{4.9}$$

4.1.4 Updating Scheduling Variables in the Fog

Let us now explain how the values of scheduling variables t_n are assigned to become parameters of an optimization instance. As no requests are processed at the beginning of the simulation, we

set $t_n = 0, \forall n \in \mathcal{F}$. For each $T_k \in \mathcal{T}$, after allocations a_n^r are determined, times t_n are updated according to when computation of requests offloaded to FNs is scheduled to finish:

$$t_n := \max(t_n, T_k + \sum_{r \in \mathcal{R}_k} a_n^r (D_{\text{ul},n}^r + D_{\text{queue},n}^{r,l} + D_{\text{cp},n}^r)), \forall n \in \mathcal{F}. \tag{4.10}$$

By using (4.10), each new instance of the optimization problem depends on results (allocations) of previous instances.

4.2 Optimization Problem

The objective of the optimization problem is to minimize total energy spent on offloading all requests arriving at the network at time T_k, that is to find:

$$(\mathbf{a}^\star, \mathbf{f}^\star) = \arg\min_{\mathbf{a},\mathbf{f}} \sum_{r \in \mathcal{R}_k} E_{\text{tot}}^r, \tag{4.11}$$

subject to:

$$\sum_{n \in \mathcal{N}} a_n^r = 1 \qquad \forall r \in \mathcal{R}_k, \tag{4.12}$$

$$\sum_{r \in \mathcal{R}_k} a_n^r \leq 1, \qquad \forall n \in \mathcal{F}, \tag{4.13}$$

$$D_{\text{tot}}^r \leq D_{\text{max}}^r, \qquad \forall r \in \mathcal{R}_k, \tag{4.14}$$

$$f_{\text{min},n} \leq f_n \leq f_{\text{max},n} \qquad \forall n \in \mathcal{F}, \tag{4.15}$$

$$a_n^r \in \{0, 1\}, \qquad \forall r \in \mathcal{R}_k, \forall n \in \mathcal{N}, \tag{4.16}$$

where $\mathbf{a}^\star = \{a_n^{r\star}\}$ and $\mathbf{f}^\star = \{f_n^\star\}$ are optimal values of the optimization variables: allocation variables a_n^r and CPU clock frequencies f_n, respectively. The constraints (4.12) and (4.13) restrict that each request must be processed at one and only one FN or CN and that each FN can process at most a single request at a time, respectively. The constraints (4.14) guarantee that the total delay D_{tot}^r must not be greater than the maximum acceptable one D_{max}^r. Moreover, according to the constraints (4.15), the CPU frequency is limited by lower and upper bound while the decision variables a_n^r take only binary values, according to constraints (4.16).

The optimization problem cannot be solved for some sets of requests $\mathcal{R}_k$, where it is impossible to satisfy all constraints (e.g., no feasible allocation of requests so that each request is processed (4.12) while fulfilling its delay requirement (4.14)). In this case, rather than terminating the optimization without any solution (which would translate to rejecting all requests $\mathcal{R}_k$), we choose to reject requests for which (4.14) cannot be fulfilled. The optimization is then performed over the set of remaining requests $\mathcal{R}_k \setminus \mathcal{R}_k'$, where $\mathcal{R}_k'$ denotes the set of rejected requests.

4.3 Proposed solution

The optimization problem defined in Section 4.2 is a MINLP problem due to binary values of the allocation variables and continuous values of the CPU clock frequencies. Nonlinearity in the

problem results from the power consumption model of the CPU and the set of constraints (4.14). Note that after substituting (4.2) into (4.1), the energy spent on processing of request $r \in \mathcal{R}_k$ at node $n \in \mathcal{F}$ is the sum of polynomial and rational functions:

$$E^r_{\text{cp},n} = \frac{L^r \theta^r}{s_n} \left[\frac{p_{n,0}}{f_n} + \sum_{q=1}^{Q} p_{n,q} f^{q-1} \right]. \tag{4.17}$$

As such, for $f_n \in \mathbb{R}^+$, the convexity of (4.17) in f_n depends on the parameters $p_{n,q}$ (except $p_{n,1}$ and $p_{n,2}$ which have no influence on convexity, since their second derivatives are zero). If $\{p_{n,0}, p_{n,3}, ..., p_{n,Q}\}$ are positive, the objective function is convex. If all these parameters are negative the function is concave. In these cases, standard optimization methods can be used to solve it [129]. However, if some of these parameters are negative, the others being positive, we deal with the difference of convex functions which is non-convex, requiring special optimization techniques. Therefore, in this section, the solution of the optimization problem, in the case of any possible values of CPU power consumption parameters is presented as follows.

Let us rewrite the objective function (4.11) with $E^r_{\text{cp},n}$ being a difference of convex functions:

$$(\mathbf{a}^\star, \mathbf{f}^\star) = \arg\min_{\mathbf{a},\mathbf{f}} \sum_{r\in\mathcal{R}_k} \sum_{n\in\mathcal{F}} a^r_n \left(\underbrace{E^{r+}_{\text{cp},n} - E^{r-}_{\text{cp},n}}_{E^r_{\text{cp},n}} + E^r_{\text{comm},n} \right), \tag{4.18}$$

where $E^{r+}_{\text{cp},n}$ is the sum components of $E^r_{\text{cp},n}$ with positive parameters $p_{n,q}$ and $E^{r-}_{\text{cp},n}$ is the negative of the sum components of $E^r_{\text{cp},n}$ with negative parameters $p_{n,q}$. We apply the SCA method [130–132] to approximate the possibly non-convex function by the series of convex ones. Since the objective function (4.18) is composed of differences of convex functions, the subtrahend $E^{r-}_{\text{cp},n}$ can be approximated with a linear function using the first-order Taylor series expansion at $\bar{\mathbf{f}} = \{\bar{f}_n\}$:

$$E^{r-}_{\text{cp},n}(f_n) \le E^{r-}_{\text{cp},n}(\bar{f}_n) + \left.\frac{\partial E^{r-}_{\text{cp},n}(f_n)}{\partial f_n}\right|_{f_n=\bar{f}_n} (f_n - \bar{f}_n) \triangleq \tilde{E}^{r-}_{\text{cp},n}. \tag{4.19}$$

After substituting $E^{r-}_{\text{cp},n}$ with $\tilde{E}^{r-}_{\text{cp},n}$ in (4.18), the objective function becomes:

$$(\mathbf{a}^\star, \mathbf{f}^\star) = \arg\min_{\mathbf{a},\mathbf{f}} \sum_{r\in\mathcal{R}_k} \sum_{n\in\mathcal{F}} a^r_n \left(\underbrace{E^{r+}_{\text{cp},n} - \tilde{E}^{r-}_{\text{cp},n}}_{\tilde{E}^r_{\text{cp},n}} + E^r_{\text{comm},n} \right). \tag{4.20}$$

This transformed optimization problem is convex for fixed allocation variables, thus it can be solved by employing primal and dual decomposition methods [129, 133]. The primal decomposition can be applied when the problem has a coupling variable such that, when fixed to some value, the rest of the optimization problem decouples into several subproblems. Thus, let us decompose the objective problem to:

$$(\mathbf{a}^\star, \mathbf{f}^\star) = \arg\min_{\mathbf{a}} \arg\min_{\mathbf{f}} \sum_{r\in\mathcal{R}_k} \sum_{n\in\mathcal{F}} a^r_n \left(\tilde{E}^r_{\text{cp},n} + E^r_{\text{comm},n} \right) \tag{4.21}$$

subject to (4.12) – (4.16). Now, according to (4.21), the solution to the optimization problem comes down to solving a two-step minimization problem. In the first step, the optimal CPU frequencies $\mathbf{f}^\star$ are determined for fixed allocation variables. First, let us define the auxiliary variables f_n^r determining the CPU frequencies of node n where request r is allocated. The relation between f_n^r and f_n is given by: $f_n = \sum_{r \in \mathcal{R}_k} a_n^r f_n^r$ while satisfying constraints (4.12) and (4.13). The optimal values of allocation variables $\mathbf{a}^\star$ are obtained in the second step based on the previously determined $\mathbf{f}^\star$ and constraints (4.12) – (4.13). Thus, we can now define the Lagrangian function of the subproblem for determining $\mathbf{f}^\star$:

$$\mathcal{L}\left(\mathbf{a}, \mathbf{f}, \boldsymbol{\mu}, \boldsymbol{\Phi}, \boldsymbol{\Psi}\right) = \sum_{r \in \mathcal{R}_k} \sum_{n \in \mathcal{F}} a_n^r \left(\tilde{E}_{\text{cp},n}^r + E_{\text{comm},n}^r \right) + \\ - \sum_{n \in \mathcal{F}} \Phi_n \left(f_{\text{min},n} - f_n^r \right) - \sum_{n \in \mathcal{F}} \Psi_n \left(f_n^r - f_{\text{max},n} \right) - \sum_{r \in \mathcal{R}_k} \mu^r \left(D_{\text{tot}}^r - D_{\text{max}}^r \right), \tag{4.22}$$

and the Lagrange dual problem:

$$\left(\mathbf{a}^\star, \mathbf{f}^\star, \boldsymbol{\mu}^\star, \boldsymbol{\Phi}^\star, \boldsymbol{\Psi}^\star\right) = \arg \min_{\boldsymbol{\mu}, \boldsymbol{\Phi}, \boldsymbol{\Psi} \geq 0} \arg \min_{\mathbf{a}} \arg \min_{\mathbf{f}} \mathcal{L}\left(\mathbf{a}, \mathbf{f}, \boldsymbol{\mu}, \boldsymbol{\Phi}, \boldsymbol{\Psi}\right) \tag{4.23}$$

subject to (4.12), (4.13), where $\boldsymbol{\mu} = \{\mu^r\}$, $\forall r \in \mathcal{R}_k$, $\mu^r \in \mathbb{R}^+$, $\boldsymbol{\Phi} = \{\Phi_n\}$, $\forall n \in \mathcal{F}$, $\Phi_n \in \mathbb{R}^+$ and $\boldsymbol{\Psi} = \{\Psi_n\}$, $\forall n \in \mathcal{F}$, $\Psi_n \in \mathbb{R}^+$ are the Lagrangian multipliers responsible for fulfilling constraints (4.14) and (4.15), respectively. The dual problem in (4.23) can be decomposed into a master problem and subproblems, and thus solved in an iterative manner. The allocation variables $\mathbf{a}$ and CPU frequencies $\mathbf{f}$ are obtained by solving subproblems and then the Lagrange multipliers $\boldsymbol{\mu}, \boldsymbol{\Phi}, \boldsymbol{\Psi}$ are updated by solving the master problem for the obtained frequencies. This process continues until convergence while satisfying constraints.

4.3.1 Solving the Subproblems

The primal problem is solved in two steps. First, the optimal values of the CPU frequencies $\mathbf{f}^\star$ for each request $r \in \mathcal{R}_k$ and node $n \in \mathcal{F}$ are obtained. Then, in the second step, the optimal values of the allocation variables $\mathbf{a}^\star$ are determined based on $\mathbf{f}^\star$. Thus, with Karush–Kuhn–Tucker (KKT) conditions, for fixed allocation variables $\mathbf{a}$, we can find the optimal CPU frequencies by taking the partial derivative of (4.22) with respect to f_n^r setting the gradient to 0:

$$\frac{\partial \mathcal{L}}{\partial f_n^r} = 0 \qquad \forall n \in \mathcal{F}, \forall r \in \mathcal{R}_k. \tag{4.24}$$

Due to the polynomial form of the objective function and constraint (4.14), there is no closed-form solution for the above equation. Therefore, the numerical method, e.g., the Newton method with the maximum number of iterations I_{num} has to be applied to solve it.

Vector $\mathbf{a}^\star$ can be obtained based on the optimal values of the CPU clock frequency determined in the first step by solving the following optimization problem.

$$\mathbf{a}^{\star} = \arg\max_{\mathbf{a}} \sum_{r\in\mathcal{R}_k}\sum_{n\in\mathcal{F}} a_n^r \left(\tilde{E}_{\text{cp},n}^{r}{}^{\star} + E_{\text{comm},n}^r\right) - \sum_{n\in\mathcal{F}} \Phi_n\left(f_{\text{min},n} - f_n^{r\star}\right) - \sum_{n\in\mathcal{F}} \Psi_n\left(f_n^{r\star} - f_{\text{max},n}\right) - \sum_{r\in\mathcal{R}_k} \mu^r\left(D_{\text{tot}}^{r}{}^{\star} - D_{\text{max}}^r\right), \quad (4.25)$$

subject to (4.12), (4.13), where $\tilde{E}_{\text{cp},n}^{r}{}^{\star} = \tilde{E}_{\text{cp},n}^r\left(f_n^{r\star}\right)$ and $D_{\text{tot}}^{r}{}^{\star} = D_{\text{tot}}^r\left(f_n^{r\star}\right)$. The optimization problem defined in (4.25) is the linear assignment problem, and can be solved by the Hungarian algorithm [134]. Let us define matrix $\mathbf{\Theta} = \left\{E_{\text{tot},n}^{r}{}^{\star}\right\}, \forall r \in \mathcal{R}_k$ and $\forall n \in \mathcal{C}$ with $|\mathcal{R}_k|$ rows and $|\mathcal{C}|$ columns, and matrix $\mathbf{\Lambda} = \left\{\tilde{E}_{\text{tot},n}^{r}{}^{\star}\right\}, \forall r \in \mathcal{R}_k$ and $\forall n \in \mathcal{F}$ with the same number of rows and $|\mathcal{F}|$ columns, where $\tilde{E}_{\text{tot},n}^{r}{}^{\star} = \tilde{E}_{\text{cp},n}^{r}{}^{\star} + E_{\text{comm},n}^r$. To reflect unlimited computational resources at each CN, we introduce matrix $\mathbf{\Omega} = \left[\mathbf{\Lambda}\ \mathbf{\Theta} \otimes \mathbf{1}_{1\times|\mathcal{R}_k|}\right]$, where $\otimes$ is the Kronecker tensor product while $\mathbf{1}_{1\times|\mathcal{R}_k|}$ is a vector of ones with one row and $|\mathcal{R}_k|$ columns. It means that the columns of $\mathbf{\Theta}$ are replicated $|\mathcal{R}_k|$ times and matrix $\mathbf{\Omega}$ has $|\mathcal{R}_k|$ rows and $|\mathcal{R}_k| \cdot |\mathcal{C}| + |\mathcal{F}|$ columns. E.g.,, in the case of three tasks $|\mathcal{R}_k| = 3$, two fog nodes $\mathcal{F} = \{1, 2\}$ and one cloud $\mathcal{C} = \{3\}$, the matrix $\mathbf{\Omega}$ is defined as follows:

$$\mathbf{\Omega} = \begin{bmatrix} \underbrace{\tilde{E}_{\text{tot},1}^{1}{}^{\star} & \tilde{E}_{\text{tot},2}^{1}{}^{\star}}_{E_{\text{tot},n}^{r}{}^{\star}\ \forall n\in\mathcal{F}} & \underbrace{E_{\text{tot},3}^{1}{}^{\star} & E_{\text{tot},3}^{1}{}^{\star} & E_{\text{tot},3}^{1}{}^{\star}}_{E_{\text{tot},n}^{r}{}^{\star}\ \forall n\in\mathcal{C}} \\ \tilde{E}_{\text{tot},1}^{2}{}^{\star} & \tilde{E}_{\text{tot},2}^{2}{}^{\star} & E_{\text{tot},3}^{2}{}^{\star} & E_{\text{tot},3}^{2}{}^{\star} & E_{\text{tot},3}^{2}{}^{\star} \\ \tilde{E}_{\text{tot},1}^{3}{}^{\star} & \tilde{E}_{\text{tot},2}^{3}{}^{\star} & E_{\text{tot},3}^{3}{}^{\star} & E_{\text{tot},3}^{3}{}^{\star} & E_{\text{tot},3}^{3}{}^{\star} \end{bmatrix} \quad (4.26)$$

Next, applying the Hungarian algorithm for matrix $\mathbf{\Omega}$ the matrix with binary values is determined, e.g.,:

$$\mathcal{H}(\mathbf{\Omega}) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.27)$$

The example above shows that the first task is computed in the second fog node while the second and the third task are computed in the CN. Thus, the optimal values of $a_n^{r\star}$ can be determined by:

$$a_n^{r^{\star}} = \begin{cases} \mathcal{H}(\mathbf{\Omega}(r,n)) & \text{for} \quad n \le |F| \\ \sum_{j=|F|+1}^{|\mathcal{F}|+|\mathcal{R}_k||\mathcal{C}|+1} \mathcal{H}(\mathbf{\Omega}(r,j)) & \text{for} \quad n > |F| \end{cases}. \quad (4.28)$$

4.3.2 Solving the Master Problem

We can fulfill constraints (4.14) and (4.15) by determining the search range of the optimal solution. Let $f_{\text{delay},n}^r$ denote the minimum value of f_n which satisfies the constraint (4.14) for request $r \in \mathcal{R}_k$ processed at node $n \in \mathcal{F}$. These can be obtained by solving the following equation:

$$D_{\text{tot},n}^r - D_{\text{max}}^r = 0, \quad \forall n \in \mathcal{F},\ \forall r \in \mathcal{R}_k. \quad (4.29)$$

Inserting $D_{\text{tot},n}^r$ from Eq. (4.9) (together with $D_{\text{cp},n}^r$ taken from Eq. (4.5)) into Eq. (4.29) we get:

$$f^r_{\text{delay},n} = \frac{L^r \theta^r}{s_n \left(D^r_{\max} - D^r_{\text{comm},n} - D^{r,l}_{\text{queue},n} \right)}. \tag{4.30}$$

If $f^r_{\text{delay},n} > f_{\max,n}$, the task r cannot be processed in FN n within the delay constraint. If $f^r_{\text{delay},n} \leq f_{\min,n}$, the minimum CPU clock frequency is kept at $f_{\min,n}$. Thus, if the obtained optimal clock frequency is in the range $f_n \in \langle \max \left\{ f_{\min,n}, f^r_{\text{delay},n} \right\}, f_{\max,n} \rangle$, the Lagrange multipliers in (4.22) and (4.25) simplify by setting $\Phi_n = 0$, $\Psi_n = 0$, $\forall n \in \mathcal{F}$ and $\mu^r = 0$, $\forall r \in \mathcal{R}_k$.

Finally, in this chapter, the algorithm called Energy-EFFicient Resource Allocation (EEFFRA) (Alg. 1) is proposed. It finds the solution (CPU clock frequencies and request allocation over the nodes) for total energy minimization with delay constraints, as discussed above. The computational complexity of the proposed algorithm results from the complexity of the Hungarian algorithm (line 8) and the iterative frequency finding procedure (lines 3-7). The complexity of the Hungarian algorithm is proportional to a cube of greater number: number of tasks or number of agents. In the proposed model, the requests (tasks) can be assigned to the fog nodes or the cloud (agents). Moreover, the cloud can process more than one request simultaneously. Therefore, in the model, we have $|\mathcal{R}_k|$ tasks which can be assigned to the $|\mathcal{F}| + |\mathcal{R}_k||\mathcal{C}|$ nodes, where $|\mathcal{R}_k||\mathcal{C}|$ represents the cloud nodes which can process more than one request.

Thus, in EEFFRA the complexity of the Hungarian algorithm equals $\mathcal{O}\left((|\mathcal{F}| + |\mathcal{R}_k||\mathcal{C}|)^3 \right)$. The second part of the computational complexity of the proposed algorithm results from the CPU frequency calculation. Let us observe that the CPU frequency has to be calculated for each node and each request i.e., $|\mathcal{R}_k|\,|\mathcal{N}|$ frequencies have to be determined. The main step of the proposed algorithm (line 4) determines the optimal CPU frequencies for a given approximation of the objective function which are then updated in the loop (line 5). This procedure is repeated until the termination conditions are met (line 7). Thus, the complexity of the CPU frequency calculation is equal to $\mathcal{O}\left(|\mathcal{R}_k|\,|\mathcal{N}|\, i_{\text{num}} i_{\text{sca}} \right)$, where i_{num} and i_{sca} are the numbers of iterations of the numerical method applied to solve (4.24) and the SCA method, respectively. Complexity of the entire EEFFRA algorithm is therefore equal to $\mathcal{O}\left((|\mathcal{F}| + |\mathcal{R}_k||\mathcal{C}|)^3 + |\mathcal{R}_k|\,|\mathcal{N}|\, i_{\text{num}} i_{\text{sca}} \right)$.

4.3.3 Low-complexity solution (LC-EEFFRA)

In the optimization solution called Low-Complexity EEFFRA (LC-EEFFRA), let us remove the Hungarian algorithm from EEFFRA leading to reduced computational complexity to $\mathcal{O}\left(|\mathcal{R}_k|\,|\mathcal{N}|\, i_{\text{num}} i_{\text{sca}} \right)$. The optimal values of frequency $\mathbf{f}^\star$ are determined in the same way as in Alg. 1 while the values of $\mathbf{a}^\star$ are obtained in a heuristic manner. In this heuristic approach, only a single request $r \in \mathcal{R}_k$ is considered at a time. It is allocated to node $n^\star$ where the energy consumption for processing r is the lowest, i.e., we find:

$$n^\star = \arg\min_n {E^r_{\text{tot},n}}^\star \; \forall r \in \mathcal{R}_k. \tag{4.31}$$

Values t_n are updated after allocation of each request to prevent multiple collisions of two or more requests at the same FN. The examination order of requests arriving at the same time is random to emulate a decentralized approach.

Algorithm 1 The EEFFRA in the fog computing networks.

1: **Inputs:** L^r, θ^r, o^r, $D^r_{\max}$ for $r \in \mathcal{R}_k$, $\{p_{n,0}, p_{n,3}, ..., p_{n,Q}\}$, $f_{\min,n}$, $f_{\max,n}$, s_n, d^n for $n \in \mathcal{F}$, γ^l_n, b^n_r for $r \in \mathcal{R}_k$ and $n \in \mathcal{F}$ and b_{back}, χ, maximum number of iterations I_{num}, I_{sca}, iteration index i_{sca}, maximum error ε and initial values of optimization variables $\bar{\mathbf{f}}$

2: **Outputs:** $E^r_{\text{tot},n}$ for $r \in \mathcal{R}_k$ and $n \in \mathcal{F}$, $\mathbf{f}^\star$, $\mathbf{a}^\star$

3: **repeat**

4: calculate $\mathbf{f}^\star$ by solving (4.24) in the range $f_n \in \langle \max\left\{f_{\min,n}, f^r_{\text{delay},n}\right\}, f_{\max,n}\rangle$ for $\Phi_n = 0$, $\Psi_n = 0, \forall n \in \mathcal{F}$ and $\mu^r = 0, \forall r \in \mathcal{R}_k$ using numerical method with max. I_{num} iterations

5: $\bar{\mathbf{f}} \leftarrow \mathbf{f}^\star$

6: $i_{\text{sca}} \leftarrow i_{\text{sca}} + 1$

7: **until** $\left|\bar{\mathbf{f}} - \bar{\mathbf{f}}\right| \leq \varepsilon$ **or** $i_{\text{sca}} = I_{\text{sca}}$

8: calculate $\mathbf{a}^\star$ using the Hungarian method for the matrix $\boldsymbol{\Omega}$ and (4.28)

9: $E^r_{\text{tot},n} \leftarrow \tilde{E}^r_{\text{tot},n}$ for $r \in \mathcal{R}_k$ and $n \in \mathcal{F}$

4.4 Results

Results obtained from computer simulations are presented in this section. Let us consider a network with $|\mathcal{F}| = 10$ FNs and $|\mathcal{C}| = 1$ CN. Simulation parameters are summarized in Tab. 4.2. The process of generating requests for simulations is as follows. At time $T_k \in \mathcal{T}$ there appear between 5 and 10 (uniform distribution) new computational requests. The value T_k is generated at a random delay after previous time instance T_{k-1}. The difference $T_k - T_{k-1}$ is chosen to be a random variable of exponential distribution with average value 50 ms (intensity = 20 s^{-1}). The requests have randomly (with uniform distribution) assigned values of their parameters (size, arithmetic intensity, delay requirement) in ranges shown in Tab. 4.2.

It is assumed that each FN uses a single Intel Core i5-2500K as its CPU. Data relating frequency, voltage, and power consumption of i5-2500K is taken from [135] and fit into Eq. (4.2) adopted from [98]. The resulting power consumption and energy efficiency are plotted in Fig. 4.2. These figures show that power consumption increases faster-than-linearly with operating frequency and that the frequency with the highest energy efficiency is around 2.6 GHz. The cloud CPUs are parameterized according to the *Intel Xeon Phi* family commonly used in computer clusters [128, 136] characterized with $s = 32$ FLOP/cycle [124], and run at a constant frequency 1.5 GHz. Transmission parameters are analogous to those used in [13]. If not stated differently, simulations for each data point are obtained over 550 time instances T_k. Results from the first 50 instances are discarded. Random number generator seeds are kept the same for each value of swept parameters for a fair comparison of results.

Our solutions, i.e., EEFFRA and LC-EEFFRA, are compared with three reference methods. The first method, called *Cloud Only*, processes all requests in the cloud. The second method, called *Fog Only*, processes all requests in the FNs. The FNs' CPU frequencies and requests-to-nodes assignments are determined using LC-EEFFRA. Finally, the third method, called *Fog Simple*, processes requests in the same FN that these requests arrived at. Still, it uses optimal FNs' CPU

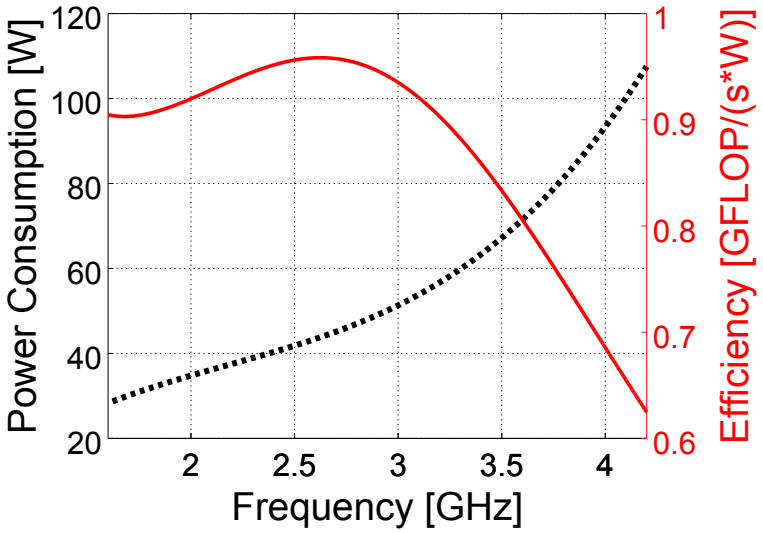


Figure 4.2: Power consumption and energy efficiency of Intel Core i5-2500K vs. CPU frequency.

Table 4.2: Simulation parameters used in Chapter 4.

Symbol	Value/Range	Symbol	Value/Range	Symbol	Value/Range
Requests, $r \in \mathcal{R}_k$					
L^r	[1,10] MB	θ^r	[1,100] FLOP/bit	o^r	[0, 0.5]
$D^r_{\max}$	[100, 1000] ms	$\lvert\mathcal{R}_k\rvert$	[5,10]	$\overline{T_k - T_{k-1}}$	50 ms
Number of nodes					
$\lvert\mathcal{F}\rvert$	10	$\lvert\mathcal{C}\rvert$	1		
Computations in the fog [98, 124, 135], $n \in \mathcal{F}$					
s_n	16 FLOP/cycle	$p_{n,3}$, $p_{n,2}$	5.222, 34.256	$f_{\min,n}$	1.6 GHz
Q	3	$p_{n,1}$, $p_{n,0}$	88.594, -47.152	$f_{\max,n}$	4.2 GHz
Computations in the cloud [124, 128], $n \in \mathcal{C}$					
f_n	1.5 GHz	s_n	32 FLOP/cycle		
Transmission [92, 118, 119]					
d^n, $n \in \mathcal{C}$	2000 km	χ	7.5 µs/km	b_{back}	1 Gbps
b^n_r, $n \in \mathcal{F}$	1 Gbps	γ^l_n, $n \in \mathcal{F}$	0.3 nJ/(bit·hop)	γ^l_n, $n \in \mathcal{C}$	10 nJ/bit

frequencies determined using LC-EEFFRA. Simulations are performed using MATLAB.

4.4.1 Convergence of Algorithms and Optimality of Solution

EEFFRA utilizes SCA for finding optimal operating frequencies and the Hungarian algorithm for assigning requests to nodes. The Hungarian algorithm is guaranteed to find the optimum in polynomial time [134]. SCA is guaranteed to converge [137]. To show the SCA convergence rate, we plot normalized energy costs resulting from offloading requests depending on the maximum number of algorithm iterations in Fig. 4.3. Apart from SCA iterations I_{sca}, we also vary the maximum number of iterations I_{num} used to find the optimum CPU frequencies in Alg. 1. For clarity of convergence analysis, we assume that there is no cloud, i.e., $|\mathcal{C}| = 0$. The operating frequency of the cloud is not adjusted (not a variable) and therefore does not influence the analysis. Normalization is obtained by plotting the relative difference between the cost achieved by EEFFRA and the

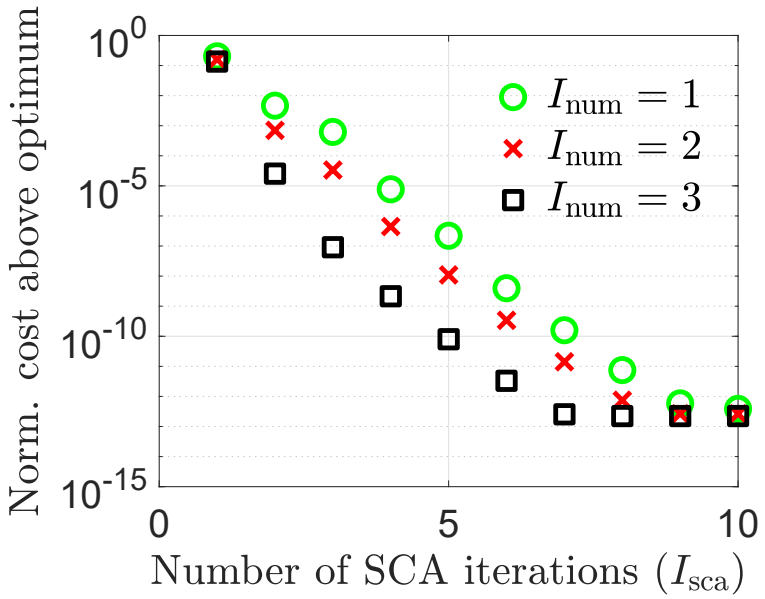


Figure 4.3: Convergence of solutions found by EEFFRA to the optimum with the number of iterations.

optimal cost found by solving the original problem without SCA. Since the degree of polynomial modeling CPU power consumption in these simulations equals 3, the optimal frequencies $f_n^{r\star}$ can be found analytically for each request $r \in \mathcal{R}_k$ and node $n \in \mathcal{F}$ which result in the lowest cost $E^r_{\text{cp},n}$ while fulfilling (4.14). Since the first derivative of $E^r_{\text{cp},n}$ over f_n from (4.17) is continuous everywhere except at singularity at $f_n = 0$, and has at most 3 real roots, the optimal frequency $f_n \in \langle \max\left\{f_{\min,n}, f^r_{\text{delay},n}\right\}, f_{\max,n}\rangle$ is either obtained for the endpoint of this interval or for one of the roots of $\frac{d}{df_n} E^r_{\text{cp},n}(f_n)$. The lowest value $E^r_{\text{cp},n}$ for these frequencies determines the optimal frequency $f_n^{r\star}$. The solution is continued as described in Section 4.3 from Eq. (4.25). It is visible in Fig. 4.3 that EEFFRA converges both quickly and to values close to the optimal ones.

4.4.2 Impact of Computational Energy Efficiency of the Cloud

First, we vary the values of the computational efficiency of the CN in the range [0.5, 5.0] GFLOPS/W (Giga Floating Point Operations per Second (GFLOPS) per Watt, as of 2020, the median value for 500 of the most powerful commercially available computer clusters is 2.962 GFLOPS/W [128]). The average energy costs per successfully processed request are shown in Fig. 4.4. At low computational efficiency of the cloud, policies utilizing only nodes in the fog tier (*Fog Simple*, *Fog Only*) perform similarly to those utilizing both the fog and the cloud. *Cloud only* approach has the highest energy consumption at low efficiency of the cloud (up to around 1.3 GFLOPS/W). Above that level *Cloud only* is characterized with lower energy consumption than *Fog Simple* and *Fog Only* solutions and similar to EEFFRA and LC-EEFFRA solutions as under these parameters it is the most efficient to process most requests in the cloud. EEFFRA is slightly more efficient than LC-EEFFRA at higher cloud efficiency values. The percentage of requests which were unable to be processed using each of the offloading policies is the following: The *Fog Simple* solution where FNs cannot "share" computational requests between themselves has the highest ratio of rejected

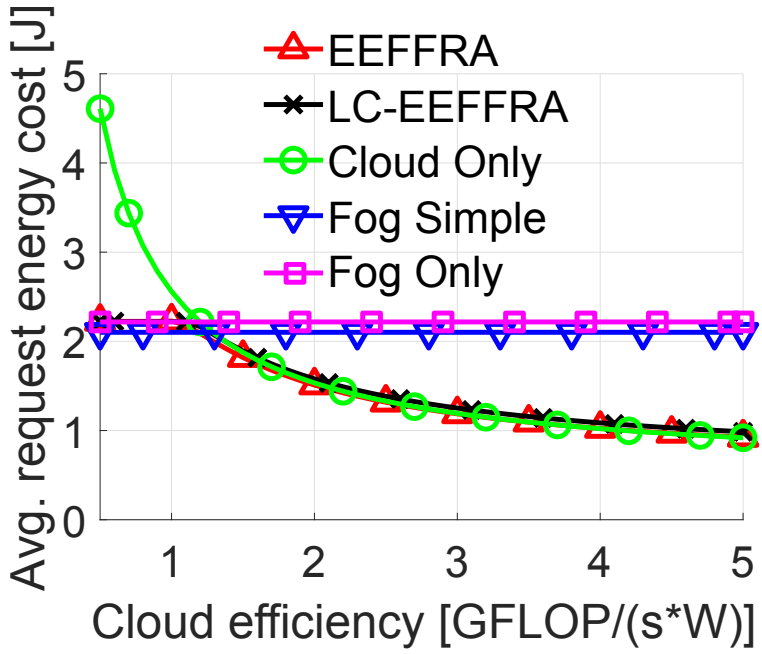
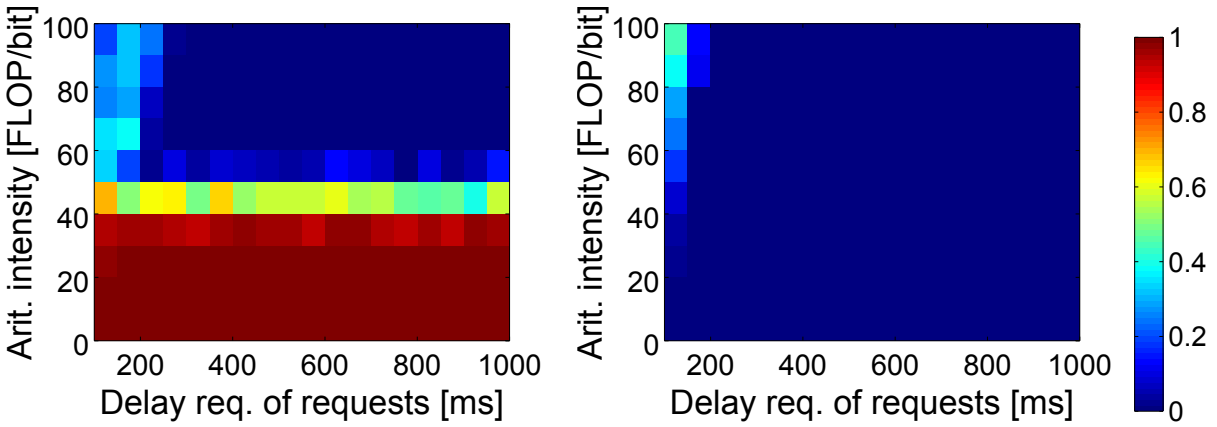


Figure 4.4: Influence of cloud energy efficiency on the average energy cost for chosen policies.

requests (8.2%), while only 1.4%-1.7% requests (percentage varies depending on cloud efficiency – lower for higher efficiency) are rejected by both proposed solutions utilizing both fog and cloud (EEFFRA and LC-EEFFRA). *Fog Only* and *Cloud Only* have rejection rates of 1.9% and 4.3% respectively. Requests which are rejected tend to have larger sizes and higher arithmetic intensities (as shown later in Figs. 4.5b and 4.8). It "artificially" decreases the average-per-request cost of methods with higher rejection rates in Fig. 4.4, e.g., causing *Fog Simple* to show lower average cost than *Fog Only*).

Let us examine more closely where EEFFRA chooses to offload computational requests and what parameters impact these decisions. Fig. 4.5 shows histograms of parameters characterizing offloaded requests obtained after running simulations for 2000 T_k instances. Fig. 4.5a and Fig. 4.5b show the probabilities of requests being processed in the fog tier of the network and those rejected due to too low delay requirement at cloud efficiency of 1.3 GFLOPS/W. A similar histogram for requests processed in the cloud would be superfluous as probabilities for results processed in the fog, in the cloud, and those rejected sum to 1. Unsurprisingly, Fig. 4.5b shows that results with strict latency requirements are less likely to be successfully processed in time. In Fig. 4.5a up to 40% of high arithmetical intensity tasks with delay requirements of around 200 ms are processed in the fog tier as a result of the cloud being unable to fulfill these requirements. In Fig. 4.5a one can see a "threshold" between 40 and 50 FLOP/bit below which requests are chosen by EEFFRA to be served by the FNs rather than the cloud. Similar histograms plotted for other values of cloud efficiencies show that this threshold increases with a less efficient cloud and decreases with a more efficient cloud. In particular, for efficiencies below 1.0 GFLOPS/W, all requests are processed in Fog. On the other hand, even for infinitely high efficiencies of the cloud around 20% of tasks remain processed in the fog tier (low-intensity ones for which the cost of transmission to the cloud outweighs computational costs in the fog and those with strict delay requirements). In Fig. 4.5a up to 40% of high arithmetical intensity tasks with delay requirements of around 200 ms are processed



(a) Requests processed in FNs

(b) Rejected requests.

Figure 4.5: Histograms of requests at 1.3 GFLOPS/W cloud efficiency. Results of EEFFRA.

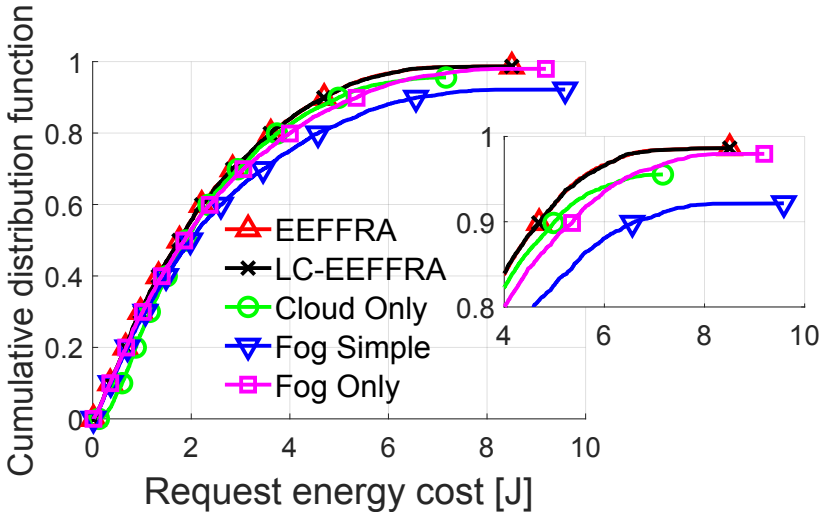
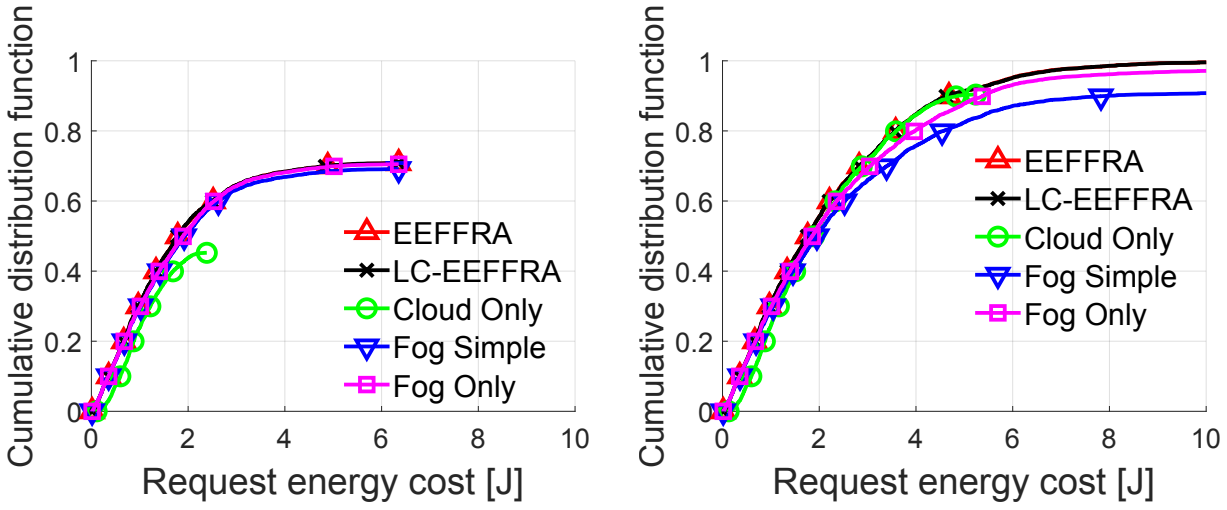


Figure 4.6: CDFs of request processing energy cost at cloud efficiency of 1.3 GFLOPS/W. Comparison of different policies.

in the fog tier as a result of the cloud being unable to fulfill these requirements.

To better illustrate differences in requests allocation policies, we plot Cumulative Distribution Function (CDF) of energy cost spent on a single request. Energy spent on rejected requests is assumed to be infinite for the purpose of CDF plots. Such results can be seen in Fig. 4.6. First, it is visible that utilization of both fog and cloud tiers of the network yields significantly better results than utilizing nodes in only one tier. The proposed EEFFRA and LC-EEFFRA provide the lowest required energy cost for each percentile of the CDFs. All methods do not reach 1 on the y-axis, i.e., some requests cannot be processed within a given delay budget. Our methods achieve the lowest rejection rate as shown in the inset of Fig. 4.6.



(a) Delay req.: 100 ms

(b) Delay req.: 200 ms

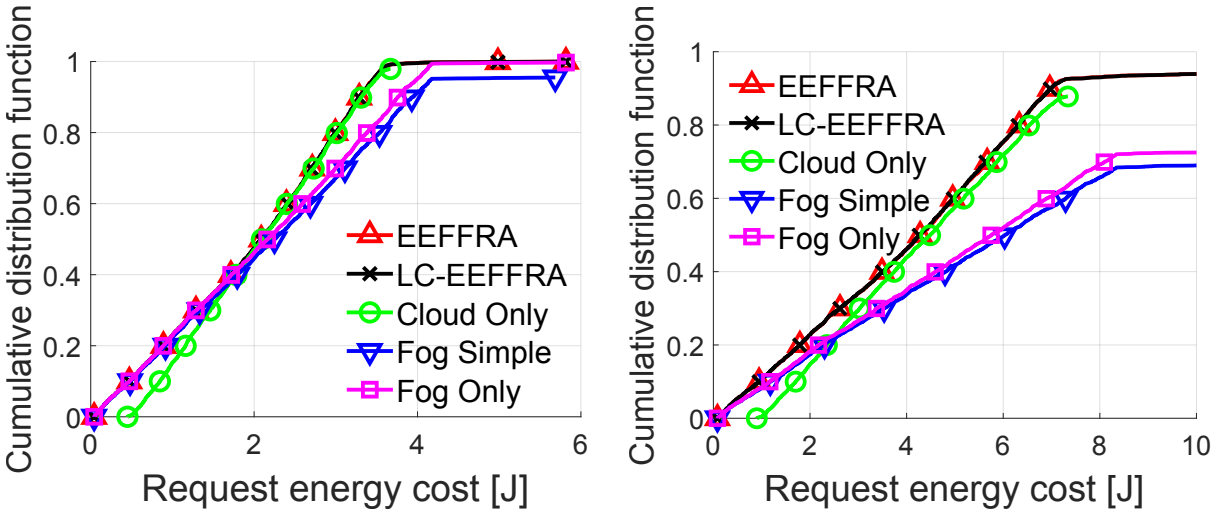
Figure 4.7: CDFs of request processing energy cost – influence of delay requirement of requests.

4.4.3 Impact of Delay Requirements and Size of Requests on the Offloading Decisions

Let us see how the energy consumption and the percentage of rejected requests change with the size and the delay requirement of computation requests. The energy efficiency of the cloud is set to 1.3 GFLOPS/W and parameter sweeps for other parameters are performed. All other parameters are generated according to Tab. 4.2.

Fig. 4.7 plots CDFs of energy consumption costs of processing single requests with delay requirements: 100 ms (Fig. 4.7a) and 200 ms (Fig. 4.7b). At the required delay of 100 ms, all methods have high rejection rates, with *Cloud Only* being clearly the worst-suited for low-latency applications. The differences between the rest of the methods are minor – the requests with such low delay requirements either can or cannot be solved in time at the receiving FN and the ability of nodes to transmit tasks between themselves does not improve performance. With a 200 ms, the differences between approaches become more profound. Utilizing both fog and cloud (EEFFRA, LC-EEFFRA) gives the lowest rejection rates and energy costs. *Fog Simple* meanwhile has the worst performance.

Fig. 4.8 plots CDFs of energy consumption costs of processing single requests with sizes 5 MB, (Fig. 4.8a) and 10 MB (Fig. 4.8b). EEFFRA, LC-EEFFRA, and *Cloud Only* show the highest energy efficiency for 5 MB requests. However, *Cloud Only* fails to process a small number of requests within a given delay requirement, while *Fog Only*, EEFFRA, and LC-EEFFRA process all requests successfully. Fig. 4.8c shows that EEFFRA and LC-EEFFRA achieve the lowest energy costs and rejection rates for 10 MB requests. It is worth observing that for both request sizes *Cloud Only* has the highest energy costs up to at least 20-th percentile (caused by energy spent for transmission), but for higher percentiles (influenced by requests with higher arithmetical intensities) its costs are lower than that of either *Fog Only* or *Fog Simple*.



(a) Size: 5 MB.

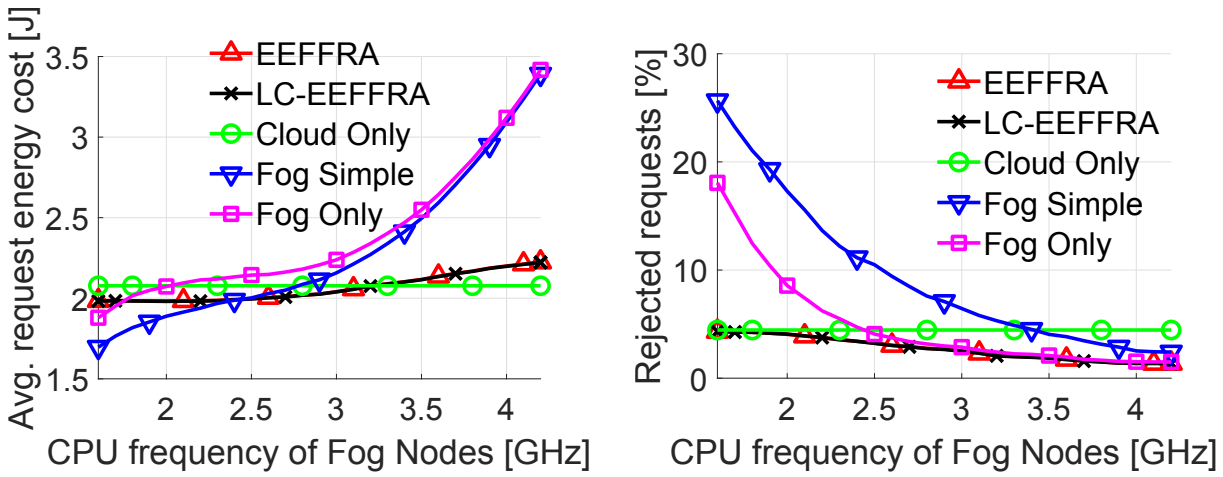
(b) Size: 10 MB.

Figure 4.8: CDFs of request processing energy cost – influence of size of requests.

4.4.4 Impact of CPU Frequency of Fog Nodes

In previous sections, it was assumed that FNs can dynamically adjust their operating frequency (and voltage) to minimize energy consumption while satisfying delay requirements. Let us assume that all FNs utilize the same, fixed CPU frequency. Fig. 4.9 shows the average energy cost and percentage of rejected requests plotted as a function of this fixed frequency (swept between 1.6 and 4.2 GHz with a 0.1 GHz step). Results for *Cloud Only* are constant as no requests are processed in FNs. *Fog Simple* and *Fog Only* methods have high rejection rates at low frequencies. Meanwhile EEFFRA and Low Complexity (LC)-EEFFRA have the lowest rejection rates while also having the lowest (considering the rejected requests are not taken into account by this metric) energy costs. As the frequency of FNs increases the rejection rates decline and average energy cost increases for all methods utilizing FNs. However, this effect is considerably smaller for EEFFRA and LC-EEFFRA (utilizing resources in both fog and cloud tiers) than for *Fog Simple* and *Fog Only*.

Let us compare yhe efficiency of the network employing EEFFRA with and without Dynamic Voltage and Frequency Scaling (DVFS). As shown in Fig. 4.9 the possibility to send requests to the cloud diminishes the impact of FNs' operating frequency on energy costs and rejection rate. Therefore, to focus on the differences, Fig. 4.10 shows the results of simulations for a network with 10 FNs and no connection to the cloud. We compare CDFs of energy costs per request achieved utilizing DVFS with the following fixed frequencies of FNs: 1.6 GHz (minimum), 2.6063 GHz (optimal frequency for maximizing energy efficiency as seen in Fig. 4.2, later referred to as 2.6 GHz), and 4.2 GHz (maximum). The range of possible arithmetic intensities of requests is increased to [1, 500] to make the requests highly variable in terms of required computations speed while the mean time between sets of requests is increased ($\overline{T_k - T_{k-1}}$ = 500 ms). Rejection rates are increasing with decreasing fixed FN frequency. On the other hand, 4.2 GHz has the highest energy cost. EEFFRA utilizing DVFS manages to maintain the lowest energy cost for every percentile.



(a) Average energy cost.

(b) Rejected requests.

Figure 4.9: Influence of fixed CPU frequency of FNs.

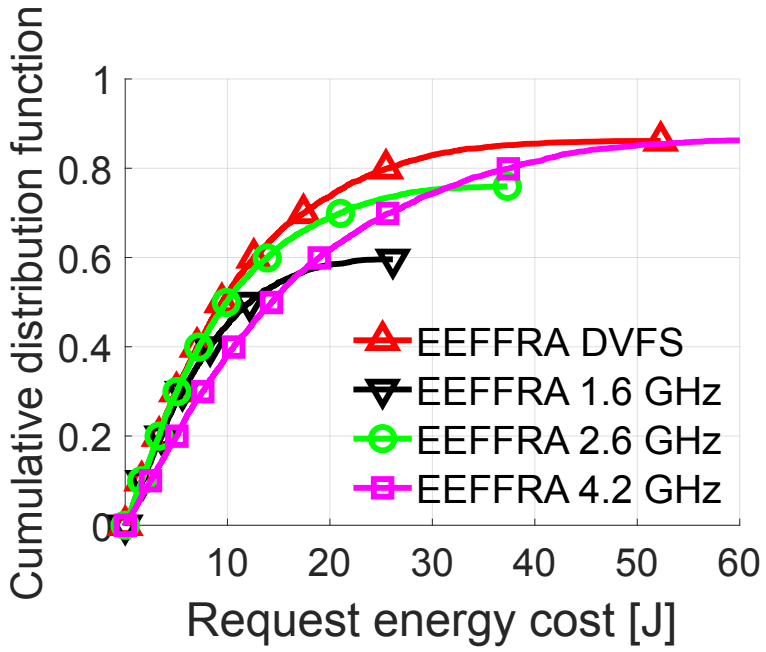


Figure 4.10: CDFs of request processing energy cost – comparison of FNs working at fixed frequencies and utilizing DVFS. Parameters: $\overline{T_k - T_{k-1}}$ = 500 ms, $\theta^r \in [1, 500]$ FLOP/bit.

4.5 Chapter Summary

In this chapter, the thesis author formulated the optimization problem of minimizing the energy consumption in the fog computing network while maintaining the latency constraints. This energy consumption is assumed to result from both transportation (wired transmission) and the processing of offloaded computational tasks (computation requests originating from the end-users). The latency and energy consumption models and their parameters are based on real-life computing and networking equipment product data sheets and measurements.

The author of this thesis formulated the optimization problem of selecting the place (node) of task execution along with its CPU frequency to minimize the energy consumption in the network while satisfying the latency constraint. The proposed EEFFRA algorithm solves the formulated optimization problem using its successive approximations for adjusting clock frequencies of CPUs in fog nodes. A sub-optimal, lower complexity solution LC-EEFFRA, which does not require coordinated decision-making, is also proposed. Results of the computer simulation show that both EEFFRA and LC-EEFFRA can significantly reduce average energy cost and the number of rejected computational requests by distributing the workload between fog and cloud nodes.

CDF of the average energy cost per task (request), i.e., the probability that the energy cost is lower than the argument, is always higher for both EEFFRA and LC-EEFFRA than for the cloud-only or fog-only task delegation. This is particularly significant in the case of more demanding requests (latency required below 100 ms, and the size of request of 10 MB) and moderate cloud efficiency (of 1.3 GFLOPS/W). Thus, the proposed solutions can be seen as promising alternatives for managing fog computing networks, as they allow for flexible utilization of communication and computing resources.

5 Optimization of Energy Consumption in Things, Fog and Cloud Tiers

This chapter is an extension of Chapter 4. It expands the fog network model by including the things-fog transmission. The optimization space includes possible places (FNs and CNs) of the execution of the tasks, and their CPU frequencies, as well as wireless connections between MDs and FNs. Thus, there is a higher degree of freedom in selecting FN to execute tasks delegated by MDs. This is because a single MD can be connected to multiple FNs (while the considerations presented in Chapter 4 were agnostic to underlying wireless connections of MDs), and FNs are interconnected and can relay the tasks to each other.

Thus, compared to Chapter 4, the energy-consumption optimization problem formulated below includes a new set of related parameters and variables, and the optimization solutions space is also expanded.

The work presented in this chapter has been published by the author of this thesis in [9].

5.1 Network Model

In the bottom tier of the network, there is a set $\mathcal{M}$ of MDs (e.g., smartphones) with specific computational requests. It is assumed that serving these tasks requires offloading them to one of the FNs or CNs, constituting the second and the third tier, respectively. The MDs cannot process these tasks on their own because of energy or computational limitations. The MDs send computational requests using wireless transmission to one of the nearby FNs. As shown in 5.1, FNs are located at Base Stations (BSs) or Access Points (APs), close to the end users. Then, each task can be processed either in one of the FNs out of set $\mathcal{F}$ or in the cloud tier (set $\mathcal{C}$ of CNs). Unlike MDs, nodes in the fog and cloud tiers of the network are interconnected with wireline communication technology.

The model shown in this work extends the one used in [10].

5.1.1 Computational Requests

Let $\mathcal{T}$ be a numbered set $\{T_1, T_2, ..., T_{|\mathcal{T}|}\}$ of all time instances at which MDs offload computational requests. Let $\mathcal{R}_k$ be a set of all requests that MDs try to offload at time T_k. The following parameters characterize each computing request $r \in \mathcal{R}_k$:

- MD $m^r \in \mathcal{M}$ which offloads the task,
- size L^r in bits,
- arithmetic intensity θ^r in FLOP/bit,

- ratio o^r of the size of the result of the processed task r to the size of the offloaded task r,
- maximum tolerated delay $D^r_{\max}$.

Let us define a binary variable a^r_n that shows where the request is computed, i.e., a^r_n equals 1 if $r \in \mathcal{R}_k$ is computed at node $n \in \mathcal{F} \cup \mathcal{C}$, and 0 otherwise. Similarly, let us define a binary variable w^r_l that indicates if request r is wirelessly transmitted from MD m^r to FN $n \in \mathcal{F}$.

5.1.2 Energy Consumption

The energy consumption model is divided into two parts: computation (processing of data) and communication (transmission of data). Energy E^r_{cp} spent on computing request $r \in \mathcal{R}_k$ equals:

$$E^r_{\text{cp}} = \sum_{n \in \mathcal{F} \cup \mathcal{C}} a^r_n E^r_{\text{cp},n} = \sum_{n \in \mathcal{F} \cup \mathcal{C}} a^r_n \frac{L^r \theta^r}{\beta_n}, \tag{5.1}$$

where $E^r_{\text{cp},n}$ is the energy spent on computing request $r \in \mathcal{R}_k$ at node $n \in \mathcal{F} \cup \mathcal{C}$ and β_n is the computational efficiency of node $n \in \mathcal{F} \cup \mathcal{C}$ given in GFLOPS per watt [138]. For CNs, constant CPU clock frequency f_n and efficiency β_n are assumed. For FNs, β_n depends on CPU frequency f_n of node $n \in \mathcal{F}$, number s_n of FLOPs performed within a single clock cycle of CPU [124], and on power consumption P_n of CPU. β_n is obtained by modeling P_n as a polynomial function of f_n using four parameters $p_{n,3}$, $p_{n,2}$, $p_{n,1}$, and $p_{n,0}$ derived from [98]:

$$\beta_n = \frac{f_n s_n}{P_n} = \frac{f_n s_n}{p_{n,3} f_n^3 + p_{n,2} f_n^2 + p_{n,1} f_n + p_{n,0}}. \tag{5.2}$$

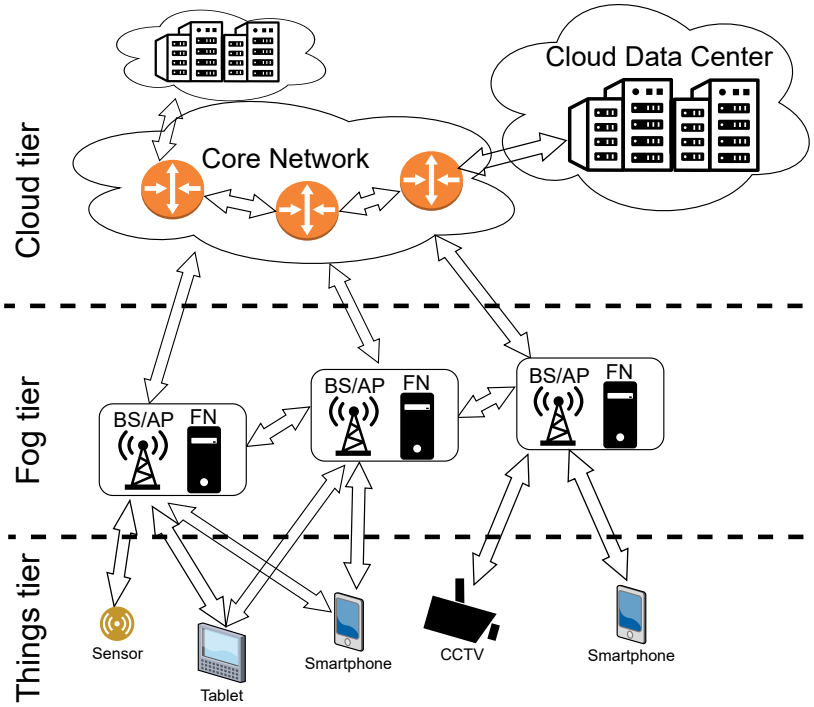


Figure 5.1: Fog network architecture.

Table 5.1: The notation used for modeling the network and defining the optimization problem in Chapter 5

Symbol	Description
$\mathcal{T}$	set $\{T_1, ..., T_{\lvert\mathcal{T}\rvert}\}$ of all considered time instances, when one or more computational requests arrive
$\mathcal{M}$	set of all Mobile/End Devices
$\mathcal{F}$	set of all Fog Nodes
$\mathcal{C}$	set of all Cloud Nodes
$\mathcal{R}_k$	set of all computational requests arriving at $T_k \in \mathcal{T}$
L^r	size of request $r \in \mathcal{R}_k$
θ^r	computational complexity of request $r \in \mathcal{R}_k$
m^r	MD which offloads the request $r \in \mathcal{R}_k$
o^r	output-to-input data size ratio of request $r \in \mathcal{R}_k$
D^r_{max}	maximum tolerated delay requirement for request $r \in \mathcal{R}_k$
T_k	time at which request $r \in \mathcal{R}_k$ arrives in the network, $k \in \{1, ..., \lvert\mathcal{T}\rvert\}$
γ^x_y	energy-per-bit cost of transmitting data between nodes x and y
s_n	number of FLOPs performed per single clock cycle at node $n \in \mathcal{N}$
b^x_y	link bitrate between nodes x and y
d^n	fiberline distance to CN $n \in \mathcal{C}$
χ	a parameter characterizing delay depending on distance
$f_{\text{min},n}$	minimum clock frequency of node $n \in \mathcal{N}$
$f_{\text{max},n}$	maximum clock frequency of node $n \in \mathcal{N}$
$p_{n,0}$, $p_{n,1}$, $p_{n,2}$, $p_{n,3}$	parameters of the power model of CPUs installed in node $n \in \mathcal{N}$
t_n	time at which node $n \in \mathcal{N}$ finishes computing its last task
a^r_n	variable showing whether request $r \in \mathcal{R}_k$ is computed at node $n \in \mathcal{N}$, $a^r_n \in \{0, 1\}$
w^r_l	variable showing whether request $r \in \mathcal{R}_k$ is transmitted wirelessly to node $l \in \mathcal{F}$, $w^r_l \in \{0, 1\}$
f_n	clock frequency of node $n \in \mathcal{N}$, $f_{\text{min},n} \leq f_n \leq f_{\text{max},n}$
β_n	energy efficiency (GFLOPS per Watt) characterizing node $n \in \mathcal{N}$
P_n	power consumption related to computations at node $n \in \mathcal{N}$
E^r_{tot}	energy spent on transmission and processing of request $r \in \mathcal{R}_k$
E^r_{cp}	energy spent in the network on processing request $r \in \mathcal{R}_k$
E^r_{comm}	energy spent on transmission of request $r \in \mathcal{R}_k$
E^r_{wl}, E^r_{wd}	energy spent on wireless/wired transmission of request $r \in \mathcal{R}_k$
$E^{r,x}_{\text{comm},y}$	energy cost for transmission of request $r \in \mathcal{R}_k$ between nodes x and y
$E^r_{\text{cp},n}$	energy cost of processing request $r \in \mathcal{R}_k$ at node $n \in \mathcal{N}$

This representation provides flexibility to cover various models of CPUs. The clock frequency f_n must lie within the range of possible frequencies of CPU in node $n \in \mathcal{F}$, i.e., $f_{\text{min},n} \leq f_n \leq f_{\text{max},n}$.

The energy spent on the transmission of request $r \in \mathcal{R}_k$ is a sum of energies resulting from

Table 5.1: The notation used for modeling the network and defining the optimization problem.

Symbol	Description
D^r_{tot}	total delay of request $r \in \mathcal{R}_k$
D^r_{comm}	delay caused by transmitting request $r \in \mathcal{R}_k$
$D^r_{\text{wl}}/D^r_{\text{wd}}$	wireless/wired delay of request $r \in \mathcal{R}_k$
$D^{r,x}_{\text{comm},y}$	delay of transmission of request $r \in \mathcal{R}_k$ between nodes x and y
$D^{r,m^r,l}_{\text{ul},n}$	uplink delay of transmitting request $r \in \mathcal{R}_k$ to node $n \in \mathcal{F}$, provided that $w^r_l = 1$
D^r_{queue}	queuing delay of request $r \in \mathcal{R}_k$
$D^{r,l}_{\text{queue},n}$	queuing delay of request $r \in \mathcal{R}_k$ at node $n \in \mathcal{N}$, provided that $w^r_l = 1$
D^r_{cp}	computational delay caused by processing request $r \in \mathcal{R}_k$
$D^r_{\text{cp},n}$	computational delay caused by processing request $r \in \mathcal{R}_k$ at node $n \in \mathcal{N}$

wireless (E^r_{wl}) and wired (E^r_{wd}) communication:

$$E^r_{\text{comm}} = E^r_{\text{wl}} + E^r_{\text{wd}}. \tag{5.3}$$

The energy spent on wireless transmission of request $r \in \mathcal{R}_k$ equals:

$$E^r_{\text{wl}} = \sum_{l \in \mathcal{F}} w^r_l E^{r,m^r}_{\text{comm},l} = \sum_{l \in \mathcal{F}} w^r_l L^r (1 + o^r) \gamma^{m^r}_l, \tag{5.4}$$

where $E^{r,m^r}_{\text{comm},l}$ is the energy required to transmit request $r \in \mathcal{R}_k$ from MD $m^r \in \mathcal{M}$ to FN $l \in \mathcal{F}$ and return the calculation result in reverse direction while $\gamma^{m^r}_l$ is the energy-per-bit cost of this transmission. $L^r o^r$ is the size (in bits) of results transmitted back to MD m^r.

The energy spent on wired transmission of request $r \in \mathcal{R}_k$ equals:

$$E^r_{\text{wd}} = \sum_{l \in \mathcal{F}} w^r_l \sum_{n \in \mathcal{F} \cup \mathcal{C}} a^r_n E^{r,l}_{\text{comm},n} = \sum_{l \in \mathcal{F}} w^r_l \sum_{n \in \mathcal{F} \cup \mathcal{C}} a^r_n L^r (1 + o^r) \gamma^l_n, \tag{5.5}$$

where $E^{r,l}_{\text{comm},n}$ is the energy required to transmit request $r \in \mathcal{R}_k$ between FN $l \in \mathcal{F}$ and node $n \in \mathcal{F} \cup \mathcal{C}$, while γ^l_n is the energy-per-bit cost of this transmission. Energy-per-bit cost can be derived from [117], where the power consumption of networking equipment increases linearly with load starting from idle power. This relation can also be seen in measurements of core routers [92, 120]. There is no wired communication between nodes if the request is processed at the same node to which it is wirelessly transmitted by the MD, i.e., $\forall l \in \mathcal{F}\ \gamma^l_l = 0$.

The total energy spent on offloading request $r \in \mathcal{R}_k$ is given by:

$$E^r_{\text{tot}} = E^r_{\text{cp}} + E^r_{\text{wl}} + E^r_{\text{wd}}. \tag{5.6}$$

5.1.3 Delay

Three components form the delay model: communication, processing, and queuing. The delay D^r_{cp} caused by computing request $r \in \mathcal{R}_k$ equals:

$$D^r_{\text{cp}} = \sum_{n \in \mathcal{F} \cup \mathcal{C}} a^r_n D^r_{\text{cp},n} = \sum_{n \in \mathcal{F} \cup \mathcal{C}} a^r_n \frac{L^r \theta^r}{f_n s_n}, \tag{5.7}$$

where $D^r_{\text{cp},n}$ is the time required to compute request $r \in \mathcal{R}_k$ at node $n \in \mathcal{F} \cup \mathcal{C}$.

The delay caused by communication can be further subdivided into wireless (D^r_{wl}) and wired (D^r_{wd}) delay:

$$D^r_{\text{comm}} = D^r_{\text{wl}} + D^r_{\text{wd}}. \tag{5.8}$$

The delay caused by wireless transmission of request $r \in \mathcal{R}_k$ equals:

$$D^r_{\text{wl}} = \sum_{l \in \mathcal{F}} w^r_l D^{r,m^r}_{\text{comm},l} = \sum_{l \in \mathcal{F}} w^r_l \frac{L^r(1+o^r)}{b^{m^r}_l}, \tag{5.9}$$

where $D^{r,m^r}_{\text{comm},l}$ is the time required to transmit request $r \in \mathcal{R}_k$ between MD $m^r \in \mathcal{M}$ and FN $l \in \mathcal{F}$ while $b^{m^r}_l$ is the bitrate of this transmission between FN l and MD m^r.

The delay caused by wired transmission of request $r \in \mathcal{R}_k$ equals:

$$D^r_{\text{wd}} = \sum_{l \in \mathcal{F}} w^r_l \sum_{n \in \mathcal{F} \cup \mathcal{C}} a^r_n D^{r,l}_{\text{comm},n}, \tag{5.10}$$

where $D^{r,l}_{\text{comm},n}$ is the time required to transmit request $r \in \mathcal{R}_k$ between FN $l \in \mathcal{F}$ and node $n \in \mathcal{F} \cup \mathcal{C}$. The model for calculation of $D^{r,l}_{\text{comm},n}$ differs depending on node n being an FN or a CN. It is assumed that cloud data centers are located away from the rest of the network (hundreds or even thousands of kilometers away) which requires the distance-related delay to be modeled. Delay caused by transmitting request $r \in \mathcal{R}_k$ between (to and from) FN $l \in \mathcal{F}$ and cloud node $n \in \mathcal{C}$ is equal:

$$D^{r,l}_{\text{comm},n} = \frac{L^r(1+o^r)}{b^l_n} + d^n \cdot \chi, \tag{5.11}$$

where b^l_n is the link bitrate in the backhaul and backbone network between nodes l and n, while d^n is the fiberline distance to CN $n \in \mathcal{C}$. The parameter χ indicates the rate at which delay increases with distance d^n [118].

For transmission between FNs, it is assumed that the delay caused by the distance between them ($d^n \cdot \chi$ in Eq. (5.11)) is negligible – well below 1 ms as we use a value of 7.5 μs/km for parameter χ [118] – and therefore we can ignore it. Delay caused by communication between FN $l \in \mathcal{F}$ and $n \in \mathcal{F}$ for request $r \in \mathcal{R}_k$ equals:

$$D^{r,l}_{\text{comm},n} = \frac{L^r(1+o^r)}{b^l_n}. \tag{5.12}$$

The special case is when the request r is received wirelessly at FN n and the same node is used for processing. In this case, no wired communication delay is expected, i.e., $D^{r,n}_{\text{comm},n} = 0, \forall n \in \mathcal{F}$.

Even more significant differences can be observed while modeling queuing delays for requests processed in the fog tier and in the cloud tier of the network. It stems from the fact that clouds are assumed to have huge (practically infinite) computational resources with parallel-computing capabilities, and there is no need for queuing multiple requests served by the CN $n \in \mathcal{C}$. They can be processed simultaneously. Meanwhile, if multiple requests are sent to the same FN $n \in \mathcal{F}$ for processing in a short time span, additional delays may occur due to congestion of computational requests (an arriving request cannot be processed until processing of all the previous requests has been completed). Let us define a scheduling variable $t_n \in \mathbb{R}^+$ to represent the point in time when

the last request scheduled at FN $n \in \mathcal{F}$ is finished processing. The queuing delay of request $r \in \mathcal{R}_k$, transmitted wirelessly to node $l \in \mathcal{F}$, for computations being carried at node $n \in \mathcal{F}$ equals:

$$D^{r,l}_{\text{queue},n} = \max(0, t_n - T_k - D^{r,m^r,l}_{\text{ul},n}), \tag{5.13}$$

where $D^{r,m^r,l}_{\text{ul},n} = \frac{1}{1+o^r}(D^{r,m^r}_{\text{comm},l} + D^{r,l}_{\text{comm},n})$ is the uplink delay of transmitting request r to node n through FN l. $D^{r,l}_{\text{queue},n}$ has nonzero values when $t_n > T_k + D^{r,m^r,l}_{\text{ul},n}$. In such a case, the request r which arrives at node n at time $T_k + D^{r,m^r,l}_{\text{ul},n}$. It is kept in a queue until time t_n, when processing of another request (or requests) ends. For each node $n \in \mathcal{C}$, $D^{r,l}_{\text{queue},n}$ always equals zero – due to the parallel processing powers of the cloud, each request may be computed right away, regardless of how many requests are already being processed. Queuing delay of request $r \in \mathcal{R}_k$ is equal:

$$D^r_{\text{queue}} = \sum_{l \in \mathcal{F}} w^r_l \sum_{n \in \mathcal{F} \cup \mathcal{C}} a^r_n D^{r,l}_{\text{queue},n}. \tag{5.14}$$

Finally, the total delay of processing request $r \in \mathcal{R}_k$ equals the sum of delays related to computation, transmission, and queuing:

$$D^r_{\text{tot}} = D^r_{\text{cp}} + D^r_{\text{comm}} + D^r_{\text{queue}}. \tag{5.15}$$

5.1.4 Updating Scheduling Variables in the Fog

Since no requests are processed when the optimization starts, we can set $t_n = 0, \forall n \in \mathcal{F}$. Then, for each $T_k \in \mathcal{T}$, after allocations a^r_n and w^r_l are determined, the times t_n are updated for every $n \in \mathcal{F}$ according to when computation of requests offloaded to FNs are scheduled to finish:

$$t_n := \max(t_n, T_k + \sum_{r \in \mathcal{R}_k} \sum_{l \in \mathcal{F}} a^r_n w^r_l (D^{r,m^r,l}_{\text{ul},n} + D^{r,l}_{\text{queue},n} + D^r_{\text{cp},n})). \tag{5.16}$$

5.2 Optimization Problem

Once again, the defined problem seeks to minimize the total energy cost of offloading all requests that enter the network at time $\mathcal{T}_k$, that is to find:

$$(\mathbf{a}^\star, \mathbf{w}^\star, \mathbf{f}^\star) = \arg\min_{\mathbf{a},\mathbf{f},\mathbf{w}} \sum_{r \in \mathcal{R}} E^r_{\text{tot}}, \tag{5.17}$$

subject to:

$$\sum_{n \in \mathcal{F} \cup \mathcal{C}} a^r_n = 1 \qquad \forall r \in \mathcal{R}_k, \tag{5.18}$$

$$\sum_{r \in \mathcal{R}_k} a^r_n \leq 1, \qquad \forall n \in \mathcal{F}, \tag{5.19}$$

$$\sum_{l \in \mathcal{F}} w^r_l = 1 \qquad \forall r \in \mathcal{R}_k, \tag{5.20}$$

$$D^r_{\text{tot}} \leq D^r_{\text{max}}, \qquad \forall r \in \mathcal{R}_k, \tag{5.21}$$

$$f_{\min,n} \leq f_n \leq f_{\max,n},\ f_n \in \mathbb{R} \qquad \forall n \in \mathcal{F}, \tag{5.22}$$
$$a_n^r \in \{0,1\}, \qquad \forall r \in \mathcal{R}_k, \forall n \in \mathcal{F} \cup \mathcal{C}, \tag{5.23}$$
$$w_l^r \in \{0,1\}, \qquad \forall r \in \mathcal{R}_k, \forall l \in \mathcal{F}, \tag{5.24}$$

where $\mathbf{a}^\star = \{a_n^{r\star}\}$, $\mathbf{w}^\star = \{w_l^{r\star}\}$ and $\mathbf{f}^\star = \{f_n^\star\}$ are the optimal values of allocation variables a_n^r and w_l^r, and CPU clock frequencies f_n, respectively. Constraints (5.18) guarantee that each request must be processed at exactly one FN or CN. Constraints (5.19) restrict that no more than a single request can be processed at a given FN at a given time. Constraints (5.20) guarantee that for each request, a single FN will be used for wireless connectivity. Constraints (5.21) guarantee that the total delay must not be greater than the maximum acceptable one. Constraints (5.22) show the lower and upper bounds of CPU frequency at each FN. Finally, according to constraints (5.23) and (5.24), decision variables a_n^r and w_l^r take only binary values.

There exist sets of requests $\mathcal{R}_k$ for which the optimization cannot be solved (e.g., there is no feasible allocation of requests so that each request is processed (5.18) while fulfilling its delay requirement (5.21)). In such a scenario, we can decide to reject requests for which (5.21) cannot be satisfied rather than ending the optimization without finding a solution (which would translate into rejecting all requests $\mathcal{R}_k$). The remaining requests (set $\mathcal{R}_k \setminus \mathcal{R}'_k$, where $\mathcal{R}'_k$ denotes the set of rejected requests) are then subjected to the optimization.

5.3 Problem Solution

In this section, the author of this thesis provides a step-by-step solution to the optimization problem. In short, first, the minimum operating frequencies at which the delay requirements of offloaded requests are met are found. Then, the optimal operating frequencies, which minimize energy consumption spent on computations for given combinations of nodes and requests, are found. At this point, combinations that cannot satisfy delay requirements are known. Then, the nodes to which wireless transmission energy costs are the lowest are found. Finally, we can assign requests to nodes for computing to minimize the total energy consumption. This linear assignment problem is solved with the Hungarian algorithm [134, 139]. The notation used in the proposed solution is summarized in Tab. 5.2. Letters in superscript are used throughout this chapter as upper indices, not exponents, e.g., m^r does **not** denote m to the power of r.

5.3.1 Auxiliary Variables

Let us define the auxiliary variable $f_{n,l}^r$ as the CPU frequency of node $n \in \mathcal{F} \cup \mathcal{C}$ where request $r \in \mathcal{R}_k$ is allocated while node $l \in \mathcal{F}$ is the node to which r is wirelessly transmitted ($w_l^r = 1$). The relation between $f_{n,l}^r$ and f_n is given by $f_n = \sum_{r \in \mathcal{R}_k} \sum_{l \in \mathcal{F}} a_n^r w_{l,n}^r f_{n,l}^r$. Similarly, $w_{l,n}^r$ determines which node $l \in \mathcal{F}$ request $r \in \mathcal{R}_k$ is wirelessly transmitted to provided that it is allocated to $n \in \mathcal{F} \cup \mathcal{C}$ ($a_n^r = 1$) and $w_l^r = \sum_{n \in \mathcal{F} \cup \mathcal{C}} w_{l,n}^r$. Moreover, let $D_{\text{tot},n}^{r,l}$ be the total delay of request $r \in \mathcal{R}_k$ provided that it is computed at node $n \in \mathcal{F} \cup \mathcal{C}$ ($a_n^r = 1$) and node $l \in \mathcal{F}$ be the node to which r is wirelessly transmitted ($w_l^r = 1$).

Table 5.2: Additional notation used in the problem solution in Chapter 5

Symbol	Description
$w^r_{l,n}$	variable showing whether request $r \in \mathcal{R}_k$ is transmitted wirelessly to node $l \in \mathcal{F}$, provided that $a^r_n = 1$
$f^r_{n,l}$	clock frequency of node $n \in \mathcal{N}$, provided that $a^r_n = 1$ and $w^r_l = 1$, $r \in \mathcal{R}_k$
$D^{r,l}_{\text{tot},n}$	total delay of $r \in \mathcal{R}_k$, provided that $a^r_n = 1$ and $w^r_l = 1$
$D^{r,l}_{\text{cp},n}$	computational delay of $r \in \mathcal{R}_k$, provided that $a^r_n = 1$ and $w^r_l = 1$
$E^r_{\text{cp},n,l}$	energy spent on processing of request $r \in \mathcal{R}_k$, provided that $a^r_n = 1$ and $w^r_l = 1$
$\mathcal{R}'_k$	set of requests rejected due to delay requirements
$\hat{\mathcal{R}}_k$	set of not rejected requests, $\hat{\mathcal{R}}_k = \mathcal{R}_k \setminus \mathcal{R}'_k$
$w^r_{l,n}$	variable showing whether request $r \in \mathcal{R}_k$ is transmitted wirelessly to node $l \in \mathcal{F}$, provided that $a^r_n = 1$
$f^r_{n,l}$	clock frequency of node $n \in \mathcal{N}$, provided that $a^r_n = 1$ and $w^r_l = 1$, $r \in \mathcal{R}_k$
$D^{r,l}_{\text{tot},n}$	total delay of $r \in \mathcal{R}_k$, provided that $a^r_n = 1$ and $w^r_l = 1$
$D^{r,l}_{\text{cp},n}$	computational delay of $r \in \mathcal{R}_k$, provided that $a^r_n = 1$ and $w^r_l = 1$
$E^r_{\text{cp},n,l}$	energy spent on processing of request $r \in \mathcal{R}_k$, provided that $a^r_n = 1$ and $w^r_l = 1$
$\mathcal{R}'_k$	set of requests rejected due to delay requirements
$\hat{\mathcal{R}}_k$	set of not rejected requests, $\hat{\mathcal{R}}_k = \mathcal{R}_k \setminus \mathcal{R}'_k$

5.3.2 Finding Optimal Frequencies

Let us rewrite (5.17) by expanding E^r_{tot} into parts caused by computations ($E^r_{\text{cp},n}$), wireless transmission ($E^{r,m^r}_{\text{comm},l}$, between MD m^r and node l) and wired transmission ($E^{r,l}_{\text{comm},n}$, between nodes l and n):

$$(\mathbf{a}^\star, \mathbf{w}^\star, \mathbf{f}^\star) = \arg\min_{\mathbf{a},\mathbf{w},\mathbf{f}} \sum_{r\in\mathcal{R}_k} \sum_{l\in\mathcal{F}} \sum_{n\in\mathcal{F}\cup\mathcal{C}} a^r_n w^r_l \left(E^r_{\text{cp},n} + E^{r,m^r}_{\text{comm},l} + E^{r,l}_{\text{comm},n} \right). \tag{5.25}$$

Out of these three parts, $E^r_{\text{cp},n}$ is the only one that depends on frequencies f_n. The goal of this step is to find ${f^r_{n,l}}^\star$, i.e., values of f_n which minimize $E^r_{\text{cp},n}$ for all possible values of a^r_n and w^r_l. The only constraints that depend on values of $f^r_{n,l}$ are (5.21) and (5.22).

The minimum values of $f^r_{n,l}$ which satisfy Constraints (5.21) can be obtained by solving the inequality $D^{r,l}_{\text{tot},n} \leq D^r_{\text{max}}$.

$$D^{r,l}_{\text{cp},n} + D^{r,m^r}_{\text{comm},l} + D^{r,l}_{\text{comm},n} + D^{r,l}_{\text{queue},n} \leq D^r_{\text{max}} \tag{5.26}$$

$$\frac{L^r\theta^r}{s_n f^r_{n,l}} + D^{r,m^r}_{\text{comm},l} + D^{r,l}_{\text{comm},n} + D^{r,l}_{\text{queue},n} \leq D^r_{\max} \tag{5.27}$$

$$f^r_{n,l} \geq \frac{L^r\theta^r}{s_n\left(D^r_{\max} - D^{r,m^r}_{\text{comm},l} - D^{r,l}_{\text{comm},n} - D^{r,l}_{\text{queue},n}\right)} \triangleq f^r_{\min,n,l} \tag{5.28}$$

Let us rewrite $E^r_{\text{cp},n}$ as a function of f_n based on (5.1) and (5.2).

$$E^r_{\text{cp},n}(f_n) = \frac{L^r\theta^r(p_{n,3}f_n^3 + p_{n,2}f_n^2 + p_{n,1}f_n + p_{n,0})}{f_n s_n} \tag{5.29}$$

Its derivative with respect to f_n equals:

$$E^r_{\text{cp},n}{}'(f_n) = \frac{L^r\theta^r}{s_n}\frac{(2p_{n,3}f_n^3 + p_{n,2}f_n^2 - p_{n,0})}{f_n^2}. \tag{5.30}$$

The function $E^r_{\text{cp},n}(f_n)$ is continuous and differentiable for positive f_n (the only discontinuity is at $f_n = 0$). Therefore, its extrema in a given interval can only be found at the bounds of this interval or for points at which the derivative equals zero. $E^r_{\text{cp},n}{}'(f_n)$ has a cubic function in the numerator, so it has at most three real roots.

Now, we can determine ${f^r_{n,l}}^\star$ for $r \in \mathcal{R}_k$, $n \in \mathcal{F}$, $w \in \mathcal{F}$ by finding the minimum of $E^r_{\text{cp},n}(f_n)$ in the interval $[\max(f^r_{\min,n,l}, f_{\min,n}), f_{\max,n}]$. The corresponding minimum energy costs are as follows:

$${E^r_{\text{cp},n,l}}^\star = E^r_{\text{cp},n}({f^r_{n,l}}^\star). \tag{5.31}$$

For values $r \in \mathcal{R}_k$, $n \in \mathcal{F}$, $w \in \mathcal{F}$ for which $f^r_{\min,n,l} > f_{\max,n}$, constraints (5.21) and (5.22) cannot both be satisfied, so we set ${E^r_{\text{cp},n,l}}^\star$ to infinity. For computations in clouds $n \in \mathcal{C}$, we do not optimize the frequency f_n ($f_n = \text{const.}$, ${E^r_{\text{cp},n,l}}^\star = E^r_{\text{cp},n}(f_n)$). For values $r \in \mathcal{R}_k$, $n \in \mathcal{C}$, $w \in \mathcal{F}$ for which $f^r_{\min,n,l} > f_n$, constraint (5.21) cannot be satisfied, i.e., we set ${E^r_{\text{cp},n,l}}^\star$ to infinity.

Each request $r \in \mathcal{R}_k$ for which the following occurs:

$${E^r_{\text{cp},n,l}}^\star = \infty, \forall n \forall l \tag{5.32}$$

cannot be fully processed within their delay requirements regardless of chosen computation/transmission nodes. All such requests are therefore rejected. The remaining optimization is performed over $\hat{\mathcal{R}}_k = \mathcal{R}_k \setminus \mathcal{R}'_k$, where $\mathcal{R}'_k$ is the set of rejected requests.

5.3.3 Transmission Allocation

The auxiliary matrix $\mathbf{w_n}^\star = \{{w^r_{l,n}}^\star\}$ can be obtained. For each task $r \in \mathcal{R}_k$ and each computing node $n \in \mathcal{F} \cup \mathcal{C}$, the goal is to choose node $l \in \mathcal{F}$, which minimizes the sum of energy spent on computations (calculated and optimized in the previous step) and transmission (depending directly on $w^r_{l,n}$), i.e., to find:

$$\mathbf{w_n}^\star = \arg\min_{\mathbf{w_n}} \sum_{l\in\mathcal{F}} w^r_{l,n}\left({E^r_{\text{cp},n,l}}^\star + E^{r,m^r}_{\text{comm},l} + E^{r,l}_{\text{comm},n}\right), \tag{5.33}$$

while satisfying (5.20) and (5.24). This is equivalent to finding nodes l, which minimize the expression $\left({E^r_{\text{cp},n,l}}^\star + E^{r,m^r}_{\text{comm},l} + E^{r,l}_{\text{comm},n}\right)$.

5.3.4 Computation Allocation

The vector $\mathbf{a}^\star$ can now be obtained by solving the simplified problem:

$$\mathbf{a}^\star = \arg\min_{\mathbf{a}} \sum_{r\in\mathcal{R}_k} \sum_{l\in\mathcal{F}} \sum_{n\in\mathcal{F}\cup\mathcal{C}} a_n^r w_{l,n}^{r\,\star} \left(E_{\text{cp},n,l}^{r\,\star} + E_{\text{comm},l}^{r,m^r} + E_{\text{comm},n}^{r,l} \right), \tag{5.34}$$

subject to (5.18), (5.19) and (5.23). This corresponds to the linear assignment problem [134]—each request $r \in \hat{\mathcal{R}}_k$ is assigned to one and only one node $n \in \mathcal{F} \cup \mathcal{C}$. The cost matrix has $\left|\hat{\mathcal{R}}_k\right|$ rows and $|\mathcal{F}| + \left|\hat{\mathcal{R}}_k\right| \cdot |\mathcal{C}|$ columns. The columns representing processing at FN are used once as each of them can serve one request at a time while the columns representing processing at CN are multiplied to ensure that multiple requests can be assigned to them simultaneously. The Hungarian algorithm [134, 139] is used to solve this problem.

5.4 Results

Results obtained from computer (MATLAB) simulations and their setup are presented in this section. While the main goal is to serve all the incoming requests within allowed latency constraints with minimum energy, requests that failed to be served are set with virtually infinite consumed energy. This facilitates a fair comparison of various request allocation strategies using only the distribution of energy consumption spent per offloaded request. Therefore, medians, percentiles, and CDF are chosen as evaluation metrics.

For the purpose of computing medians and other percentiles in this section, the energy costs related to rejected requests are equal to positive infinity—such an approach (as well as using other fixed values or omitting them entirely) has a considerably larger impact on the averages. Medians and percentiles avoid bias that unserved requests have with respect to average values.

5.4.1 Scenario Overview

Let us consider a network with $|\mathcal{F}| = 10$ FNs and $|\mathcal{C}| = 1$ CN. Simulation parameters are summarized in Tab. 5.3. Fig. 5.2 shows a connection diagram between these FNs and the cloud. The examined environment represents a commercial facility such as an airport, where the end users (MD) want to have their requests processed. Moreover, Fig. 5.2 presents three examples of requests being calculated: (i) in the same FN as the utilized AP, (ii) being calculated in another FN and (iii) being offloaded to the cloud. Appropriate values of binary variables a_n^r and w_l^r are presented in Fig. 5.2. The following assumptions have been made for the considered scenario.

Requests—between 5 and 10 new computational requests with uniform distribution at time $T_k \in \mathcal{T}$ appear. These requests appear at random locations within the area of the examined network (with uniform distribution in both dimensions). The value T_k is generated as a random delay after the previous time instance T_{k-1}. The difference $T_k - T_{k-1}$ is chosen to be a random variable of exponential distribution with an average value of 200 ms. The requests have randomly assigned values of their parameters (size, arithmetic intensity, delay requirement) in ranges shown in Tab. 5.3 with uniform distribution.

Computations—each FN has computational resources and a frequency–power relationship of a single Intel Core i5-2500K as its CPU. Data relating frequency, voltage and power consumption

Table 5.3: Simulation parameters used in Chapter 5

Symbol	Value/Range	Symbol	Value/Range
Requests , $r \in \mathcal{R}_k$			
L^r	[1, 5] MB	θ^r	[1, 500] FLOP/bit
o^r	[0.01, 0.2]	D^r_{max}	[500, 3000] ms
$\lvert\mathcal{R}_k\rvert$	[5, 10]	$\overline{T_k - T_{k-1}}$	200 ms
Computations in fog [98, 124, 135], $n \in \mathcal{F}$			
$p_{n,3}$, $p_{n,2}$	5.222, 34.256	$p_{n,1}$, $p_{n,0}$	88.594, −47.152
$f_{\text{min},n}$	1.6 GHz	$f_{\text{max},n}$	4.2 GHz
s_n	16 FLOP/cycle		
Computations in cloud [124, 128], $n \in \mathcal{C}$			
f_n	1.5 GHz	s_n	32 FLOP/cycle
Wired Transmission [92, 118, 140]			
d^n, $n \in \mathcal{C}$	2000 km	χ	7500 ns/km
b^l_n, $n \in \mathcal{C}$	10 Gbps	b^l_n, $n \in \mathcal{F}$	1 Gbps
γ^l_n, $n \in \mathcal{F}$	$\{2, 3\} \times 2$ nJ/(bit)	γ^l_n, $n \in \mathcal{C}$	12 nJ/bit
Wireless Transmission [75, 141, 142]			
$\gamma^{m^r}_l$, $l \in \mathcal{F}$, $m^r \in \mathcal{M}$	depends on rate and path loss	$b^{m^r}_l$, $l \in \mathcal{F}$, $m^r \in \mathcal{M}$	{0, 6.5, 13, 18.5, 26, 39, 52, 58.5, 65} Mbps

of i5-2500K are taken from [135] and inserted into Eq. (5.2) adapted from [98] to obtain values for $p_{n,3}$, $p_{n,2}$, $p_{n,1}$ and $p_{n,0}$. The parameter s equals 16 for this CPU [124]. The resulting computational efficiency β is the highest (0.9586 GFLOPS/W) at frequency $f = 2.6063$ GHz.

To simulate a scenario with varying computational efficiencies of nodes, we multiply the resulting computational efficiency (5.2) by random values from the range $[0.5, 1.5]$ generated independently for each node $n \in \mathcal{F}$.

As for the computational capability of the cloud, its CPUs are parameterized according to the *Intel Xeon Phi* family commonly used in computer clusters [128, 136] run at constant frequency $f = 1.5$ GHz characterized with $s = 32$ [124].

Wireless transmission—the power consumption model of the wireless transmission is based on [75] and depends on the data rate and path loss. We use values derived for ASUS USB-N10 WiFi card. The path loss values are determined using the model from Section 3.1 of [142] for a commercial area and frequency closest to 2.4 GHz (20 dB for frequency 2.1 GHz). The wireless link uses a maximum available rate that depends on the minimum sensitivity specified in Section 19.3.19.2 of [141] for a given modulation and coding scheme. It ranges from 6.5 Mbps (BPSK, 1/2) at -82 dBm to 65 Mbps (64-QAM, 5/6) at -64 dBm. The energy-per-bit cost $\gamma^{m^r}_l$ is obtained by dividing the power by the wireless link data rate.

Wired transmission—in order to derive energy-per-bit cost of transmitting requests from one

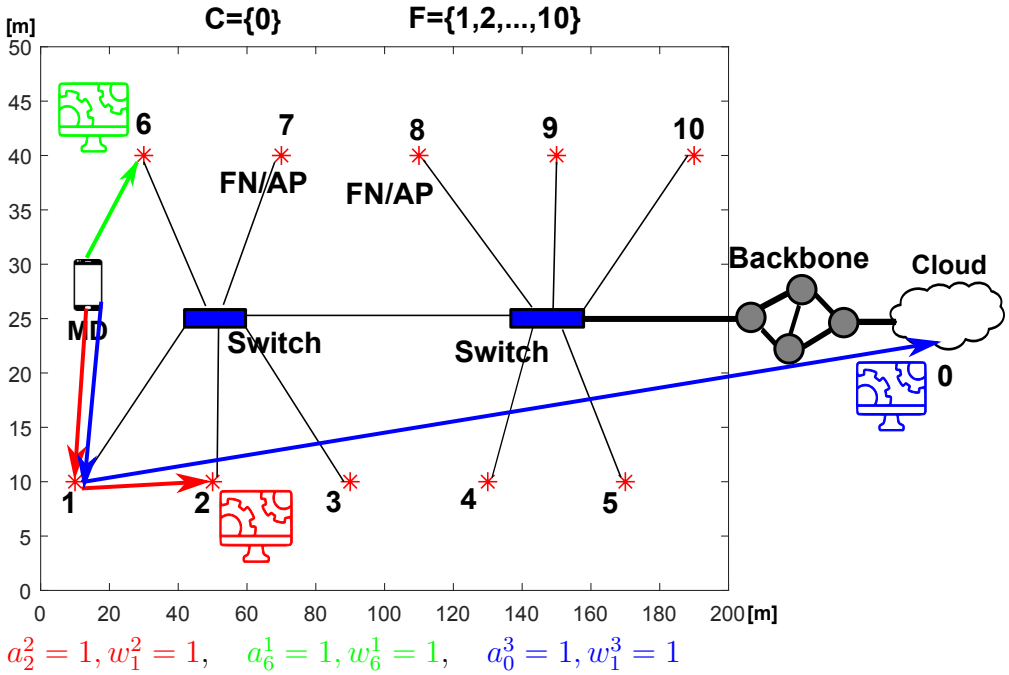


Figure 5.2: Diagram of the considered network composed of 10 FNs and a cloud with three examples of request allocations.

node to another (i.e., γ_n^l from $l \in \mathcal{F}$ to $n \in \mathcal{F} \cup \mathcal{C}$), we need to add costs induced in all devices through which it flows. For the power consumption of a single networking device, the linear model from [117] is used. It includes idle power P_{idle} and active power that scales with load C (in bits/second) by parameter γ (in Joules/bit):

$$P = P_{idle} + C\frac{P_{max} - P_{idle}}{C_{max}} = P_{idle} + C\gamma, \tag{5.35}$$

where P_{max} denotes maximum power consumption and C_{max} denotes maximum load. Energy-per-bit cost of transmitting data γ_n^l is equal to the sum of γ parameters of all network devices through which the data flows between nodes l and n. In this work, it is assumed that $\gamma_n^l = \gamma_l^n$. It is assumed for the connections between FN that they are connected with 1 G Ethernet. The power consumption of Ethernet switches is set according to [93, 140]. Each switch can serve up to 6 FN on the LAN side with 1 Gbps links (star topology) and can be connected to the 10 G EPON on the WAN side. Cost-per-bit of transmission through these switches is equal to 2 nJ/bit (82 W at 1 Gbps throughput, 80 W with no traffic). The configuration can be seen in Fig. 5.2 showing 10 FNs connected with 2 switches.

For the connection between the fog tier of the network and the cloud, it is assumed that the data flow through multiple nodes. Olbrich et al. [118] use geographically locatable nodes (over 250 nodes around the globe) to derive multiple path characteristics. Their results show that the RTT of a packet is, on average, 1.5 times longer than an estimation based only on fiberline distance (the speed of light in optic fiber $\approx 2 \times 10^8$m/s, in vacuum $c \approx 3 \times 10^8$m/s). The measured RTT has a slope of 7.5 µs/km. Lrt us assign this 7.5 µs/km value to parameter χ. The CN is assumed to be located 2000 km away from the rest of the network. It is estimated that the energy-per-bit cost of transmitting data through the backbone network to the Cloud is equal to 12.66 nJ/bit based

on 12 Juniper T1600 routers—each with cost-per-bit equal 1.03 nJ/bit [13, 120] and a 10G EPON gateway with 0.3 nJ/bit cost [119]. While there is other equipment through which the data flow within the core network (e.g., optical amplifiers), the value 12.66 nJ/bit is chosen to represent the whole energy spent on transmission. Therefore, $\gamma_n^l = 12.66 + \{2, 3\} \times 2$ nJ/bit for $n \in \mathcal{C}$ (2 or 3 depending on the logical distance between l and the switch with the WAN connection).

5.4.2 Baseline/Suboptimal Solutions

To test the effectiveness of the proposed algorithm (*Full Optimization*, shortened on plots to *Full Optim*), let us compare it with four simpler task allocation methods. A summary of these methods is shown in Tab. 5.4.

Table 5.4: Comparison of the examined algorithms.

Name	Limitation	Optimization Variables
Full Optimization	None	Computing allocation—**a**, transmission allocation—**w**, computing frequency—**f**
Exhaustive Search	None	**a**, **w**, **f**
Cloud Only	$\sum_{n\in\mathcal{C}} a_n^r = 1, \quad \forall r \in \mathcal{R}_k$	**w**, **a** (if there are multiple Cloud Nodes)
No Migrate	$\sum_{l=n\in\mathcal{F}} w_l^r a_n^r = 1, \quad \forall r \in \mathcal{R}_k$	**a** interdependently on **w**, **f**
Closest Wireless	$w_l^r = \arg\min_{w_l^r} E_{\text{wl}}^r, \quad \forall r \in \mathcal{R}_k$	**a**, **f**

Exhaustive Search—all possible variations of allocations are verified. While this baseline approach finds the optimal solution, its running time scales exponentially with the number of requests. The optimal frequencies of CPU are calculated as in *Full Optimization*.

Cloud Only—all requests are transmitted to and processed in the cloud tier of the network. The optimal transmission allocation is obtained using a simplified version of the *Full Optimization*.

No Migrate—the nodes in the fog tier and cloud tier of the network cannot transmit tasks between themselves, i.e., the FN to which the request r is sent from the MD is the one that computes it ($a_n^r w_l^r = 1 \iff l = n$).

Closest Wireless—in this approach, requests are always transmitted wirelessly to the closest node (the one with the lowest path loss). Then, the rest of the optimization is performed as in *Full Optimization*. The difference lies mostly in the step described in Eq. (5.33)—in Full Optimization the set of allocation variables **w** is found to minimize total transmission + computation costs, while in *Closest Wireless* each w_l^r is found separately, minimizing "only" the wireless transmission costs.

Not all of these solutions are plotted on every graph for clarity in this section. The results of *Closest Wireless* in many configurations overlap with the results of *Full Optimization*. In other words, the results of *Closest Wireless* are indistinguishable (within 0.1%) from the optimal results

of *Full Optimization* for the vast majority of tested parameter setups. Therefore, they are omitted from all plots except Fig. 5.7, where the difference between these two is visible. Shaded areas around results for each solution show 95% confidence intervals.

5.4.3 Comparison with Exhaustive Search and All Possible Allocations

First, let us compare results obtained from *Full Optimization* with those resulting from *Exhaustive Search* to validate the ability of the proposed algorithm to find the total minimum energy cost. A set of four computational requests is considered. The size of this set is limited due to the high computational complexity of *Exhaustive Search*. These requests have to be allocated among 10 FNs (allocation in the cloud is not considered in this example to highlight the importance of optimization within the fog tier). There are 50,400,000 possible allocations (10^4 for transmission, $\frac{10!}{(10-4)!} = 5040$ for computation) in total with energy consumption varying from 18.3 J to more than 29.4 J, as presented in Fig. 5.3.

The results obtained by *Full Optimization* (red dashed line) and *No Migrate* (black dotted–dashed line) are also shown. *Full Optimization* does indeed find the same solution as *Exhaustive Search*. The solution found by *No Migrate* results in slightly higher energy cost. Still, both solutions provide energy costs significantly lower than the average cost of all possible allocations. It is clear that an algorithm which assigns requests to nodes randomly would not be efficient in terms of energy cost.

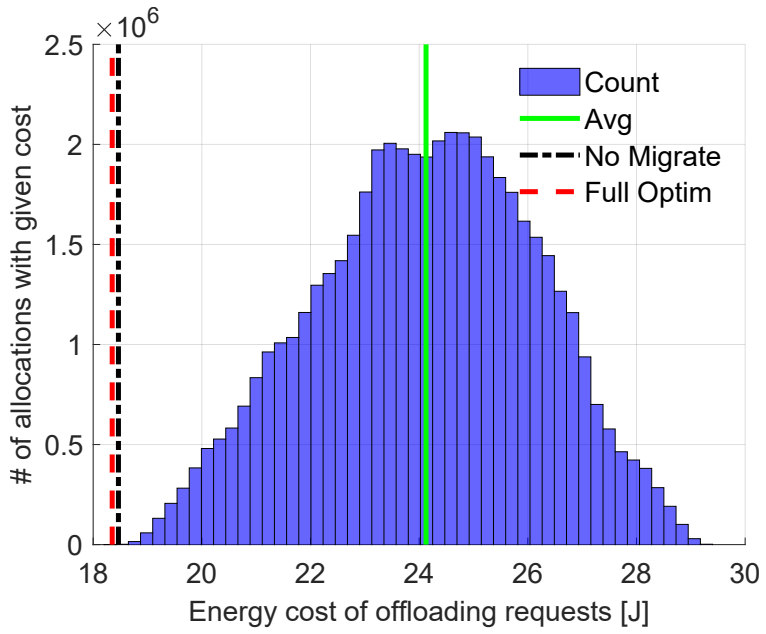


Figure 5.3: Comparison of Full Optim solution with the No Migrate solution and all possible allocations from exhaustive search (blue bars; average value marked with solid green line).

5.4.4 Impact of Network Parameters

Now let us examine the impact of the computational efficiency of the cloud on energy costs and allocations in the full network. Let us sweep this efficiency from 0.8 to 3.0 GFLOPS/W (efficiency of the 500 most powerful commercially available computer clusters ranges from 0.19 GFLOPS/W to 39.4 GFLOPS/W with 4.04 GFLOPS/W as the median [143]). Fig. 5.4 shows the median and the 90th percentile of the total energy costs spent on transmission and computation of offloaded requests. It can be seen that the energy costs of *Cloud Only* are significantly higher than those of *Full Optimization* for the lowest values of cloud efficiency, while differences between *No Migration* and *Full Optimization* are small. In all cases, the proposed solution requires a smaller amount of energy for a single request calculation than *No Migration*. As cloud efficiency increases, the cost of *Cloud Only* allocation decreases. In parallel, this allows *Full Optimization* to offload more tasks to the cloud, decreasing the energy consumption. The differences between the 90th percentiles are significantly higher than those between medians, showing the highest gains of *Full Optimization* for the most difficult requests. It is obvious that for the extremely high or low efficient cloud, the requests will be mostly calculated in the cloud or in the fog nodes, respectively. Therefore, for other results in this section, cloud efficiency is chosen to be 1.3 GFLOPS/W. This is a value of cloud efficiency that results in offloading decisions being not as straightforward as for values significantly higher or lower.

Another network parameter that can impact the costs and offloading decisions is the physical size of the network. The network shown in Fig. 5.2 (10 FN distributed over a 200 m $\times$ 50 m hall) is used by default. Now let us vary the physical size of the network while maintaining the same number of FNs. This has an effect on the distance between MDs and FNs. The greater the distance, the higher the path loss and the energy-per-bit cost of wireless transmission. At the same time, the higher the path loss the lower the wireless transmission rate. In Fig. 5.5 the length of the area covered by the network is swept up to 1000 m from the initial 200 m. With changing length

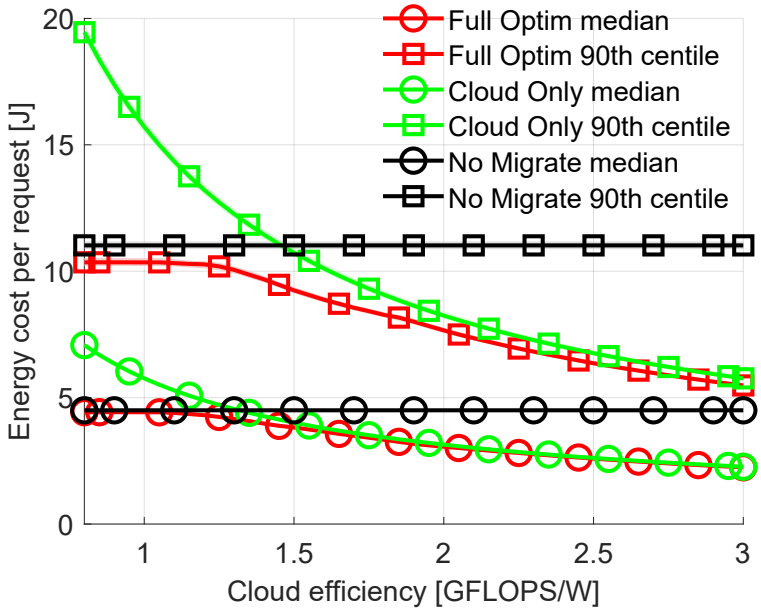


Figure 5.4: Comparison of energy cost per request with varied computational efficiency of cloud.

(the longer of the two dimensions) the ratios of distances between all FNs and the area perimeter remain constant. The results in Fig. 5.5 clearly show that the energy cost per request increases with the increasing size of the network. The increase is significant for *No Migrate* as MD is often "forced" to wirelessly send requests to more distant nodes if the close nodes are busy processing other requests or are not efficient enough. The rejection rates also increase from 3.3% at 200 m to 8.6% at 1000 m For *Full Optimization*, from 3.8% to 21.8% for *No migrate* and from 6.5% to 23.7% for *Cloud Only*. The difference in energy costs between *Full Optimization* and other methods becomes more apparent with increasing distances within the network.

5.4.5 Impact of Traffic Parameters

Let us vary parameters characterizing the requests offloaded to the network. For previous results, the parameters characterizing offloaded requests are random, as shown in Tab. 5.3. First, let us look at the impact of the delay requirement. It is fixed for all the incoming requests. The other parameters (e.g., arrival rate, arithmetic intensity) are generated in the same way as described in Section 5.4.1. Fig. 5.6 plots the median and the 75th percentile of energy costs spent on offloading requests as a function of the delay requirement (between 500 and 1000 ms) of these requests. There are a few key observations: (i) the percentage of rejected requests increases with stricter delay requirements, (ii) the energy cost increases with stricter delay requirements, (iii) *Cloud Only* is particularly poorly suited for delay-sensitive applications. Observation (i) is self-explanatory. The shorter the time-constraint, the harder it is to successfully offload the task, compute it and transmit the results back within this time. This can be seen on the plot where the respective lines terminate in the middle of a plot as a result of the virtually infinite energy cost of a request that is unsuccessfully calculated. For example, the green line representing the 75th percentile of *Cloud Only* terminates at 800 ms. This means that for delay requirements lower than 800 ms more than

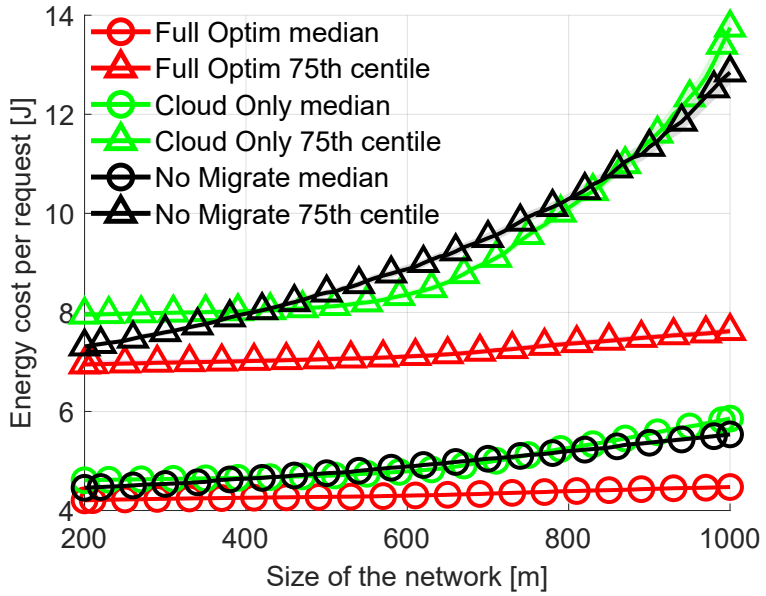


Figure 5.5: Comparison of energy consumption per request vs. the size of the area (area radius) covered by the network.

25% of requests are rejected. Observation (ii) is an effect of the higher CPU frequency required at FN to fulfill stricter delay requirements. This results in decreased CPU efficiency and increased energy consumption. Observation (iii) stems from the additional transmission delay caused by sending requests to the distant cloud.

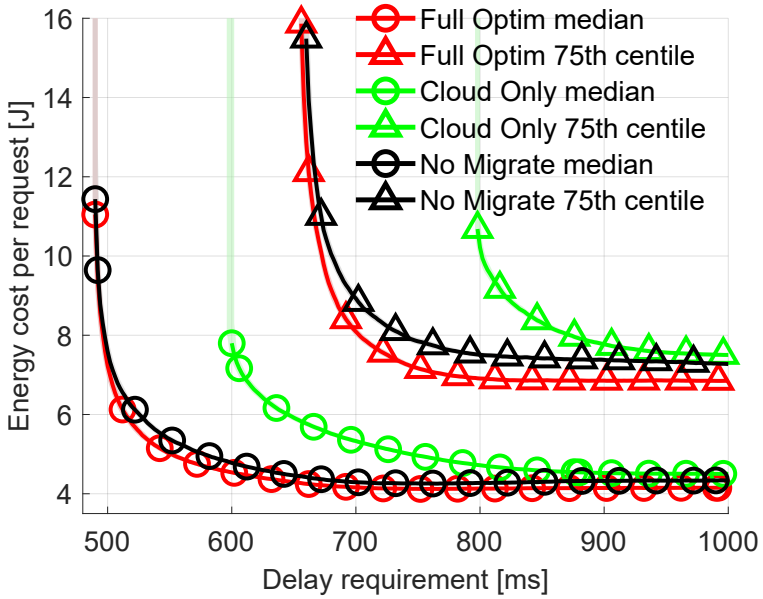


Figure 5.6: Comparison of energy consumption per request with varied delay requirement of requests.

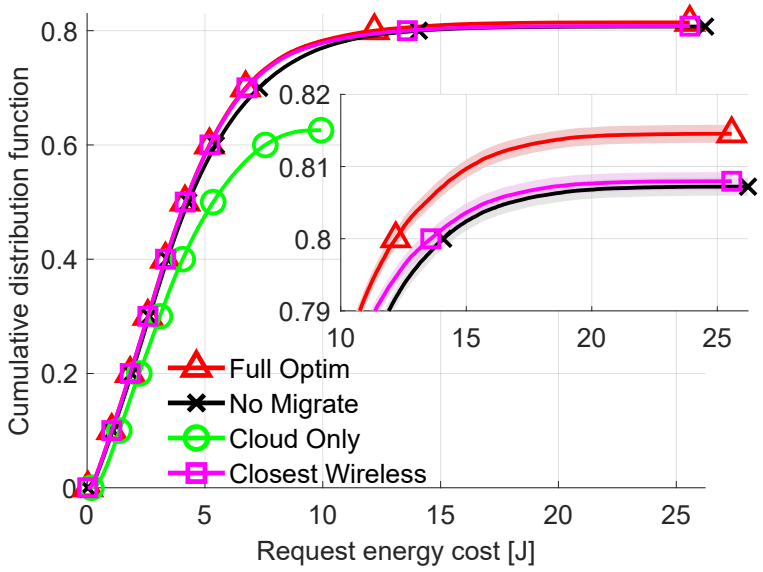


Figure 5.7: Comparison of energy consumption per request (CDF). Delay requirement: 700 ms.

To further analyze the difference between allocation strategies CDFs of energy costs are plotted in Fig. 5.7 for fixed delay requirement of all requests equal to 700 ms. Unlike previous plots, Fig. 5.7 includes results from the *Closest Wireless* algorithm. In all previous plots, the resulting

energy costs of *Closest Wireless* are not shown, since they are either identical to those of *Full Optimization* or are within 0.1% of it. Lowering the delay requirement created a scenario where sending the request wirelessly to the nearest (cheapest) AP/FN and then finding the optimal node for computation may not result in the optimal solution. This shows that *Full Optimization* manages to successfully offload nearly 81% of all requests. This is the most out of all the compared methods, about 0.5 percentage point more than *Closest Wireless*. It is visible that all the methods are differentiated mostly for high percentiles of energy costs. The worst solution is *Cloud Only*, which rejects nearly 40% of all requests. While the difference between *Closest Wireless* and *Full Optimization* is relatively small, this can be treated as a promising suboptimal solution which decreases algorithm complexity while maintaining efficiency. This can change if the considered wireless technology, e.g., 5G NR, provides a higher data rate and higher energy efficiency. However, this requires energy consumption models of 5G terminals to be available.

Finally, the impact of the arithmetic intensity of offloaded requests is examined. This parameter determines how many computations are needed to process a given request relative to its size. The median and 75th percentile of energy costs for arithmetic intensity swept in the range $\langle 1,\ 1000 \rangle$ FLOP/bit are plotted in Fig. 5.8. As expected, the energy cost increases with rising intensity. Higher values resulting from *Cloud Only* allocation at low intensity can be attributed to costs related to transmission (which do not directly depend on arithmetic intensity). Such requests can be more efficiently calculated in FNs, being commonly the result of the *Full Optimization* method. Energy costs (both median and 75th percentile) of *No Migrate* are within 10% of *Full Optimization* except for the values above 300 FLOP/bit where *No Migrate* steeply inclines. Rejection rates for *Full Optimization* are 1.1% for 1 FLOP/bit, 1.8% for 100 FLOP/bit and 11.7% for 1000 FLOP/bit. For *Cloud Only*, the values equal 2.9%, 4.4% and 20%, respectively. For *No Migrate*, they also start at 1.1% for 1 FLOP/bit and 1.8% for 100 FLOP/bit but reach 46.7% for 1000 FLOP/bit.

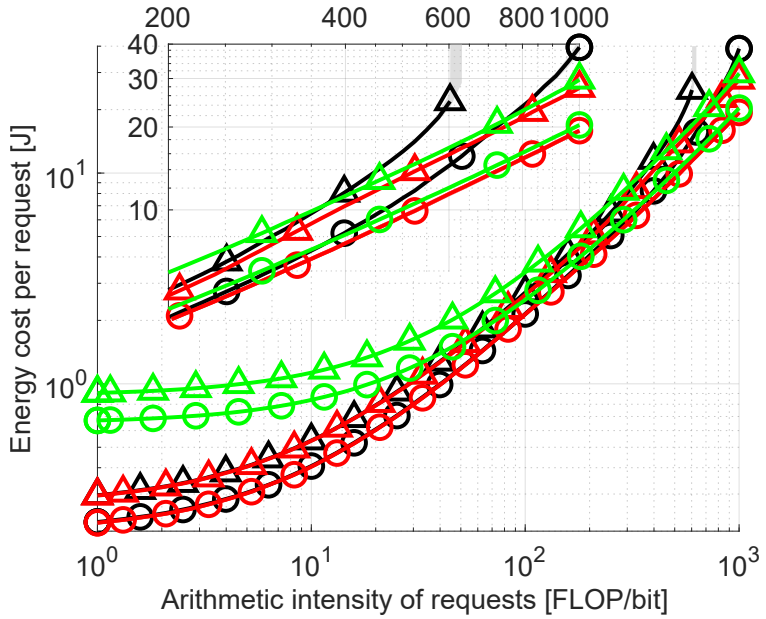


Figure 5.8: Comparison of energy consumption per request with varied arithmetic intensity. Same legend as in Figures 5.4 and 5.5.

5.5 Chapter Summary

Here above, the author investigated the minimization of energy spent on offloading computational tasks in fog networks. The considered network model includes delay and energy costs resulting from computation as well as wireless and wired transmission. The optimization problem formulated seeks the selection of the wireless connections, FNs and CNs involved in the task execution as well as their CPU frequencies to minimize the energy consumption in the network, while satisfying the latency constraint.

The computational task allocation algorithm, *Full Optimization*, proposed by the author of this thesis, successfully minimizes energy consumption while satisfying delay constraints. All the considered degrees of freedom, i.e., in AP selection, computing node selection, and FN CPU frequency tuning increase system performance (reduce energy-consumption) when compared with baseline allocation algorithms (*Cloud Only* and *No Migrate*). However, precise gain characterization depends on a specific network configuration and on the parameterization of computational requests.

When compared with the *No Migrate* solution, the biggest performance improvements of the *Full Optimization* can be seen when offloaded tasks have high arithmetic intensity or when a large area covered by the network causes higher path loss. In such a case, the observed gain is up to 50% energy saving. Compared with performing all computations in the cloud (*Cloud Only* algorithm), the proposed *Full Optimization* is much better suited for requests with strict delay requirements and low arithmetic intensity.

A heuristic approach that independently allocates wireless transmission called *Closest Wireless* is also proposed. This simplified algorithm provides optimal solutions for almost all considered scenarios. Its performance is slightly worse for requests with strict delay requirements—it manages to satisfy delay constraints of 0.8% fewer requests compared to *Full Optimization* at 700 ms.

6 Optimization of Energy Consumption – Allocation of Tasks Modeled as Directed Graphs

In this chapter, let us consider tasks that can be divided into subtasks. Each subtask can be computed at a different network node. The problem of offloading such composite tasks can be seen through the lenses of Virtual Network Embedding (VNE). The main contribution of this chapter is the mechanism to map these tasks called Virtual Network Requests (VNRs) to a fog-cloud substrate network. The challenge of such mapping is not only the computation requirement posed by VNRs, but also transmission limitations of links of the substrate network. Results of the proposed Clustered Network Embedding (CNE) algorithm are compared with Particle Swarm Optimization (PSO) and an exhaustive search over a variety of scenarios. The work presented in this chapter has been published by the author of this thesis in [11]. The original idea of modeling the offloaded tasks as directed graphs and the definition of the optimization problem came from Wei-Che Chen and Chun-Ting Chou, i.e., the co-authors of [11]. The main contribution of the author of this thesis is in researching the solutions and their verification by means of computer simulations.

6.1 Network Model

6.1.1 Virtual Network Requests with the Chain Topology

General VNE problems consider arbitrary topologies of VNRs and substrate networks. In this chapter, the focus is on VNRs with chain topologies, where traffic flows through these processes in a predefined order. This corresponds for example to an Augmented Reality (AR) application divided into many processes such as compression, decompression, video rendering, etc. Each VNR is represented as a set of tasks. Each task is represented by a Virtual Node (VN). The tasks are computed sequentially, but not necessarily at the same physical machine – a Substrate Node (SN).

A VNR is modeled as a weighted directed graph $G_V = (N_V, E_V)$, where N_V is the set of VNs and E_V is the set of Virtual Links (VLs) (compare with Fig. 6.1 illustrating an example of a VNR with its networking and computing requirements). The first VN in each VNR is called the edge node. Each VN $i \in N_V$ except for the edge node is characterized with the required computing workload weight value $W(i)$, where $W(i)$ is given in millions of FLOPs. Each VL is denoted as $(i, j) \in E_V$, and is characterized with the size of data $D(i, j)$, where the value of $D(i, j) \in E_V$ is given in Megabits. Next, let us define the latency demand of a VNR. Let t_0 be the time when a VNR arrives in the network and t_f be the completion time of its last VN. The difference $t_f - t_0$ must be lower or equal to latency demand L. L is given in number of time-slots.

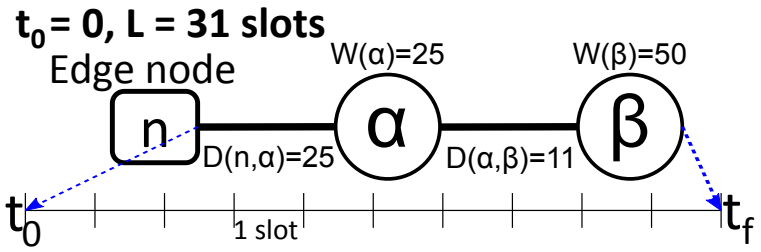


Figure 6.1: An example of a VNR with chain topology.

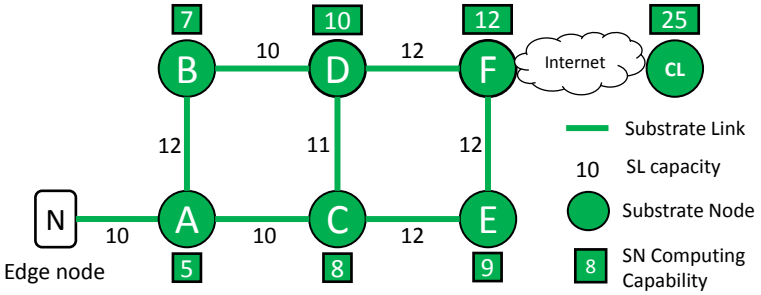


Figure 6.2: A Fog-Cloud substrate network.

6.1.2 A Fog-Cloud Substrate Network

The substrate network is modeled as a weighted undirected graph $G_S = (N_S, E_S)$, where N_S is the set of SNs and E_S is the set of Substrate Links (SLs). Each SN $p \in N_S$ is characterized with the computing capability $C(p)$ in GFLOPS. Among these SNs, there exists a specific SN that is called the edge node. The edge node is where all VNRs originate from and it does not have its own computing capability. Moreover, one of the SNs represents a Cloud data center – it has the highest computing capability while also being the furthest away from the edge node. In addition, each SN except for the edge node has its own scheduling table $T_N(p)$ providing information about occupancy of computing resources. A VN processed at a given SN utilizes all its computational resources.

An SL between a pair of SNs p and q is denoted as $(p, q) \in E_S$. It is characterized by its link capacity $B(p, q)$ in Gbps. Among these SLs, there exists a specific SL that represents the connection with the Cloud – the Internet. In addition, each SL except for the Internet has its own scheduling table $T_E(p, q)$ providing information whether bandwidth slots of (p, q) are occupied or not.

Fig. 6.2 illustrates a substrate network, where the numbers over the SLs represent bandwidth and the numbers in rectangles represent computing capability of a given node. An example of the scheduling table is shown in Fig. 6.3. Occupied slots are blue while unoccupied slots are white.

6.1.3 Node and Link Embedding

When a VNR arrives, network operators have to determine a suitable embedding for the VNR, and allocate resources on SNs and SLs for this embedding. The embedding of a VNR onto the substrate network is divided into two primary Sub-Problems (SubPs).

1) SubP1 – Virtual Node Embedding: Each VN of a given VNR must be embedded into one

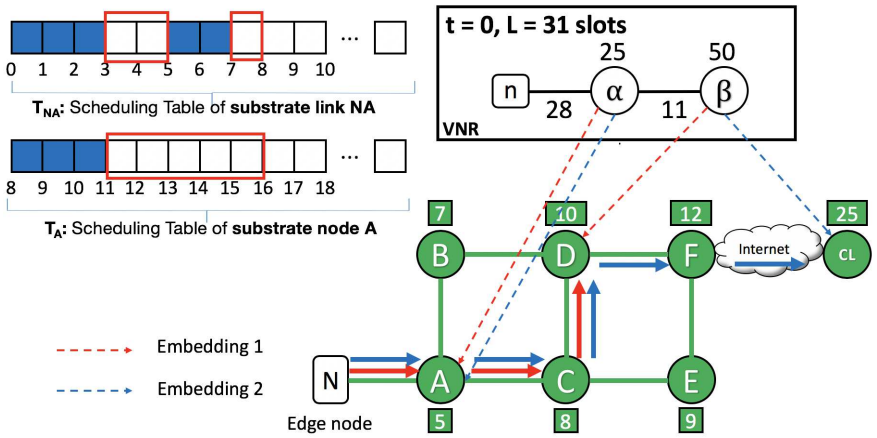


Figure 6.3: An example of LA-VNE.

of the SNs. x_p^i denotes whether VN i is mapped on SN p.

$$x_p^i = \begin{cases} 1 & \text{if VN } i \in N_V \text{ is mapped to SN } p \in N_S \\ 0 & \text{otherwise} \end{cases}$$

2) SubP2 – Virtual Link Embedding: Each VL is mapped to a substrate path consisting of one or more SLs between the corresponding SNs that host the end VNs of that VL. f_{pq}^{ij} denotes whether VL (i,j) is mapped on SL (p,q). g^{ij} denotes whether VL (i,j) goes through the Internet.

$$f_{pq}^{ij} = \begin{cases} 1 & \text{if VL}(i,j) \in E_V \text{ is mapped to SL}(p,q) \in E_S \\ 0 & \text{otherwise} \end{cases}$$

$$g^{ij} = \begin{cases} 1 & \text{if VL } (i,j) \in E_V \text{ goes through the Internet} \\ 0 & \text{otherwise} \end{cases}$$

Fig. 6.3 shows two exemplary embeddings of a single VNR. The VNR can be served by the Embedding 1 (red dashed arrow) $\{\alpha \to A, \beta \to D\}$ or the Embedding 2 (blue dashed arrow) $\{\alpha \to A, \beta \to CL\}$. Embedding 1 makes the following mappings of the VNR to SLs: $(n,\alpha) \to (N,A)$ and $(\alpha,\beta) \to \{(A,C),(C,D)\}$, while the Embedding 2 maps it to $(n,\alpha) \to (N,A)$ and $(\alpha,\beta) \to \{(A,C),(C,D),(D,F),(F,CL)\}$.

6.1.4 Latency Model

Networking latency $L_{net}(i,j)$ in VLs and computing latency $L_{com}(i)$ in VNs contribute to the total latency $L(G_V)$ of VNR G_V:

$$L(G_V) = \sum_{(i,j)\in E_V} L_{net}(i,j) + \sum_{i \in N_V} L_{com}(i). \tag{6.1}$$

For successful embedding, the total latency $L(G_V)$ must satisfy the latency demand L:

$$L(G_V) \le L. \tag{6.2}$$

Two slots functions are defined to compute $L(G_V)$:

$$U_{p,q}^{net}(i,j) = \begin{cases} 0 & \text{if } p = q, \\ \lceil \frac{D(i,j)}{B(p,q)} \rceil & \text{otherwise}, \end{cases} \tag{6.3}$$

$$U_p^{com}(i) = \lceil \frac{W(i)}{C(p)} \rceil. \tag{6.4}$$

They represent the networking slots function and the computing slots function, respectively as explained next based on the example from Fig. 6.3.

In the Embedding 1, the VL (n, α) only goes via the SL (N, A). The slots needed to be used on the SL (N, A) are $U_{N,A}^{net}(n, \alpha) = \lceil \frac{D(n,\alpha)=28}{B(N,A)=10} \rceil = 3$. Next, we check the scheduling table $T_E(N, A)$ in Fig. 6.3 at t=0. The slots are occupied by other VNRs in the interval $[0, 3]$ and $[5, 7]$. According to the scheduling table, these 3 slots can be successfully transmitted from N to A at $t = 8$. Therefore, the networking latency is $L_{net}(n, \alpha) = 8$.

α is embedded into SN A. The computing slots to be used on A are $U_{\alpha}^{com}(A) = \lceil \frac{W(\alpha)=25}{C(A)=5} \rceil = 5$. Likewise, we check the scheduling table $T_N(A)$ in Fig. 6.3 at $t = 8$. According to the table, these 5 slots can be successfully processed in A by t=16. Therefore, the computing latency $L_{com}(\alpha)$ is 8.

For VL (α, β), the required slots on AC and CD are $U_{A,C}^{net}(\alpha, \beta) = 2$ and $U_{C,D}^{net}(\alpha, \beta) = 1$, respectively. Therefore, the networking latency $L_{net}(\alpha, \beta)$ is 3. In β, the required slots on D are $U_D^{com}(\beta) = 5$ slots. Therefore, the computing latency $L_{com}(\beta)$ is 5. Finally, the completion time of the Embedding 1 is $L_{net}(n, \alpha) + L_{com}(\alpha) + L_{net}(\alpha, \beta) + L_{com}(\beta) = 24$, and it is lower than the latency demand $L = 31$.

In the Embedding 2, α is also embedded on A. Therefore, the networking latency is $L_{net}(n, \alpha) = 8$ and the computing latency $L_{com}(\alpha)$ is 8. Meanwhile, β is embedded on CL, and the path between A and CL is $\{(A, C), (C, D), (D, F), (F, CL)\}$.

Here, let us define the latency in the Internet link connecting the Cloud with the rest of the network. The latency of VL $(i, j) \in N_v$ in the Internet $U_I^{net}(i, j)$ consists of the propagation latency and the transmission latency:

$$U_I^{net}(i, j) = \theta + \lceil \frac{D(i, j)}{B_I} \rceil. \tag{6.5}$$

θ is the constant that represents the propagation latency caused by fiberline distance from the Cloud, and B_I represents the link capacity. In this example, let us assume that $\theta = 7$ ms and $B_I = 10$ Gbps. Therefore, $U_I^{net}(\alpha, \beta) = 7 + \lceil 11/10 \rceil = 9$ slots. Networking latency $L_{net}(\alpha, \beta) = U_{A,C}^{net}(\alpha, \beta) + U_{C,D}^{net}(\alpha, \beta) + U_{D,F}^{net}(\alpha, \beta) + U_I^{net}(\alpha, \beta) = 13$. The computing latency $L_{com}(\beta) = \lceil 50/25 \rceil = 2$. Finally, the completion time of the Embedding 2 is $L_{net}(n, \alpha) + L_{com}(\alpha) + L_{net}(\alpha, \beta) + L_{com}(\beta) = 31$ slots. It is equal the latency demand. Therefore, both Embedding 1 and 2 satisfy the latency demand.

6.1.5 Cost Model

Since both Embedding 1 and 2 satisfy the latency demand, we should determine which one is better for the network operator. The goal is to minimize the total energy consumption used for serving the VNR. Eq. (6.6) defines the energy cost model $Y(G_V)$ consisting of computing costs $Y_{com}(i)$ (see Eq. (6.7)) and networking costs $Y_{net}(i, j)$ (Eq. (6.8)).

$$Y(G_V) = \sum_{(i,j) \in E_V} Y_{net}(i, j) + \sum_{i \in N_V} Y_{com}(i) \tag{6.6}$$

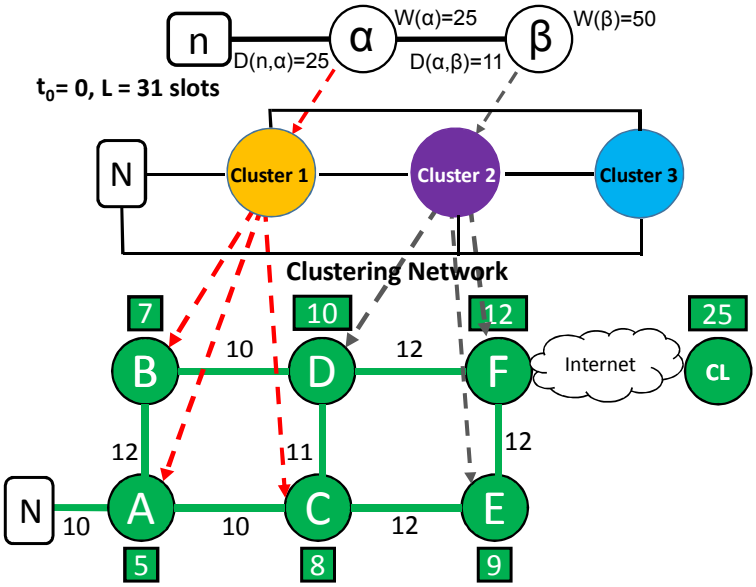


Figure 6.4: Key concept of the CNE algorithm.

$$Y_{com}(i) = \sum_{p \in N_S} x_p^i \cdot U_i^{com}(p) \cdot \frac{C(p)}{\sigma_p} \tag{6.7}$$

$$Y_{net}(i,j) = g^{ij} \cdot I_C(i,j) + \sum_{(p,q) \in E_S} f_{pq}^{ij} \cdot U_{p,q}^{net}(i,j) \cdot \omega_p \cdot B(p,q) \tag{6.8}$$

$$I_C(i,j) = \omega_I \cdot D(i,j) \tag{6.9}$$

$\frac{C(p)}{\sigma_p}$ represents the computing cost per slot (in mJ/ms, as $[\frac{\text{GFLOPS}}{\text{GFLOPS/W}} = \text{W} = \frac{\text{mJ}}{\text{ms}}]$) of SN p and σ_p represents energy efficiency of SN p in GFLOPS per Watt. $\omega_p \cdot B(p,q)$ represents the networking cost per slot on a SL. ω_p is the energy-per-transmitted-bit parameter characterizing networking equipment in SN p. The networking cost on the Internet is defined in Eq. (6.9), where ω_I denotes energy-per-transmitted-bit cost of sending data to the Cloud.

The optimization objective is, for each arriving request G_V, to minimize the total cost (Eq. (6.6)) while satisfying the latency constraint (Eq. (6.2)).

6.2 Proposed Solution

In this section, we describe the proposed CNE Algorithm. Instead of directly embedding the VNR into the substrate network, we first embed it into the clustered network for preliminary evaluation in terms of latency and cost. The necessary steps are discussed below.

Step 1: Building the Clustered Network: In Step 1, we want to group SNs into multiple clusters. At first, we transform SNs in the substrate network (except for the Cloud) into two-dimensional datasets based on number of transmission hops from the edge node and their computing capability.

After creating these datasets, let us use an unsupervised k-means clustering approach to group them. In order to dynamically choose the value k representing the number of clusters, the elbow method [144] is chosen to find appropriate value of k.

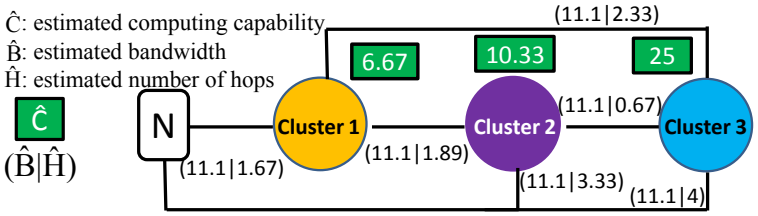


Figure 6.5: Estimated components in a clustered network.

After generating the clusters we determine their estimated computing capabilities by averaging the computing capabilities of all SNs within a cluster. The clustered network is modeled as a weighted undirected graph $\hat{G}_S = \left(\hat{N}_S, \hat{E}_S\right)$, where $\hat{N}_S$ is the set of clusters and $\hat{E}_S$ is the set of SLs between clusters. Each cluster $n \in \hat{N}_S$ is characterized by its estimated computing capability $\hat{C}(n)$ and energy efficiency $\hat{\sigma}_n$. Each pair of clusters n and m has a SL between them denoted as $(n,m) \in \hat{E}_S$, and is characterized with the estimated bandwidth $\hat{B}(n,m)$ and average number of transmission hops $\hat{H}(n,m)$. For simplicity, let us set the estimated bandwidth in each SL between clusters as the average value of bandwidth in all SLs in the substrate network. After building the clustered network, we can enter the next step – the preliminary evaluation.

Step 2: Preliminary Evaluation By Embedding VNRs onto the clustered network: The embedding variables are defined as follows:

$$a_n^i = \begin{cases} 1 & \text{if VN } i \in N_V \text{ maps to cluster } n \in \hat{N}_S \\ 0 & \text{otherwise} \end{cases}$$

$$o_{nm}^{ij} = \begin{cases} 1 & \text{if VL}(i,j) \in E_V \text{ maps to SL}(n,m) \in \hat{E}_S \\ 0 & \text{otherwise} \end{cases}$$

$$r^{ij} = \begin{cases} 1 & \text{if VL } (i,j) \in E_V \text{ goes through the Internet} \\ 0 & \text{otherwise} \end{cases}$$

Now we can define total estimated latency $\hat{L}$ in Eq. (6.10). The estimated networking latency $\hat{L}_{net}(i,j)$ and the computing latency $\hat{L}_{com}(i)$ are defined in Eqs. (6.11) and (6.12) respectively. Next, we define the estimated cost $\hat{Y}$ in Eq. (6.13). The networking estimated cost $\hat{Y}_{net}(i,j)$ and the computing estimated cost $\hat{Y}_{com}(i)$ are defined in Eqs. (6.14) and (6.15) respectively.

$$\hat{L} = \sum_{(i,j)\in E_V} \hat{L}_{net}(i,j) + \sum_{i \in N_V} \hat{L}_{com}(i) \tag{6.10}$$

$$\hat{L}_{net}(i,j) = r^{ij} \cdot I_L(i,j) + \sum_{(n,m)\in \hat{E}_S} o_{nm}^{ij} \cdot \frac{D(i,j)}{\hat{B}(n,m)} \cdot \hat{H}(n,m) \tag{6.11}$$

$$\hat{L}_{com}(i) = \sum_{n \in \hat{E}_S} a_n^i \cdot \frac{W(i,j)}{\hat{C}(n)} \tag{6.12}$$

$$\hat{Y} = \sum_{(i,j)\in E_V} \hat{Y}_{net}(i,j) + \sum_{i \in N_V} \hat{Y}_{com}(i) \tag{6.13}$$

$$\hat{Y}_{net}(i,j) = r^{ij} \cdot I_C(i,j) + \sum_{(n,m)\in \hat{E}_S} o_{nm}^{ij} \cdot D(i,j) \cdot \hat{H}(n,m) \cdot \omega \tag{6.14}$$

Table 6.1: An example of the PE table.

	α	β	$\hat{L}$	$\hat{Y}$	Rank	Success	Y
PE (1)	1	1	15.44	87.37	2	Yes	91.84
PE (2)	1	2	14.73	91.6	3	Yes	89.18
PE (3)	1	3	18.55	84.53	1	Yes	**87.67**
PE (4)	2	1	20.17	116.76	7	–	–
PE (5)	2	2	15.64	97.49	5	–	
PE (6)	2	3	20.15	94.21	4	–	–
PE (7)	3	1	**34.26**	–	–	–	–
PE (8)	3	2	29.96	118.45	8	–	–
PE (9)	3	3	24.46	103.13	6	–	–

$$\hat{Y}_{com}(i) = \sum_{n \in \hat{E}_S} a_n^i \cdot \frac{W(i)}{\hat{\sigma}_n} \tag{6.15}$$

After calculating the estimated latency and cost, we start embedding the VNR into the clustered network to evaluate the actual performance. Since the scale of the network is not large, we use the exhaustive search to embed the VNR. We denote these node embedding variations as Preliminary Embeddings (PEs). We calculate the estimated latency and cost of each PE by using Eq. (6.1) and Eq. (6.6), respectively.

Step 3: Embedding VNRs into the Substrate Network: Tab. 6.1 illustrates an exemplary table of PEs ranked by their estimated costs $\hat{Y}$. We choose the PE with the lowest $\hat{Y}$, i.e., PE (3). In this example, α in PE (3) must be embedded into Cluster 1 (SNs A, B, or C), while β must be embedded into Cluster 2 (SNs D, E, or F) – we now have to solve the Latency-Aware Virtual Network Embedding (LA-VNE) problem with these constraints. We divide it into 2 stages as seen in Section 6.1.3. During both virtual node embedding (SubP1) and virtual link embedding (SubP2), we consider the scheduling tables of SNs and SLs. By solving this problem, we find the energy cost Y for the chosen PE.

For SubP1, depending on the size and complexity of the network, we consider one of the two following algorithms within the proposed CNE algorithm. The first algorithm is the exhaustive search, and the second algorithm is the discrete PSO algorithm described in [109].

After embedding VNs into the SN in the node embedding stage, we enter the link embedding stage. We embed VLs of a single VNR starting from the edge node and finishing at the last VN. When embedding VLs, we use Dijkstra's algorithm to find the shortest paths according to networking latency while considering scheduling tables. After embedding all VLs, we calculate total latency L and cost Y related to processing a single VNR according to Eq. (6.1) and Eq. (6.6). Node and link embedding combination is a viable solution if the total latency is lower or equal to the latency demand.

We continue embedding VNR until we find 3 solutions fulfilling latency constraints. For each next solution, we take the PE from the PE table with the next best rank. Finally, we choose the solution with the lowest energy cost. The VNE is rejected if no solution can be found.

6.3 Results

6.3.1 Performance Evaluation

Substrate networks: The substrate network is configured to have either 10 FNs and one Cloud (small network scenario - Section 6.3.2.1) or 50 FNs and one Cloud (large network scenario - Sections 6.3.2.2 and 6.3.2.3). SLs connecting SNs (FNs) are randomly added following these rules: (1) each node must be directly connected with at least 2 other nodes for resilience reasons, (2) there exists at least one path between any node pair, (3) nodes should not be directly connected to more than 5 other nodes. Then two SNs that are the furthest away from each other (in terms of hops) are chosen as a fog gateway and a Cloud gateway. VNRs arrive at one node (i.e., the fog gateway connected to the edge node gathering VNRs from data sources), while the other node (the Cloud gateway) is connected to the Cloud via the Internet.

It is assumed that FNs further away from the user have higher computational power and that the CN has significantly higher computational power (and lower energy-per-instruction cost) than any of the nodes in the fog tier of the network. This corresponds to multi-tier Fog deployment proposed in [19]. The computational capability of FNs is assumed to be between 10 and 50 GFLOPS, while the computational capability of the Cloud is 200 GFLOPS. The energy efficiency parameter given in GFLOPS per Watt is used to determine energy costs related to computing requests. It is used for measuring the efficiency of supercomputers [128]. The cloud SN has an efficiency of 5 GFLOPS per Watt (each single operation amounts to 0.2 nJ of energy spent). Fog SNs have an efficiency between 1 and 1.25 GFLOPS per Watt. Link capacity of SLs between the FNs follows a uniform distribution between 5 and 10 Gbps. The energy cost of transmitting data through SLs is equal to 0.2 nJ/bit for each link between the FNs. This value is equal to the per-bit energy consumption of a 10/10G Ethernet Passive Optical Network (EPON) gateway [119]. Requests sent through the Internet connection between Cloud and SN serving as Cloud gateway are transmitted at 10 Gbps bitrate. For this connection, it is assumed that the Cloud is located 2000 km away from the rest of the network and that the data goes through multiple core network routers (with 1 nJ/bit cost each [120]). The energy cost of transmitting data to the Cloud through the Internet is therefore assumed to be 8 nJ/bit, and this transmission introduces additional 15 ms of delay (using 7.5 μs/km Round-Trip Time (RTT) [145]).

The aforementioned scenarios are inspired by a facial recognition access system installed in the Department of Electrical Engineering in National Taiwan University. A camera, representing the edge node, captures the raw videos while the tasks including video compression, image pre-processing, deep learning-based face detection, and identity recognition are carried out in the local FNs connected by Ethernet Local Area Network (LAN). Finally, the CN is assumed to be virtual machines of the Amazon Web Services servers located in Japan (roughly 2000 km away from Taipei).

Virtual Network Requests: The number of VNs in a single VNR is randomly determined by a uniform distribution between 2 and 3 (low load scenario) or between 2 and 6 (high load scenario). An example of a VNR with 2 VNs is shown in Fig. 6.1. The arrival rate of VNR follows a Poisson process with mean 0.1 or 0.5 requests per time slot in low load and high load scenarios respectively. Each time slot lasts for 1 ms. The computing workload in VNs follows a uniform distribution from 50 to 500 millions of FLOPs. The raw data sent through VLs follows a uniform distribution from 10 to 100 Mb. The latency demand on VNRs follows a uniform distribution from 200 to 500 ms.

Table 6.2: Simulation parameters used in Chapter 6.

Parameter	Symbol	Value	Unit
The # of SNs (small/large network)	$\|N_S\|$	11; 51	–
Computing capability of FNs	$C(p)$	10 - 50	GFLOPS
Computing capability of Cloud	$C(p)$	200	GFLOPS
Computing cost in FNs	$1/\beta_p$	0.8 - 1	nJ/FLOP
Computing cost in Cloud	$1/\beta_p$	0.2	nJ/FLOP
Transmission cost through SLs	ω_p	0.2	nJ/bit
Transmission cost through the Internet	ω_I	8	nJ/bit
Link capacity of SLs	$B(p,q)$	5 - 10	Gbps
Propagation latency through Internet	θ	15	ms
No. of VNs in a VNR (low/high load)	$\|N_V\|$	2 - 3; 2 - 6	–
Data transmitted through VLs	$D(i,j)$	10 - 100	Mbit
Workload of VNs	$W(i)$	50 - 500	MFLOP
Latency demand	L	200 - 500	ms
Arrival rate (low/high load)	–	P(0.1); P(0.5)	1/ms
Simulation time	–	5000	ms

The simulation time is set at 5000 time slots (5000 ms). Simulation parameters are summarized in Tab. 6.2 and taken from [119, 120, 128, 145].

6.3.2 Evaluation Results

The proposed CNE algorithm can be adjusted depending on the size and complexity of a studied network. For small networks (Section 6.3.2.1), CNE algorithm utilizing exhaustive search for node embedding (labelled as CNE[EX]) is used. While for larger networks studied in Section 6.3.2.2 and 6.3.2.3 the CNE algorithm utilizing PSO is chosen. Single-threaded simulations are performed on a Personal Computer (PC) equipped with Intel® Core™ i7-860 processor. The code is written in C++.

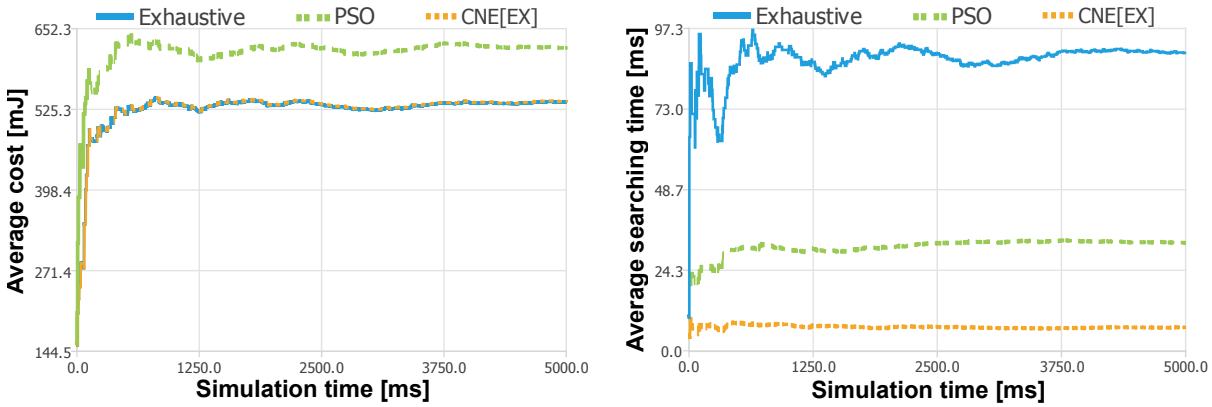
6.3.2.1 Scenario 1: Small Networks and Low Load

Since the scale of the substrate network and VNRs is small, we can compare the proposed CNE algorithm with exhaustive search to verify the performance in terms of cost and execution time. In addition, let us contrast it with general discrete PSO algorithm proposed in [109].

As shown in Fig. 6.6a, the proposed algorithm (orange dotted line) has near-optimal performance in terms of cost compared to the exhaustive search (blue solid line). The difference is lower than 1% in this setting. Our proposed solution outperforms both the exhaustive search and the general PSO in terms of execution time (about 92% and 78% respectively – see Fig. 6.6b).

6.3.2.2 Scenario 2: Large Networks and Low Load

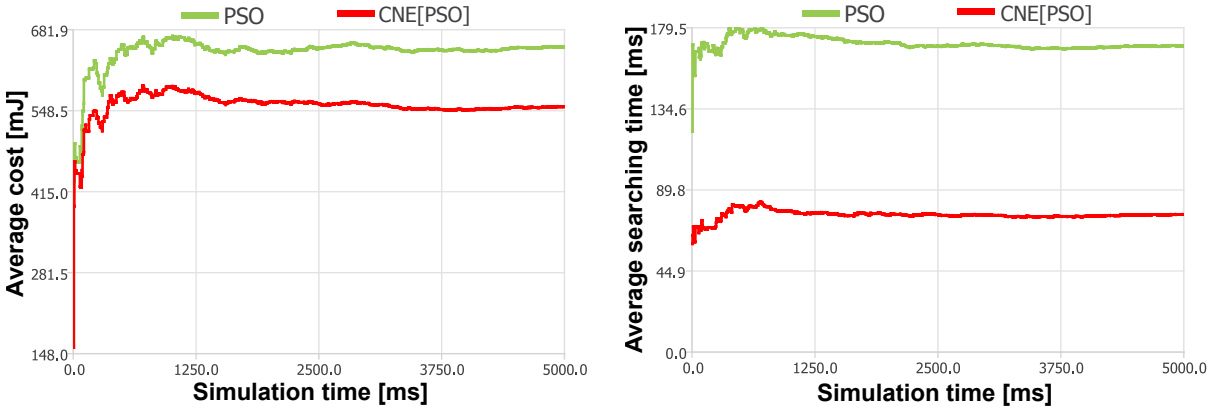
Exhaustive search is computationally infeasible in larger networks. Therefore, let us compare the proposed solution (CNE algorithm utilizing PSO – red line) with the pure PSO (green line) in terms of cost and execution time in Fig. 6.7a and Fig. 6.7b respectively. Our proposed solution has better performance – lower cost by about 15% and shorter execution time by about 55%. It stems from the fact that grouping similar nodes into clusters greatly reduces the search space which directs PSO to better solutions.



(a) Average cost.

(b) Average execution time.

Figure 6.6: Simulation results in small networks with low computation load (Scenario 1).



(a) Average cost.

(b) Average execution time.

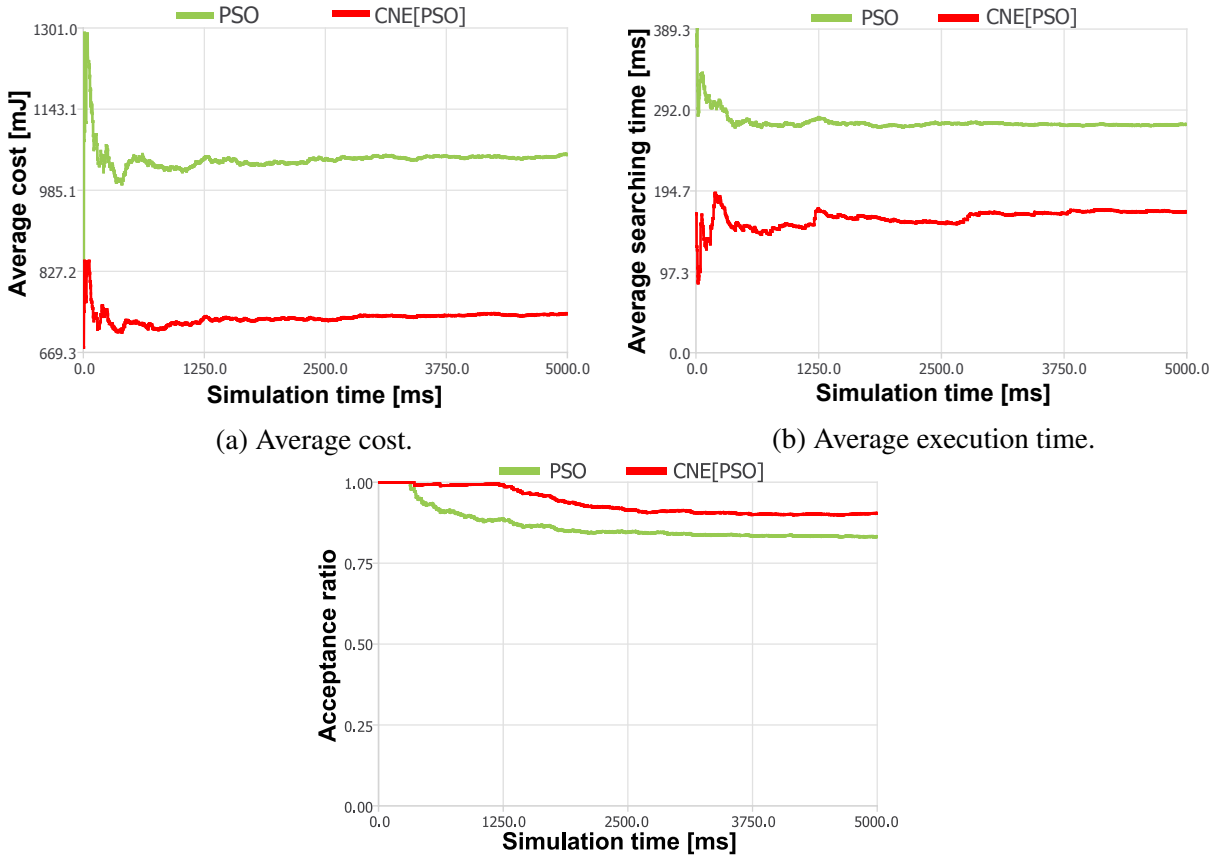
Figure 6.7: Simulation results in large networks with low computation load (Scenario 2).

6.3.2.3 Scenarios 3&4: Large Networks and High Load

As the complexity and the number of arriving VNRs increases, it becomes more likely that no feasible embedding is found. Let us compare the proposed solution (CNE algorithm utilizing PSO) with PSO in terms of cost, acceptance ratio, and execution time in large networks with high computation load in Fig. 6.8 (*Scenario 3*). The acceptance ratio is defined as the number of successfully embedded VNRs to the total number of VNRs that arrive at the network.

Our solution has 29% lower energy consumption (Fig. 6.8a), and a noticeably higher (7 percentage points) acceptance ratio as shown in Fig. 6.8c. According to these results, it can be concluded that CNE algorithm efficiently utilizes computing and networking resources and allows more VNRs to be successfully served. It is still considerably faster (by 38%) than the general PSO algorithm. It is worth noting that this execution time is significantly higher than in the low load scenario (Fig. 6.7b). It is caused by the fact that embedding VNRs with higher number of nodes requires more computations.

Results shown in Figs. 6.6, 6.7, and 6.8 are acquired simulating network where fog SNs further away from the end user have higher computational capability. This follows the concept of hier-



(a) Average cost.

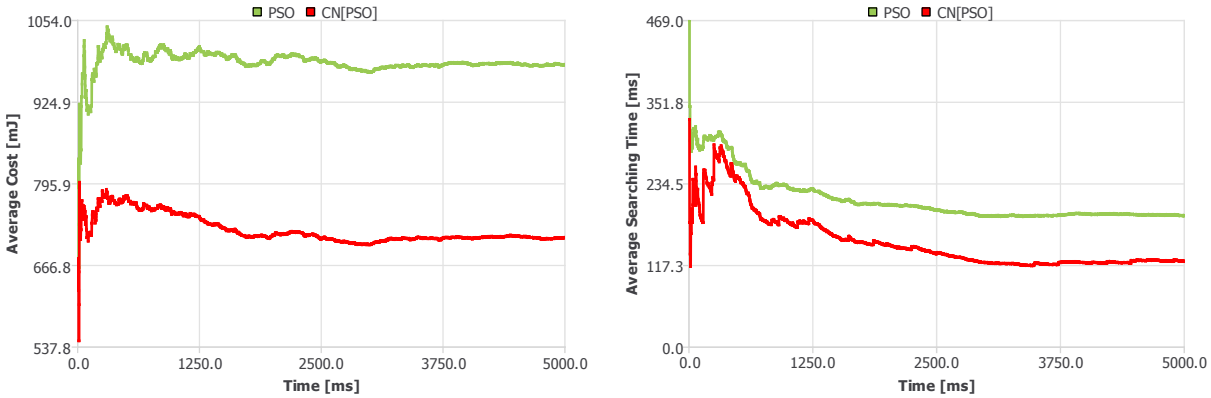
(b) Average execution time.

(c) Acceptance ratio.

Figure 6.8: Simulation results in large networks with high computation load (Scenario 3).

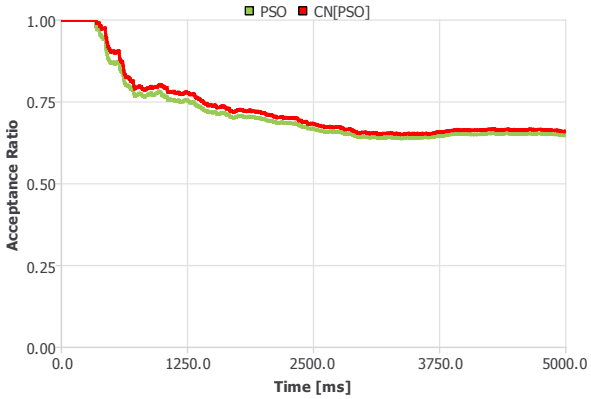
archical, tiered fog computing networks as seen in [19]. In contrast, Fig. 6.9 (*Scenario 4*) shows results of simulating a network, where computational capabilities of fog SNs are independent of their distance to the fog gateway. All other parameters are the same as in *Scenario 3* shown in Fig. 6.8. It can be seen that the percentage of accepted cases (as well as the difference between algorithms) is significantly lower in Fig. 6.9 than in Fig. 6.8. However, CNE algorithm is still faster and finds more energy-efficient embeddings than pure PSO. It demonstrates that clustering nodes with similar parameters remains effective even when their distribution does not follow a hierarchical fog network structure. Interestingly, while acceptance ratio is lower in a scenario with more "randomized" network, average cost and searching time are also lower. It stems from the fact that these averages are calculated over successfully embedded requests, and requests which fail to be embedded tend to be longer and more costly to embed.

Values plotted in this chapter show high variance at the beginning of the simulation and then stabilize at certain levels. It is caused by the fact that the average values over all embedded requests are plotted. Each new request can significantly change the average value in the transient period (when only a few requests have been processed). As the total number of requests increases, the volatility decreases.



(a) Average cost.

(b) Average execution delay.



(c) Acceptance ratio

Figure 6.9: Simulation results in large networks with high computation load when computational power of SNs does not increase with its distance from the end user (Scenario 4).

6.4 Chapter Summary

In this chapter, the optimization problem of allocating resources to processing offloaded tasks is formulated. These tasks consist of subtasks and are modeled as directed graphs. Each of these subtasks can be processed at a different node but the computations have to be performed sequentially. The problem is expressed in the form of VNE. The aim is to minimize energy costs while satisfying the latency demands. A heuristic algorithm called the CNE algorithm is proposed to find close-to-optimal solutions in a reasonable time. Our algorithm can efficiently reduce the dimension of the search space while retaining better feasible solutions. The simulation results show that the proposed algorithm has near-optimal performance in small networks, and has a clear 29% improvement in terms of energy cost, 7 percentage points higher acceptance ratio, and 38% lower execution time over the PSO algorithm in large networks.

7 Conclusions

7.1 Work summary

In this dissertation, the problem of energy consumption minimization in fog computing networks has been studied. First, the author analyzed published works with regard to modeling, parameterizing, and optimizing fog networks in Chapter 2. The author's original research is then focused on the most commonly cited application of fog: offloading computational tasks. Chapter 3 considers modeling of the energy consumption and computational task delay related to task offloading in the fog and cloud tiers of a network. The models are parameterized to represent real-world equipment. Moreover, the impact of the network parameters on the task average energy consumption and latency is presented. In the following Chapters 4–6, three optimization problems are formulated and subsequently solved. Each concerns computational task offloading in fog networks and aims to minimize energy consumption while fulfilling the tasks' latency requirements.

In Chapter 4, the optimization problem involves the allocation of tasks to fog and cloud nodes as well as the adjustment of their clock frequency. The formulated optimization problem is a non-convex Mixed-Integer Non-Linear Programming (MINLP), which is solved using the problem decomposition and a series of Successive Convex Approximation (SCA) as well as the Hungarian algorithms. A sub-optimal, low-complexity solution is also proposed and examined in the computer simulation experiments.

Chapter 5 expands on the problem presented in Chapter 4 by adding wireless things-to-fog transmission to the network model and the problem formulation. It changes the objective function (by adding energy cost for things-fog transmission) and constraints (by adding the wireless transmission-associated delay), and includes a new set of decision variables (extending the optimization solution search space). The exact solution is found without the use of SCA and compared against baseline solutions.

Finally, Chapter 6 examines the offloading of sequential tasks, which are broken down into subtasks. Each subtask can be computed at a different node, however, they must be executed in the proper order. The model and problem are defined in terms of Virtual Network Requests (VNRs) represented by directed graphs. The solution proposed as the Clustered Network Embedding (CNE) algorithm involves decreasing the search space size by clustering nodes with similar parameters.

Chapters 3 through 6 provide results for varied networking scenarios and parameter sweeps.

7.2 Key findings

Based on the original research done by the author of this thesis, as well as reviewing other published works in the area of energy-efficient fog networks, the following conclusions can be drawn.

Fog computing can improve the performance of the standard cloud-exclusive task delegation networks in terms of energy consumption and task execution latency. Utilization of Fog Nodes (FNs) located close to end users allows a network to serve these users more efficiently. This is especially true for tasks with strict latency requirements. On the other hand, using a standard cloud delegation is preferable for computationally intensive tasks without latency requirements and when the cloud servers are particularly computationally powerful. Therefore, a network with distributed and heterogeneous nodes can provide the best results in minimizing energy consumption if flexibility in task allocation is possible. It must be stressed that these results heavily depend on the choice of particular network models and their parameters.

Proper allocation of resources, transmission, and computing needs to be made to maximize the effectiveness of a fog network. Many different optimization problems and solutions have been proposed related to this issue and energy consumption is often used as an objective, constraint, or both. To solve these problems, varied optimization methods have been harnessed. The most common solutions include convex optimization methods, nature-inspired metaheuristics, and machine learning—particularly deep reinforcement learning. Researchers show that these solutions achieve either the optimal or "good enough" results.

The solutions proposed by the author of this thesis efficiently allocate offloaded computations between fog and cloud nodes. Simulation results in Chapter 4 show that each degree of freedom (allowing FNs to cooperate, utilizing both FNs and cloud, dynamically adjusting clock frequency of FNs servers) improves the efficiency of the network. This corresponds to lowering the overall energy consumption and maximizing the number of requests that are processed within the given delay requirements. Results in Chapter 5 lead to the same conclusions. Moreover, in both Chapter 4 and 5, the author proposes a solution with lower complexity by replacing one optimization step with a simpler heuristic. Results show that they are capable of finding optimal and close-to-optimal solutions. Similarly, Chapter 6 demonstrates that a clever reduction in the size of the search space can significantly reduce the time complexity of the offloading algorithm without noticeable erosion in results.

7.3 Final conclusion

In conclusion, the author believes that the thesis of the dissertation has been proved i.e., there exist optimal solutions to computational task offloading problems in fog networks, minimizing energy consumption while maintaining required levels of latency. The author has formulated such optimization problems, proposed solutions to them, and evaluated these solutions in the extended computer simulation experiments. In the case of particularly computationally complex solutions, heuristic optimization methods have been found and proved to be sufficiently effective.

Appendices

A Appendix

A.1 Summary of optimization methods and results

Below, there is a summary of each work from Tab. 2.4 and a discussion of the results achieved by the authors. Works are divided by the role that energy consumption plays in their optimization problems. Within these sections, they are ordered by year of publication – the oldest first.

A.1.1 Energy as a sole objective

Huang et al. [22] examine a network with a single Mobile Device (MD) offloading parts of tasks (which are modeled as directed graphs) to the cloud (through a Base Station (BS) or Wi-Fi Access Point (AP)) or processing them locally. The optimization problem is to minimize the average energy consumption of the MD while satisfying the stability of the queues and keeping the percentage of tasks that fail to be executed within a given time below a certain threshold. The proposed solution (H1) uses the Lyapunov optimization and 1-opt algorithm which is based on local search by changing only a single subtask allocation variable at a time. The authors compare their solution with the one from [108] and with two trivial baseline solutions: sending all to the cloud and processing all tasks locally. Fig. 4 from [22] compares the solutions and shows that the proposed one achieves the lowest energy consumption and the local processing has the highest energy costs. Meanwhile, Fig. 5 compares the proposed solution with local processing depending on the density of Wi-Fi network deployment. The results show that the denser the network, the more significant the drop in energy consumption is.

Ouesis et al. [38] examine a network with multiple FNs receiving tasks from MDs. They can process these tasks themselves or transmit them (or parts of them) to another MD for processing (the authors call this clustering). The original problem of minimizing energy consumption is non-convex due to the form of latency constraints. After reformulating the problem to be convex, it is solved through the Lagrange method. The proposed solution (C2) is compared with the following baseline solutions: No Clustering – all tasks are processed in the FN they are sent to, Static Clustering – there is a predetermined way in which FNs share the computational load, and Successive Clustering which is a greedy approach that optimizes each task sequentially. Counter-intuitively, the proposed solution has the highest energy costs. It stems from the fact the baseline solutions (especially Static Clustering) have higher percentages of failure to meet the latency constraints of tasks. This effect intensifies with an increasing number of MDs offloading tasks.

Muñoz et al. [24] examine a network with a single MD offloading a task (also partially) to a single FN or processing it locally. Unlike other researchers, Muñoz et al. separately consider UpLink (UL) and DownLink (DL) transmission (C7.1 and C7.2 respectively) and show their optimization prior to proposing the main problem. The objective of the main problem is to minimize the energy spent by the MD while satisfying task delay constraint. The solution (C7.3) involves re-

defining the problem with a smaller number of decision variables (two down from four) and convex optimization methods. They also propose a simpler scenario with no delay constraints in which a closed-form solution (C8) is found. The performance of the proposed solution is checked for different channel gains and delay constraint levels. Four transmission methods (e.g., Multiple-Input and Multiple-Output (MIMO) 4x4) are checked. However, the only baseline solution to compare to is no offloading. The results show that better channel conditions, longer delay constraints, and higher MIMO levels decrease the energy consumption related to transmission and, in turn, the total energy spent by the MD.

Sardatelli et al. [23] examine a network with one or more MDs offloading tasks through one or more BSs to a single FN. Apart from these MDs, there can also be non-offloading MDs which are included in the allocation of channel resources and calculation of interference. The authors propose multiple different solutions to the non-convex problem of minimizing the energy consumption spent by the MD transmission. For a single MD problem, an equivalent convex problem is proposed and solved with a water-filling algorithm (C3). Solutions to a problem with multiple MDs and FNs involve SCA. One centralized (C4) and multiple decentralized solutions are proposed. Decentralized solutions include one based on dual decomposition (C5) and one based on adding slack variables (C6). The latter one also has two versions, one of which utilizes the second-order information. The results achieved through the centralized solution are compared against a single baseline solution, where the FN computing resources are allocated proportionally. Decentralized solutions are compared against themselves. The centralized solution achieves lower energy costs than the baseline one. However, for tasks with high arithmetic intensity, the difference is negligible. All decentralized solutions converge to the same result, with the second-order one converging the fastest.

Deng et al. [49] examine a network with multiple FNs and clouds that share workload offloaded by the MDs. The problem is to minimize power consumption while keeping total delay below a certain level. The proposed solution (MA1) is achieved by approximating the problem (MA1.1) with 3 subproblems solving allocation in the fog tier (MA1.2), in the cloud tier (MA1.3), and choosing transmission parameters between the tiers (MA1.4) separately. Its effectiveness is not compared with any baseline solutions. The results show only the trade-off between delay and power consumption.

Gao et al. [42] examine a network with multiple FNs divided into 2 tiers and a single cloud. Lower-tier FNs receive offloaded workload and can process it locally or send it (or part of it) to higher-tier FNs. Similarly, higher-tier FNs can process workload themselves or offload it to the cloud. The proposed solution (C20) to an optimization problem predicts future workloads and is based on Lyapunov optimization. The authors examine the impact of multiple parameters on their solution: prediction window size, weighting the trade-off between power consumption and backlog in queues, and number of higher-tier FNs chosen for possible allocation. It is compared against the following baseline solutions: no offloading, offloading of all workload to higher-tier FNs, offloading of all workload to the cloud, random. All solutions converge to low power consumption (indistinguishable from 0 W and one another on the graph) with the all-to-cloud approach having the lowest power consumption at the beginning of the simulation. The proposed solution does not seem to optimize the objective function (average power consumption) any better than the simple baseline solutions. The fact that power consumption for all solutions appears to be converging to zero may be an artifact of the chosen model (see Tab. 2.1) in which e.g., transmission to the cloud and processing by the cloud incurs no delay and no power consumption.

You et al. [27] examine a network with multiple MDs and a single FN called edge cloud. MDs can offload their tasks or process them locally. They propose optimization problems for two types of access systems: Time Division Multiple Access (TDMA) and Orthogonal Frequency-Division Multiple Access (OFDMA). First, they solve a simpler problem (C12) with an FN of infinite capacity. For TDMA system they propose two algorithms: an optimal one requiring two-dimensional search (C13) and a heuristic suboptimal one (H2) allocating time slots to MDs using a greedy approach based on priority (function of channel gain and task arithmetic intensity). Equal allocation of time slots is chosen as the baseline solution. Results are shown for different time slot duration, FN computing capacity, and number of users. They show that suboptimal allocation results in energy costs close to that of optimal allocation. They both significantly outperform equal allocation under all examined scenarios. Their results show that energy costs grow much faster than linearly with the number of users. For OFDMA they find an optimal solution (C14) based on relaxation-and-rounding. They also propose a faster suboptimal solution (H3) based on sequential allocation of subchannels to MDs based on priority. As a baseline solution, they use greedy allocation which does not consider the characteristics of offloaded tasks (size, arithmetic intensity). Results are shown for a different number of subchannels and number of users. One can see that suboptimal allocation causes energy costs close to that of optimal allocation while costs achieved via greedy allocation are significantly higher. It is also clear that the time needed to find the optimal allocation is too high for application in real-life scenarios, i.e., close to 6 minutes for 20 users on average.

Feng et al. [29] examine a network with multiple MDs and a single FN. MDs can offload their tasks or process them locally. Unlike most other works, they do not minimize total or average energy consumption but rather minimize the highest (worst-case) consumption of an MD. The baseline solutions include full offloading of all tasks to the FN and the algorithms proposed in [22] and [24]. Their proposed solution (C15, based on the Lagrange method and the subgradient projection) achieves lower energy consumption than [22] and significantly lower energy consumption than full offloading. Their Fig. 3 shows that on average, it has higher energy consumption than the algorithm proposed in [24], however, its worst-case energy cost is significantly lower. Interestingly, another plot (Fig. 4 displaying results of a scenario with twice the number of subcarriers) shows that the algorithm from [24] achieves both lower average energy consumption and lower worst-case energy consumption.

Sun et al. [54] examine a network with multiple MDs getting content from content servers through multiple different transmission paths including through Remote Radio Heads (RRHs) or Device-to-Device (D2D) transmission. The optimization problem is to minimize the overall energy consumption. The proposed solution (M4) is based on deep Reinforcement Learning (RL) and convex optimization. Various parameter sweeps are performed to show their impact on total energy consumption. It is also contrasted with baseline solutions: random, Q-learning-based, D2D only, and cloud only. It achieves lower energy costs than all these baseline solutions.

Kopras et al. [11] examine a network with a single MD offloading tasks to multiple FNs and a cloud. The tasks are modeled as Directed Acyclic Graphs (DAGs) (see Fig. 2.2) with each graph node representing a computational subtask possible to be processed by another FN or cloud. The optimization problem is to minimize energy consumption while satisfying the delay constraints of tasks. Two solutions are proposed depending on the size of the network. Both utilize clustering of nodes to decrease the search space. One (H4) then uses exhaustive search to find the optimal allocations for smaller networks and one (MA6) utilizes Particle Swarm Optimization (PSO) for

larger networks. The chosen baseline solutions are exhaustive search and PSO based on [109]. In a smaller network, the proposed solution has close-to-optimal energy cost when compared with exhaustive search, while being significantly faster to execute. In a larger network, the proposed solution (utilizing PSO) has lower energy costs, shorter execution time, and lower percentage of tasks violating delay constraint than "pure" PSO.

Vu et al. [35] examine a network with multiple MDs and multiple FNs. MDs can offload their tasks to the FNs or a cloud, or process them locally. Unlike in other works, there is a possibility of a direct MD-cloud transmission. They propose three main solutions: one based on relaxation-and-rounding (C21), one based on Branch and Bound (BB) (C22), and one based on Feasibility Finding Benders Decomposition (FFBD) (C23). They are further subdivided: BB based on optimal node selection order (Local Fog Cloud or Local Cloud Fog) and FFBD into Slow and Fast variants. FFBD-Slow/Fast can also include resource allocation results from a relaxation-and-rounding solution. They also consider the following baseline solutions: all offloading and no offloading. The authors show results for a vast range of changing parameters. The no offloading as well as the relaxation-and-rounding fail to find feasible solutions (satisfying latency constraints) in up to 90% cases and up to 30% cases, respectively. FFBD, BB, and all offloading always find feasible solutions, but all offloading tends to have significantly higher energy consumption. There is no straightforward answer as to which of the proposed solutions is the best at achieving the lowest energy consumption while FFBD-Fast (with or without relaxation-and-rounding) tends to be the fastest.

Kopras et al. [9, 10] examine a network with multiple FNs and a cloud. In [10] the problem is to allocate tasks offloaded from MDs between FNs and the cloud and choose the operating frequencies for the FNs. [9] adds another decision variable – choosing FNs for MD-FN transmission. Proposed solutions in both works (MA8, MA10) rely on sequential solving of subproblems and finishing with a matching problem solved by a Hungarian algorithm. Suboptimal, lower complexity versions of their solutions (MA9, MA11) are also proposed with one part of the problem solved by a simple heuristic. In [10] the following baseline solutions are used: offloading all-to-cloud, offloading only between FNs, and no offloading from FN which receives the task. The results show that, depending on chosen parameters and compared with the baselines, the proposed solutions achieve lower energy cost, lower percentage of tasks unable to meet their delay constraint, or both. They show that the difference between results achieved by the full solution and the lower complexity one is small and also highlight the gain from using Dynamic Voltage and Frequency Scaling (DVFS). In [9] the authors also compare their solutions with no offloading and all-to-cloud and find that they achieve lower energy costs. The lower complexity solution achieves virtually identical results to the full one for almost all input parameters. Both works verify that their solution finds the optimum when compared with an exhaustive search and both works show that tasks with high arithmetic intensity are best served by the cloud and those with low intensity – by the FNs.

A.1.2 Energy as one of a few objectives

Do et al. [48] examine a network with multiple FNs and a single cloud. The cloud can stream video content to the FN providing utility to the end-users in the process. The stated problem is to maximize this utility subtracting costs related to power consumption in the cloud. This is solved iteratively using a proximal algorithm and Alternating Direction Method of Multipliers (ADMM) (C1). The authors do not compare their solution to any baseline solutions. All their plots show

performance as a function of algorithm iterations.

Dinh et al. [26] examine a network with a single MD. It can offload its tasks to multiple FNs or process them locally. The results are shown for various numbers of tasks and nodes, as well as for varying weights assigned to delay and energy consumption in the objective function. Baseline solutions include exhaustive search, local processing, random assignment, and offloading all tasks to the cloud (which is added only in the model for this baseline solution). Linear Programming (LP) based algorithm (C9) causes higher costs (weighted sum of energy and delay) at larger numbers of offloaded tasks than the SemiDefinite Relaxation (SDR) based algorithms (C10, C11), which achieve close to optimal solutions. They all significantly outperform local processing and random assignment. Comparison with all-to-cloud depends on the number of offloaded tasks - the lower the number, the bigger the difference in cost in favor of LP and SDR based algorithms.

Xu et al. [39] examine a network with a single FN and a single cloud. The FN is powered by battery, renewable energy, as well as non-renewable energy, and can turn its servers from active to inactive. It gets offloaded workload and can process it in one of its servers or send it to the cloud. The problem of minimizing the total expected cost is formulated as a Markov Decision Process (MDP) and solved using RL. The proposed solution (ML1) utilizes Post-Decision State (PDS) and is compared with Q-learning as well as with three simple schemes: one that utilizes all available battery energy in each time slot and two with fixed power consumption. The results show that the PDS-based RL achieves the lowest total cost, with Q-learning having the second lowest cost. Interestingly, both PDS-based RL and Q-learning have higher costs related to delay than the three other methods. However, they are significantly better at saving battery and minimizing costs related to usage of non-renewable energy, which offsets longer delays.

Liu et al. [28] examine a network with a single FN and a single cloud. There are multiple Energy Harvesting (EH)-capable MDs offloading tasks to an FN or a cloud (through FN) or processing them locally. MDs are connected by social ties which influence the costs related to offloading. Their proposed solution (MA3) uses partial penalization to transform the original generalized Nash Equilibrium Problem (NEP) into a "classical" NEP. Then, using Karush–Kuhn–Tucker (KKT) conditions they transform the problem into a system of non-smooth equations. Finally, they use the semi-smooth Newton method with Armijo line search to solve the system. The authors focus on showing the impact of varying task arrival rates on values such as task execution rates and costs spent on different parts of the network. The baseline solutions (whose results are shown only on a single plot) are taken from other works namely [23] and [25]. Implementation details are not provided. Liu et al. show that their solution has a lower average cost (delay and dropped tasks penalty increased (penalized) by corresponding costs of socially linked MDs) than those proposed in [23, 25].

Cui et al. [30] examine a network with a single FN located at a BS and multiple small cell BSs which relay tasks from MDs. MDs can also process tasks locally. The multi-objective problem (delay and MDs energy consumption) is solved using modified Non-dominated Sorting Genetic Algorithm II (NSGAII) (MH2). Results are plotted for various values of parameters: number of small cell BSs, task arrival rate, MD transmission power, FN computing capability, and MD computing capability. They show the impacts on energy consumption, delay, and the trade-off between them. The authors do not compare the achieved results with any baseline solutions.

Wang et al. [53] examine a network with a single FN and multiple MDs. MDs can offload tasks to an FN, to other MD, or process them locally. The authors propose two solutions, one based on deep RL scheduling (ML2) and one based on Deep Dynamic Scheduling (DDS) with deep Q-

learning (ML3). They choose the following baseline solutions: no offloading, offloading all tasks to other MDs, and offloading all tasks to the FN. The objective function is to minimize a weighted sum of energy consumption, delay, and penalties for exceeding task deadlines. Meanwhile, the results show energy and delay separately. The proposed solutions have lower energy costs than the baseline solutions with "no offloading" one having the highest costs. On the other hand, "no offloading" and "offloading to MDs" achieve lower delay than the proposed solutions except for tasks with large size, either in bits or number of instructions. From the two proposed solutions, DDS achieves slightly lower delay and energy costs than deep RL scheduling.

Djemai et al. [55] examine a network with multiple FNs and cloud nodes. MDs receive (from sensors and actuators) tasks modeled as directed graphs, and each node of these graphs represents a computational task that can be processed locally or in the fog or cloud nodes. The authors propose a solution (MH1) based on discrete PSO and compare it with 6 different baseline solutions including 4 simple ones like "cloud only" and 2 complex ones: binary PSO and the one from [16]. While the optimization problem is to minimize energy consumption increased by violations of delay constraints, these values are only plotted separately. The proposed discrete PSO achieves lower energy costs and delay violations than the "simple" solutions. Comparison with binary PSO and the solution from [16] is less clear. Depending on the size of tasks, it can achieve both higher and lower energy consumption and delay violations.

Abbasi et al. [56] examine the same system as proposed by Deng et al. [49]. Unlike [49], they do not clearly state their objective function. They also miss some of the constraints used in [49] e.g., those requiring that offloaded tasks are processed. Still, their solution (MH4) to the problem is drastically different – they use NSGAII. They do not compare their solution against baseline solutions or those found in other works. Instead, they show results for three scenarios of possible workload allocation: only to FNs, only to clouds, and both to FNs and cloud. There does not appear to be a clear gain from utilizing nodes in both tiers, as only FNs have the lowest delay and only clouds have the lowest energy consumption with both FNs and clouds achieving intermediate results.

He et al. [31] examine a network with multiple MDs and one or more FNs. MDs can offload their tasks or process them locally. However, the focus of optimization is on a single MD user whose offloaded tasks can be detected by adversaries. Scenarios with both colluding (C18) and non-colluding (C19) adversaries are examined. Similar solutions based on Lyapunov optimization are used for both scenarios to minimize energy consumption while preventing the detection of "feature" tasks. Baseline solutions include privacy-ignoring solution and "naive" solution which prohibits the offloading of "feature" tasks. Results show the performance gain (lower energy cost) of the proposed solution when compared with the naive one as well as the price of privacy (higher cost) when compared with the privacy-ignoring one. They also show that a higher number of FNs to which tasks can be offloaded decreases costs for all solutions.

Nath and Wu [33] examine a network with multiple MDs, one or more FNs, and a cloud. MDs can offload their tasks or process them locally and FNs can cache tasks that they process or fetch them from the cloud or other FNs. They propose two solutions based on Deep Deterministic Policy Gradient (DDPG): one for a network with a single FN (ML4) and a decentralized one for multiple cooperating FNs (ML5). For a network with a single FN, they compare the average cost (weighted sum of delay, energy, and fetching) with a total of 8 baseline solutions. For all input parameters, their proposed solution achieves the lowest cost. They also compared their solution with a non-cooperative one and a centralized cooperative one for a network with multiple FNs.

Understandably, their solution has higher costs than the centralized one and significantly lower than the non-cooperative one.

Roy et al. [57] examine a network with multiple MDs, multiple FNs divided into two hierarchical tiers (dew and edge) and a central cloud. Each task (called subservice) can be offloaded to one of the FNs or cloud. The proposed solution (H5) based on a Genetic Algorithm (GA) and adaptive PSO is compared with solutions proposed in other works: one based on self-adaptive PSO [110], one based on binary PSO [112], one based on GA (NSGAII) [56], and one from [113]. Results of simulations show that the proposed solution achieves higher fitness (chosen objective to be maximized) than the baseline solutions. It also achieves both lower energy consumption and delay (i.e., parameters used to calculate fitness).

Zhang et al. [61] examine a network with multiple MDs and a single FN consisting of multiple computational servers of two different types. MDs can offload workload through RRHs or process it locally. The objective is to minimize the energy consumption divided by the total size of the processed workload (the authors call it the number of tasks rather than the size but it is given in bits, and the whole model follows that of computational load rather than discrete tasks – see Section 2.1.3). The proposed multi-step solution (MA5) utilizes Lyapunov optimization to split the problem into 4 sequentially-solved subproblems. The results achieved by the proposed solution are compared with those achieved by the following baseline solutions: no offloading, random, exhaustive search, and one where all steps follow the one from the proposed solution but the two types of FN servers cannot share computation resources. The results show that the proposed solution achieves close-to-optimal energy efficiency (when compared with an exhaustive search) and significantly better efficiency than no offloading or random offloading. The difference resulting from different FN servers not being able to cooperate has no impact at lower arriving workloads and a small impact on energy efficiency as well as a significant impact on average delay at higher workloads.

Vakilian et al. [43–45] examine a network with multiple FNs and a cloud sharing workload offloaded from MDs. Differences between the models and optimization problems of these three works are relatively small. [43] uses a solution based on Splitting Conic Solver (SCS) (C17), and in the other works metaheuristics are chosen: Artificial Bee Colony (ABC) in [44] (MH5), and the cuckoo algorithm in [45] (MH6). ABC is shown to converge to optimal values found by SCS. Meanwhile, the cuckoo algorithm is shown to have lower energy consumption and delay than the solution proposed in [146]. All three works show that an increased number of cooperating FNs decreases the costs (weighted sum of energy consumption and delay) for a given workload.

Abdel-Basset et al. [46] examine a network with a single FN that allocates offloaded tasks to Virtual Machines (VMs). While the authors do not clearly state their optimization problem, it can be assumed that their objective is to minimize fitness – a weighted sum of energy consumption and makespan (highest delay among VMs). The authors propose a solution based on Marine Predators Algorithm (MPA) adapted to solve the discrete problem of task allocation. In total, they propose 3 versions: MPA (MH7), modified MPA (MH8) which changes the process of updating positions, and improved modified MPA (MH9) which adds ranking-based reinitialization and mutation. Proposed algorithms are compared with other metaheuristics: sine cosine algorithm, whale optimization algorithm, salp swarm algorithm, equilibrium optimizer algorithm, and GA. The results show that improved modified MPA achieves the best results (lowest fitness, energy consumption, makespan, and CO2 emissions) for all studied scenarios. The second best is GA, better than the proposed modified MPA and MPA, which in turn perform better than the rest of the tested

metaheuristics.

Bai and Qian [34] examine a network with multiple MDs and multiple FNs. MDs can offload their tasks to the FNs or a cloud, or process them locally. The problem of allocation of tasks to nodes as well as channel and computing resources to tasks is solved using deep RL-based Advantage Actor-Critic (A2C) algorithm (ML6). The chosen baseline solution is random assignment. The proposed A2C algorithm at the beginning of the simulation achieves a similar cost (weighted sum of delay and MD energy) to the random assignment but with more episodes its cost drops to less than 20% of that of the random assignment. They also show that the cost decreases with a lower number of MDs and a higher number of FNs, but there are diminishing returns after 20 FNs. The plot that shows these relations is inconsistent with its description in the text – there are data points that do not correspond to the described values and there is an unexplained dip at 40 MDs.

Ghanavati et al. [62] examine a network with a broker, multiple FNs, and a cloud. The broker (gateway or AP) receives jobs (sets of tasks) from MDs and offloads them to FNs or a cloud. They propose an Ant Mating Optimization (AMO) algorithm (MH10) to solve the problem of minimizing the cost – the weighted sum of energy spent by FNs and total delays for each job. They choose the following metaheuristics used in other works as a comparison: bees life algorithm [114], GA [115], and PSO utilizing fuzzy logic [116]. The results show that the proposed solution achieves lower cost than the ones based on other metaheuristics. Its improvement in reducing delays is more significant than in reducing energy consumption. The bees life algorithm achieves the second-best results.

Sun and Chen [47] examine a network with multiple FNs and MDs, each MD connected to a single FN. An MD can offload a task to an FN by paying it a certain cost decided by the FN. The optimization problem for each FN is to maximize its utility, i.e., price paid by MDs for offloading tasks subtracting various costs which include energy. The baseline solutions include one that is Contract-based under Symmetric information (CS), one that is based on a Stackelberg game, and one that is based on linear pricing. The CS one achieves the highest utility for an FN. As it has all the information about offloading MDs, it can extract maximum revenue from them so that their utilities are zero. It can be seen as an upper bound. The proposed solution (MA12) has, therefore, lower utility than CS, but higher than the other baseline solutions. It also achieves the highest utility of MDs.

A.1.3 Energy only as a constraint

Mao et al. [25] examine a network with a single MD capable of EH and an FN. The optimization problem is to minimize the average over time delay increased by dropped task penalty while constrained by battery level of MD. In the proposed solution (MA2), the authors utilize Lyapunov optimization and find the optimal values for each decision variable step-by-step. They compare their method with three baseline solutions: no offloading, all offloading, and dynamic offloading which chooses whether to offload based on what decision incurs lower delay. For all studied scenarios the proposed solution achieves the lowest cost and the dynamic offloading achieves the second lowest. All offloading achieves the lowest average delay is for most scenarios. However, it is compensated by higher dropped task percentage – up to 50% for a scenario with large distance (80 m) between the MD and the FN caused by lower channel gain.

Wang and Chen [58] examine a network with multiple MDs offloading tasks to a single FN or

processing them locally. The goal of minimizing the total delay while satisfying the delay and energy constraints of each task is solved using Hybrid Genetic Simulated Annealing (HGSA) (MH3) – a combination of GA and simulated annealing. The baseline solutions include no offloading, full offloading and a version of HGSA used in [111]. The results show that the proposed solution achieves lower total delay than simple no offloading and full offloading approaches. However, it has slightly higher total delay than the solution from [111] and the difference between the two increases with an increasing number of MDs. On the other hand, the proposed solution has lower total energy consumption than the one from [111]. Still, it is the delay, not the energy, that is the optimization objective.

Chen et al. [40] examine a network with multiple FNs which share workload offloaded by the MDs. The problem is to minimize delay over time with constraints on energy over time as well as per time slot energy and delay (see the definition of "over time" in Section 2.3.3). The authors propose a centralized optimization algorithm (C16) using Lyapunov optimization and a decentralized one (G1) reaching a Nash equilibrium through a best-response algorithm. The chosen baseline solutions are: one without FN-FN offloading, one ignoring over time energy constraint, and one transforming over time energy constraint into a stricter per time slot one. Results show that both centralized and decentralized algorithms can satisfy over time energy constraints. They have a lower average delay than the baseline solutions except for the energy-ignoring one. The performance gain of the centralized solution in relation to the decentralized one is clearly presented. It grows with higher variation in task arrival rate.

Bian et al. [36, 37] examine a system with multiple MDs and a single FN. The network performs distributed training and the FN dynamically chooses MDs for each round of training. Their proposed solution (MA7) combines Lyapunov optimization with Upper Confidence Bound (UCB)-based graphical bandit learning. They compare their solution with baseline solutions: random MD selection and "classical" UCB-based graphical bandit algorithm. Moreover, they do propose 3 slightly modified versions of their solution: one that does not utilize the graph information, one that ignores the energy constraint, and one that ignores the fairness constraint (MD minimum selection rate). All variations of the proposed solution achieve lower average latency than baseline solutions with the fairness-ignoring one having the lowest average latency and energy consumption by a large margin. However, it expectedly fails to reach the required fairness threshold. The solution without the graph information produces only slightly inferior results to the main one. The authors also study the effects of multiple network parameters and weights used by the solution on results such as average latency and testing accuracy.

As shown in this section, some authors chose algorithms proposed in other research works as baseline solutions. In a vast majority of cases (except for when [29] compares itself with [24], and to a lesser extent when [58] compares itself with [111]) the proposed solutions show improvement over the ones cited. Still, most works use trivial baseline solutions such as no offloading or random assignment,

8 Bibliography

[1] C. Freitag, M. Berners-Lee, K. Widdicks, B. Knowles, G. S. Blair, and A. Friday, "The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations," *Patterns*, vol. 2, no. 9, p. 100340, 2021.

[2] F. Montevecchi, T. Stickler, R. Hintemann, and S. Hinterholzer, "Energy-efficient cloud computing technologies and policies for an eco-friendly cloud market. final study report." EUROPEAN COMMISSION, Directorate-General for Communications Networks, Content and Technology, Tech. Rep., 2020. [Online]. Available: https://data.europa.eu/doi/10.2759/3320

[3] R. Rogers, "AI's energy demands are out of control. welcome to the internet's Hyper-Consumption era," July 2024, last accessed on 19.09.2024. [Online]. Available: https://www.wired.com/story/ai-energy-demands-water-impact-internet-hyper-consumption-era/

[4] S. Luccioni, Y. Jernite, and E. Strubell, "Power hungry processing: Watts driving the cost of ai deployment?" in *FAccT '24: Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 85–99. [Online]. Available: https://doi.org/10.1145/3630106.3658542

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 13–16.

[6] B. Kopras, F. Idzikowski, and H. Bogucka, "A survey on reduction of energy consumption in fog networks - communications and computations," *Sensors*, vol. 24, no. 18, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/18/6064

[7] H. Bogucka, B. Kopras, F. Idzikowski, B. Bossy, and P. Kryszkiewicz, "Green time-critical fog communication and computing," *IEEE Communications Magazine*, vol. 61, no. 12, pp. 40–45, 2023.

[8] H. Bogucka and B. Kopras, "Uberization of telecom networks for cost-efficient communication and computing," *IEEE Communications Magazine*, vol. 61, no. 7, pp. 74–80, July 2023.

[9] B. Kopras, F. Idzikowski, B. Bossy, P. Kryszkiewicz, and H. Bogucka, "Communication and computing task allocation for energy-efficient fog networks," *Sensors*, vol. 23, no. 2, 2023.

[10] B. Kopras, B. Bossy, F. Idzikowski, P. Kryszkiewicz, and H. Bogucka, "Task allocation for energy optimization in fog computing networks with latency constraints," *IEEE Transactions on Communications*, vol. 70, no. 12, pp. 8229–8243, 2022.

[11] B. Kopras, F. Idzikowski, W.-C. Chen, T.-J. Wang, C.-T. Chou, and H. Bogucka, "Latency-aware virtual network embedding using clusters for green fog computing," in *2020 IEEE Globecom Workshops*, 2020.

[12] P. Kryszkiewicz, F. Idzikowski, B. Bossy, B. Kopras, and H. Bogucka, "Energy savings by task offloading to a fog considering radio front-end characteristics," in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2019.

[13] B. Kopras, F. Idzikowski, and P. Kryszkiewicz, "Power consumption and delay in wired parts of fog computing networks," in *2019 IEEE Sustainability through ICT Summit (StICT)*, Montreal, Canada, June 2019.

[14] H. Bogucka, F. Idzikowski, P. Kryszkiewicz, B. Bossy, and B. Kopras, "Mgła – nowa architektura sieci dla zrównoważonego rozwoju Internetu Rzeczy," *Przegląd Telekomunikacyjny - Wiadomości Telekomunikacyjne*, no. 7, pp. 505–511, 2019.

[15] B. Kopras and F. Idzikowski, "Porównanie efektywności energetycznej mgły i chmury obliczeniowej - przegląd," *Przegląd Telekomunikacyjny - Wiadomości Telekomunikacyjne*, no. 6, pp. 307–310, 2019.

[16] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in fog-cloud computing paradigm," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, pp. 1222–1228.

[17] P. Hosseinioun, M. Kheirabadi, S. R. Kamel Tabbakh, and R. Ghaemi, "A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm," *Journal of Parallel and Distributed Computing*, vol. 143, pp. 88–96, 2020.

[18] A. U. Rehman, Z. Ahmad, A. I. Jehangiri, M. A. Ala'Anzy, M. Othman, A. I. Umar, and J. Ahmad, "Dynamic energy efficient resource allocation strategy for load balancing in fog environment," *IEEE Access*, vol. 8, pp. 199 829–199 839, 2020.

[19] OpenFog Consortium, "Openfog reference architecture for fog computing, OPFRA001.020817," 2017. [Online]. Available: https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf

[20] S. B. Nath, H. Gupta, S. Chakraborty, and S. K. Ghosh, "A survey of fog computing and communication: Current researches and future directions," *CoRR*, vol. abs/1804.04365, 2018. [Online]. Available: http://arxiv.org/abs/1804.04365

[21] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 416–464, Q1 2018.

[22] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, 2012.

[23] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.

[24] O. Muñoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738–4755, 2015.

[25] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[26] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

[27] C. You, K. Huang, H. Chae, and B. H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[28] L. Liu, Z. Chang, and X. Guo, "Socially aware dynamic computation offloading scheme for fog computing system with energy harvesting devices," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1869–1879, June 2018.

[29] J. Feng, L. Zhao, J. Du, X. Chu, and F. R. Yu, "Energy-efficient resource allocation in fog computing supported IoT with min-max fairness guarantees," in *2018 IEEE International Conference on Communications (ICC)*, 2018.

[30] L. Cui, C. Xu, S. Yang, J. Z. Huang, J. Li, X. Wang, Z. Ming, and N. Lu, "Joint optimization of energy consumption and latency in mobile edge computing for Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4791–4803, 2019.

[31] X. He, R. Jin, and H. Dai, "PEACE: Privacy-preserving and cost-efficient task offloading for mobile-edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1814–1824, 2020.

[32] A. Shahidinejad and M. Ghobaei-Arani, "Joint computation offloading and resource provisioning for edge-cloud computing environment: A machine learning-based approach," *Software: Practice and Experience*, vol. 50, no. 12, pp. 2212–2230, 2020.

[33] S. Nath and J. Wu, "Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems," *Intelligent and Converged Networks*, vol. 1, no. 2, pp. 181–198, 2020.

[34] W. Bai and C. Qian, "Deep reinforcement learning for joint offloading and resource allocation in fog computing," in *2021 IEEE 12th International Conference on Software Engineering and Service Science (ICSESS)*, 2021, pp. 131–134.

[35] T. T. Vu, D. N. Nguyen, D. T. Hoang, E. Dutkiewicz, and T. V. Nguyen, "Optimal energy efficiency with delay constraints for multi-layer cooperative fog computing networks," *IEEE Transactions on Communications*, vol. 69, no. 6, pp. 3911–3929, 2021.

[36] S. Bian, S. Wang, Y. Tang, and Z. Shao, "Social-aware edge intelligence: A constrained graphical bandit approach," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 6372–6377.

[37] S. Wang, S. Bian, Y. Tang, and Z. Shao, "Social-aware edge intelligence: A constrained graphical bandit approach," ShanghaiTech University, Tech. Rep., 2022. [Online]. Available: http://faculty.sist.shanghaitech.edu.cn/faculty/shaozy/GC22-GRIND.pdf

[38] J. Oueis, E. C. Strinati, S. Sardellitti, and S. Barbarossa, "Small cell clustering for efficient distributed fog computing: A multi-user case," in *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, Sep. 2015.

[39] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 3, pp. 361–373, Sep. 2017.

[40] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1619–1632, 2018.

[41] F. Murtaza, A. Akhunzada, S. ul Islam, J. Boudjadar, and R. Buyya, "QoS-aware service provisioning in fog computing," *Journal of Network and Computer Applications*, vol. 165, p. 102674, 2020.

[42] X. Gao, X. Huang, S. Bian, Z. Shao, and Y. Yang, "Pora: Predictive offloading and resource allocation in dynamic fog computing systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 72–87, 2020.

[43] S. Vakilian and A. Fanian, "Enhancing users' quality of experienced with minimum energy consumption by fog nodes cooperation in Internet of Things," in *2020 28th Iranian Conference on Electrical Engineering (ICEE)*, 2020.

[44] S. Vakilian, S. V. Moravvej, and A. Fanian, "Using the Artificial Bee Colony (ABC) algorithm in collaboration with the fog nodes in the internet of things three-layer architecture," in *2021 29th Iranian Conference on Electrical Engineering (ICEE)*, 2021, pp. 509–513.

[45] ——, "Using the cuckoo algorithm to optimizing the response time and energy consumption cost of fog nodes by considering collaboration in the fog layer," in *2021 5th International Conference on Internet of Things and Applications (IoT)*, 2021.

[46] M. Abdel-Basset, R. Mohamed, M. Elhoseny, A. K. Bashir, A. Jolfaei, and N. Kumar, "Energy-aware marine predators algorithm for task scheduling in IoT-based fog computing applications," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 5068–5076, 2021.

[47] Z. Sun and G. Chen, "Contract-Optimization Approach (COA): A new approach for optimizing service caching, computation offloading, and resource allocation in mobile edge computing network," *Sensors*, vol. 23, no. 10, 2023.

[48] C. T. Do, N. H. Tran, C. Pham, M. G. R. Alam, J. H. Son, and C. S. Hong, "A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing," in *2015 International Conference on Information Networking (ICOIN)*, Jan. 2015, pp. 324–329.

[49] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.

[50] S. Sarkar and S. Misra, "Theoretical modelling of fog computing: a green computing paradigm to support IoT applications," *IET Networks*, vol. 5, no. 2, pp. 23–29, 2016.

[51] W. Zhang, Z. Zhang, and H.-C. Chao, "Cooperative fog computing for dealing with big data in the Internet of Vehicles: Architecture and hierarchical resource management," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 60–67, 2017.

[52] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of Internet of Things," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 46–59, Jan. 2018.

[53] Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo, "Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 976–986, 2019.

[54] Y. Sun, M. Peng, and S. Mao, "Deep reinforcement learning-based mode selection and resource management for green fog radio access networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1960–1971, 2019.

[55] T. Djemai, P. Stolf, T. Monteil, and J.-M. Pierson, "A discrete particle swarm optimization approach for energy-efficient IoT services placement over fog infrastructures," in *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*, 2019, pp. 32–40.

[56] M. Abbasi, E. Pasand, and M. Khosravi, "Workload allocation in IoT-fog-cloud architecture using a multi-objective genetic algorithm," *Journal of Grid Computing*, vol. 18, pp. 43–56, 03 2020.

[57] A. Roy, S. Midya, K. Majumder, and S. Phadikar, "Distributed resource management in dew based edge to cloud computing ecosystem: A hybrid adaptive evolutionary approach," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 8, p. e4018, 2020.

[58] Q. Wang and S. Chen, "Latency-minimum offloading decision and resource allocation for fog-enabled Internet of Things networks," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 12, p. e3880, 2020.

[59] N. Khumalo, O. Oyerinde, and L. Mfupe, "Reinforcement learning-based computation resource allocation scheme for 5G fog-radio access network," in *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, 2020, pp. 353–355.

[60] P. Gazori, D. Rahbari, and M. Nickray, "Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach," *Future Generation Computer Systems*, vol. 110, pp. 1098–1115, 2020.

[61] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, and F. Tian, "Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3282–3299, 2020.

[62] S. Ghanavati, J. Abawajy, and D. Izadi, "An energy aware task scheduling model using ant-mating optimization in fog computing environment," *IEEE Transactions on Services Computing*, vol. 15, no. 4, pp. 2007–2017, 2022.

[63] M. Othman and S. Hailes, "Power conservation strategy for mobile computers using load sharing," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 2, no. 1, pp. 44–51, Jan. 1998.

[64] A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning, "Saving portable computer battery power through remote process execution," *Mobile Computing and Communications Review*, vol. 2, no. 1, p. 19–26, Jan. 1998.

[65] T. N. Gia, M. Jiang, A. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "Fog computing in healthcare Internet of Things: A case study on ECG feature extraction," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, Oct. 2015, pp. 356–363.

[66] H. Dubey, J. Yang, N. Constant, A. M. Amiri, Q. Yang, and K. Makodiya, "Fog data: Enhancing telehealth big data through fog computing," in *Proceedings of the ASE BigData & SocialInformatics 2015*, ser. ASE BD& SI '15. New York, NY, USA: ACM, 2015, pp. 14:1–14:6.

[67] A. Vakali and G. Pallis, "Content delivery networks: status and trends," *IEEE Internet Computing*, vol. 7, no. 6, pp. 68–74, Nov. 2003.

[68] B. Bossy, P. Kryszkiewicz, and H. Bogucka, "Energy-efficient OFDM radio resource allocation optimization with computational awareness: A survey," *IEEE Access*, vol. 10, p. 94100–94132, 2022.

[69] G. Miao, N. Himayat, and G. Y. Li, "Energy-Efficient Transmission in Frequency-Selective Channels," in *IEEE Global Telecommunications Conference 2008 (IEEE GLOBECOM 2008)*, Nov. 2008.

[70] ——, "Energy-Efficient Link Adaptation in Frequency-Selective Channels," *IEEE Transactions on Communications*, vol. 58, no. 2, pp. 545–554, Feb. 2010.

[71] C. Isheden and G. P. Fettweis, "Energy-Efficient Multi-Carrier Link Adaptation with Sum Rate-Dependent Circuit Power," in *IEEE Global Telecommunications Conference 2010 (IEEE GLOBECOM 2010)*, Dec. 2010.

[72] ——, "Energy-Efficient Link Adaptation with Transmitter CSI," in *IEEE Wireless Communications and Networking Conference 2011 (IEEE WCNC 2011)*, Mar. 2011, pp. 1381–1386.

[73] B. Bossy and H. Bogucka, "Optimization of Energy Efficiency in Computationally-Aware Adaptive OFDM Systems," in *IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications 2016 (IEEE PIMRC 2016)*, Sep. 2016.

[74] P. Kryszkiewicz and A. Kliks, "Modeling of Power Consumption by Wireless Transceivers for System Level Simulations," in *European Wireless 2017; 23th European Wireless Conference*, May 2017.

[75] P. Kryszkiewicz, A. Kliks, Ł. Kułacz, and B. Bossy, "Stochastic Power Consumption Model of Wireless Transceivers," *Sensors*, vol. 20, no. 17, p. 4704, Aug 2020.

[76] K. Gomez, T. Rasheed, R. Riggio, D. Miorandi, C. Sengul, and N. Bayer, "Achilles and the Tortoise: Power Consumption in IEEE 802.11n and IEEE 802.11g Networks," in *2013 IEEE Online Conference on Green Communications (OnlineGreenComm)*, Oct. 2013, pp. 20–26.

[77] A. Rice and S. Hay, "Measuring mobile phone energy consumption for 802.11 wireless networking," *Pervasive and Mobile Computing*, vol. 6, no. 6, pp. 593–606, 2010, Special Issue PerCom 2010.

[78] D. Halperin, B. Greenstein, A. Sheth, and D. Wetherall, "Demystifying 802.11n Power Consumption," in *International Conference on Power Aware Computing and Systems 2010 (HotPower 2010)*, Oct. 2010.

[79] A. Mezghani, N. Damak, and J. A. Nossek, "Circuit Aware Design of Power-Efficient Short Range Communication Systems," in *7th International Symposium on Wireless Communication Systems 2010 (ISWCS 2010)*, Sep. 2010, pp. 869–873.

[80] A. Mezghani and J. A. Nossek, "Power Efficiency in Communication Systems from a Circuit Perspective," in *IEEE International Symposium of Circuits and Systems 2011 (IEEE ISCAS 2011)*, May 2011, pp. 1896–1899.

[81] ——, "Modeling and Minimization of Transceiver Power Consumption in Wireless Networks," in *International ITG Workshop on Smart Antennas 2011 (WSA 2011)*, Feb. 2011.

[82] Y. Chen, J. A. Nossek, and A. Mezghani, "Circuit-Aware Cognitive Radios for Energy-Efficient Communications," *IEEE Wireless Communications Letters*, vol. 2, no. 3, pp. 323–326, June 2013.

[83] Y. Li, B. Bakkaloglu, and C. Chakrabarti, "A System Level Energy Model and Energy-Quality Evaluation for Integrated Transceiver Front-Ends," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 1, pp. 90–103, Jan. 2007.

[84] R. V. R. Kumar and J. Gurugubelli, "How green the LTE technology can be?" in *2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology 2011 (Wireless VITAE 2011)*, Feb. 2011.

[85] E. Björnson, L. Sanguinetti, J. Hoydis, and M. Debbah, "Optimal Design of Energy-Efficient Multi-User MIMO Systems: Is Massive MIMO the Answer?" *IEEE Transactions on Wireless Communications*, vol. 14, no. 6, pp. 3059–3075, June 2015.

[86] E. Björnson, M. Matthaiou, and M. Debbah, "Massive MIMO with non-ideal arbitrary arrays: Hardware scaling laws and circuit-aware design," *IEEE Transactions on Wireless Communications*, vol. 14, no. 8, pp. 4353–4368, Aug. 2015.

[87] P. H. Y. Wu, "On the Complexity of Turbo Decoding Algorithms," in *IEEE 53rd Vehicular Technology Conference 2001 (IEEE VTC 2001-Spring)*, vol. 2, May 2001, pp. 1439–1443 vol.2.

[88] C. Desset and A. Fort, "Selection of Channel Coding for Low-Power Wireless Systems," in *IEEE 57th Semiannual Vehicular Technology Conference 2003 (IEEE VTC 2003-Spring)*, vol. 3, Apr. 2003, pp. 1920–1924 vol.3.

[89] J. Lorandel, J. Prévotet, and M. Hélard, "Dynamic Power Evaluation of LTE Wireless Baseband Processing on FPGA," in *Conference on Design and Architectures for Signal and Image Processing 2015 (DASIP 2015)*, Sep. 2015.

[90] ——, "Fast Power and Performance Evaluation of FPGA-Based Wireless Communication Systems," *IEEE Access*, vol. 4, pp. 2005–2018, 2016.

[91] M. Maaz, J. Lorandel, P. Mary, J. Prévotet, and M. Hélard, "Energy efficiency analysis of hybrid-ARQ relay-assisted schemes in LTE-based systems," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, pp. 1–13, Dec. 2016.

[92] W. Van Heddeghem, F. Idzikowski, W. Vereecken, D. Colle, M. Pickavet, and P. Demeester, "Power consumption modeling in optical multilayer networks," *Photonic Network Communications*, vol. 24, no. 2, pp. 86–102, 2012.

[93] C. Gunaratne, K. Christensen, B. Nordman, and S. Suen, "Reducing the energy consumption of ethernet with Adaptive Link Rate (ALR)," *IEEE Transactions on Computers*, vol. 57, no. 4, pp. 448–461, Apr. 2008.

[94] A. Morea, O. Rival, N. Brochier, and E. Le Rouzic, "Datarate adaptation for night-time energy savings in core networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 31, no. 5, pp. 779–785, Mar. 2013.

[95] J. C. C. Restrepo, C. G. Gruber, and C. Mas Machuca, "Energy profile aware routing," in *Proc of the ICC workshop on Green Communications, Dresden, Germany*, June 2009.

[96] F. Idzikowski, E. Bonetto, L. Chiaraviglio, A. Cianfrani, A. Coiro, R. Duque, F. Jiménez, E. Le Rouzic, F. Musumeci, W. Van Heddeghem, J. López Vizcaíno, and Y. Ye, "TREND in Energy-Aware Adaptive Routing Solutions," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 94–104, Nov. 2013.

[97] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, 1992.

[98] S. Park, J. Park, D. Shin, Y. Wang, Q. Xie, M. Pedram, and N. Chang, "Accurate modeling of the delay and energy overhead of dynamic voltage and frequency scaling in modern microprocessors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 5, pp. 695–708, 2013.

[99] H. Lin, Y. Shih, A. Pang, and C. Chou, "Virtual local-hub: A service platform on the edge of networks for wearable devices," *IEEE Network*, vol. 32, no. 4, pp. 114–121, July 2018.

[100] S. J. Edwards, S. D. Forsyth, J. Stanback, and A. Saremba, "Standard: Portable Game Notation specification and implementation guide," 1994, last accessed on 17.07.2024. [Online]. Available: http://www.saremba.de/chessgml/standards/pgn/pgn-complete.htm

[101] B. Varghese, N. Wang, D. Bermbach, C.-H. Hong, E. D. Lara, W. Shi, and C. Stewart, "A survey on edge performance benchmarking," *ACM Comput. Surv.*, vol. 54, no. 3, apr 2021.

[102] T. Pfandzelter and D. Bermbach, "Towards a benchmark for fog data processing," in *2023 IEEE International Conference on Cloud Engineering (IC2E)*, 2023, pp. 92–98.

[103] J. Martins and A. Ning, *Engineering Design Optimization*. Cambridge University Press, 10 2021.

[104] A. Avan, A. Azim, and Q. H. Mahmoud, "A state-of-the-art review of task scheduling for edge computing: A delay-sensitive application perspective," *Electronics*, vol. 12, no. 12, 2023.

[105] I. Osman and G. Laporte, "Metaheuristics: A bibliography," *Annals of Operational Research*, vol. 63, pp. 513–628, Oct. 1996.

[106] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Transactions on Services Computing*, pp. 1–1, 2018.

[107] N. N. Khumalo, O. O. Oyerinde, and L. Mfupe, "Reinforcement learning-based resource management model for fog radio access network architectures in 5G," *IEEE Access*, vol. 9, pp. 12 706–12 716, 2021.

[108] S. Ou, K. Yang, and J. Zhang, "An effective offloading middleware for pervasive services on mobile devices," *Pervasive and Mobile Computing*, vol. 3, no. 4, pp. 362–385, 2007, middleware for Pervasive Computing.

[109] Z. Zhang, X. Cheng, S. Su, Y. Wang, K. Shuang, and Y. Luo, "A unified enhanced particle swarm optimization-based virtual network embedding algorithm," *International Journal of Communication Systems*, vol. 26, no. 8, pp. 1054–1073, 2013.

[110] X. Zuo, G. Zhang, and W. Tan, "Self-adaptive learning pso-based deadline constrained task scheduling for hybrid iaas cloud," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 564–573, 2014.

[111] H. Wang, R. Li, L. Fan, and H. Zhang, "Joint computation offloading and data caching with delay optimization in mobile-edge computing systems," in *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, 2017.

[112] I.-H. Chuang, R.-C. Sun, H.-J. Tsai, M.-F. Horng, and Y.-H. Kuo, "A dynamic multi-resource management for edge computing," in *2019 European Conference on Networks and Communications (EuCNC)*, 2019, pp. 379–383.

[113] M. S. Hossain Khan, P. Roy, F. Khanam, F. H. Hera, and A. K. Das, "An efficient resource allocation mechanism for time-sensitive data in dew computing," in *2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT)*, 2019, pp. 506–510.

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

[115] Y. Xu, K. Li, J. Hu, and K. Li, "A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues," *Information Sciences*, vol. 270, pp. 255–287, 2014.

[116] N. Mansouri, B. Mohammad Hasani Zade, and M. M. Javidi, "Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory," *Computers & Industrial Engineering*, vol. 130, pp. 597–633, 2019.

[117] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S. Tucker, "Fog computing may help to save energy in cloud computing," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1728–1739, May 2016.

[118] M. Olbrich, F. Nadolni, F. Idzikowski, and H. Woesner, "Measurements of path characteristics in PlanetLab," TU Berlin, Tech. Rep. TKN-09-005, July 2009.

[119] P. Bertoldi, "EU code of conduct on energy consumption of broadband equipment: Version 6," 2017.

[120] W. Van Heddeghem, F. Idzikowski, E. Le Rouzic, J. Y. Mazeas, H. Poignant, S. Salaun, B. Lannoo, and D. Colle, "Evaluation of power rating of core network equipment in practical deployments," in *OnlineGreenComm*, Sep. 2012.

[121] W. Feng and T. Scogland, "Green500 list for November 2018," 2018, last accessed on 21.03.2024. [Online]. Available: https://www.top500.org/green500/lists/2018/11/

[122] G. Almes, S. Kalidindi, and M. Zekauskas, "A round-trip delay metric for IPPM," 1999, RFC 2681.

[123] "GNU octave." [Online]. Available: www.gnu.org/software/octave/

[124] R. Dolbeau, "Theoretical peak FLOPS per instruction set: a tutorial," *The Journal of Supercomputing*, vol. 74, no. 3, pp. 1341–1377, 2018.

[125] Y. Wang, T. Zhao, L. Li, Z. Hou, and J. Gu, "Roofline model based performance-aware energy management for scientific computing," in *2018 9th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, 2018.

[126] N. M. Allayla and S. A. Dawwd, "Performance optimization on GPGPU & multicore CPU using roofline model," *IOP Conference Series: Materials Science and Engineering*, vol. 1152, no. 1, May 2021.

[127] P. Cai, F. Yang, J. Wang, X. Wu, Y. Yang, and X. Luo, "Jote: Joint offloading of tasks and energy in fog-enabled IoT networks," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3067–3082, 2020.

[128] E. Strohmaier, J. Dongarra, H. Simon, and M. Martin, "Green500 list for June 2020," last accessed on 21.03.2024. [Online]. Available: https://www.top500.org/lists/green500/2020/06/

[129] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[130] B. Bossy, P. Kryszkiewicz, and H. Bogucka, "Energy efficient wireless relay networks with computational awareness," *IEEE Transactions on Communications*, vol. 68, no. 2, pp. 825–840, 2020.

[131] ——, "Energy efficient resource allocation in multiuser DF relay interference networks," in *2018 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2018.

[132] T. Wang and L. Vandendorpe, "Successive convex approximation based methods for dynamic spectrum management," in *Proc. IEEE ICC*, Jun. 2012.

[133] D. P. Palomar and Mung Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439–1451, 2006.

[134] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[135] H. Wong, "A comparison of Intel's 32nm and 22nm Core i5 CPUs: Power, voltage, temperature, and frequency," Oct. 2012, last accessed on 22.02.2024. [Online]. Available: http://blog.stuffedcow.net/2012/10/intel32nm-22nm-core-i5-comparison/

[136] Intel, "Intel delivers new architecture for discovery with intel xeon phi coprocessor," Nov. 2012, last accessed on 22.02.2024. [Online]. Available: https://www.intc.com/news-events/press-releases/detail/538/intel-delivers-new-architecture-for-discovery-with-intel

[137] A. Zappone, E. Björnson, L. Sanguinetti, and E. Jorswieck, "Globally optimal energy-efficient power control and receiver design in wireless networks," *IEEE Transactions on Signal Processing*, vol. 65, no. 11, pp. 2844–2859, June 2017.

[138] E. Strohmaier, J. Dongarra, H. Simon, and M. Martin, "Green500 list for November 2022," last accessed on 21.03.2024. [Online]. Available: https://www.top500.org/lists/green500/list/2022/11/

[139] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *J. ACM*, vol. 19, no. 2, p. 248–264, apr 1972. [Online]. Available: https://doi.org/10.1145/321694.321699

[140] C. Gunaratne, K. Christensen, and B. Nordman, "Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed," *International Journal on Network Management*, vol. 15, no. 5, pp. 297–310, 2005.

[141] IEEE, "IEEE standard for information technology–telecommunications and information exchange between systems - local and metropolitan area networks–specific requirements - part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," pp. 1–4379, 2021.

[142] ITU, "Propagation data and prediction methods for the planning of indoor radiocommunication systems and radio local area networks in the frequency range 300 MHz to 450 GHz," 2019.

[143] E. Strohmaier, J. Dongarra, H. Simon, and M. Martin, "Green500 list for November 2021," last accessed on 21.03.2024. [Online]. Available: https://www.top500.org/lists/green500/list/2021/11/

[144] D. J. Ketchen and C. L. Shook, "The application of cluster analysis in strategic management research: An analysis and critique," *STL*, vol. 17, no. 6, pp. 441–458, June 1996.

[145] M. Olbrich, F. Nadolni, F. Idzikowski, and H. Woesner, "Measurements of path characteristics in planetlab," Telecommunication Networks Group, Technical University Berlin, TKN Technical Report Series TKN-09-005, July 2009. [Online]. Available: http://www.tkn.tu-berlin.de/fileadmin/fg112/Papers/TKN_Technical_Report_TKN-09-005.pdf

[146] Y. Dong, S. Guo, J. Liu, and Y. Yang, "Energy-efficient fair cooperation fog computing in mobile edge networks for smart city," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7543–7554, 2019.