Mathematical Models for Task Assignment with Service Level Agreement in Fog Computing Networks

by

Ahmed Salem, B.Eng

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science in Electrical and Computer Engineering

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario

Abstract

Fog computing was proposed to bridge the gap between the cloud computing capabilities and the new requirements introduced by 5G and IoT applications. Assigning clients' requests to fog nodes for processing while meeting the quality of service requirements is still a challenge. In this thesis, we propose two mathematical models for the task assignment problem in fog networks. The main objective of both models is to maximize the fog service provider's profit while satisfying the service level agreement requirements of the offloaded tasks. In addition, each model addresses multiple requirements to satisfy various service provider's needs. The first model addresses green computing requirements through an energy efficient fog nodes operation. The second model addresses load balancing requirements by minimizing the difference between the node's utilization across the fog network. Since these models provide optimal solutions, they can be useful with historical data and for benchmarking various real-time algorithms.

To my children who inspired me to go through this journey.

Acknowledgements

I would like to express my deep and sincere gratitude to Professor Marc St-Hilaire for the invaluable contribution, support and patience he provided me during my study. The development and completion of this thesis could not have been accomplished without his guidance, vision, motivation and immense knowledge.

Table of Contents

List of Tables

List of Figures

List of Acronyms

5G	Fifth Generation Mobile Networks
AP	Access Point
AR	Augmented Reality
BS	Base Station
CAPEX	Capital Expenditure
CDN	Content Delivery Networks
CLI	Command Line Interface
CoS	Class of Service
CPS	Cyber Physical System
CSP	Cloud Service Provider
DC	Data Center
D2D	Device-to-Device
ETSI	European Telecommunication Standards Institute
E2E	End-to-End
FSP	Fog Service Provider
GPS	Global Positioning System
GPU	Graphics Processing Unit
GUI	Graphical User Interface
IDE	Integrated Development Environment
ILP	Integer Linear Programming
IoT	Internet of Things
IP	Internet Protocol
ITS	Intelligent Transportation System
KPI	Key Performance Indicators
LAN	Local Area Network
LBF	Load Balancing Factor
MEC	Multi-Access Edge Computing
M2M	Machine-to-Machine

OPEX	Operating Expenses
OPL	Optimization Programming Language
QoS	Quality of Service
RAN	Radio Access Network
RNC	Radio Network Controller
SDN	Software Defined Networks
SLA	Service Level Agreement
TASLA	Task Assignment with Service Level Agreement in fog networks
TASLA-GC	Task Assignment with Service Level Agreement in fog networks-Green Computing
TASLA-LB	Task Assignment with Service Level Agreement in fog networks-Load Balancing
vCPU	Virtual Central Processing Unit
VFC	Vehicular Fog Computing
VM	Virtual Machine
WAN	Wide Area Network
WSN	Wireless Sensor Network

Chapter 1: Introduction

Data Centers (DC) have traditionally consisted of a large collection of physical compute, memory, storage and networking resources. Over time, DC with different computing paradigms have significantly evolved till the point where cloud computing has become a major player in the current Internet. With the advancements in virtualization and Software Defined Networks (SDN) technologies, cloud computing has been able to provide elasticity, automation and scalability. Today, Machine-to-Machine (M2M) or Internet of Things (IoT) applications are rapidly growing and massively deployed across different domains to improve our life. Fifth Generation (5G) mobile networks are being widely deployed and providing a whole new set of features like massive connectivity, high mobile traffic volumes, ubiquitous access for clients, automated provisioning and reliability. According to Cisco, it is expected that there will be 29.3 billion Internet Protocol (IP) networked devices with 14.7 billion M2M connections and 1.4 billion 5G devices by 2023 [1]. As a result, cloud computing has been expanding enormously and widely implemented to provide a solution for such massive traffic volumes and processing. However, cloud computing also comes with some challenges. For example, the long distance between the things/end user devices (hereinafter referred to as end user) and the centralized cloud data center (hereinafter referred to as cloud DC) can cause significant delay that affects the performance and reliability of the applications that require low latency. In addition, more applications will require location-awareness and mobility support. Towards that end, fog computing, a novel edge computing paradigm, can offer a promising solution for such challenges. Fog computing can support large and ubiquitous volumes of latency-sensitive traffic produced by IoT. Fog computing allows the provisioning of resources at the network edge, with a proximity to the end users. Fog nodes allow computing and storage offloading of the cloud DC. Fog computing can provide distributed processing at the network edge while interacting with the cloud DC. This leads to an efficient resource utilization and energy consumption, while satisfying the Service Level Agreement (SLA) in terms of performance and delay. Fog computing can have n number of tiers architecture with the 3-tier architecture as the most basic and popular architecture which consists of the cloud layer, the fog layer and the end user layer [2][3].

1.1 Problem Statement

Task assignment is a major research topic which has been shown to be an NP-hard optimization problem [4]. The task assignment problem is presented when the end user starts offloading one or more tasks to the network, requesting computing resources for processing. Tasks are then queued, prioritized, scheduled and assigned to the processing node where the computing resources are allocated for task processing. Task assignment in cloud computing is present across the application, virtualization and deployment layers. In addition, different requirements such as Quality of Service (QoS), cost efficiency, energy conservation, load balancing and service placement add more complexity to the task assignment problem in cloud computing.

The decentralized nature of fog computing introduces a new level of complexity to this problem by extending the task assignment to include the network edge resources, where tasks can be assigned to the fog nodes or cloud DC. This requires the task assignment models to consider the unique characteristics of the fog nodes such as geographical distribution and interaction with cloud DC. A model must have the capability to differentiate between the fog nodes, cloud DC and incorporate different node characteristics such as location, availability, resource type, capacity and utilization. This allows the model to address different requirements such as SLA, green computing or load balancing. In addition, task assignment models in fog networks must be able to address new requirements introduced by IoT and 5G applications such as location-awareness, mobility and low latency.

Most of the proposed task assignment models in fog computing have often tackled few aspects of this problem by only addressing a single requirement. Furthermore, these proposals have ignored some of the realistic attributes of these requirements and made strong assumptions when applying the models. For example, SLA has been usually reduced to only include processing delay requirements or maximum allowed distance between the end user and the fog node. Another example is breaking down the SLA into multiple independent requirements to overcome the problem complexity. In such an approach, the task requests separate requirements for distance, processing delay, completion time, etc. This requires additional attributes in the task request which results in inefficient bandwidth and compute resources utilization. In addition, such approach can lead to a constant resource allocation rate which restricts the ability for variable workload rate over time according to the network state and resource availability. Finally, this approach pushes the

complexity towards the end user side to breakdown the overall SLA requirements into multiple separate requirements. This limits the ability of these models to address real-life use cases.

The task assignment models in fog networks must be able to address multiple requirements. Furthermore, these requirements should be modeled in a comprehensive and realistic approach to address service providers and end users needs such as End-to-End (E2E) task completion time and Class of Service (CoS). This also requires the ability to extract task requirements without introducing unnecessary overhead. Finally, the model should allow Fog Service Providers (FSP) to maximize profit while satisfying other model requirements such as SLA, green computing or load balancing.

An exact task assignment optimization model can address these problems by optimally assigning the tasks in the fog network while satisfying multiple requirements. The exact model runs periodically for a pre-determined cycle duration. This requires a complete knowledge of the set of tasks that are offloaded and submitted for processing prior to the model execution. Despite some of the real-time restrictions of the exact model implementation in large-scale fog networks, it can still be used as a benchmark to evaluate real-time task assignment models and algorithms in fog networks.

1.2 Research Objectives and Contributions

The main objective of this thesis is to propose two mathematical models for the task assignment problem in 3-tier fog networks. These models provide optimal solutions and can be used for benchmarking. The models allow FSP to maximize the profit while optimally satisfying the SLA and other requirements. More precisely, we will:

- Consider realistic SLA parameters such as CoS and E2E task completion time. The task length, CoS, release time and end user location are used to calculate the maximum E2E task completion time and the expected revenue that can be generated by processing the request. The node location and resource capacity are used to calculate the optimal task assignment in terms of the number and duration of the allocated resources which satisfy the SLA requirements and maximize the FSP revenue.
- Develop a first mathematical model to maximize the profit of FSP by satisfying the SLA requirements (as described above) while considering green computing aspects. Green

computing requirements will be satisfied by minimizing the associated operational, startup and shutdown costs of the fog nodes.

- Develop a second mathematical model to maximize the profit of FSP by satisfying the SLA requirements while simultaneously considering load balancing task assignment. The load balancing requirements will be satisfied by minimizing the difference between the resource utilization of all the fog nodes in the network.

Once the above listed research objectives are achieved, the following contributions to the task assignment in fog networks research area can be claimed:

- The development of two new mathematical models that can be used for optimal task assignment in 3-tier fog networks with multiple FSP and end user's requirements.
    - The models calculate the optimal assignment for a complete set of tasks in a fog network.
    - The optimal solution provided by the models ensures task assignment to satisfy SLA requirements which maximize FSP profit.
    - In addition to the SLA requirements, each model introduces a different objective function and constraints. This allows the FSP to implement different strategies including green computing or load balancing.
    - The models can further be used as a benchmark to validate and verify real-time task assignment models and algorithms in large-scale fog networks.

1.3 Methodology

This section describes the methodology that will be used to achieve the research objectives stated in Section 1.2. This methodology is summarized as:

Study and research Fog Computing: The starting point in this thesis is to understand fog computing as a new paradigm. This includes the evolution of fog computing and the main reasons why it has emerged to complement cloud computing. In addition, the study explores the main use cases, applications and requirements. Furthermore, understand the main characteristics of fog networks, architecture, layers interactions and how it's different from other edge computing paradigms. Finally, a literature review for different fog computing research areas with a focus on task assignment in fog networks, the proposed models and algorithms to identify the gaps.

Develop the mathematical models: In this step, and based on the research outcomes, the development of the mathematical models includes the following:

- Develop a mathematical representation of the fog network operations. This includes the formulation of different task and fog node attributes. The main goal is to provide a flexible and simple way to represent the different task requirements and the fog node attributes such as task length, SLA requirements, node resource capacity, location, etc. This allows the model to address different task assignment requirements and can be further extended to include new attributes and requirements for future use cases. This formulation is used to define the input data sets for the mathematical models.
- Define the main objective functions and constrains using the linear programming technique for each mathematical model to represent the different requirements such as profitability, SLA, green computing and load balancing. Applying the linear programing calculations on the input data set will determine the optimal task assignment in fog network based on the defined objective function and constraints.

Implement the mathematical models: To implement the proposed mathematical models in any fog network, the optimization software called IBM ILOG CPLEX is used to calculate the optimal task assignment according to each mathematical model against the input data sets of the simulated tasks and fog network elements.

Validate the mathematical models: To validate the proposed mathematical models, manual calculations will be used to determine optimal results based on data sets that simulate very simple and small fog networks. The next step is to execute the model in CPLEX using the same data sets and make sure that the CPLEX output is identical to the manual calculations in terms of optimal profit value and task assignment.

Simulate and test the mathematical models: The next step will be to develop a test plan and generate input data sets to simulate fog networks with various sizes. The input data sets will be generated randomly from a range of values for each attribute. This will ensure that the generated task data sets will represent multiple task length, release time, location, SLA requirements, etc. Similarly, the generated fog network elements data sets will include multiple node locations and resource capacity values. The input data sets will then be used by CPLEX to run the implemented mathematical model and generate optimal task assignment results.

Analyze the test results: Finally, we need to present and analyze the test results for each proposed mathematical model.

1.4 Thesis Outline

The remaining of this thesis will be divided as follows:

- Chapter 2: Background and Related Work. This chapter will describe the main concepts of fog computing, its characteristics, applications and current research status with a focus on task assignment, green computing and load balancing in fog networks.
- Chapter 3: Formulation of the Task Assignment Problem in Fog Networks. This chapter will first introduce the notation that is used to model the problem and then, the formulation of the task assignment problem in fog networks will be presented.
- Chapter 4: Results and Analysis. This chapter will demonstrate and analyze the detailed validation, evaluation and test results.
- Chapter 5: Conclusion and Future Work. This chapter will summarize the thesis outcomes and the recommended future work.

Chapter 2: Background and Related Work

This chapter will provide an overview of the fog computing related research work. More precisely, Section 2.1 will describe the fog computing background. Section 2.2 will provide a literature review on the preceding and current efforts in the field of task assignment, load balancing and green computing in fog networks. Finally, Section 2.3 will summarize the chapter.

2.1 Fog Computing Background

This section will introduce the fog computing evolution, main concepts, characteristics and applications.

Cloud computing promotes automation, elasticity, flexibility and scaling for different applications. Despite these benefits, one of the major challenges for cloud computing is the latency requirements of the end users. Internet is typically used to provide the connectivity between the cloud DC and end users. Internet connectivity is not suitable for latency-sensitive applications [5]. Moreover, Cloud Service Providers (CSP) such as Amazon, Google and Microsoft usually deploy the cloud DC in centralized locations. The long distance between the cloud DC and the end users can result in a high E2E delay. In addition, cloud computing usually processes the applications requests over multiple components in a distributed fashion and the application components can be deployed and processed in multiple data centers [6][7][8]. The inter-data center communications can add more overhead that results in an increased service delay. Furthermore, the large amount of the horizontal and/or vertical traffic can cause links congestion, delay and QoS degradation due to limited bandwidth. Another major challenge is the support of the increasing number of applications that requires location-awareness. The cloud computing centralized architecture doesn't support the location-awareness required for such applications. Some examples of latency-sensitive and location-aware applications can include autonomous vehicles, smart grid [9], public safety networks [10], eHealth, drones, cognitive assistance [11] and Content Delivery Networks (CDN) [12].

With the increasing awareness of cyber security and information privacy, cloud computing is facing multiple security and regulation compliance challenges. In many cases, the governments regulations require that certain types of data must be processed and/or stored within specific geographical area where the CSP doesn't have any deployed data centers [12]. In addition, some

applications cannot send data to the cloud DC due to privacy concerns. Finally, the energy consumption and the carbon footprint of the cloud DC is another major challenge. It is estimated that data centers around the world consume about 26GW which is around 1.4% of the global electrical energy consumption, with an annual increase rate of 12% [13]. Data centers around the globe were responsible for 78.7 million metric ton of CO2 emissions, which is equal to 2% of global emissions in 2011 [14]. Despite the fact that many of the CSP have introduced green and sustainable data centers in their expanding infrastructure, the carbon emissions from data centers will still dominate the global carbon footprint.

Edge computing paradigms have emerged to bridge the gap between the centralized model of cloud computing and the increased demand for ultra-low latency, high bandwidth and location-aware applications. Resource provisioning at the network edge brings these resources closer to the end users and enables the support of new applications such as mobile data offloading. Cyber foraging is considered to be one of the early paradigms to tackle edge computing model that brings computing and resource storage closer to the edge. It has been later superseded by other paradigms such as cloudlet, Multi-Access Edge Computing (MEC) and fog computing [15].

Satyanarayanan et al. [16] have introduced the cloudlet model in 2009. Cloudlets are also referred to as cloudlet-based cyber foraging [17][18]. Cloudlet is a DC that can be considered as a smaller version of the cloud DC (mini cloud). MEC is an industry initiative that has been introduced by the European Telecommunication Standards Institute (ETSI) in 2014. MEC can be deployed near mobile users at the Radio Access Network (RAN), Base Stations (BS) and Radio Network Controller (RNC) [19].

Fog computing was introduced by Cisco in 2012 as an attempt to address the challenges that face cloud computing such as latency, location-awareness, mobility and security. The fog computing concept was proposed as a new architecture that extends the cloud computing capabilities to the network edge. The term ‘fog’ is used in the sense that “fog is a cloud which is close to the ground” [20]. Fog computing provides virtualized computing, storage, networking resources and data management services with close proximity to the end users [20]. Fog network can be defined as “a scenario where a huge number of heterogeneous (wireless and sometimes autonomous) ubiquitous and decentralized devices communicate and potentially cooperate among them and with the network to perform storage and processing tasks without the intervention of third parties. Users leasing part of their devices to host these services get incentives for doing so.” [21].

The OpenFog consortium is a workgroup that consists of leading technology companies and university researchers that are working towards creating an open architectural framework for fog computing. The group has released its reference architecture for fog computing in 2017. OpenFog has used the term SCALE to describe the capabilities of fog computing, which are security, cognition, agility, latency and efficiency. OpenFog defines fog computing as "a horizontal system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum.". In such architecture, the horizontal level can distribute the computing capabilities to different networks, applications and devices with close proximity. Although the cloud computing vertical model provides strong capabilities that can address the requirements for a specific type of networks or applications, it still may not be able to support other applications that require multiple interactions across a heterogenous network. The fog computing horizontal model allows the flexibility to satisfy different requirements for end users, services and applications [22].

Fog computing is expected to support low latency, location-awareness, geographical distribution, end user mobility, high number of processed nodes, wireless access, real-time applications and heterogeneity. Different application components that require low latency, can be processed by the fog nodes at the network edge, close to the end user. Other components that are delay tolerant and require high computational capabilities can still be processed on the cloud DC. Fog computing can also provide the flexibility to process some application components at distributed locations in the case of high traffic density. This is achieved by utilizing the computational capabilities of different types of equipment such as Access Points (AP), network proxies and other networking devices located at the edge of the network (e.g., switches and routers). By bringing the computing resources to the network edge closer to the end users, fog computing can support IoT and 5G latency-sensitive applications and satisfy SLA requirements that cloud computing cannot satisfy due to its previously stated limitations. In addition, fog computing can improve the overall system efficiency by integrating the end user devices computational capabilities [23].

There are number of differences between fog computing and cloud computing in terms of scalability, efficiency, location, etc. The first major difference is the decentralized nature of the fog network compared to the centralized cloud architecture. Fog nodes are usually distributed over different geographical locations at the network edge and close to the access network. Cloud DC is centralized in the network core. Fog computing requires security to be run at the network edge by

the fog nodes while cloud computing require security to be run in the network core. Fog computing allows different kind of devices to have multiple roles according to their available computational capabilities. As an example, connected vehicles can act as fog nodes that provide computing resources at the network edge or as clients to be served by other fog nodes. Another major difference is the hardware capabilities of fog nodes vs the cloud DC. Fog computing is expected to provide moderate resource availability while the cloud DC is required to provide high resource availability. Fog nodes can vary from mini DCs, network devices to end user devices. Cloud DCs are built of huge number of physical computing machines. This requires the cloud DC to consume much more power compared to fog nodes. Cloud DC also requires much more space than the space required for a fog node. This results in more flexibility when positioning fog nodes closer to end users. In terms of connectivity, cloud computing requires a highly available Internet Wide Area Network (WAN) connection to serve end users. Fog nodes can operate with low or no Internet connection, as the devices can access the network edge through Local Area Network (LAN). Fog nodes can send any required requests to the cloud DC once the Internet connection is available. Cloud computing is considered to have better overall reliability due to its massive resource capabilities and high availability.

As discussed earlier in Chapter 1, fog computing can have n number of tiers architecture with the 3-tier architecture as the most basic and popular architecture. Figure 2.1 demonstrates the 3-tier fog architecture that consists of the cloud layer, fog layer and end user layer.

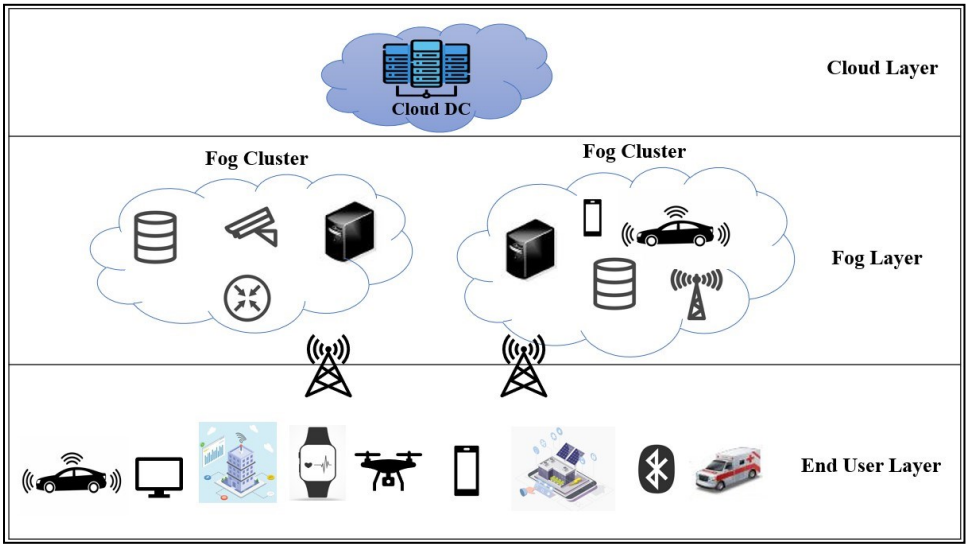

Figure 2.1: Fog 3-Tier Architecture

The cloud layer consists of the centralized cloud data center(s). This layer has massive resource capacity. The fog layer can consist of one or more clusters. Each of these clusters can be managed by the same or different FSP. Each fog cluster consists of one or more fog nodes (e.g., routers, switches, AP, end user devices, etc.). The end user layer can consist of IoT devices, cellular devices, etc. The fog network supports interactions between end users, fog nodes and cloud DC [20]. The communication between the fog and cloud layers typically happens over WAN, while the communication between the end user and fog layers usually happens over LAN. In addition to the vertical communication between the *n*-tiers, different fog nodes can corporate horizontally by sharing their computing resources to better serve end user requests [24]. This reduces the volume of requests and traffic being sent to the cloud layer. If the fog node is unable to process a request due to limited resources or specific requirements, the request is sent to the cloud DC for processing. The horizontal corporation and communication between different fog nodes and domains, allows the fog layer to send requests to the cloud layer through a single point without the need to replicate the same request from different fog nodes or domains. This works also in the opposite direction where the cloud DC needs only to send a request or communicate with the fog layer through a single point. This results in reducing the amount of inter-layer traffic and eliminating the unnecessary overhead between the cloud and fog layers.

Fog computing allows computing to happen anywhere across the data path down from the end user layer up to the cloud layer, preferably closer to the end users. For example, in the Intelligent Transportation Systems (ITS), Global Positioning System (GPS) information can be processed at the network edge prior to sending it to the cloud DC [25]. In addition, the end user can establish a direct connection with fog nodes using Device-to-Device (D2D) communication [26] or cellular small cell [27].

2.1.1 Fog Computing Characteristics

While fog computing shares a lot of common characteristics with cloud computing and other edge computing paradigms, authors in [12] and [20] have listed the main specific characteristics that need to be supported in fog computing. These characteristics can be used to identify fog nodes as a significant extension to the cloud at the network edge, as well as differentiate fog computing from other edge computing paradigms. Some of these characteristics include:

Edge location and location-awareness: Deploying the fog node resources at the network edge will allow efficient network resource utilization and satisfying the low latency and location-awareness requirements.

QoS management: One of the major requirements for fog computing is to support latency-sensitive and real-time applications such as mobile gaming. Fog must be able to satisfy the QoS requirements for different fog computing applications.

Geographical distribution: This allows fog computing to support different applications in a distributed model. As an example, fog computing can support video streaming for mobile end users by utilizing wireless AP that are located on highways.

Mobility support: Fog computing is required to support applications that run on cellular devices, connected vehicles, etc.

Real-time interactions: Fog computing is required to support real-time interactions instead of the batch processing that can be unsuitable for real-time applications.

Scalability: Fog computing must be able to support the massive numbers of end user devices deployed in IoT, 5G, smart cities and smart grid networks.

Heterogeneity: Fog computing must be able to support different types of virtualized resources across heterogenous networks. Fog nodes are expected to be deployed over virtualized wireless AP, BS, Wireless Sensor Networks (WSN), mobile gateways, proxy servers, routers, switches, etc.

Interoperability and Federation: Due to the different deployment scenarios, fog computing must be able to support interoperability between different types of fog nodes as well as fog nodes that are part of multiple federations or clusters. This is essential for resource management and service provisioning in association with mobility support and geographical distribution.

2.1.2 Fog Computing Applications

The OpenFog is expecting that fog computing can address the requirements, improve the QoS and performance of many applications that require high bandwidth, ultra-low latency, location-awareness and mobility such as Cyber Physical Systems (CPS), smart cities, eHealth, ITS, mission critical, surveillance, Augmented Reality (AR) and remote gaming [28][29][30][31][32][33]. In the following sub-sections, we will review some of the proposed fog computing applications.

2.1.2.1 Smart Cities

Fog-enabled smart cities can support different use cases and applications [34]. This include surveillance, smart agriculture [35], smart transportation [36], water management and smart waste management [37]. The authors in [38] proposed a fog-enabled real-time multi-target vehicle tracking system. In smart agriculture, fog computing can support location-awareness and real-time response required to monitor the weather condition and the agriculture process, using sensor-enabled devices [39].

2.1.2.2 Smart Grid

Smart grids aim to reduce the energy consumption through smart energy management systems [40]. Energy management systems consist of sensors, actuators, gateways and computing platforms [41]. Computing platforms are required to gather, process, analyze and store the data collected by the sensors and provide programmability to control the actuators used for energy management. Major requirements for energy management systems include high reliability and low latency [42]. Fog computing can address these requirements by processing the latency-sensitive data [20]. The authors in [43] proposed fog computing as a solution to address the energy management requirements such as scalability, service customization, interoperability and interaction between different devices. The proposed fog-enabled system can support energy consumption monitoring, metering and management through efficient device control.

2.1.2.3 Intelligent Transportation System

Vehicular networks and ITS have been proposed to address road safety and traffic management. Different applications include vehicle speed control, traffic flow control, parking assistance and surveillance. Cloud computing provided the base for implementing ITS. In some cases, cloud computing cannot meet the real-time and low latency requirements. Fog computing can be an alternate solution to address these requirements [44]. In addition, vehicular ad hoc networks require the support of high mobility, location and context awareness. Fog nodes can be used to process the location and other related information [45]. Another major requirement for ITS is the support of heterogenous networks, which requires a high level of communication and corporation between different fog clusters. The authors in [46] proposed a fog-enabled adaptive resource

scheduler to support real-time vehicular communication. The authors in [47] proposed a fog-based vehicular crowd sensing infrastructure, where fog nodes are used for sensor-enabled vehicular data management. To meet the growing demand for computational resources for vehicular applications, the authors in [31] proposed to utilize individual vehicles computational resources to act as fog nodes that can support near end users.

2.1.2.4 eHealth

Fog computing can address the latency-sensitivity and context awareness requirements for different eHealth and emergency applications. The authors in [48] proposed to use fog nodes to process electrocardiogram data in order to meet the low bandwidth and latency requirements. The authors in [49] proposed fog-enabled remote patient monitoring system. The system collects data from the IoT devices, and then enables data analytics partitioning and processing for decision making. Cloud DC is used to process tasks that require high computational resources. Fog nodes are used to process periodic light-weighted tasks. A fog-based neuro-imaging system is proposed in [50]. Mobile devices are used for data collection while fog nodes are used for primary brain state processing and classification in real-time. Cloud DC is used for more in-depth analysis and scalable storage of the brain state information. The authors in [51] introduced emergency mobile system where fog nodes are used to offload the mobile user's data, share the location and other emergency details to the first responders. The information is then sent to the cloud DC for more analysis, storage and future early notification.

2.1.2.5 Public Safety

Emergency and natural disaster management highly relies on real-time, location-awareness and latency-sensitive data processing and communications. Fire detection and fighting systems are mission critical and essential for public safety. A fog-enabled fire detection and fighting system which monitors different areas and responds in case of fire detection is proposed in [10]. The system uses sensor-enabled devices for data collection. The collected data is processed and analyzed on fog nodes in real-time for evaluation and decision making. In case a fire is detected, the system dispatch robots fire fighters to the fire location. Fire detectors and robot dispatchers are hosted on the fog layer for real-time and reliable processing and response. Firefighting strategies are yet hosted on the cloud layer to implement the required procedure.

2.1.2.6 Remote Gaming

In remote gaming, end users aren't required to download or install the gaming application on their devices. Instead, the user can play over the Internet by connecting to a server that hosts the game. Usually, these servers are located in the cloud layer, where the application is hosted, executed and the user data is processed. Despite the numerous benefits of this approach, many challenges can affect the user experience and results in less revenue for the gaming company. These challenges include limited bandwidth, high latency, poor QoS and end user device high power consumption. The authors in [52] proposed a lightweight fog-based system. Cloud DC is used to process the heavy computational state of a new game then sends the updated game state to the fog node which renders the game videos and stream it to the end user.

2.2 Related Work

Now that we have a better understanding of fog computing concepts and typical applications, this section will focus on the related work dealing specifically with task assignment, load balancing and green computing in fog computing.

2.2.1 Task Assignment in Fog Computing

The task assignment is a fundamental combinatorial optimization problem of two separate sets, a set of tasks and a set of agents. Each task can be assigned to an agent subject to a cost that may vary depending on the task and agent attributes. The task assignment objective is to assign as many tasks as possible while minimizing the cost [53]. As illustrated in Figure 2.2, the task assignment problem can be presented using a weighted-bipartite graph where each task in one partition must be matched with at most one agent in the other partition with the objective to maximize the summation of the weights of the matched edges of the bipartite graph [54]. The task assignment has been shown to be NP-hard problem.

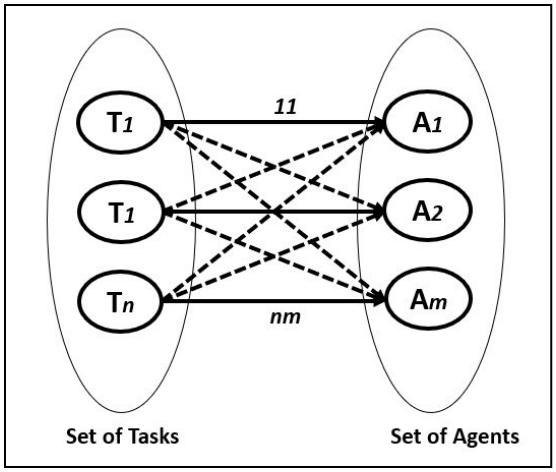

Figure 2.2: Task Assignment Bipartite Graph

The task assignment has been widely studied in cloud computing. The task assignment problem in cloud computing is presented when multiple tasks are offloaded to the network for processing, each with a specific amount of workload and an execution deadline. CSP supplies a large pool of computing resources using a massive number of Virtual Machines (VMs). The main goal of the task assignment is to specify VM(s) and allocate the minimum number of resources on which the assigned task(s) can be processed successfully within the deadlines [55]. The task assignment objective function should be optimized while different execution constraints are respected. Different objectives can include the total execution time, load balancing, reliability, fault tolerance, green computing and profit.

The task assignment problem also presents itself in fog networks. Although quite similar to the task assignment in cloud computing, the decentralized nature and heterogeneity of fog networks are adding to the overall complexity. Compared to cloud computing, tasks can now be assigned to fog nodes or to the cloud DC. The fog nodes can vary in type, resource capacity, availability, link bandwidth, distance to the end user, etc. Task assignment in fog computing must consider multiple nodes and task attributes to address these challenges. In order for fog service providers to generate more revenue, tasks with low latency requirements that cannot be addressed by the cloud layer must be assigned to the fog layer else the SLA will be violated. This comes with a challenge due to the limited resource capacity and higher associated costs of the fog layer when compared to the cloud layer. Task assignment models in fog computing must be able to address these challenges by introducing new problem formulations and satisfying multiple requirements.

The task assignment problem can be modeled as a linear program [56]. Researchers have been proposing different optimization algorithms and exact models to address the task assignment problem. Examples include linear programming techniques, ant colony optimization, artificial bee colony and particle swarm optimization [57]. Different terminologies in literature are usually used to refer to the task assignment problem. Examples include task scheduling, task allocation, service provisioning, service placement, workload and resource allocation. In this section, we provide a review for research work related to task assignment in fog computing with relevant references to task assignment in cloud computing.

2.2.1.1 Task Assignment Models in Cloud Computing

Many task assignment models in cloud computing are trying to address latency, QoS and SLA objectives. Kliazovich et al. [58] proposed a task scheduling model in cloud computing that satisfies multiple separate task requirements such as latency, computing resources, etc. Requirements are grouped into two main categories, certain (e.g., memory resources) and uncertain (e.g., communication resources) requirements. The model prioritizes the requests, optimizes task scheduling and resource allocation based on each category of the requirements. Another SLA-based scheduling model in cloud computing is proposed by Schwiegelshohn et al. [59]. The model is optimized based on the cost and generated income of the task allocation. The algorithm prioritizes the incoming requests based on the maximum allowed latency required to satisfy the SLA requirements.

Wei et al. [60] proposed a task and resource allocation model in cloud computing that consider costs as well. The problem is modeled using binary integer programing where the objective function includes costs that represents the requested computing resources. Cost and time constraints are used for an exact task and resource allocation. The optimization of the objective function tries to minimize the costs.

2.2.1.2 Task Assignment Models in Fog Computing

Many task assignment models in fog computing have used approaches similar to the task assignment models in cloud computing while considering the decentralized nature of fog computing. Agarwal et al. [61] proposed a resource provisioning algorithm in fog networks. The main goal is to provide efficient resource allocation. This is achieved by distributing the incoming

requests over both fog nodes and cloud DC. The algorithm is implemented in the fog layer. Once a user request is received from the end user layer, the algorithm will allocate fog node(s) resources based on the user's computational requirement while considering the time constraints. In case that there are not enough resources, the request can be partially processed by the fog layer and the remaining or all the components are sent to the cloud layer for processing.

Mathematical modeling can be used to address different objectives while respecting multiple constraints such as resource capacity, delay, etc. Intharawijitr et al. [62] defined a mathematical model for task scheduling in fog-enabled 5G networks. The main goal is to minimize the computing and communication delay. To achieve low latency, the problem is defined with each group of users are represented as a source node and each source node is associated with a fog node. In addition, different elements such as workload, capacity and roundtrip delay are used. The main objective is to minimize the blocking probability while respecting the latency constraints.

Integer Linear Programming (ILP) is widely used to formulate the task assignment problem in fog computing to model different attributes and objectives specific to fog computing. Souza et al. [63] formulate the service allocation problem in fog networks to achieve low latency for mobile services and address future requirements of IoT services. The problem is modeled as an ILP problem which include different metrics such as the number of services, accessible resources, node capacity, service requirements, resource allocation time and resource allocation delay for a service. Another example is a task allocation model in fog networks based on the user's dynamic resource and QoS requirements proposed by Lai et al. [64]. The authors use ILP to formulate the objective function which tries to maximize the level of satisfying the QoS requirements of the end users based on the allocated resources on fog nodes.

Similar to cloud computing, cost efficiency is one of the main objectives for task assignment in fog computing. GU et al. [65] proposed an algorithm for service placement in fog networks for medical CPS. The main goal is to achieve cost efficient resource management. The authors formulate the problem as a mixed-integer non-linear program and then linearize it into a mixed ILP. This is done using the user to BS association, task distribution, VM placement and QoS constraints.

Cost objective functions allow the task assignment model to address multiple requirements such as efficient resource utilization or QoS. This can be done by assigning costs for resource consumption or QoS adherence. Skarlat et al. [66] proposed an optimization model for IoT service

placement in fog networks while considering QoS requirements. The authors formulate the problem using ILP with a mapping between the applications and resources that meet the QoS requirements. The main objective is to maximize the fog network resource utilization. To meet the QoS requirements, different constraints are included such as application deployment time and execution deadlines. The optimized cost of the execution and QoS satisfaction is used to evaluate the model performance.

Fog service provider's profit is another major objective that needs to be addressed. In addition to cost-efficiency, profit objective functions can include FSP revenue calculations while respecting constraints related to latency, resource utilization, etc. Mahmud et al. [67] proposed an application placement model in fog computing that aims to maximize the service provider's profit while satisfying QoS requirements. The authors use ILP for problem formulation. The model tries to maximize the placement objective function based on the estimated profit. The profit calculations include the revenue generated by successful application placement within its deadline, the operational cost of the processing fog node and the cost to compensate any SLA violation. The model constraints include placement, QoS, budget and resource constraints.

2.2.2 Load Balancing

Load balancing can refer to the equal workload distribution over multiple processors where no single processor is overloaded. Load balancing is essential for efficient resource utilization and higher performance in distributed networks.

Different models and algorithms are proposed to allow efficient task and resource allocation in cloud and fog computing with the aim to enhance network performance and user experience simultaneously while considering load balancing. Load balancing models can be classified either static or dynamic. Static load balancing requires a prior knowledge of the network status and nodes information. Static load balancing is better suited for homogeneous and stable networks. Dynamic load balancing doesn't require any prior knowledge of the network status, instead it relies on real-time information. Dynamic load balancing is more flexible, can adapt to different network types, attributes and provides better performance in real-time [68].

Load balancing models face many challenges such as efficiency, overhead, fault tolerance, etc. [69]. For example, spatial distribution of the network elements can impose many challenges due to the variation in links bandwidth, delay between end users and different network elements.

Heterogeneity is another major challenge due to the variation in the resource capabilities and capacity of network elements. Finally, efficient load balancing approaches require the gathering of a lot of information and network status monitoring which increase the model complexity [68]. Different approaches have been proposed to achieve load balancing such as hill-climbing [70] and tabu search-based algorithms [71]. Load balancing approaches and strategies that have been applied in cloud computing can be used in fog computing by adapting to the characteristics of fog networks [72]. This section provides an overview of the research work related to load balancing models in fog computing.

Many task assignment models in fog computing have addressed the load balancing objectives. In addition, many models have combined load balancing with other objectives such as QoS. Fan et. al. [73] proposed a distributed workload balancing model in fog computing while minimizing the service delay. The problem is formulated using the communication latency between the BS acting as fog node and IoT devices. The authors developed two separate algorithms, the BS algorithm estimates the workload and resource utilization, while the IoT algorithm is used to assign tasks to BS based on the workload and latency information. Load balancing is achieved based on the latency ratio of the entire fog network. Another task offloading policy for fog nodes that have combined load balancing and service delay objectives was proposed for IoT applications by Yousefpour et al. [74]. The proposed policy considers the vertical interactions across cloud, fog and end user layers. Fog nodes horizontal interactions are used to support load balancing. The policy considers different characteristics, such as queue length, estimated waiting time and different request types with varying processing times for load balancing. Task assignment is determined based on the number of requested resources, scheduler queue status and fog node(s) resource capacity. The policy uses the estimated processing time for task prioritization.

Ningning et al. [75] proposed a load balancing algorithm for big data in fog computing based on dynamic graph partitioning. The main goal is to minimize the idle task waiting time and the communication overhead using parallel workload allocation. The system is modeled as a graph where fog nodes are represented with specific resource capacity connected with weighted edge links representing the communications bandwidth. Task assignment load balancing is based on the distance between the end user and the node on the graph.

2.2.3 Green Computing

Green computing aims to reduce the energy consumption, carbon emissions of different network elements and data centers while maintaining high performance and efficiency. Both researchers and industry have proposed and implemented multiple approaches for efficient energy utilization. This includes smart location, enhanced cooling systems, energy efficient computing machines, energy recycle, monitoring/metering systems, dynamic deactivation [76]. Task assignment can play a major role in implementing green computing by adopting approaches such as sleeping, hibernation and on demand shutdown. In such approaches, the task scheduler tries to group and assign the maximum amount of workload to the minimum number of processors, allowing idle machines to be put into sleep or shutdown. However, this can introduce challenges and complexity to leverage variable workload, minimize transition overhead while minimizing delay to satisfy SLA requirements [77].

Multiple task assignment models have been proposed to address the trade-off between latency, QoS and energy-efficiency. Deng et al. [78] proposed a model for workload allocation in fog networks that considers the trade-off between power consumption and delay. The main objective is to minimize the power consumption. The model constraints include end user delay, resource capacity, link bandwidth, workload utilization in fog and cloud layers. Xiao et al. [79] proposed a distributed optimization algorithm for workload allocation in fog computing. The main goal is to satisfy the end user QoS requirements while considering power efficiency. This is achieved through cooperative offloading where the fog nodes jointly offload workload from the cloud layer. Each fog node can send unprocessed workload to other fog nodes for processing instead of sending it to the cloud DC. The fog nodes cooperative offloading must respect the QoS and power constraints.

Zhu et al. [80] proposed an energy-aware task scheduling model for cloud computing. Task execution deadlines are used for service prioritizing. Different costs are set based on the VM status where resources are scaled up or down in real-time for optimization. Yi-Ju et al. in [81] proposed power saving policies to reduce machine idle power by optimizing the cost function while maintaining service guarantee based on arrival time. The cost function is formulated using the power consumption, system congestion and server startup costs.

More recently, Daigneault et al. [82] proposed an exact mathematical model for task assignment in fog networks to maximize FSP profit while considering green computing. The problem is formulated and optimized using linear programing. The profit objective function calculates the revenue generated by task assignment and the fog nodes costs. Task requirements include the number of computing resources, maximum latency and execution deadline. Costs include fog nodes startup, operational and shutdown costs which allow the model to optimize fog nodes status through an on demand turn on/off. The model tries to determine optimal task assignment for maximum profit. This is achieved by satisfying the different task requirements while minimizing different fog node costs which satisfies the green computing requirements. For future work, the authors have proposed to introduce SLA. The model presented in [82] is used as a starting point for the work presented in this thesis.

2.3 Summary

This chapter provided an overview on the fog computing background and related research work. It presented the fog computing evolution, concepts, architecture, characteristics and applications. The literature review has presented the main concepts and related work of task assignment, load balancing and green computing in fog computing.

Task assignment has been shown to be an NP-hard optimization problem. The decentralized and heterogenous nature of fog networks impose additional challenges on task assignment in fog computing. Moreover, the different application requirements such as ultra-low latency and end user mobility add more complexity to the task assignment problem. Many models have been proposed to address these different requirements and challenges.

Task requirements have been usually represented using multiple individual attributes such as the number of computing resources, execution deadline, maximum latency between the end user and processing node. While this approach can reduce the model complexity, it can result in inefficient utilization of bandwidth and compute resources. In addition, it restricts the ability for dynamic workload allocation over time according to the network status and resource availability. Finally, this approach pushes the complexity towards the end user side to breakdown the overall QoS requirements into multiple separate requirements which limits the ability of these models to address real-life use cases.

Task assignment models in fog computing must be able to address more comprehensive SLA requirements such as E2E task completion time and CoS. This requires the model to be able to breakdown these requirements into different attributes and metrics such as workload, processing delay, resource utilization, latency between end user and processing node. In addition, the model must be able to support different horizontal/vertical interactions between different layers in the fog network based on the model objective. Based on the review, it has been shown that SLA, profit, load balancing and green computing concepts are important aspects in fog computing. To our best knowledge, most of the proposed work have focused on addressing a single requirement such as QoS. Some of the interesting proposals, have used the objective function to address two requirements simultaneously. The authors in [82], have formulated the task assignment problem using a profit objective function. The objective function in such approach can incorporate different incentives and costs to address additional requirements. By maximizing the profit, the model determines an optimal task assignment that satisfies multiple requirements such as profit, latency, energy-efficiency, etc. This approach can be used to address multiple requirements within a single model. This is what we will tackle in the next chapter.

Chapter 3: Formulation of the Task Assignment Problem in Fog Networks

In this chapter, two mathematical models for the Task Assignment with SLA in fog networks (hereinafter referred to as TASLA) are formulated. While both models try to optimize FSP profit by satisfying SLA requirements, each model focuses on an important aspect from the service provider perspective. As discussed earlier in Chapter 2, the energy consumption and the carbon footprint are major challenges for different computing paradigms. As a result, the first model, called TASLA-Green Computing (hereinafter referred to as TASLA-GC), addresses SLA and green computing requirements. The model satisfies the green computing requirements mainly by reducing the fog node operations through an on-demand turn on/off fashion based on the requested resources, node state and resources utilization over time.

While the green computing requirement can be a priority when managing an entire fog network, it might not be the case when the fog network consists of different federations managed by different providers. In addition, third parties and individuals can lease network resources and devices to FSP. This allows the fog service provider to utilize these resources to expand network capacity and proximity towards the end users. However, the FSP doesn't have full control over the leased network elements. As a result, the green computing requirement cannot be addressed due to the FSP's lack of ability to turn on/off third-party network elements on demand. In addition, the lease agreement between the FSP and third party can include fixed rate charges regardless of the actual utilization of the network elements. In this case, a different approach such as load balancing can allow the FSP to efficiently utilize the fog network resources. The second model, called TASLA-Load Balancing (hereinafter referred to as TASLA-LB), addresses SLA and load balancing requirements.

The rest of this chapter is organized as follow. Section 3.1 describes the basic concepts of both models. Section 3.2 describes the system model. Finally, Sections 3.3 and 3.4 describe the formulation for the green computing model and the load balancing model respectively.

3.1 Basic Concepts

In this section, we introduce the basic concepts that will help in the understanding of the two models. We will first discuss how the SLA requirements are modeled and how the profit is calculated.

3.1.1 SLA Requirements

SLA requirements can refer to multiple Key Performance Indicators (KPI). This thesis uses the maximum E2E task completion time (hereinafter referred to as latency) to represent the task SLA requirements. Many factors can contribute to latency such as transmission delay, propagation delay, queuing delay and processing delay. In this thesis, we have considered the propagation and processing delay only.

The propagation delay is the time required for the data to travel with specific speed for a specific distance from one endpoint to another. The distance between the end user and the processing node is the main factor that contribute to the propagation delay.

This thesis defines the processing delay as the time required to successfully process the task instructions on the processing node. The task length, the amount and the duration of the allocated resources are the main factors that contribute to the processing delay.

TASLA doesn't require the end user to include extra attributes to explicitly define the latency requirements in the task request. Instead, the model calculates the task latency requirements using the requested task length and CoS. Class of service commonly refers to a service or payload parameter that is used to differentiate and prioritize traffic types across the network. For example, real-time and latency-sensitive traffic such as voice and signaling usually have higher CoS than other types of traffic across the network. The mathematical models use the CoS to extract the latency requirements where *max latency = length/*CoS. This formula ensures that higher CoS translates into lower latency requirements. For example, when $task_1$ with length 100, CoS 5 and $task_2$ with length 100, CoS 1 are offloaded to the fog network, the model determines the required maximum latency for $task_1$ as 20 and for $task_2$ as 100. The latency calculation can be set by the FSP to reflect their business rules. This allows for an efficient resource utilization and helps FSP to add new requirements to the model in the future without requiring any changes in the task attributes. The model uses the node resource capacity and location to determine whether it can satisfy the latency requirements or not. This provides flexibility for the model to manage and utilize different type of network elements.

As discussed in Chapter 2, one of the main objectives for introducing fog computing is to satisfy the latency requirements that cloud computing is unable to satisfy. For such reason, TASLA doesn't favor the fog node over the cloud DC by default. Instead, the task assignment decision is

based on the ability to satisfy the specific latency requirement for each request. For example, serving latency-sensitive and real-time applications such as mobile gaming and AR requires tasks to be assigned to fog nodes within proximity and with enough resource capacity to satisfy the SLA requirements. Other applications that are not (or less) delay sensitive can safely be served by the cloud DC without the need to overload the limited-resource fog nodes with such requests. In the case where both the fog node(s) and the cloud DC can satisfy the latency requirements and generate the same revenue, the model will assign the task to the node with the least associated cost.

If the model is unable to satisfy the task SLA requirements, the model will assign the task to the cloud DC for processing as a last resort without generating any revenue since the SLA is violated.

3.1.2 Workload Allocation

TASLA determines the workload required for each task assignment as the number of allocated resources on the processing node per time unit. The workload allocation allows for task processing over different time units with variable rates without exceeding the maximum allowed latency. This allows for efficient resource utilization and holistic optimal decisions based on the overall volume of requests and available network resources. For example, a task with length 25, can be processed on a node with resource capacity 15 over two time units, where the node process workload of 15 in the first time unit and workload of 10 in the second time unit. The node will then have 5 free resources in the second time unit that can be used to process additional task(s).

3.1.3 Profit Calculation

The model maximizes the profit objective function by prioritizing task assignments that generate more revenue and less cost (while still satisfying the model constraints). The revenue is calculated based on the task CoS, the volume of the requested resources (task length) and the service rate (allocated workload). Combining the CoS, service volume and service rate in the revenue calculation allows the model to prioritize service requests. For example, in the case where two tasks request the same service volume, the task with higher CoS will be prioritized as it will generate more revenue.

In general, the associated costs and expenses can include both Capital Expenditure (CAPEX) and Operating Expenses (OPEX) related to planning, deployment, operations, maintenance of different

network elements, networking and power infrastructure. TASLA excludes CAPEX (since the infrastructure is already in place) and only calculates OPEX according to each model objective.

TASLA-GC calculates the startup, operation and shutdown costs of the fog nodes. The model tries to minimize the operating costs to maximize the overall profit. This ensures that the model will turn on/off and operate the fog node in an optimal on-demand fashion. This is expected to reduce the carbon footprint of the fog network which achieves the green computing requirements.

TASLA-LB introduces the Load Balancing Factor (LBF) as the difference in the node utilization between every pair of nodes in the fog network. The node utilization is the ratio of the total allocated workload to the node resource capacity. The optimal LBF should be minimum. The model tries to minimize the cost of deviating from the optimal LBF values across the fog network. This ensure that tasks are assigned in a load balancing fashion for an efficient resource utilization of the fog network.

Figure 3.1 shows a high-level overview of the profit objective function in terms of revenue calculation and the specific cost calculation for TASLA-GC and TASLA-LB.

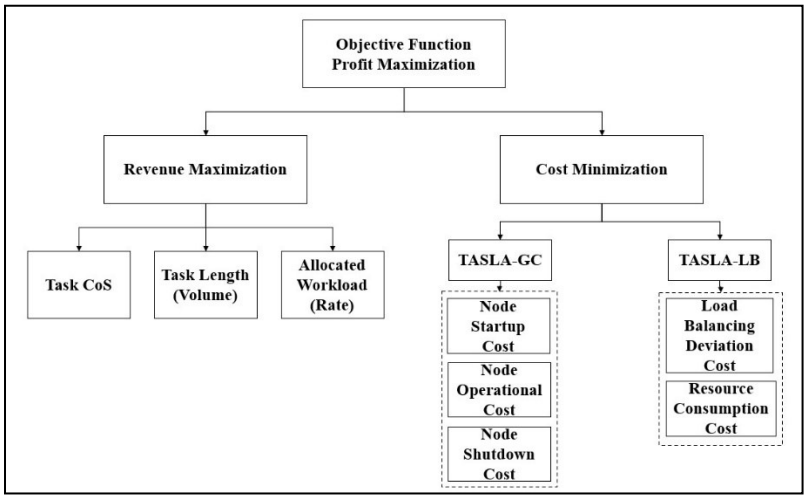

Figure 3.1: TASLA Profit Objective Function

3.2 System Model

In the 3-tier fog network shown in Figure 3.2, the end user can request different type of resources including virtual memory, Virtual Central Processing Unit cores (vCPU), Graphics Processing Unit (GPU), network bandwidth and storage. For simplicity and without loss of generality, this

thesis only considers vCPU resources for task processing. The size of the requested resources or the task length is presented in terms of the number of instructions that are offloaded for processing.

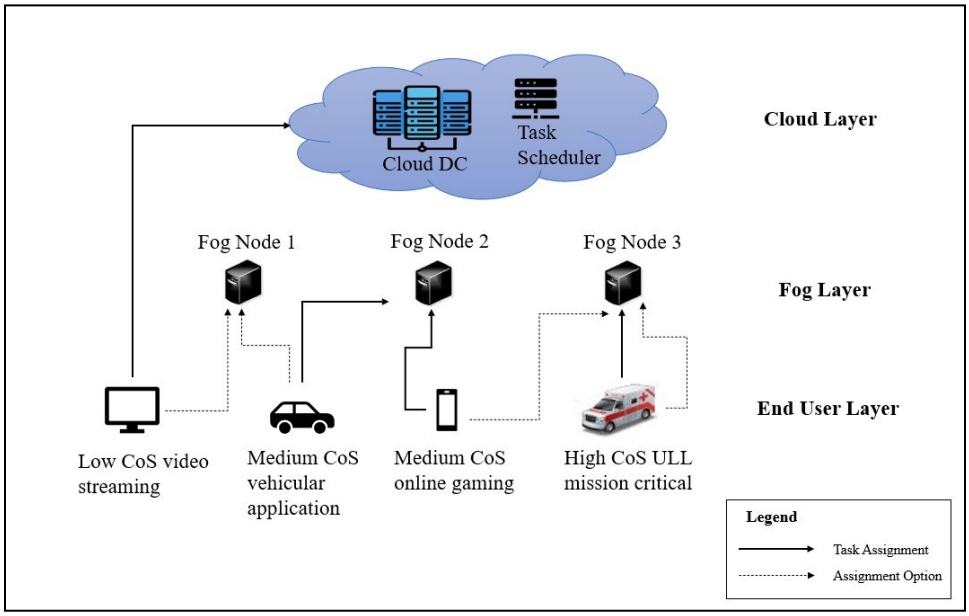

Figure 3.2: System Model

A fog network consists of multiple fog nodes and one cloud DC in different locations with different resource capacity. Each node, whether it's a fog node or the cloud DC, is represented in terms of resource capacity, node coordinates and node type. Following the same approach as in the task representation, only vCPU cores are used to represent node capabilities. The overall node vCPU cores capacity is described as the total number of instructions that can be processed per time unit.

The task scheduler is located in the cloud layer and is responsible for monitoring, gathering information and communicating with different fog network elements. In addition, the task scheduler communicates with the end user to receive task requests and update request status. The task scheduler queues, schedules and assigns tasks based on the model calculations. The task scheduler is required to maintain all the required information of active tasks, fog nodes and cloud DC. The task scheduler leverages the cloud layer massive processing capabilities, high availability to support large-scale and real-time task assignment. However, this might come with the price of a significant delay in the control plane between the task scheduler, fog nodes and end users. This thesis neglects this delay as it is only present during the service setup phase and doesn't impact the actual processing delay. In the case of ultra-low latency applications, network clustering can

be used where a light-weighted version of the task scheduler is installed on a dedicated fog node in each cluster. All task schedulers can be managed by a centralized scheduler in the cloud DC.

3.2.1 General Assumptions

This section enlists the general assumptions for TASLA (i.e., they apply to both models) used in this thesis.

3.2.1.1 Nodes Assumptions

- The cloud DC is operationally up, reachable and available for use 100% of the time. Following this assumption, no turn on/off costs are considered for the cloud DC.
- The cloud DC has unlimited resource capacity.
- The fog nodes have a finite resource capacity.
- The operational and resource consumption costs of the cloud DC are significantly lower than the fog nodes and are assumed to be equal to zero.
- The number, locations and resource capacity of the fog nodes and cloud DC are known (defined during the planning phase of the network) and are used as an input to the model.
- The fog network, including all fog nodes and cloud DC, is assumed to be stable with no unplanned outages.
- All the nodes in the fog network are assumed to be 100% reliable. Once the task is assigned to a node for processing, the task processing will be completed successfully. Reliability or fault tolerance requirements are not considered in this thesis.

3.2.1.2 Communications Assumptions

- There are no bandwidth limitations in the network infrastructure interconnecting the end users, fog nodes, cloud DC and task scheduler.
- The communications between the task scheduler, fog nodes, cloud DC and end users are assumed to be 100% reliable with no disruption.
- The control and signaling traffic between the end users and the task scheduler include the time to authenticate end users, validate request parameters and setup the service. The delay

introduced during this phase is insignificant and neglected in comparison with the overall latency.

- The control and signaling communications between the fog nodes, cloud DC and the task scheduler include the time to gather information and calculations of the node state, available resources, current workload, etc. The delay introduced during this phase is insignificant and neglected in comparison with the overall latency.
- The transmission rate is assumed to be fixed across the entire fog network and is neglected.
- The propagation delay is equal to the cartesian distance between the end user and the fog node or cloud DC.

3.2.1.3 Tasks Assumptions

- The distance between any end user and any fog node is always shorter than the distance between any end user and the cloud DC.
- Task specifications and attributes are pre-defined and are used as an input to the model.
- The tasks, fog nodes and cloud DC use common metrics only for all resources in terms of number of instructions to be processed per time using vCPU cores.
- Any task can be assigned either to a fog node or the cloud DC if the required resources can be supplied and the task latency requirements can be satisfied within the model simulation period.
- Task queuing and scheduling delay is insignificant and neglected in comparison with the overall latency.
- The task can only be processed by a single node for the whole model simulation period. Task migration requirements are not considered in this thesis.
- VM instantiation and allocation within the cloud DC or fog nodes is excluded from the task assignment problem and not considered in this thesis.
- The end user, fog nodes and cloud DC locations are static. Mobility requirements are not considered in this thesis.

3.3 TASLA-GC: Model Formulation

This section describes the first model based on the concept of green computing (TASLA-GC). We will first present the specific assumptions required for this model, followed by the notation and the model formulation. Figure 3.3 shows the main building blocks of the mathematical model.

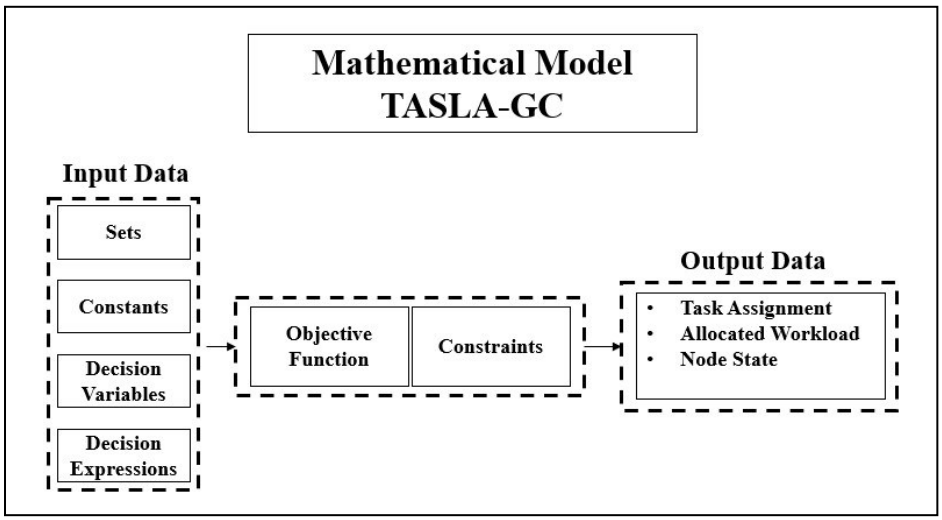

Figure 3.3: TASLA-GC Model Notation

3.3.1 Specific Assumptions for TASLS-GC

On top of all the general assumptions listed in Section 3.2.1, the following specific assumptions are made in the context of green computing.

- A fog node must be turned on to process tasks. If the fog node is turned off, the task scheduler sends a startup request to turn on the fog node.
- A fog node requires one complete time unit to go through the full startup cycle. In other words, when the task scheduler sends a startup request to the fog node, the fog node is operationally up in the next time unit.
- A fog node shuts down within the same time unit. In other words, when the task scheduler sends a shutdown request to the fog node, the fog node is shut down in the same time unit.

3.3.2 Input Data

3.3.2.1 Sets

$I = i_0(\alpha_0, \beta_0, \gamma_0, \delta_0), \ldots\ i_n(\alpha_n, \beta_n, \gamma_n, \delta_n)$ is the set of tasks that are offloaded by the end users to the fog network for processing. Each task is represented by the following attributes:

- α_i, the number of the offloaded task instructions.
- β_i, the CoS requested for offloading task i.
- γ_i, a two-tuple value corresponding to the (x, y) coordinates of task i.
- δ_i, the release time of task i.

$J = j_0(\varepsilon_0, \zeta_0, \eta_0), \ldots\ j_n(\varepsilon_m, \zeta_m, \eta_m)$ is the set of nodes in the network including the fog nodes and cloud DC. Each node has the following attributes:

- ε_j, the type of node $j \in \{0$ is cloud DC, 1 is fog node$\}$.
- ζ_j, resource capacity of node j. This represents the vCPU capabilities of node j in term of the number of instructions that can be processed per time unit.
- η_j, a two-tuple value corresponding to the (x, y) coordinates of node j.

$T = t_0, \ldots\ t_k$ is the set of time units. It contains a set of iterative values between 0 and k, where k is the time required to complete the model simulation. t is used as the time unit for different time-based variables and functions.

3.3.2.2 Constants

The following constants are set by the FSP to determine the fog network costs:

- CU is a constant indicating the cost (in currency unit) to change the fog node state from down to up.
- CD is a constant indicating the cost (in currency unit) to change the fog node state from up to down.
- OC is a constant indicating the fog node operational cost (in currency unit).

3.3.2.3 Decision Variables

The following decision variables are defined.

- x_{ij} is a binary variable indicating if task i is assigned to node j or not.
- y_{ijt} is a binary variable indicating if task i is being allocated resources on node j at time t or not.
- w_{ijt} is a positive integer variable indicating the workload as the number of resources allocated for task i on node j at time t.
- θ_{jt} is a binary variable that represents the operational state of node j at time t (1 is up, 0 is down).
- λ_{jt} is a binary variable that represents the startup state of node j at time t (1 node is going through a startup, 0 is not).
- μ_{jt} is a binary variable that represent the shutdown state of node j at time t (1 node is going through a shutdown, 0 is not).

Representing the node state using three different variables (θ_{jt}, λ_{jt}, and μ_{jt}) allows the objective function to apply different costs for each state (startup, operational and shutdown).

3.3.2.4 Decision Expressions

Eqns. 3.1 and 3.2 calculate the parameters used by the objective function and constraints defined in Sections 3.3.3 and 3.3.4.

- ν_i is a positive integer variable used to calculate the maximum allowed latency for task i in terms of time units.

$$\nu_i = \alpha_i / \beta_i \tag{3.1}$$

- ξ_{ij} is a positive integer variable used to calculate the distance between task i and node j that represents the propagation delay.

$$\xi_{ij} = \sqrt{\left(\gamma_i(x) - \eta_j(x)\right)^2 + \left(\gamma_i(y) - \eta_j(y)\right)^2} \tag{3.2}$$

3.3.3 Objective Function

The objective function of TASLA-GC tries to maximize profit generated by the fog service provider. As a general statement, the profit is the difference between the generated revenue (R) and associated costs (C). Eqn. 3.3 calculates the revenue.

$$R = \sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{t=0}^{k}(\alpha_i * \beta_i * w_{ijt}) \tag{3.3}$$

The revenue calculation combines the task CoS used for prioritization, the task length that represents the service volume and the allocated workload that represents the service rate. Eqn. 3.4 calculates the associated cost of operating the fog network. The cost calculation combines the operational, startup and shutdown costs for the fog nodes.

$$C = \sum_{j=0}^{m}\sum_{t=0}^{k}((\theta_{jt} * OC\ ) + (\lambda_{jt} * CU) + (\mu_{jt} * CD)) \tag{3.4}$$

Linear programing is used to maximize the profit (P) as shown in Eqn. 3.5.

$$\max P = max\{R - C\} \tag{3.5}$$

The objective function is used to calculate P generated by assigning the set of tasks I to the set of nodes J within the set of time units T. The maximization of the profit objective presented in Eqn. 3.5 can be achieved by maximizing revenue R presented in Eqn. 3.3 and minimizing costs C presented in Eqn. 3.4. The constraints defined in Section 3.3.4 ensures that a successful task assignment satisfies the SLA requirements. In other words, the model prioritizes and tries to satisfy the latency requirements for the tasks with higher CoS and longer task length as they generate more revenue. The model simultaneously assigns tasks to nodes with lower operational costs and without enduring unnecessary startup or shutdown actions to minimize the overall costs to achieve the green computing requirement.

In case the task latency requirements cannot be satisfied, the task is assigned to the cloud DC as a last resort without generating any revenue since the SLA is violated.

3.3.4 Constraints

This section describes the constraints used to model the problem. The constraints are divided into six main categories as illustrated in Figure 3.4. The node state constraints are specific to TASLA-GC while the other five categories are common in TASLA.

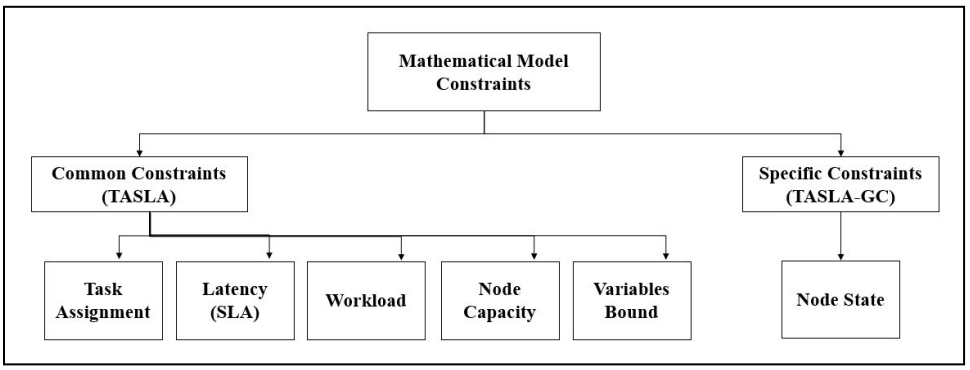

Figure 3.4: TASLA-GC Model Constraint Categories

3.3.4.1 Task Assignment Constraints

- *Task Assignment:* Eqn. 3.6 ensures that each task i can only be assigned to at most one node (either fog node or cloud DC).

$$\sum_{j \in J} x_{ij} \leq 1 \quad \{ i \in I \} \tag{3.6}$$

- *Resource Allocation:* Eqn. 3.7 ensures that node j can allocate resources (y_{ijt}) to task i only if task i is already assigned (x_{ij}) to node j.

$$y_{ijt} \leq x_{ij} \quad \{ i \in I,\ j \in J,\ t \in T \} \tag{3.7}$$

- *Task Assignment Start Time:* Eqn. 3.8 ensures that task i cannot be allocated resources (y_{ijt}) before its release time (δ_i).

$$y_{ijt} = 0 \quad \{ i \in I,\ j \in J,\ t \leq \delta_i \in T \} \tag{3.8}$$

3.3.4.2 Latency Requirements Constraints

- *Task Assignment End Time:* Eqn. 3.9 ensures that task i cannot be allocated resources (y_{ijt}) after the maximum allowed latency (E2E overall task completion time $\delta_i + \nu_i$). This

constraint ensures that the task assignment satisfies the latency requirements with respect to the actual task release time ($\delta_i + \nu_i$).

$$y_{ijt} = 0 \quad \{ i \in I,\ j \in J,\ t > (\delta_i + \nu_i) \in T\} \tag{3.9}$$

- *Latency Requirements:* Eqn. 3.10 ensures that the task assignment satisfies the latency requirements. More precisely, it makes sure that the sum of the processing delay ($\sum_{t \in T} y_{ijt}$) and the propagation delay (ξ_{ij}) between assigned task i and node j over the model simulation period T doesn't exceed the latency requirements (ν_i) of task i.

$$\sum_{t \in T} y_{ijt} \leq (x_{ij} * (\nu_i - \xi_{ij})) \quad \{ i \in I, j \in J\} \tag{3.10}$$

3.3.4.3 Workload Constraints

- *Workload Lower Bound:* Eqn. 3.11 ensures that task i can only consume workload (w_{ijt}) when the node resources have been allocated (y_{ijt}). This constraint sets the minimum value for the workload (w_{ijt}) to zero.

$$w_{ijt} \geq y_{ijt} \quad \{ i \in I,\ j \in J,\ t \in T\} \tag{3.11}$$

- *Workload Upper Bound:* Eqn. 3.12 ensures that the workload (w_{ijt}) consumed by task i at time t cannot exceed the total task length (α_i).

$$w_{ijt} \leq (y_{ijt} * \alpha_i) \quad \{ i \in I,\ j \in J,\ t \in T\} \tag{3.12}$$

- *Workload Allocation:* Eqn. 3.13 ensures that the total workload ($\sum_{t \in T} w_{ijt}$) consumed by task i over the model simulation period T must be equal to the task length (α_i). This constraint ensures that the task is completely processed using the total allocated workload over the duration of the model simulation.

$$\sum_{t \in T} w_{ijt} = (x_{ij} * \alpha_i) \quad \{ i \in I,\ j \in J\} \tag{3.13}$$

3.3.4.4 Node Capacity Constraints

- *Node Resource Capacity:* Eqn. 3.14 ensures that the total workload of all assigned tasks to node j at time t ($\sum_{i \in I} w_{ijt}$) cannot exceed the node resource capacity (ζ_j).

$$\sum_{i \in I} w_{ijt} \leq \zeta_j \quad \{ j \in J,\ t \in T \} \tag{3.14}$$

3.3.4.5 Node State Constraints

- *Node State:* Eqn. 3.15 ensures that each node can only be in one valid state at time t (startup, up, down or shutdown).

$$\theta_{jt} + \lambda_{jt} + \mu_{jt} \leq 1 \quad \{ j \in J, t \in T \} \tag{3.15}$$

  This constraint ensures the following node states:

  - Node is going through a startup: $\lambda_{jt} = 1$.
  - Node is operationally up: $\theta_{jt} = 1$.
  - Node is going through a shutdown: $\mu_{jt} = 1$.
  - Node is operationally down: $\theta_{jt} = 0$.

- *Node Startup:* Eqn. 3.16 ensures the valid values for the node state and startup variables. This constraint ensures that if a node is down and going through a startup at time t, the node will be up at time "$t + 1$". For example: $\theta_{j2} = 0$, $\lambda_{j2} = 1$, $\theta_{j3} = 1$.

$$\theta_{jt} + \lambda_{jt} \geq \theta_{j(t+1)} \quad \{ j \in J, t \in T \} \tag{3.16}$$

- *Node Shutdown:* Eqn. 3.17 ensures the valid values for the node state and shutdown variables. This constraint ensures that if a node was up at time "$t - 1$" and went through a shutdown at time t, the node will be operationally down at the same time t. For example: $\theta_{j2} = 0$, $\mu_{j2} = 1$, $\theta_{j1} = 1$.

$$\theta_{jt} + \mu_{jt} \geq \theta_{j(t-1)} \quad \{ j \in J, t \in T \} \tag{3.17}$$

3.3.4.6 Variables Bound Constraints

- *Variables Bound:* Eqns. 3.18 to 3.22 determine the bounds for variables x_{ij}, y_{ijt}, θ_{jt}, λ_{jt} and μ_{jt}.

$$x_{ij} \in \{0,1\} \quad \{ i \in I,\ j \in J \} \tag{3.18}$$

$$y_{ijt} \in \{0,1\} \quad \{ i \in I,\ j \in J,\ t \in T \} \tag{3.19}$$

$$\theta_{jt} \in \{0,1\} \quad \{j \in J,\ t \in T\} \tag{3.20}$$

$$\lambda_{jt} \in \{0,1\} \quad \{j \in J,\ t \in T\} \tag{3.21}$$

$$\mu_{jt} \in \{0,1\} \quad \{j \in J,\ t \in T\} \tag{3.22}$$

3.4 TASLA-LB: Model Formulation

This section describes the second model based on the concept of load balancing (TASLA-LB). Similar to the previous model, we will first present how the load balancing concept was formulated followed by the specific assumptions, the notation, and the model formulation. Figure 3.5 show the main building blocks of the mathematical model.

It is important to note that since TASLA-GC and TASLA-LB share some common details, some of the details that have been previously described in Section 3.3 will be described again in this section for the sake of clarity and completeness. This way, each model can be presented without having to combine equations from different sections.

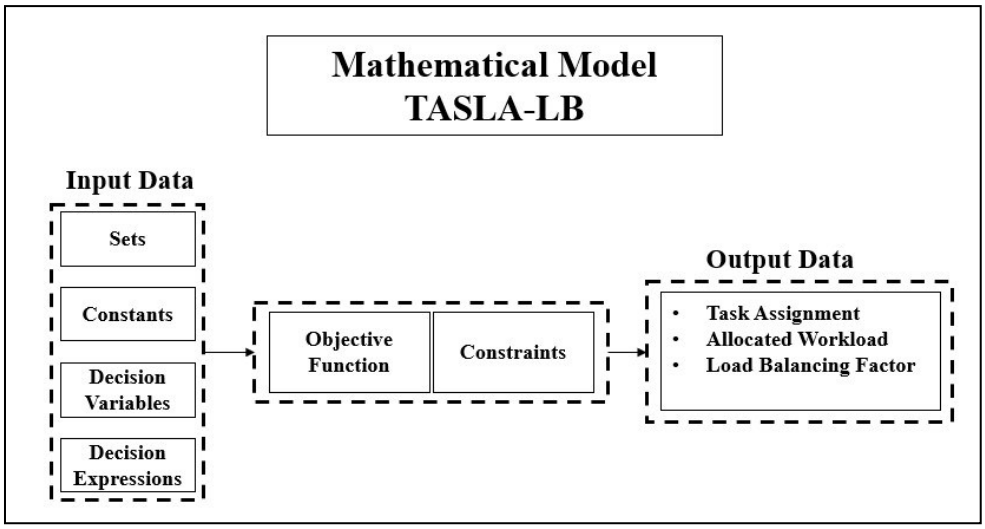

Figure 3.5: TASLA-LB Model Notation

3.4.1 Load Balancing Formulation

The load balancing requirement can be formulated and achieved using multiple approaches. For example, the load balancing can be based on the total number of assigned tasks per node. In such case, a round robin approach can be used to achieve load balancing. However, if tasks have different sizes, this could lead to an imbalance across the nodes.

To avoid this issue, TASLA-LB assign tasks in a load balancing fashion based on the node utilization. The main goal is to maintain an equal level of node utilization across the fog network while simultaneously satisfying the tasks latency requirements. The load balancing is determined based on the difference in utilization between all nodes in the fog network. The optimal load balance means that the absolute difference of node utilization between every pair of nodes should be minimum. To give the model an incentive to minimize the difference between nodes utilization, a deviation cost is introduced. The deviation cost is proportional to and associated with the absolute difference term in the objective function. The model optimization aims to minimize the absolute difference term in order to minimize the endured deviation cost which leads to maximizing the profit. However, introducing the absolute difference term will result in a nonlinear profit objective function. According to [83][84], the following steps can be performed to linearize the objective function and minimize the absolute difference term:

- The term which represents the absolute difference of node utilization between two nodes (e.g., $|A - B|$) must be replaced by new decision variable (e.g., LBF) in the objective function.
- The newly introduced decision variable LBF will be used in the profit objective function as part of the cost calculations alongside with the deviation cost.
- The constraint: $LBF = |A - B|$ is required to link the LBF optimization to the absolute difference term.
- The equality constraint can then be changed to an inequality constraint:
    - $LBF \geq |A - B|$
    - As the model is optimizing the costs by minimizing LBF, the solution is optimal if: $LBF = |A - B|$
- $|A - B|$ can then be described as: $\max \{A - B, B - A\}$.
- This can further be formulated into three constraints:
    - $LBF \geq A - B$
    - $LBF \geq B - A$
    - $LBF \geq 0$

3.4.2 Specific Assumptions for TASLA-LB

On top of all the general assumptions listed in Section 3.2.1, the following specific assumption is made in the context of load balancing.

- All the fog nodes and cloud DC in the fog network are up, reachable, operational 100% of the time and available for task assignment without any turn on/off actions.

3.4.3 Input Data

3.4.3.1 Sets

$I = i_0(\alpha_0, \beta_0, \gamma_0, \delta_0), \ldots\ i_n(\alpha_n, \beta_n, \gamma_n, \delta_n)$ is the set of tasks that are offloaded by the end users to the fog network for processing. Each task is represented by the following attributes:

- α_i the number of the offloaded task instructions.
- β_i the CoS requested for offloading task i.
- γ_i a two-tuple value corresponding to the (x, y) coordinates of task i.
- δ_i the release time of task i.

$J = j_0(\varepsilon_0, \zeta_0, \eta_0), \ldots\ j_n(\varepsilon_m, \zeta_m, \eta_m)$ is the set of nodes in the network including the fog nodes and cloud DC. Each node has the following attributes:

- ε_j the type of node $j \in \{0$ is cloud DC, 1 is fog node$\}$.
- ζ_j resource capacity of node j. This represents the vCPU capabilities of node j in term of the number of instructions that can be processed per time unit.
- η_j a two-tuple value corresponding to the (x, y) coordinates of node j.

$T = t_0, \ldots\ t_k$ is the set of time units. It contains a set of iterative values between 0 and k, where k is the time required to complete the model simulation. t is used as the time unit for different time-based variables and functions.

3.4.3.2 Constants

The following constants are set by the FSP to determine the fog network costs:

- DC is a constant indicating the deviation cost (in currency unit) endured for deviating from the optimal LBF value.
- RC is a constant indicating the resource consumption cost (in currency unit).

3.4.3.3 Decision Variables

The following decision variables are defined.

- x_{ij} is a binary variable indicating if task i is assigned to node j or not.
- y_{ijt} is a binary variable indicating if task i is being allocated resources on node j at time t or not.
- w_{ijt} is a positive integer variable indicating the workload as the number of resources allocated for task i on node j at time t.
- Z_{lq} is a positive integer variable indicating the LBF between node l and node q.

3.4.3.4 Decision Expressions

Eqns. 3.23, 3.24 and 3.25 calculate the parameters used by the objective function and constraints defined in Sections 3.4.4 and 3.4.5.

- ν_i is a positive integer variable used to calculate the maximum allowed latency for task i in terms of time units.

$$\nu_i = \alpha_i/\beta_i \tag{3.23}$$

- ξ_{ij} is a positive integer variable used to calculate the distance between task i and node j that represents the propagation delay.

$$\xi_{ij} = \sqrt{\left(\gamma_i(x) - \eta_j(x)\right)^2 + \left(\gamma_i(y) - \eta_j(y)\right)^2} \tag{3.24}$$

- ρ_j is a positive integer variable used to calculate the node utilization. The node utilization is the sum of all the workload w_{ijt} consumed by all tasks I over the model simulation period T divided by the node resource capacity ζ_j.

$$\rho_j = \sum_{i=0}^{n} \sum_{t=0}^{k} w_{ijt} / \zeta_j \tag{3.25}$$

3.4.4 Objective Function

The objective function of TASLA-LB tries to maximize profit generated by the fog service provider. The profit is the difference between the generated revenue (R) and associated cost (C). Eqn. 3.26 calculates the revenue R.

$$R = \sum_{i=0}^{n} \sum_{j=0}^{m} \sum_{t=0}^{k} (\alpha_i * \beta_i * w_{ijt}) \tag{3.26}$$

The revenue calculation combines the task CoS used for prioritization, task length that represents the service volume and the allocated workload that represents the service rate. Eqn. 3.27 calculates the associated cost C. The cost calculation combines the deviation cost and the resource consumption cost.

$$C = \sum_{i=0}^{n} \sum_{j=0}^{m} \sum_{t=0}^{k} (y_{ijt} * RC\ ) + \sum_{\substack{l \in J \\ l<q}} \sum_{q \in J} (DC * Z_{lq}) \tag{3.27}$$

The cost calculation combines the operational costs and the deviation costs. The deviation cost is associated with the LBF (Z_{lq}) between node l and node q where $l, q \in J : l < q$. This ensures that the LBF is not calculated twice for the same pair of nodes.

Linear programing is used to maximize the profit (P) as shown in Eqn. 3.28.

$$\max P = max \{R - C\} \tag{3.28}$$

In case the task latency requirements cannot be satisfied, the task is assigned to the cloud DC as a last resort without generating any revenue since the SLA is violated.

3.4.5 Constraints

This section describes the constraints used to model the problem. The constraints are divided into six main categories as illustrated in Figure 3.6. The load balancing constraints are specific to TASLA-LB while the other five categories are common in TASLA.

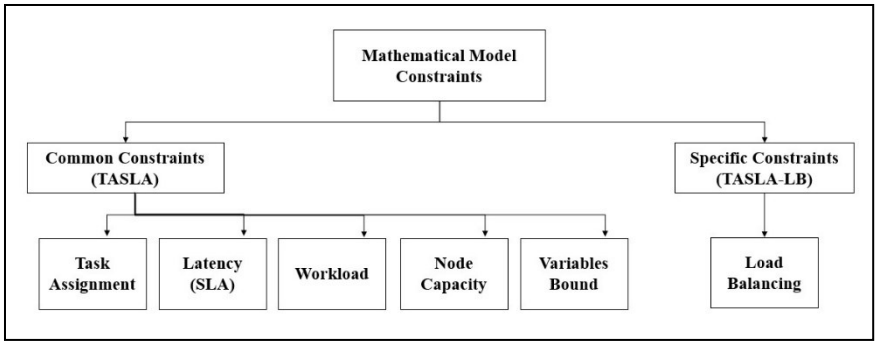

Figure 3.6: TASLA-LB Model Constraint Categories

3.4.5.1 Task Assignment Constraints

- *Task Assignment:* Eqn. 3.29 ensures that each task i can only be assigned to at most one node (either fog node or cloud DC).

$$\sum_{j \in J} x_{ij} \leq 1 \quad \{ i \in I \} \tag{3.29}$$

- *Resource Allocation:* Eqn. 3.30 ensures that node j can allocate resources (y_{ijt}) to task i only if task i is already assigned (x_{ij}) to node j.

$$y_{ijt} \leq x_{ij} \quad \{ i \in I,\ j \in J,\ t \in T \} \tag{3.30}$$

- *Task Assignment Start Time:* Eqn. 3.31 ensures that task i cannot be allocated resources (y_{ijt}) before its release time (δ_i).

$$y_{ijt} = 0 \quad \{ i \in I,\ j \in J,\ t \leq \delta_i \in T \} \tag{3.31}$$

3.4.5.2 Latency Requirements Constraints

- *Task Assignment End Time:* Eqn. 3.32 ensures that task i cannot be allocated resources (y_{ijt}) after the maximum allowed latency (E2E overall task completion time $\delta_i + \nu_i$). This constraint ensures that the task assignment satisfies the latency requirements with respect to the actual task release time ($\delta_i + \nu_i$).

$$y_{ijt} = 0 \quad \{ i \in I,\ j \in J,\ t > (\delta_i + \nu_i) \in T \} \tag{3.32}$$

- *Latency Requirements:* Eqn. 3.33 ensures that the task assignment satisfies the latency requirements. More precisely, it makes sure that the sum of the processing delay ($\sum_{t\in T} y_{ijt}$) and the propagation delay (ξ_{ij}) between assigned task i and node j over the model simulation period T doesn't exceed the latency requirements (ν_i) of task i.

$$\sum_{t\in T} y_{ijt} \leq (x_{ij} * (\nu_i - \xi_{ij})) \quad \{ i \in I, j \in J\} \tag{3.33}$$

3.4.5.3 Workload Constraints

- *Workload Lower Bound:* Eqn. 3.34 ensures that task i can only consume workload (w_{ijt}) when the node resources have been allocated (y_{ijt}). This constraint sets the minimum value for the workload (w_{ijt}) to zero.

$$w_{ijt} \geq y_{ijt} \quad \{ i \in I,\ j \in J,\ t \in T\} \tag{3.34}$$

- *Workload Upper Bound:* Eqn. 3.35 ensures that the workload (w_{ijt}) consumed by task i at time t cannot exceed the total task length (α_i).

$$w_{ijt} \leq (y_{ijt} * \alpha_i) \quad \{ i \in I,\ j \in J,\ t \in T\} \tag{3.35}$$

- *Workload Allocation:* Eqn. 3.36 ensures that the total workload ($\sum_{t\in T} w_{ijt}$) consumed by task i over the model simulation period T must be equal to the task length (α_i). This constraint ensures that the task is completely processed using the total allocated workload over the duration of the model simulation.

$$\sum_{t\in T} w_{ijt} = (x_{ij} * \alpha_i) \quad \{ i \in I,\ j \in J\} \tag{3.36}$$

3.4.5.4 Node Capacity Constraints

- *Node Resource Capacity:* Eqn. 3.37 ensures that the total workload of all assigned tasks to node j at time t ($\sum_{i\in I} w_{ijt}$) cannot exceed the node resource capacity (ζ_j).

$$\sum_{i\in I} w_{ijt} \leq \zeta_j \quad \{ j \in J,\ t \in T\} \tag{3.37}$$

3.4.5.5 Load Balancing Constraints

- As discussed in Section 3.4.1, the constraints presented in Eqns. 3.38 to 3.40 are used to linearize the profit objective function presented in Eqn. 3.28 and link the LBF (Z_{lq}) value to the absolute difference in utilization (ρ_j) between node l and node q.

$$Z_{lq} \geq \rho_l - \rho_q \quad \{l, q \in J : l < q\} \tag{3.38}$$

$$Z_{lq} \geq \rho_q - \rho_l \quad \{l, q \in J : l < q\} \tag{3.39}$$

$$Z_{lq} \geq 0 \quad \{l, q \in J : l < q\} \tag{3.40}$$

3.4.5.6 Variables Bound Constraints

- *Variables Bound:* Eqns. 3.41 and 3.42 determine the bounds for variables x_{ij} and y_{ijt}.

$$x_{ij} \in \{0,1\} \quad \{i \in I,\ j \in J\} \tag{3.41}$$

$$y_{ijt} \in \{0,1\} \quad \{i \in I,\ j \in J,\ t \in T\} \tag{3.42}$$

Chapter 4: Results and Analysis

This chapter describes the detailed validation, test results evaluation and analysis of TASLA-GC and TASLA-LB. More precisely, Section 4.1 describes the simulation environment used for developing and testing the models. Section 4.2 describes the validation procedures and presents detailed examples. Finally, Section 4.3 describes the test results, computational complexity, performance evaluation and analysis of each model.

4.1 Simulation Environment

The mathematical models were developed using Optimization Programming Language (OPL). IBM ILOG CPLEX Optimization Studio version 20.1 with default settings was used to execute the models. Solving the model in CPLEX can be performed using CPLEX Integrated Development Environment (IDE) or interactive CPLEX. CPLEX IDE uses Graphical User Interface (GUI) while interactive CPLEX uses oplrun Command Line Interface (CLI). When using the GUI, the OPL conversion layers and the required time to load the input data, mathematical model and present the output data in IDE, significantly increase the total execution time. The CLI performance has proven to be better than the GUI in terms of the execution time and the ability to automate the execution of the test plan. For such reason, all the tests developed to validate and evaluate the performance of the mathematical models were executed using oplrun CLI.

The computer that was used to generate input data sets and execute the test plan has the following specifications:

- Operating System: Microsoft Windows 10 64-bit
- Processor: AMD 2.6 GHz 2 Cores
- Memory: 8 GB
- Storage: 931 GB HDD

4.2 Mathematical Models Validation

The validation of TASLA-GC and TASLA-LB was accomplished through three main steps:

1. Test Plan Generation
    a. Multiple tests were designed to individually evaluate every single constraint of the mathematical models. The values of the input data sets were carefully selected to ensure that each individual test neutralizes all the constraints used in the model except for one constraint in order to determine whether the model perform as expected in term of the task assignment and other decision variables or not.
        i. For example, to test the task release time and its impact, the task data set was generated to include multiple tasks with different release times and the same value for other task attributes such as the task length, CoS and location. Similarly, the node data set was generated to include multiple fog nodes with the same values for resource capacity and location.
        ii. Another example was to test the node location and its impact. The node data set was generated to include multiple nodes with different coordinates and the same resource capacity.
    b. The input data sets used for the test plan were designed to simulate a simple and small fog network with multiple task sizes, node sizes and simulation periods.
2. Manual Calculations
    a. Manual calculations against each input data set were performed. The results of the manual calculations determine the profit and the associated task assignment according to the equations used in each mathematical model.
3. Model Execution in CPLEX
    a. The next step was to execute each mathematical model in CPLEX using the same input data sets of each test that were used for manual calculations.
    b. The CPLEX output of each test includes the value of the optimal profit and the associated decision variables.
    c. The CPLEX output of each test was compared to manual calculations against the same input data sets. The output from CPLEX was identical to the manual calculations for all the performed tests.

The above procedure was performed to test all the constraints in each model. Both mathematical models were validated to perform as per the expected behaviour.

Table 4.1 and Table 4.2 show the constants pre-set in TASLA-GC and TASLA-LB respectively. These constants are used for all the validation and evaluation testing of the mathematical models in this thesis unless explicitly stated otherwise. These constants can be used by the FSP to set the different costs in the network and determine the optimal profit values as described in Chapter 3.

As shown in Table 4.1, the startup cost of a fog node is significantly higher than the operational cost (5:1). This is to ensure that the model will always avoid turning on a fog node unless it will generate more revenue by satisfying the SLA requirements for a task that cannot be satisfied on other operational nodes. In addition, the shutdown cost is double the operational cost. This is to ensure that the model will avoid turning off a fog node unless it is idle for two consecutive time units.

Table 4.1: TASLA-GC Constants

Constant	**Value**
Operational cost (*OC*)	1
Startup cost (*CU*)	5
Shutdown cost (*CD*)	2

Table 4.2: TASLA-LB Constants

Constant	**Value**
Resource consumption cost (*RC*)	1
Deviation cost (*DC*)	1

4.2.1 TASLA-GC Detailed Example

This section describes a detailed example used to validate TASLA-GC. In addition, this example shows how the model implements the task assignment, satisfy the SLA and green computing requirements. More precisely, this example is designed to demonstrate the impact of task, node attributes and different costs on the propagation delay, service prioritization, variable workload allocation rates and different node states change over the model's simulation period. Table 4.3 shows the set of tasks I and Table 4.4 shows the set of nodes J. These input data sets are pre-generated and used for the model iterations using set of time units $|T| = 11$ (0-10).

Table 4.3: Set of Tasks I

$\|I\|$	Task i	Length (α_i)	CoS (β_i)	(x, y) Coordinates (γ_i)	Release Time (δ_i)	Latency (v_i) *
5	1	200	20	(0, 0)	0	10
	2	200	5	(0, 0)	0	40
	3	200	1	(0, 0)	0	200
	4	100	20	(5, 0)	1	5
	5	100	5	(5, 0)	1	20

* The maximum allowed latency v_i is not a task attribute. Instead v_i is a decision expression calculated by the model using Length α_i and CoS β_i , where $\boldsymbol{v_i = \alpha_i/\beta_i}$.

Table 4.4: Set of Nodes J

$\|J\|$	Node j	Type (ε_j)	Resource Capacity (ζ_j)	(x, y) Coordinates (η_j)
3	1	0 (Cloud DC)	Unlimited	(100, 100)
	2	1 (Fog node)	25	(10, 0)
	3	1 (Fog node)	50	(5, 0)

As it can be noticed, the task set I includes tasks with multiple CoS values. Tasks 1 and 4 have the highest CoS, tasks 2 and 5 have medium CoS and task 3 has the lowest CoS. When comparing tasks 1, 2 and 3, it can be noticed that they all have the same task length, location and release time. The model is expected to prioritize tasks with higher CoS to maximize the profit according to the objective function presented in Eqn. 3.5 which calculates the revenue based on task length, CoS

and allocated workload. When comparing tasks 1 and 3, although they both have the same task length of 200, task 1 has a higher CoS of 20. Accordingly, the model is expected to prioritize task 1 which will generate more revenue. Finally, the model calculates the task latency requirements using Eqn. 3.1 where $\boldsymbol{\nu_i = \alpha_i/\beta_i}$.

The simulated fog network consists of two fog nodes and the cloud DC in three different locations. Both fog nodes are much closer than the cloud DC to the tasks. The model calculates the propagation delay between each task and the candidate processing node using Eqn. 3.2. The model uses the propagation delay and the node resource capacity to determine which nodes can process the tasks based on the task requirements.

Table **4.5** shows the tasks latency requirements and the calculated propagation delay for each task and node pair.

Table 4.5: Propagation Delay

		Node *j*		
Task *i*	**Latency (ν_i)**	1 (Cloud DC)	2 (Fog node)	3 (Fog node)
1	10	141	10	5
2	40	141	10	5
3	200	141	10	5
4	5	138	5	0
5	20	138	5	0

Based on the above information, below are few notes on the expected task assignment results:

- Task 3 can be assigned to the cloud DC as the maximum required latency is 200 which is larger than the propagation delay between task 3 and the cloud DC of 141. This allows a room for a maximum processing delay of 59.
    - Assigning task 3 to the cloud DC will generate more profit when compared to the fog nodes as the latency requirements can be satisfied with no associated costs on the cloud DC since it doesn't have any operational, startup or shutdown costs according to the assumptions in Section 3.2.1.
    - This allows the model to save the limited resources on the fog nodes to be able to process other tasks which have lower latency requirements and cannot be assigned to the cloud DC.

- Tasks 1, 2, 4 and 5 will not generate any profit if assigned to the cloud DC as the latency requirements will not be satisfied due to the high propagation delay between the tasks and the cloud DC.
- Tasks 1 will not generate any profit if assigned to fog node 2 as the max required latency is 10, while the propagation delay between task 1 and fog node 2 is 10. In other words, the required processing delay for task 1 on fog node 2 must be zero, which is impossible to achieve.
- Task 4 cannot be assigned to fog node 2 as the maximum allowed latency is 5 while the propagation delay between task 4 and fog node is 5, this means that the required processing delay to meet the SLA must be zero, which is impossible to achieve.
- In the case where the maximum allowed latency exceeds the actual model duration, the model will still determine a task assignment based on the task CoS, where the task processing is completed within the model duration if there is enough resource capacity. For example, task 3 has a maximum allowed latency of 200 time units while the actual model runs for 11 time units only. In such case, the task will be assigned for processing and the task processing must be completed by t_{10}. This allows the model to satisfy the latency requirements and maximize the profit.
- In summary, the model is expected to assign task 1 to fog node 3 and task 3 to the cloud DC. Task 4 is expected to be assigned to fog node 3. Finally, fog node 3 will not have enough resource capacity within the model simulation period to address the SLA for both task 2 and 5 simultaneously. As task 2 is released before task 5, task 2 is expected to be assigned to fog node 3 which is already turned on, while task 5 will be assigned to fog node 2 which has to be turned on.

After solving the model with CPLEX, the results shown in Table **4.6** to Table **4.9** were obtained. The optimal solution returned a profit of 1,289,971 units and was solved in 0.19 seconds (sec). Table **4.6** shows that only task 3 has been assigned to the cloud DC. Task 5 has been assigned to fog node 2 while tasks 1, 2 and 4 have been assigned to fog node 3.

Table 4.6: Task Assignment (x_{ij})

	Node *j*		
Task *i*	1 (Cloud DC)	2 (Fog node)	3 (Fog node)
1	0	0	1
2	0	0	1
3	1	0	0
4	0	0	1
5	0	1	0

Table 4.7: Resource Allocation

Task *i*	**Node *j***	**Time Unit *t***	**Resource Allocation (y_{ijt})**	**Workload (w_{ijt})**
1	3	1	1	50
1	3	2	1	50
1	3	4	1	50
1	3	5	1	50
2	3	7	1	50
2	3	8	1	50
2	3	9	1	50
2	3	10	1	50
3	1	2	1	200
4	3	3	1	50
4	3	6	1	50
5	2	4	1	25
5	2	5	1	13
5	2	6	1	12
5	2	7	1	12
5	2	8	1	13
5	2	9	1	13
5	2	10	1	12

* Only non-zero values are included

As an example, the results show that task 1 has been allocated 50 resources at t_1, t_2, t_4 and t_5 on fog node 3 for a total of 200 as initially requested. In result, the processing delay for task 1 is equal to 5 since the task release time was t_0. If we add the propagation delay between task 1 and fog node

3 (also equal to 5), the actual latency is equal to 10. This satisfies the latency requirements of 10 for task 1. In addition, the results show that the allocated workload is variable over different time units. For example, task 5 is allocated a workload at t_5 equal to 13, while the workload at t_6 is equal to 12.

Table 4.8 and Table 4.9 show the node different states over the model simulation time.

Table 4.8: Node Startup State (λ_{jt})

	Time Unit *t*	
Node *j*	0	3
2 (fog node)	0	1
3 (fog node)	1	0

Table 4.9: Node Operational State (θ_{jt})

	Time Unit *t*										
Node *j*	0	1	2	3	4	5	6	7	8	9	10
1 (cloud DC)	1	1	1	1	1	1	1	1	1	1	1
2 (fog node)	0	0	0	0	1	1	1	1	1	1	1
3 (fog node)	0	1	1	1	1	1	1	1	1	1	1

As expected, the results show that only fog nodes 2 and 3 have gone through a startup. The cloud DC didn't go through any startup as it is already operationally up 100% of the time. As task 1 was released at t_0 and was assigned to fog node 3, fog node 3 had gone through a startup at t_0 in order to be operationally up at t_1 and process task 1. Similarly, task 5 was released at t_1 with a latency requirement of 20, was assigned to fog node 2 which had gone through a startup at t_3 to process the task.

Finally, both fog nodes 2 and 3 were operationally up after their initial startup for the entire simulation period of the model. In other words, the fog nodes didn't go through a shutdown in order to process all the assigned task and satisfy the SLA requirement to maximize the profit.

4.2.2 TASLA-LB Detailed Examples

This section describes a detailed example used to validate TASLA-LB. In addition, this example is used to show how the model implements the task assignment, satisfies the SLA and load balancing requirements. More precisely, this example is designed to demonstrate how the model can assign tasks in an optimal load balancing fashion over nodes that only satisfies the SLA requirements. The example includes an odd number of tasks and odd number of nodes including the cloud DC. Table 4.10 shows the set of tasks *I* and Table 4.11 shows the set of nodes *J*. These input data sets are pre-generated and used for the model iterations using set of time units $|T| = 5$ (0-4).

Table 4.10: Set of Tasks *I*

$\|I\|$	Task i	Length (α_i)	CoS (β_i)	(x, y) Coordinates (γ_i)	Release Time (δ_i)	Latency (v_i)
3	1	100	20	(0, 0)	0	5
	2	100	5	(0, 0)	0	20
	3	100	1	(0, 0)	0	100

* The maximum allowed latency v_i is not a task attribute. Instead v_i is a decision expression calculated by the model using Length α_i and CoS β_i , where $v_i = \alpha_i/\beta_i$.

Table 4.11: Set of Nodes *J*

$\|J\|$	Node j	Type (ε_j)	Resource Capacity (ζ_j)	(x, y) Coordinates (η_j)
3	1	0 (Cloud DC)	Unlimited	(90, 0)
	2	1 (Fog node)	100	(1, 0)
	3	1 (Fog node)	100	(1, 0)

The task set *I* includes 3 different tasks with the same task length, location and release time values but different CoS. The network consists of two fog nodes and the cloud DC. Both fog nodes have the exact resource capacity and location. The model calculates the propagation delay between the task and the candidate processing node using Eqn. 3.24. The model uses the propagation delay and the node resource capacity to determine which nodes can process the tasks based on the task requirements. Table **4.12** shows the tasks latency requirements and the calculated propagation delay for each task and node pair.

Table 4.12: Propagation Delay

		Node *j*		
Task *i*	**Latency (v_i)**	1 (Cloud DC)	2 (Fog node)	3 (Fog node)
1	5	90	1	1
2	20	90	1	1
3	100	90	1	1

Based on the previous information, below are few notes on the expected task assignment results:

- The cloud DC can only satisfy the latency requirements for task 3.
- Fog node 2 or fog node 3 can satisfy the latency requirements for all the three tasks combined within the model simulation period.
- In order for the model to achieve optimal load balancing, task 1 and task 2 must be assigned to a different fog node each while task 3 must be assigned to the cloud DC.
- Node utilization ρ_j is calculated using Eqn. 3.25 where $\rho_j = \sum_{i=0}^{n} \sum_{t=0}^{k} w_{ijt} / \zeta_j$.
- As the cloud DC has unlimited resource capacity, its node utilization is always equal to zero.
- The cloud DC resource consumption cost (RC) is always equal to zero.
- The load balancing factor Z_{lq} between node l and node q is determined by the model in relation with each node utilization value ρ_j using Eqns. 3.38 to 3.40.
- The revenue $R = \sum_{i=0}^{n} \sum_{j=0}^{m} \sum_{t=0}^{k} (\alpha_i * \beta_i * w_{ijt})$.
- The cost $C = \sum_{i=0}^{n} \sum_{j=0}^{m} \sum_{t=0}^{k} (y_{ijt} * RC) + \sum_{\substack{l \in J \\ l<q}} \sum_{q \in J} (DC * Z_{lq})$.
- $\max P = max\{R - C\}$.

The below manual calculations for the profit *P* objective function in Eqn. 3.28 show the three main permutations for the task assignment for this example:

1. All three tasks are assigned to one fog node (e.g., fog node 2):
   a. $R = \sum_{i=0}^{n} \sum_{j=0}^{m} \sum_{t=0}^{k} (\alpha_i * \beta_i * w_{ijt}) = (100 * 20 * 100) + (100 * 5 * 100) + (100 * 1 * 100)$
   b. For each task assignment, the term $(y_{ijt} * RC) = 1 * 1 = 1$.

c. For each pair of nodes the term $Z_{lq} = \left|\rho_l - \rho_q\right| = |\left(\sum_{i=0}^{n}\sum_{t=0}^{k} w_{ilt}/\zeta_l\right) - (\sum_{i=0}^{n}\sum_{t=0}^{k} w_{iqt}/\zeta_q)|$.

d. If all tasks are assigned to fog node 2, then $\rho_2 = \frac{100+100+100}{100} = 3$, while ρ_1 and $\rho_3 = 0$.

e. Then, $Z_{12} = |0-3|$, $Z_{13} = |0-0|$ and $Z_{23} = |3-0|$.

f. $C = \sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{t=0}^{k}(y_{ijt} * RC) + \sum_{\substack{l\in J \\ l<q}}\sum_{q\in J}\left(DC * Z_{lq}\right) = (1+1+1) + |0\text{-}3| + |0\text{-}0| + |3\text{-}0|$

g. $P = (100*20*100) + (100*5*100) + (100*1*100) - (1+1+1) - (3+0+3) = 259{,}991$

2. Tasks 1 and 2 are assigned to one fog node (e.g., fog node 2) and task 3 is assigned to the cloud DC:

a. $R = \sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{t=0}^{k}(\alpha_i * \beta_i * w_{ijt}) = (100*20*100) + (100*5*100) + (100*1*100)$

b. $C = \sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{t=0}^{k}(y_{ijt} * RC) + \sum_{\substack{l\in J \\ l<q}}\sum_{q\in J}\left(DC * Z_{lq}\right) = (1+1+1) + |0\text{-}2| + |0\text{-}0| + |2\text{-}0|$

c. $P = (100*20*100) + (100*5*100) + (100*1*100) - (1+1+1) - (2+0+2) = 259{,}993$

3. Task 1 is assigned to fog node 2, task 2 to fog node 3 and task 3 to cloud DC (optimal load balance):

a. $R = \sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{t=0}^{k}(\alpha_i * \beta_i * w_{ijt}) = (100*20*100) + (100*5*100) + (100*1*100)$

b. $C = \sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{t=0}^{k}(y_{ijt} * RC) + \sum_{\substack{l\in J \\ l<q}}\sum_{q\in J}\left(DC * Z_{lq}\right) = (1+1+1) + |0\text{-}1| + |0\text{-}1| + |1\text{-}1|$

c. $P = (100*20*100) + (100*5*100) + (100*1*100) - (0+1+1) - (1+1+0) = 259{,}996$ (max profit)

The manual calculations are aligned with the expected results where the optimal maximum profit is only achieved when the tasks are assigned in a load balancing fashion to the nodes that can satisfies the latency requirements.

After solving the model with CPLEX, the results shown in Table **4.13** to Table **4.15** were obtained. The optimal solution returned a profit of 259,996 units which matches the manual calculations and was solved in 0.01 sec.

Table 4.13: Load Balancing Factor (Z_{lq})

	Node *q*		
Node *l*	1 (Cloud DC)	2 (Fog node)	3 (Fog node)
1 (Cloud DC)	0	1	1
2 (Fog node)	0	0	0
3 (Fog node)	0	0	0

Table 4.14: Task Assignment (x_{ij})

	Node *j*		
Task *i*	1 (Cloud DC)	2 (Fog node)	3 (Fog node)
1	0	1	0
2	0	0	1
3	1	0	0

Table 4.15: Resource Allocation

Task *i*	**Node *j***	**Time Unit *t***	**Resource Allocation (y_{ijt})**	**Workload (w_{ijt})**
1	2	4	1	100
2	3	3	1	100
3	1	3	1	100

* Only non-zero values are included

The results shows that task 1 has been assigned to fog node 2, task 2 has been assigned to fog node 3 and task 3 has been assigned to cloud DC. Task 1 has been allocated 100 resources at t_4 while task 2 and task 3 each has been allocated 100 resources at t_3. The task assignment satisfies the load balancing requirements where each task is assigned to a single node that satisfies the latency requirements.

4.3 Mathematical Models Evaluation

The test plan used to evaluate the mathematical models is designed to include various size simulation networks, task sets with randomly generated values for task and node attributes. The same exact test plan in terms of data input set sizes and values was used to evaluate both TASLA-GC and TASLA-LB. Both models were initially executed in CPLEX without setting any execution time limits. The analysis of the executed test results and CPLEX logs has shown that most of the executed tests have been completed successfully in less than 36000 sec with few exceptions. Further analysis of the tests with an actual execution time that exceed 36000 sec, have shown that most of the iterations through the optimization tree branches performed after 36000 sec have returned identical values to the final optimal results determined by the solver. For such reason, it has been decided to set an execution time limit of 36000 sec in CPLEX (hereinafter referred to as full model). We believe that the full model with an execution time limit of 36000 sec is a good trade-off to search for optimal solutions with a high degree of confidence. The full model is used to find optimal solutions since CPLEX has enough time (in most cases) to iterate through all possible solutions according to the problem size.

Furthermore, in order to determine the model's performance within a relatively shorter execution time, both models were executed again in CPLEX but with an execution time limit set to 180 sec (hereinafter referred to as limited model). By setting the time limit to 180 sec in the limited model, the analysis and comparison to the full model results has shown that most of the CPLEX processing time is used to validate that the solution is optimal by comparing the results to all other possible solutions. If the time limit has been reached before completing this process, CPLEX will return the best solution found by that point of time. This limited model could also be seen as a heuristic algorithm with a trade-off between the solution quality and the execution time.

Each test consists of input data with specific set size in terms of the number of tasks ($|I|$) and nodes ($|J|$) of the simulated network. For each problem size, three different test instances were generated with the task and node parameters were randomly generated within a specified range. Each test instance was loaded and executed using both the full and limited model in CPLEX for the required simulation period which is the set of time units T. The test results in terms of profit and execution time are captured and sorted according to the minimum (min), median (med) and maximum (max) values of each test.

The test results presented in the next sub-sections include the optimal profit value determined by the full model and the relative profit determined by the limited model if applicable. The relative profit is represented as a ratio to the optimal profit value in the full model where a relative profit of 1 indicates that the limited model was able to determine the same exact optimal solution as the full model. A relative profit of 0 indicates that no solution has been found within the execution time (180 sec) of the limited model. The details of the test plan are summarized in Table 4.16 and the range of possible values for tasks i and nodes j are shown in Table 4.17 and Table 4.18 respectively.

Table 4.16: Parameters of the Test Plan

Number of tasks ($\|I\|$)	{10, 20, 30, 40, ..., 200}
Number of nodes ($\|J\|$)	{5, 10, 15, 20, 25}
Number of instances	3
Time limit of full model	36000 seconds
Time limit of limited model	180 seconds

Table 4.17: Range of parameters for Task i

	Range
Length (α_i)	{100, 150, 200, 250, 300}
CoS (β_i)	{1, 5, 20}
(x, y) Coordinates (γ_i)	({0, 1, 2, 3, 4, …, 99}, 0)
Release Time (δ_i)	{0, 1, 2, 3, 4}

Table 4.18: Range of parameters for nodes j

	Range
Type (ε_1)	{0} for Cloud DC
Type ($\varepsilon_{2\text{-}m}$)	{1} for Fog Node
Resource Capacity (ζ_1)	{Unlimited}
Resource Capacity ($\zeta_{2\text{-}m}$)	{50, 100, 150, 200, 250}
(x, y) Coordinates (η_1)	(200, 0)
(x, y) Coordinates ($\eta_{2\text{-}m}$)	({0, 1, 2, 3, 4, …, 99}, 0)

As it can be noticed in Table 4.17 and Table 4.18, the *y* coordinate value for of all the tasks and nodes is always fixed to zero. As previously stated in Section 3.2.1, the propagation delay is calculated by the model using the cartesian distance between the task and the processing node. For such reason and for the sake of simplicity when validating the model calculations, it has been decided to fix the *y* coordinate values to zero and use random values for the *x* coordinate of all tasks and node locations.

In summary, each individual test was developed to represent a specific problem size. Three different instances for each problem size with randomly generated values according to the ranges in Table 4.17 and Table 4.18 were used. Each test instance was executed in CPLEX using both the full model and the limited model where the execution time limit for the full model is set to 36000 sec and the limited model is set to 180 sec.

4.3.1 TASLA-GC Evaluation

Based on the test results, this section will first evaluate the model computational complexity followed by the performance analysis.

4.3.1.1 Computational Complexity

As mentioned previously, the classical assignment problem has already been shown to be NP-hard [4]. In order to evaluate the model computational complexity and its impact on the execution time, we perform tests using ten different set of tasks $|I| \in \{10, 20, \ldots, 100\}$, five different set of nodes $|J| \in \{5, 10, 15, 20, 25\}$ and two different set of time units $|T| \in \{10, 20\}$. Three different test instances of each set size were generated using random values within the ranges described in Table 4.17, Table 4.18 and were executed using the full model in CPLEX. Figure 4.1 shows the median execution time for the computation complexity evaluation tests. For readability reasons, the detailed test results for the computational complexity evaluation are shown in Appendix A.1.

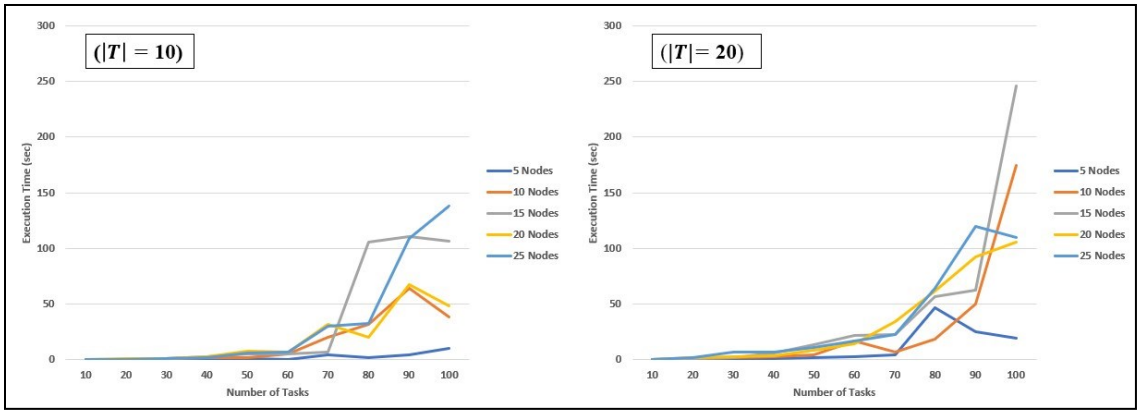

Figure 4.1: Median Execution Time for TASLA-GC Evaluation Tests

The results show that for a fixed number of nodes, the execution time increases exponentially with respect to the number of tasks. For a specific number of tasks, the execution time also increases when more fog nodes are available. Moreover, increasing the model simulation period has significantly increased the execution time as the optimization tree for the same number of tasks and nodes has increased. This is due to the fact that the model tries to optimize the node status over time to achieve the green computing requirement. These observations can be contributed to the fact that the more available options for task assignment, the more execution time will be required by the solver to determine the optimal task assignment. In addition, some variations in the trend of the execution time can be noticed especially for $|T|$=10. This can be a result of the algorithm used by CPLEX to search the optimization tree.

The analysis of the CPLEX logs shows that the optimal solution was determined approximately after 20-30% of the execution time on average. The remaining time was used to validate the solution by iterating through the remaining branches of the optimization tree. This allows for a high level of confidence when predicting the minimum execution time required to determine the optimal profit and task assignment.

To add strength to our analysis about complexity, we also looked at how the number of decision variables and constraints is affected by the various input sizes. Eqn. 4.1 and Eqn. 4.2 calculate the number of decision variables and the maximum number of constraints in TASLA-GC respectively.

$$Decision\ Variables = |J|(|I|(2|T| + 1) + 3|T|) \quad (4.1)$$

$$Constraints \leq 4|J||T| + |I|(2|J|(3|T| + 1) + 1) \quad (4.2)$$

Eqns. 4.1 and 4.2 show that the size of the input data sets of tasks I, nodes J and time units T increases the number of the decision variables and constraints in TASLA-GC. In addition, the number of constraints is significantly higher when compared to the number of the decision variables. As it can be noticed, the size of the time units T is the major factor that contributes to the computational complexity of the model.

For example, the results show that for ($|I| = 50$, $|J| = 25$ and $|T| = 10$), the median execution time is 5.97 sec, the number of decision variables is 27000 and the number of constraints is 62773. For the same network size over a different simulation period ($|I| = 50$, $|J| = 25$ and $|T| = 20$), the median execution time 11.09 sec, the number of decision variables is 52750 and the number of constraints is 119971. These results are aligned with the computational complexity calculations which indicate that the increase in the size of the set of time units $|T|$ will significantly increase the model computational complexity and accordingly the execution time.

4.3.1.2 Performance Analysis

To evaluate and analyse the model performance, we used ten different set of tasks $|I| \in \{110, 120, \ldots, 200\}$, five different set of nodes $|J| \in \{5, 10, 15, 20, 25\}$ and two different set of time units $|T| \in \{10, 20\}$. Three different instances of each set size were generated using random values within the ranges described in Table 4.17 and Table 4.18. All test instances were executed using both the full and the limited model in CPLEX. Figure 4.2 shows the median profit values for the performance analysis tests and all the detailed results are shown in Appendix A.2.

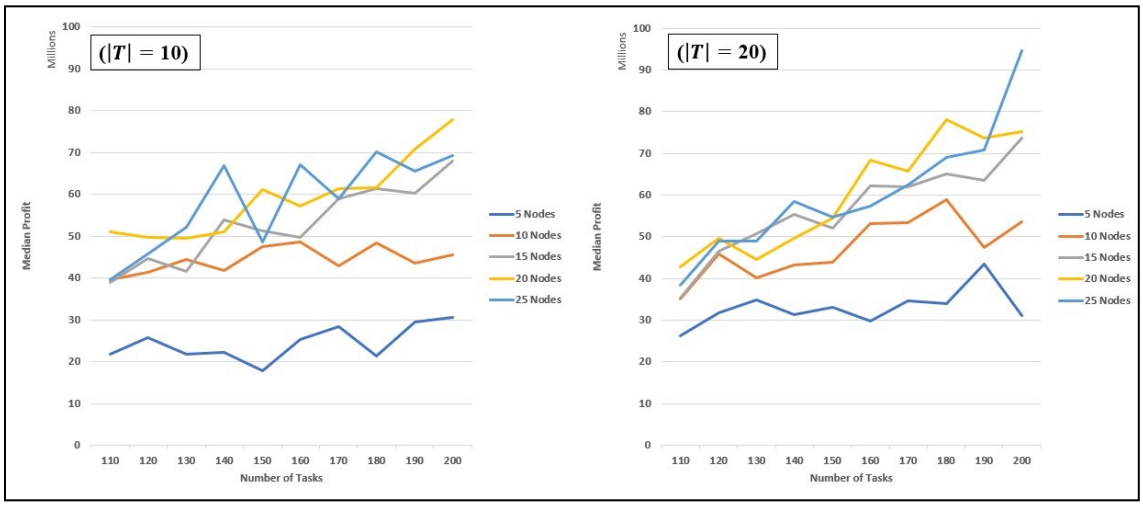

Figure 4.2: Median Profit for TASLA-GC Performance Analysis Tests

The results show that different factors contribute to the profit. For the same number of tasks, the increase in the number of nodes improves the model's ability to address the SLA for more tasks and increase the profit accordingly. In addition, for the same number of nodes, increasing the model simulation period has slightly increased the profit for larger number of tasks as the model was able to assign more tasks and satisfy the SLA requirements within the longer simulation period.

Figure 4.3 shows a break down for the execution time of the full model test results. As can be seen, most test instances (202/300) were solved within 3600 seconds and only a limited number (98/300) took more than 3600 and less than or equal 36000 (time limit for full mode).

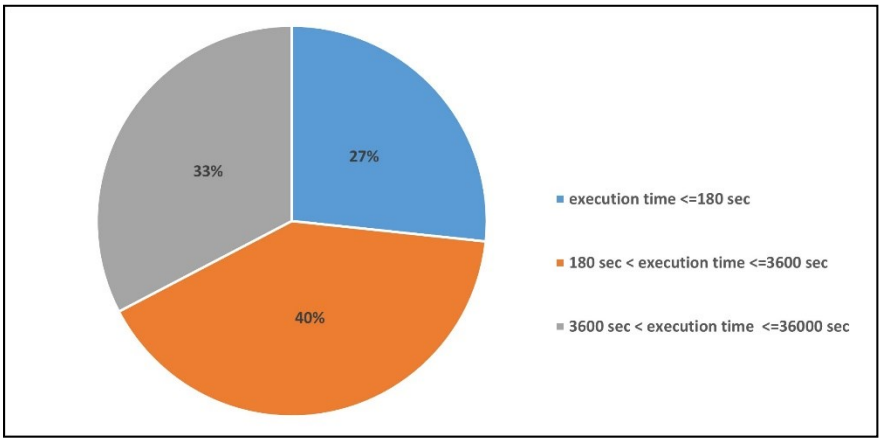

Figure 4.3: Breakdown for TASLA-GC Execution Time using Full Model

The detailed results, presented in Appendix A.2, show that for 96.3% of the tests, the limited model with an execution time limit of 180 sec provided an identical result in terms of relative profit when compared to the optimal profit returned by the full model. Further analysis shows that the relative profit is 100% of the optimal profit in networks where the number of tasks and nodes are smaller than or equal to $|I| = 140$ and $|J| = 25$. For larger networks where number of tasks and nodes are greater than or equal to $|I| = 160$ and $|J| = 25$, the limited model was able to determine a relative profit equal to 96.7% of the optimal profit on average. Finally, the results show that the size of the time units set $|T|$ doesn't influence the performance of the limited model or the relative profit as a ratio to the optimal profit determined by the full model. These results provide a high level of confidence when using the limited model to determine the profit.

Figure 4.4 shows the median relative profit as determined by the limited model for the set of tasks $|I| \in \{110, 120, \ldots, 200\}$ over the set of nodes $|J| = 25$ and the set of time units $|T| = 20$.

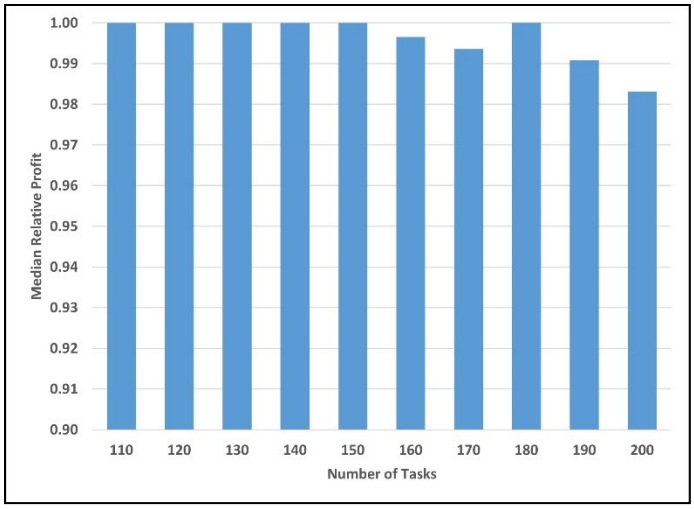

Figure 4.4: Median Relative Profit for TASLA-GC using set of nodes $|J|$= 25 and $|T| = 10$.

As mentioned earlier, the relative profit determined by the limited model (execution time limit of 180 sec) is presented as a ratio of the optimal profit value determined by the full model (execution time limit of 36000 sec). A relative profit of 1 indicates that the limited model was able to determine the same exact optimal solution as the full model while a relative profit of 0 indicates that no solution has been found by the limited model. The results in Figure 4.4 show that the limited model was able to provide identical results to the full model for $|I| \leq 150$, $|J|$=25 and $|T|$=10. In addition, the relative profit has slightly decreased with the increase in the number of tasks for the same number of nodes and simulation period. This indicates that the limited model was still able to provide results with more than 98% of the value of the results provided by the full model for $|I| \geq 160$, $|J|$=25 and $|T|$=10.

4.3.1.3 TASLA-GC Evaluation Summary

This section presented the computational complexity and the performance analysis of TASLA-GC based on the test results of both the full model (36000 sec) and the limited model (180 sec).

While the size of the input data sets of tasks I, nodes J and time units T determines the number of the decision variables and constraints, it has been shown that the increase in size of the time units T plays a major role in the increase of the model computational complexity and execution time. The number of nodes J and time units T are the main factor that allow the model to address the SLA requirements and maximize the profit.

The performance of the limited model has shown to provide reliable results in terms of the relative profit when compared to the optimal profit determined by the full model. This can be contributed to the fact that the full model can find an optimal result after 20-30% of the actual execution time which allows for more confidence when estimating the minimum required time to find optimal solution whether using the full or the limited model.

4.3.2 TASLA-LB Evaluation

This section describes the model computational complexity and the performance analysis of the second model based on load balancing.

4.3.2.1 Computational Complexity

To evaluate the model's computational complexity and its impact on the execution time, we perform tests using ten different set of tasks $|I| \in \{10, 20, \ldots, 100\}$, five different set of nodes $|J| \in \{5, 10, 15, 20, 25\}$ and two different set of time units $|T| \in \{10, 20\}$. Three different test instances of each set size were generated using random values within the ranges described in Table 4.17 and Table 4.18 and were executed using the full model in CPLEX. Figure 4.5 shows the median execution time for the computation complexity evaluation tests. For readability reasons, the detailed test results for the computational complexity evaluation are shown in Appendix B.1.

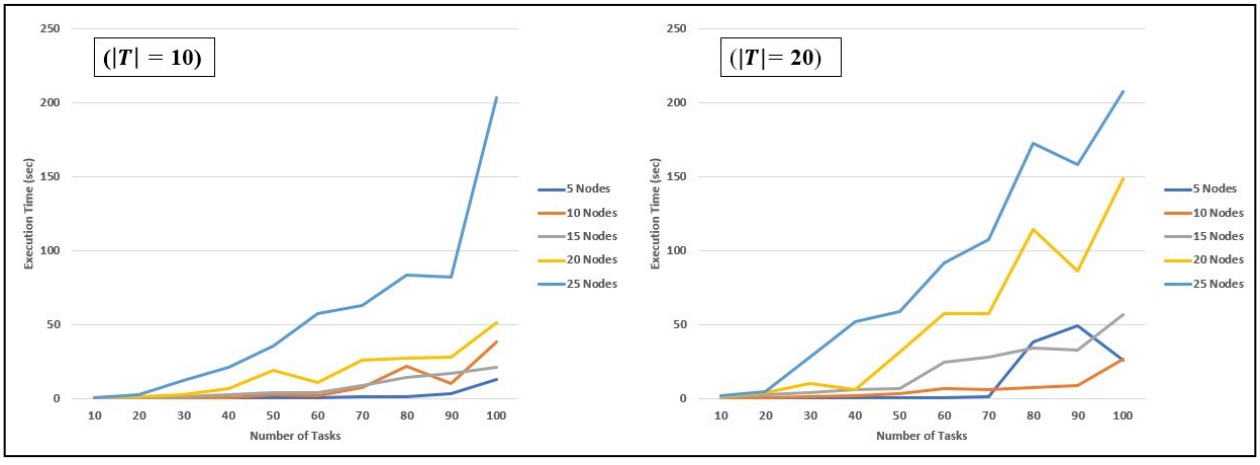

Figure 4.5: Median Execution Time for TASLA-LB Evaluation Tests

The results show that for a fixed number of nodes, the execution time increases exponentially with respect to the number of tasks. For a specific number of tasks, the execution time also increases when more fog nodes are available. In addition, increasing the model simulation period has

significantly increased the execution time as the optimization tree for the same number of tasks and nodes has increased. Finally, few variations in the trend of the execution time can be noticed especially for $|T|$=20. This can be a result of the algorithm used by CPLEX to search the optimization tree. In summary, the increase in the problem size will result in more task assignment options which leads to an increase in the execution time required by the solver to iterate all the optimization tree branches to determine the optimal result.

The analysis of the CPLEX logs shows that the optimal solution was determined approximately after 80-90% of the execution time on average. The remaining time was used to validate the solution by iterating through the remaining branches of the optimization tree. This allows for a high level of confidence when predicting the minimum execution time required to determine the optimal profit and task assignment.

To add strength to our analysis about complexity, we also looked at how the number of decision variables and constraints is affected by the various input sizes. Eqn. 4.3 and Eqn. 4.4 calculate the number of decision variables and the maximum number of constraints in TASLA-LB respectively.

$$Decision\ Variables = |J|\left(|I|(2|T|+1) + \frac{|J|-1}{2}\right) \tag{4.3}$$

$$Constraints \leq |J|\left(5|I||T| + 2|I| + |T| + \frac{3(|J|-1)}{2}\right) + |I| \tag{4.4}$$

Eqns. 4.3 and 4.4 shows that the size of the input data sets of tasks I, nodes J and time units T increases the number of the decision variables and constraints in TASLA-LB. In addition, the number of constraints is significantly higher when compared to the number of the decision variables. Finally, the size of the set of nodes J is the major factor that contributes to the computational complexity of the model due to the required calculations for the load balancing which is determined based on the node utilization.

For example, the results show that for ($|I|$ = 50, $|J|$ = 20 and $|T|$ = 10), the median execution time is 19.08 sec, the number of decision variables is 21190 and the number of constraints is 40389. When increasing the number of nodes for the same number of tasks over the same simulation period ($|I|$ = 50, $|J|$ = 25 and $|T|$ = 10), the median execution time is 35.91 sec, the number of decision variables is 26550 and the number of constraints is 50457. These results show the impact of increasing the number of nodes which increases the model complexity and the execution time.

The results for similar network size over different simulation period ($|I|$ = 50, $|J|$ = 25 and $|T|$ = 20), shows that the median execution time is 58.89 sec, the number of decision variables is 51550 and the number of constraints is 94804. These results are aligned with the computational complexity calculations which indicates that both the size of the set of nodes $|J|$ and the size of the set of time units $|T|$ contributes significantly to the model computational complexity and the execution time accordingly.

4.3.2.2 Performance Analysis

In order to evaluate and analyse the model performance, we used ten different set of tasks $|I| \in \{110, 120, \ldots, 200\}$, five different set of nodes $|J| \in \{5, 10, 15, 20, 25\}$ and two different set of time units $|T| \in \{10, 20\}$. Three different test instances of each set size were generated using random values within the ranges described in Table 4.17 and Table 4.18. All test instances were executed using both the full and the limited model in CPLEX. Figure 4.6 shows the median profit values for the performance analysis tests and all the detailed results are shown in Appendix B.2.

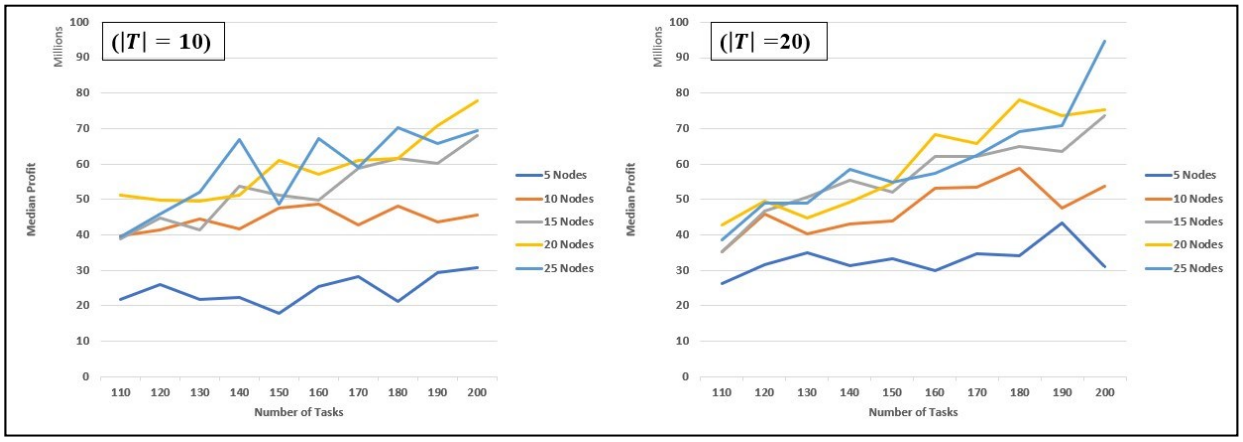

Figure 4.6: Median Profit for TASLA-LB Performance Analysis Tests

The results show that increasing the size of the set of time units $|T|$ has slightly increased the profit for larger networks as the model was able to assign more tasks and satisfy the SLA requirements within the longer simulation period. For the same number of tasks, the increase in the number of nodes improves the model's ability to address the SLA for more tasks and increases the profit accordingly. In addition, for the same number of nodes, increasing the model simulation period has slightly increased the profit for larger number of tasks as the model was able to assign more tasks and satisfy the SLA requirements within the longer simulation period.

Figure 4.7 shows a break down of the execution time of the full model test results. As can be seen, most test instances (222/300) were solved within 3600 seconds and only a limited number (78/300) took more than 3600 seconds and less than or equal 36000 (time limit for full mode).

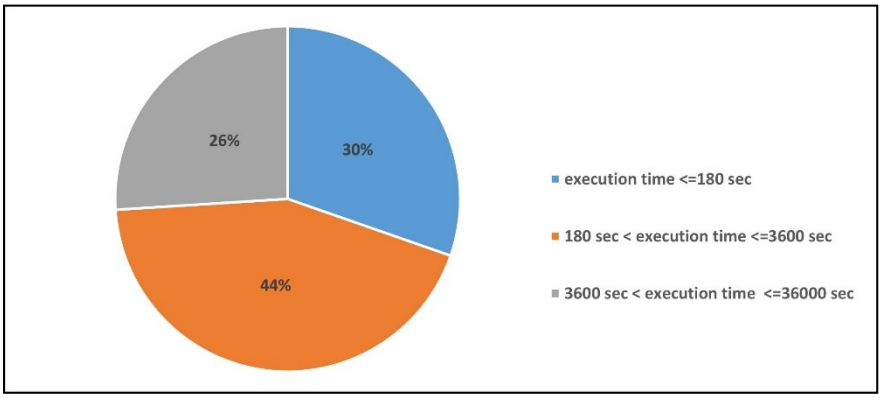

Figure 4.7: Breakdown for TASLA-LB Execution Time using Full Model

The detailed results presented in Appendix B.2 show that for 58% of the tests, the limited model with an execution time limit of 180 sec provided identical results in terms of relative profit when compared to the optimal profit results of the full model. In addition, the limited model was not able to determine any solution at 5% of the time where it returned a relative profit of 0. Further analysis shows that the relative profit is 100% of the optimal profit in networks where the number of tasks and nodes are smaller than or equal to $|I| = 100$ and $|J| = 15$. For larger networks where the number of tasks and nodes are greater than or equal to $|I| = 120$ and $|J| = 15$, the limited model was able to determine a relative profit equal to 75.5% of the optimal profit on average. Finally, the results show that the size of the time units set $|T|$ doesn't influence the performance of the limited model or the relative profit as a ratio to the optimal profit determined by the full model. These results provide a guideline for using the limited model to determine the profit.

Figure 4.8 shows the median relative profit as determined by the limited model for the set of tasks $|I| \in \{110, 120, \ldots, 200\}$ over the set of nodes $|J| = 25$ and the set of time units $|T| = 10$.

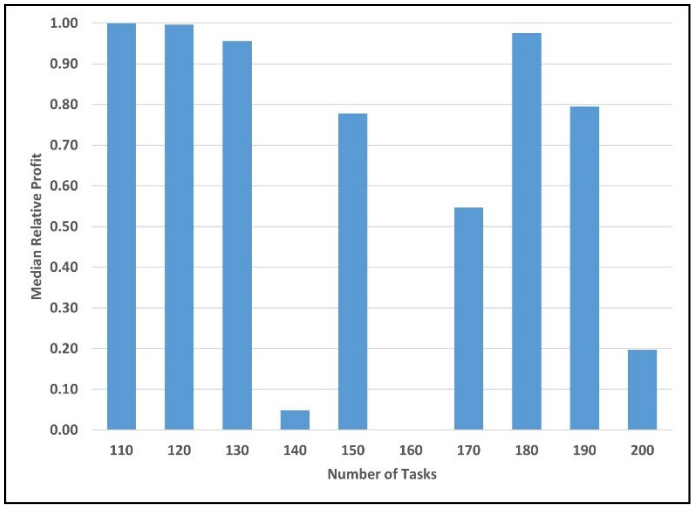

Figure 4.8: Median Relative Profit for TASLA-LB using set of nodes $|J|$= 25 and $|T| = 10$.

As mentioned earlier, the relative profit determined by the limited model (execution time limit of 180 sec) is presented as a ratio of the optimal profit value determined by the full model (execution time limit of 36000 sec). A relative profit value of 1 indicates that the limited model was able to determine the same exact optimal solution as the full model while a relative profit value of 0 indicates that no solution has been found by the limited model. The results in Figure 4.8 show that the limited model was able to provide identical results to the full model for $|I| \leq 120$, $|J|$=25 and $|T|$=10. In addition, relative profit has significantly decreased with the increase in the number of tasks for the same number of nodes and simulation period. For example, for $|I|$ =160, the ratio is equal to zero, which indicates that the limited model was not able to determine any solution for this problem size. Moreover, Figure 4.8 show that for $|I| \geq 130$, there are variations in the relative profit values across the test sample. This can be contributed to the fact that these results represent the median result of only 3 test instances of each problem size with randomly generated task parameters. Increasing the number of the test instances may provide more stable results but it comes with the price of increasing the time required to test all instances for the entire test plan using both the full model (36000 sec) and limited model (180 sec) for each test instance.

4.3.2.3 TASLA-LB Evaluation Summary

This section presented the computational complexity and performance analysis of TASLA-LB based on the test results of both the full model (36000 sec) and the limited model (180 sec).

While the size of the input data sets of tasks I, nodes J and time units T determines the number of the decision variables and constraints, it has been shown that the increase in size of the set of nodes J plays a major role in the increase of the model computational complexity and execution time. In addition, the increase in the number of nodes and time units allows the model to address the SLA requirements for more tasks and maximize the profit.

The performance of the limited model has shown to provide less reliable results in terms of the relative profit when compared to the optimal profit determined by the full model. This can be contributed to the fact that the full model can take up to 80-90% of the total execution time to find an optimal solution. For such reason, with the increase in the size of the simulation network, there can be a significant gap between the relative and optimal profit. However, the results can be used to provide a guideline to determine the minimum required execution time to determine reliable results based on the network size.

Chapter 5: Conclusion and Future Work

5.1 Summary

Fog computing is an edge computing paradigm that has been proposed to bridge the gap between the centralized architecture of cloud computing and the new requirements introduced by IoT and 5G applications such as ultra-low latency, location-awareness, mobility, etc. Fog computing extends the cloud computing capabilities to the network edge. Task assignment is an essential factor that allows fog computing to address the requirements that cloud computing cannot satisfy. The decentralized nature and heterogeneity of fog networks are adding to the overall complexity of the task assignment problem. Therefore, task assignment models in fog networks are required to address multiple requirements such as SLA, green computing, load balancing, profitability, etc.

In this thesis, we have developed two mathematical models for task assignment with SLA in fog networks. The main objective of both models is to maximize the fog service provider's profit while satisfying the offloaded task(s) SLA requirements. The SLA combines the task class of service and end-to-end latency requirements. The profit objective function of both models calculates the revenue realized by satisfying the task(s) SLA requirements and the associated costs. Both models try to determine the optimal task assignment that satisfies the SLA requirements according to the task(s) CoS, service volume, latency and node resource capacity in order to maximize the revenue. In addition to the profit and SLA requirements, each model addresses additional requirements to satisfy fog service provider's needs.

The first model addresses the green computing requirements by implementing an on-demand turn on/off strategy for fog nodes. The optimization of the profit objective function tries to minimize the fog node(s) startup, operational and shutdown costs to satisfy the green computing requirements. The second model addresses the load balancing requirements based on the node resource utilization as a ratio of the allocated workload to node resource capacity. The model tries to minimize the difference in the node utilization between nodes across the fog network. Both models determine the optimal task assignment that satisfies the model's objective function and constraints based on a complete pre-defined set of tasks, nodes and time units.

Both models were validated and evaluated using various size data input sets. In general, it has been observed that increasing the problem size results in more available options for task assignment

which leads to an increase in the execution time required by the solver to iterate all the branches of the optimization tree and determine the optimal task assignment. More precisely, the green computing model results have shown that the model computational complexity increases exponentially with the increase in number of tasks and nodes. However, the increase in the model simulation period has a significant impact on increasing the model execution time. This can be contributed to the fact that the model optimizes the node status over time to minimize the costs and achieve green computing requirements. Further analysis has shown that the optimal solution was determined by the solver approximately after 30% of the total execution time while the remaining time was used by the solver to validate the solution. For such reason, executing the model with a limited execution time limit of 180 sec has provided identical results to the full model which uses an execution time limit of 36000 sec for smaller networks where the number of tasks and nodes are smaller than or equal to $|I| = 140$ and $|J| = 25$. For larger networks where number of tasks and nodes are greater than or equal to $|I| = 160$ and $|J| = 25$, the limited model was able to provide a relative profit equal to 96.7% of the optimal profit determined by the full model. These results provide a high level of confidence when predicting the minimum execution time required to determine the optimal solution.

The results of the load balancing model have shown that the increase in the number of nodes and time units is the main contributor to the model complexity and the increase of the execution time. This can be contributed to the fact that the model is trying to determine and minimize the difference of the node utilization across the network to achieve load balancing.

When compared to the green computing model, the load balancing model requires significantly higher percentage of the total execution time to determine the optimal solution before validation. The solver log analysis for the load balancing model has shown that the optimal solution was determined after 90% of the total execution time while the remaining time was used to validate the solution. For such reason, the limited model with 180 sec execution time limit was able to provide identical results to the full model only for smaller network where the number of tasks and nodes are smaller than or equal to $|I| = 100$ and $|J| = 15$. For larger networks with the number of tasks and nodes are greater than or equal to $|I| = 120$ and $|J| = 15$, the limited model was able to determine a relative profit equal to 75.5% of the optimal profit determined by the full model. These results can be used as guidelines by service providers to predict the execution time based on the volume of requests and the size of the fog network.

The two proposed models could be useful to provide optimal solutions in fog networks with complete knowledge of future task requests that will be offloaded to the network. The models can run periodically for a pre-determined cycle duration and determine the optimal profit, task assignment and node(s) state based on the information of the tasks to be processed within each cycle. Based on the problem size, execution time limits can be set to provide reliable results. In addition, fog service providers can use both models to plan their networks according to the forecasted end users' numbers, applications, traffic patterns to determine the capacity, location and number of nodes required to maximize the profit while satisfying green computing or load balancing. Furthermore, the models can be processed against historical data and predicted task requests patterns to provide a benchmark to determine the upper bound of the task assignment problem and evaluate the performance of various approximate task assignment algorithms for larger and real-time fog networks.

5.2 Limitations

The test results of the mathematical models proposed in this thesis have shown that the increase in the problem size will significantly increase the computational complexity and the execution time. Fog service providers can set an execution time limit to overcome this limitation. However, the deployment of the mathematical models in large-scale fog networks will not be practical. In addition, both mathematical models require knowledge of the complete set of tasks and nodes to be processed prior to the execution. This is a major limitation that restricts the use of the mathematical models in real-life networks with real-time and dynamic nature in terms of the incoming task requests and network status. However, as discussed above, the models can still be used to determine the upper bound values for task assignment and profit value. This allows the service providers to have a benchmark to evaluate real-time task assignment models.

Finally, the proposed models have a lot of assumptions related to input data sets, latency calculations, mobility requirements, reliability, etc. These assumptions can add a lot of restrictions on using the mathematical model in real-life networks.

5.3 Future Work

To overcome the previously listed limitations, the following sub-sections will recommend future work that can eliminate some of these restrictions and extend the model's capabilities to address real-life fog network requirements.

5.3.1 Input Data Sets Improvement

In order for the mathematical models to address more realistic requirements, some of the assumptions that have been listed in this thesis could be removed.

Firstly, the end-to-end latency calculation was based only on the propagation and processing delay. Additional types of delay can be included such as transmission, queuing and scheduling delay. This could be improved by introducing more practical measurements based on different access technologies and communication links.

Secondly, additional resource types can be introduced to both the task requirements and node capabilities. This can include storage and communications bandwidth requirements. Moreover, different tasks can have different resource type requirements. For example, some tasks can request processing resources while other tasks can request storage resources.

Finally, in this thesis, the revenue calculation was a combination of the class of service, service volume and service rate. The revenue and cost calculations can be adapted to reflect more realistic, flexible billing and business models.

5.3.2 Addressing New Requirements

This thesis didn't consider some of the main requirements for fog computing such as mobility and reliability. Reliability is a major challenge in distributed architectures. Reliability can be defined as the ability of the system to perform the required tasks within a certain amount of time while providing accurate results. Reliability has a huge impact on the SLA and can be directly impacted by failures that occur in different layers of the fog architecture. To achieve high reliability, fault tolerant techniques such as task distribution and/or migration can be implemented.

Mobility support is also a major requirement that allows fog computing to support different applications such as cellular devices and vehicular networks. Due to the dynamic nature of fog networks, mobility challenges can include load balancing, QoS degradation, service interruption,

energy and cost efficiency. To address these challenges, several issues need to be tackled like mobility prediction, traffic routing, network management, signaling and synchronization.

The models proposed in this thesis can be adapted to support mobility by changing the end user/task location over different time units within the model simulation period. Task handover across different fog nodes can then be enabled to support mobility and satisfy the SLA requirements for larger mobile tasks that are processed over longer duration.

5.3.3 Real-Time Optimization Model

As previously discussed, the exact mathematical models require knowledge of the complete set of task requests prior to the model execution in order to determine the optimal task assignment and nodes state changes. In addition, the mathematical models don't adapt to real-time changes in the network status such as nodes or communication links failures.

A real-time optimization model must assign task without a pre-existing knowledge of the number, frequency, location and detailed requirements of the tasks that will be offloaded to the fog network. In addition, the real-time model must be able to efficiently support large-scale networks.

One of the major aspects that a real-time optimization model must tackle is task assignment with uncertainty. For example, in the case of the green computing model, the fog node state change can be a major challenge for a real-time model. This is due to the fact that the model is required to predict the frequency, volume and requirements of future offloaded tasks. Different approaches can be used to overcome the uncertainty in task assignment and node state change such as stochastic scheduling, probabilistic distribution or prediction based on historical patterns. For example, machine learning can be used to analyze the historical data including network status changes and patterns of the task requests to provide an enhanced prediction mechanism.

List of References

[1] Cisco Annual Internet Report (2018–2023). Available online at: “https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html” [Accessed 24 February 2022].”.

[2] Yannuzzi, Marcelo, et al. "Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing." 2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD). IEEE, 2014.

[3] Sarkar, Subhadeep, Subarna Chatterjee, and Sudip Misra. "Assessment of the Suitability of Fog Computing in the Context of Internet of Things." IEEE Transactions on Cloud Computing 6.1 (2015): 46-59.

[4] Zhan, Z. H., Liu, X. F., Gong, Y. J., Zhang, J., Chung, H. S. H., & Li, Y. (2015). Cloud computing resource scheduling and a survey of its evolutionary approaches. ACM Computing Surveys (CSUR), 47(4), 1-33.

[5] Jiao, L., Friedman, R., Fu, X., Secci, S., Smoreda, Z., & Tschofenig, H. (2013, July). Cloud-based computation offloading for mobile devices: State of the art, challenges and opportunities. In 2013 Future Network & Mobile Summit (pp. 1-11). IEEE.

[6] Yangui, S., & Tata, S. (2015). The SPD approach to deploy service-based applications in the cloud. Concurrency and Computation: Practice and Experience, 27(15), 3943-3960.

[7] Pop, D., Iuhasz, G., Craciun, C., & Panica, S. (2016, July). Support services for applications execution in multi-clouds environments. In 2016 IEEE international conference on autonomic computing (ICAC) (pp. 343-348). IEEE.

[8] Di Martino, B. (2014). Applications portability and services interoperability among multiple clouds. IEEE Cloud Computing, 1(1), 74-77.

[9] Stojmenovic, I. (2014, November). Fog computing: A cloud to the ground support for smart things and machine-to-machine networks. In 2014 Australasian telecommunication networks and applications conference (ATNAC) (pp. 117-122). IEEE.

[10] Yangui, S., Ravindran, P., Bibani, O., Glitho, R. H., Hadj-Alouane, N. B., Morrow, M. J., & Polakos, P. A. (2016, June). A platform as-a-service for hybrid cloud/fog environments. In

2016 IEEE international symposium on local and metropolitan area networks (LANMAN) (pp. 1-7). IEEE.

[11] Zhang, B., Mor, N., Kolb, J., Chan, D. S., Lutz, K., Allman, E., ... & Kubiatowicz, J. (2015). The Cloud is Not Enough: Saving {IoT} from the Cloud. In 7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15).

[12] Zhu, X., Chan, D. S., Hu, H., Prabhu, M. S., Ganesan, E., & Bonomi, F. (2015). IMPROVING VIDEO PERFORMANCE WITH EDGE SERVERS IN THE FOG COMPUTING ARCHITECTURE. Intel Technology Journal, 19(1).

[13] Uchechukwu, A., Li, K., & Shen, Y. (2014). Energy consumption in cloud computing data centers. International Journal of Cloud Computing and Services Science (IJ-CLOSER), 3(3), 31-48.

[14] Shuja, J., Gani, A., Shamshirband, S., Ahmad, R. W., & Bilal, K. (2016). Sustainable cloud data centers: a survey of enabling techniques and technologies. Renewable and Sustainable Energy Reviews, 62, 195-214.

[15] Mouradian, C., Naboulsi, D., Yangui, S., Glitho, R. H., Morrow, M. J., & Polakos, P. A. (2017). A comprehensive survey on fog computing: State-of-the-art and research challenges. IEEE communications surveys & tutorials, 20(1), 416-464.

[16] Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The case for vm-based cloudlets in mobile computing. IEEE pervasive Computing, 8(4), 14-23.

[17] Lewis, G. A., Echeverría, S., Simanta, S., Bradshaw, B., & Root, J. (2014, May). Cloudlet-based cyber-foraging for mobile systems in resource-constrained edge environments. In Companion Proceedings of the 36th International Conference on Software Engineering (pp. 412-415).

[18] Ali, A. M. M., Ahmad, N. M., & Amin, A. H. M. (2014, December). Cloudlet-based cyber foraging framework for distributed video surveillance provisioning. In 2014 4th World Congress on Information and Communication Technologies (WICT 2014) (pp. 199-204). IEEE.

[19] Giust, F., Verin, G., Antevski, K., Chou, J., Fang, Y., Featherstone, W., ... & Zhou, Z. (2018). MEC deployments in 4G and evolution towards 5G. ETSI White paper, 24(2018), 1-24.

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

[21] Vaquero, L. M., & Rodero-Merino, L. (2014). Finding your way in the fog: Towards a comprehensive definition of fog computing. ACM SIGCOMM computer communication Review, 44(5), 27-32.

[22] ETSI, G. (2017). Mobile edge computing (MEC). Mobile Edge Management, (Part 1), 010-1.

[23] Bader, A., Ghazzai, H., Kadri, A., & Alouini, M. S. (2016). Front-end intelligence for large-scale application-oriented internet-of-things. IEEE Access, 4, 3257-3272.

[24] Aryafar, E., Keshavarz-Haddad, A., Wang, M., & Chiang, M. (2013, April). RAT selection games in HetNets. In 2013 Proceedings IEEE INFOCOM (pp. 998-1006). IEEE.

[25] ElSawy, H., Hossain, E., & Alouini, M. S. (2014). Analytical modeling of mode selection and power control for underlay D2D communication in cellular networks. IEEE Transactions on Communications, 62(11), 4147-4161.

[26] Feng, D., Lu, L., Yuan-Wu, Y., Li, G. Y., Feng, G., & Li, S. (2013). Device-to-device communications underlaying cellular networks. IEEE Transactions on communications, 61(8), 3541-3551.

[27] Wildemeersch, M., Quek, T. Q., Kountouris, M., Rabbachin, A., & Slump, C. H. (2014). Successive interference cancellation in heterogeneous networks. IEEE transactions on communications, 62(12), 4440-4453.

[28] Bastug, E., Bennis, M., Médard, M., & Debbah, M. (2017). Toward interconnected virtual reality: Opportunities, challenges, and enablers. IEEE Communications Magazine, 55(6), 110-117.

[29] Yi, S., Li, C., & Li, Q. (2015, June). A survey of fog computing: concepts, applications and issues. In Proceedings of the 2015 workshop on mobile big data (pp. 37-42).

[30] Yi, S., Hao, Z., Qin, Z., & Li, Q. (2015, November). Fog computing: Platform and applications. In 2015 Third IEEE workshop on hot topics in web systems and technologies (HotWeb) (pp. 73-78). IEEE.

[31] Hou, X., Li, Y., Chen, M., Wu, D., Jin, D., & Chen, S. (2016). Vehicular fog computing: A viewpoint of vehicles as the infrastructures. IEEE Transactions on Vehicular Technology, 65(6), 3860-3873.

[32] Hu, P., Ning, H., Qiu, T., Zhang, Y., & Luo, X. (2016). Fog computing based face identification and resolution scheme in internet of things. IEEE transactions on industrial informatics, 13(4), 1910-1920.

[33] Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., ... & Jue, J. P. (2019). All one needs to know about fog computing and related edge computing paradigms: A complete survey. Journal of Systems Architecture, 98, 289-330.

[34] Perera, C., Qin, Y., Estrella, J. C., Reiff-Marganiec, S., & Vasilakos, A. V. (2017). Fog computing for sustainable smart cities: A survey. ACM Computing Surveys (CSUR), 50(3), 1-43.

[35] Perera, C., Jayaraman, P. P., Zaslavsky, A., Georgakopoulos, D., & Christen, P. (2014, March). Sensor discovery and configuration framework for the internet of things paradigm. In 2014 IEEE World Forum on Internet of Things (WF-IoT) (pp. 94-99). IEEE.

[36] Perera, C., Ranjan, R., Wang, L., Khan, S. U., & Zomaya, A. Y. (2015). Big data privacy in the internet of things era. IT Professional, 17(3), 32-39.

[37] Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2014). Sensing as a service model for smart cities supported by internet of things. Transactions on emerging telecommunications technologies, 25(1), 81-93.

[38] Chen, N., Chen, Y., Song, S., Huang, C. T., & Ye, X. (2016, October). Smart urban surveillance using fog computing. In 2016 IEEE/ACM Symposium on Edge Computing (SEC) (pp. 95-96). IEEE.

[39] Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2013). Context aware computing for the internet of things: A survey. IEEE communications surveys & tutorials, 16(1), 414-454.

[40] Distefano, S., Merlino, G., & Puliafito, A. (2012, August). Sensing and actuation as a service: A new development for clouds. In 2012 IEEE 11th International Symposium on Network Computing and Applications (pp. 272-275). IEEE.

[41] Merlino, G., Bruneo, D., Distefano, S., Longo, F., & Puliafito, A. (2014, November). Stack4things: Integrating iot with openstack in a smart city context. In 2014 International Conference on Smart Computing Workshops (pp. 21-28). IEEE.

[42] Mukherjee, M., Shu, L., & Wang, D. (2018). Survey of fog computing: Fundamental, network applications, and research challenges. IEEE Communications Surveys & Tutorials, 20(3), 1826-1857.

[43] Al Faruque, M. A., & Vatanparvar, K. (2015). Energy management-as-a-service over fog computing platform. IEEE internet of things journal, 3(2), 161-169.

[44] Kai, K., Cong, W., & Tao, L. (2016). Fog computing for vehicular ad-hoc networks: paradigms, scenarios, and issues. the journal of China Universities of Posts and Telecommunications, 23(2), 56-96.

[45] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., & Steggles, P. (1999, September). Towards a better understanding of context and context-awareness. In International symposium on handheld and ubiquitous computing (pp. 304-307). Springer, Berlin, Heidelberg.

[46] Shojafar, M., Cordeschi, N., & Baccarelli, E. (2016). Energy-efficient adaptive resource management for real-time vehicular cloud services. IEEE Transactions on Cloud computing, 7(1), 196-209.

[47] Ni, J., Zhang, A., Lin, X., & Shen, X. S. (2017). Security, privacy, and fairness in fog-based vehicular crowdsensing. IEEE Communications Magazine, 55(6), 146-152.

[48] Gia, T. N., Jiang, M., Rahmani, A. M., Westerlund, T., Liljeberg, P., & Tenhunen, H. (2015, October). 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 (pp. 356-363). IEEE.

[49] Azimi, I., Anzanpour, A., Rahmani, A. M., Pahikkala, T., Levorato, M., Liljeberg, P., & Dutt, N. (2017). HiCH: Hierarchical fog-assisted computing architecture for healthcare IoT. ACM Transactions on Embedded Computing Systems (TECS), 16(5s), 1-20.

[50] Zao, J. K., Gan, T. T., You, C. K., Méndez, S. J. R., Chung, C. E., Te Wang, Y., ... & Jung, T. P. (2014, June). Augmented brain computer interaction based on fog computing and linked data. In 2014 International conference on intelligent environments (pp. 374-377). IEEE.

[51] Aazam, M., & Huh, E. N. (2015, March). E-HAMC: Leveraging Fog computing for emergency alert service. In 2015 ieee international conference on pervasive computing and communication workshops (percom workshops) (pp. 518-523). IEEE.

[52] Lin, Y., & Shen, H. (2016). CloudFog: Leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service. IEEE Transactions on Parallel and Distributed Systems, 28(2), 431-445.

[53] Zavlanos, M. M., Spesivtsev, L., & Pappas, G. J. (2008, December). A distributed auction algorithm for the assignment problem. In 2008 47th IEEE Conference on Decision and Control (pp. 1212-1217). IEEE.

[54] Toroslu, I. H., & Arslanoglu, Y. (2007). Genetic algorithm for the personnel assignment problem with multiple objectives. Information Sciences, 177(3), 787-803.

[55] Dhall, S. K., & Liu, C. L. (1978). On a real-time scheduling problem. Operations research, 26(1), 127-140.

[56] Alfaro, C. A., Perez, S. L., Valencia, C. E., & Vargas, M. C. (2018). The equivalence between two classic algorithms for the assignment problem. arXiv preprint arXiv:1810.03562.

[57] Elhady, G. F., & Tawfeek, M. A. (2015, December). A comparative study into swarm intelligence algorithms for dynamic tasks scheduling in cloud computing. In 2015 IEEE Seventh international conference on intelligent computing and information systems (ICICIS) (pp. 362-369). IEEE.

[58] Kliazovich, D., Pecero, J. E., Tchernykh, A., Bouvry, P., Khan, S. U., & Zomaya, A. Y. (2016). CA-DAG: Modeling communication-aware applications for scheduling in cloud computing. Journal of Grid Computing, 14(1), 23-39.

[59] Schwiegelshohn, U., & Tchernykh, A. (2012, May). Online scheduling for cloud computing and different service levels. In 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (pp. 1067-1074). IEEE.

[60] Wei, W., Fan, X., Song, H., Fan, X., & Yang, J. (2016). Imperfect information dynamic stackelberg game based resource allocation using hidden Markov for cloud computing. IEEE transactions on services computing, 11(1), 78-89.

[61] Agarwal, S., Yadav, S., & Yadav, A. K. (2016). An efficient architecture and algorithm for resource provisioning in fog computing. International Journal of Information Engineering and Electronic Business, 8(1), 48.

[62] Intharawijitr, K., Iida, K., & Koga, H. (2016, March). Analysis of fog model considering computing and communication latency in 5G cellular networks. In 2016 IEEE international conference on pervasive computing and communication workshops (PerCom workshops) (pp. 1-4). IEEE.

[63] Souza, V. B. C., Ramírez, W., Masip-Bruin, X., Marín-Tordera, E., Ren, G., & Tashakor, G. (2016, May). Handling service allocation in combined fog-cloud scenarios. In 2016 IEEE international conference on communications (ICC) (pp. 1-5). IEEE.

[64] Lai, P., He, Q., Cui, G., Xia, X., Abdelrazek, M., Chen, F., ... & Yang, Y. (2019, October). Edge user allocation with dynamic quality of service. In International Conference on Service-Oriented Computing (pp. 86-101). Springer, Cham.

[65] Gu, L., Zeng, D., Guo, S., Barnawi, A., & Xiang, Y. (2015). Cost efficient resource management in fog computing supported medical cyber-physical system. IEEE Transactions on Emerging Topics in Computing, 5(1), 108-119.

[66] Skarlat, O., Nardelli, M., Schulte, S., & Dustdar, S. (2017, May). Towards qos-aware fog service placement. In 2017 IEEE 1st international conference on Fog and Edge Computing (ICFEC) (pp. 89-96). IEEE.

[67] Mahmud, R., Srirama, S. N., Ramamohanarao, K., & Buyya, R. (2020). Profit-aware application placement for integrated fog–cloud computing environments. Journal of Parallel and Distributed Computing, 135, 177-190.

[68] Al Nuaimi, K., Mohamed, N., Al Nuaimi, M., & Al-Jaroodi, J. (2012, December). A survey of load balancing in cloud computing: Challenges and algorithms. In 2012 second symposium on network cloud computing and applications (pp. 137-142). IEEE.

[69] Chandak, A., & Ray, N. K. (2019, December). A review of load balancing in fog computing. In 2019 International Conference on Information Technology (ICIT) (pp. 460-465). IEEE.

[70] Zahid, M., Javaid, N., Ansar, K., Hassan, K., KaleemUllah Khan, M., & Waqas, M. (2018, October). Hill climbing load balancing algorithm on fog computing. In International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (pp. 238-251). Springer, Cham.

[71] Téllez, N., Jimeno, M., Salazar, A., & Nino-Ruiz, E. (2018). A tabu search method for load balancing in fog computing. Int. J. Artif. Intell, 16(2), 1-30.

[72] Manju, A. B., & Sumathy, S. (2019). Efficient load balancing algorithm for task preprocessing in fog computing environment. In Smart Intelligent Computing and Applications (pp. 291-298). Springer, Singapore.

[73] Fan, Q., & Ansari, N. (2018). Towards workload balancing in fog computing empowered IoT. IEEE Transactions on Network Science and Engineering, 7(1), 253-262.

[74] Yousefpour, A., Ishigaki, G., Gour, R., & Jue, J. P. (2018). On reducing IoT service delay via fog offloading. IEEE Internet of things Journal, 5(2), 998-1010.

[75] Ningning, S., Chao, G., Xingshuo, A., & Qiang, Z. (2016). Fog computing dynamic load balancing mechanism based on graph repartitioning. China Communications, 13(3), 156-164.

[76] Ahuja, S. P., & Muthiah, K. (2016). Survey of state-of-art in green cloud computing. International Journal of Green Computing (IJGC), 7(1), 25-36.

[77] Hu, L., Zhao, J., Xu, G., Ding, Y., & Chu, J. (2013). A Survey on Green Computing Based on Cloud Environment. International Journal Of Interactive Mobile Technologies.

[78] Deng, R., Lu, R., Lai, C., & Luan, T. H. (2015, June). Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing. In 2015 IEEE International Conference on Communications (ICC) (pp. 3909-3914). IEEE.

[79] Xiao, Y., & Krunz, M. (2017, May). QoE and power efficiency tradeoff for fog computing networks with fog node cooperation. In IEEE INFOCOM 2017-IEEE Conference on Computer Communications (pp. 1-9). IEEE.

[80] Zhu, X., Yang, L. T., Chen, H., Wang, J., Yin, S., & Liu, X. (2014). Real-time tasks oriented energy-aware scheduling in virtualized clouds. IEEE Transactions on Cloud Computing, 2(2), 168-180.

[81] Chiang, Y. J., Ouyang, Y. C., & Hsu, C. H. (2014). An efficient green control algorithm in cloud computing for cost optimization. IEEE Transactions on Cloud Computing, 3(2), 145-155.

[82] Daigneault, J., & St-Hilaire, M. (2021). Profit Maximization Model for the Task Assignment Problem in 2-Tier Fog/Cloud Network Environments. IEEE Networking Letters, 3(1), 19-22.

[83] Mangasarian, O. L. (2014). Absolute value equation solution via linear programming. Journal of Optimization Theory and Applications, 161(3), 870-876.

[84] Hernandez, A. S., Lucas, T. W., & Carlyle, M. (2012). Constructing nearly orthogonal Latin hypercubes for any nonsaturated run-variable combination. ACM Transactions on Modeling and Computer Simulation (TOMACS), 22(4), 1-17.

Appendices

Appendix A TASLA-GC Evaluation Test Results

This appendix presents the detailed test results used for the evaluation of TASLA-GC mathematical model.

A.1 TASLA-GC Computational Complexity Test Results

Table A-1 presents the complete computational complexity test results described in Section 4.3.1.1.

Table A-1: TASLA-GC Computational Complexity Test Results

$\|T\|$	$\|J\|$	$\|I\|$	**Execution Time (sec) Full Model (36000 sec)**		
			Max	**Med**	**Min**
10	5	10	0.16	0.11	0.11
		20	0.47	0.39	0.28
		30	1.23	0.47	0.34
		40	3.56	0.56	0.50
		50	1.58	1.11	0.73
		60	2.11	0.70	0.50
		70	7.38	4.55	2.55
		80	18.19	2.28	2.02
		90	9.61	4.58	1.38
		100	99.91	10.38	9.50
	10	10	0.20	0.19	0.17
		20	0.23	0.23	0.22
		30	0.97	0.94	0.52
		40	2.45	1.69	0.94
		50	31.83	1.80	0.14
		60	8.23	5.17	4.73
		70	56.20	20.47	3.75
		80	468.88	32.16	24.94
		90	344.78	63.92	8.19
		100	44.50	38.30	8.48
	15	10	0.27	0.25	0.19
		20	0.53	0.39	0.38

$\|T\|$	$\|J\|$	$\|I\|$	Execution Time (sec) Full Model (36000 sec)		
			Max	**Med**	**Min**
		30	1.20	1.03	0.63
		40	9.20	2.97	1.48
		50	6.39	4.92	4.03
		60	17.00	5.59	5.13
		70	8.23	7.03	4.14
		80	438.53	105.89	13.48
		90	195.30	110.31	70.77
		100	158.33	106.86	25.94
	20	10	0.23	0.23	0.14
		20	0.80	0.80	0.66
		30	1.31	1.25	0.67
		40	3.17	3.11	2.73
		50	11.80	8.00	6.73
		60	8.13	6.77	6.02
		70	62.69	31.61	13.59
		80	50.84	20.31	22.08
		90	69.70	67.55	35.94
		100	168.28	48.19	41.30
	25	10	0.84	0.42	0.28
		20	1.09	0.52	0.48
		30	1.64	1.30	1.17
		40	3.48	2.28	2.33
		50	10.72	5.97	4.09
		60	11.42	7.38	3.53
		70	34.16	30.11	29.73
		80	46.94	32.28	23.30
		90	124.20	108.78	49.95
		100	156.91	138.17	87.80
20	5	10	0.30	0.19	0.17
		20	0.55	0.38	0.24
		30	1.08	0.52	0.34
		40	2.81	1.31	0.76
		50	3.75	2.09	1.83
		60	3.95	3.22	1.20
		70	13.39	4.67	3.25

$\|T\|$	$\|J\|$	$\|I\|$	Execution Time (sec) Full Model (36000 sec)		
			Max	**Med**	**Min**
		80	68.27	46.66	44.77
		90	30.03	25.33	6.01
		100	43.95	19.19	13.28
	10	10	0.22	0.20	0.19
		20	1.06	0.75	0.42
		30	1.17	2.27	0.94
		40	5.30	2.59	1.94
		50	6.97	4.73	0.49
		60	25.36	17.06	13.55
		70	18.45	7.09	6.36
		80	66.63	18.58	20.19
		90	58.55	50.23	18.09
		100	212.84	174.61	39.11
	15	10	0.38	0.26	0.25
		20	1.27	1.03	0.74
		30	2.41	1.99	1.89
		40	20.11	5.91	4.02
		50	16.24	13.75	4.69
		60	82.45	21.69	16.36
		70	47.14	22.63	12.11
		80	63.36	56.94	35.36
		90	127.53	62.88	39.41
		100	417.97	245.80	53.11
	20	10	0.67	0.66	0.66
		20	2.03	1.91	1.78
		30	3.06	2.91	2.61
		40	5.16	3.80	3.63
		50	11.58	8.89	8.59
		60	19.51	14.41	10.63
		70	42.66	34.50	26.02
		80	312.81	61.47	52.83
		90	343.89	92.23	73.83
		100	294.09	105.56	38.08
	25	10	0.88	0.70	0.69
		20	4.22	1.88	0.92

|*T*|	|*J*|	|*I*|	Execution Time (sec) Full Model (36000 sec)		
			Max	Med	Min
		30	12.45	7.25	2.83
		40	8.39	6.56	5.44
		50	12.39	11.09	8.77
		60	20.23	17.09	12.25
		70	49.67	23.05	11.13
		80	116.38	63.92	46.70
		90	429.13	119.69	100.58
		100	657.39	109.95	106.19

A.2 TASLA-GC Performance Test Results

Table A-2 presents the complete performance analysis test results described in Section 4.3.1.2.

Table A-2: TASLA-GC Performance Analysis Test Results

|*T*|	|*J*|	|*I*|	Profit (10^6 Units) Full Model (36000 sec)			Execution Time (sec) Full Model (36000 sec)			Relative Profit Limited Model (180 sec)		
			Max	Med	Min	Max	Med	Min	Max	Med	Min
10	5	110	33.92	21.69	14.58	276.75	9.95	9.55	1.00	1.00	1.00
		120	31.04	25.86	20.71	62.05	26.25	18.66	1.00	1.00	1.00
		130	25.32	21.77	20.09	17.61	17.25	3.95	1.00	1.00	1.00
		140	27.10	22.24	21.11	28.02	17.55	3.84	1.00	1.00	1.00
		150	23.30	17.83	16.98	20.77	17.81	12.52	1.00	1.00	1.00
		160	34.66	25.40	18.48	8086.33	55.78	0.80	1.00	1.00	1.00
		170	32.69	28.36	20.70	28.69	23.14	2.01	1.00	1.00	1.00
		180	24.60	21.26	20.46	22.88	7.42	1.86	1.00	1.00	1.00
		190	31.58	29.46	17.29	2109.95	59.48	1.72	1.00	1.00	1.00
		200	32.32	30.63	24.32	40.52	35.38	24.42	1.00	1.00	1.00
	10	110	40.62	39.60	32.69	36000.00	281.73	123.55	1.00	1.00	1.00
		120	44.60	41.31	35.45	36000.00	1027.52	263.34	1.00	1.00	1.00
		130	46.47	44.54	35.04	36000.00	36000.00	2080.39	1.00	1.00	1.00
		140	41.93	41.81	39.99	36000.00	36000.00	445.25	1.00	1.00	1.00
		150	49.66	47.60	40.54	30907.09	25473.11	4321.75	1.00	1.00	1.00
		160	49.26	48.56	45.12	36000.00	36000.00	14784.59	1.00	1.00	1.00
		170	44.23	42.88	38.16	36000.00	36000.00	10522.58	1.00	1.00	1.00

|*T*|	|*J*|	|*I*|	Profit (10^6 Units) Full Model (36000 sec)			Execution Time (sec) Full Model (36000 sec)			Relative Profit Limited Model (180 sec)		
			Max	**Med**	**Min**	**Max**	**Med**	**Min**	**Max**	**Med**	**Min**
		180	48.54	48.36	43.68	30367.95	18679.30	4014.80	1.00	1.00	1.00
		190	48.28	43.66	37.91	36000.00	1271.22	986.30	1.00	1.00	1.00
		200	46.23	45.64	43.63	164.38	149.09	27.56	1.00	1.00	1.00
	15	110	39.69	39.00	30.61	36000.00	102.67	66.78	1.00	1.00	1.00
		120	52.49	44.65	41.17	2235.80	180.56	55.83	1.00	1.00	1.00
		130	41.90	41.50	39.62	36000.00	133.73	97.66	1.00	1.00	1.00
		140	59.45	53.81	51.04	32144.42	1836.52	39.06	1.00	1.00	1.00
		150	58.73	51.31	45.57	386.11	181.42	57.55	1.00	1.00	1.00
		160	53.51	49.76	41.21	36000.00	36000.00	9687.05	1.00	1.00	1.00
		170	64.10	58.90	57.48	36000.00	25816.05	11674.86	1.00	1.00	1.00
		180	66.97	61.44	60.83	36000.00	6298.38	1005.25	1.00	1.00	1.00
		190	69.97	60.35	56.84	36000.00	3899.77	20965.36	1.00	1.00	1.00
		200	72.31	67.96	65.51	23634.05	1449.28	98.16	1.00	1.00	1.00
	20	110	52.43	51.08	36.16	104.70	102.23	92.33	1.00	1.00	1.00
		120	54.85	49.77	46.44	127.64	92.17	90.78	1.00	1.00	1.00
		130	49.78	49.54	47.03	178.31	96.78	68.80	1.00	1.00	1.00
		140	56.17	51.14	46.16	233.63	103.41	72.89	1.00	1.00	1.00
		150	61.19	61.08	59.81	866.02	172.66	151.98	1.00	1.00	1.00
		160	59.56	57.10	42.90	2811.94	642.94	373.28	1.00	1.00	1.00
		170	61.06	61.43	58.65	36000.00	374.84	242.05	1.00	1.00	1.00
		180	66.04	61.59	52.01	10968.08	1320.19	404.48	1.00	1.00	1.00
		190	85.13	70.88	68.69	36000.00	36000.00	13090.81	1.00	1.00	1.00
		200	78.53	77.77	69.69	36000.00	11115.06	256.23	1.00	1.00	1.00
	25	110	41.37	39.52	35.29	176.56	154.61	45.08	1.00	1.00	1.00
		120	51.12	45.76	42.44	225.84	202.36	118.09	1.00	1.00	1.00
		130	56.57	52.17	52.02	259.80	178.31	136.08	1.00	1.00	1.00
		140	66.81	66.81	57.29	272.81	251.95	214.55	1.00	1.00	1.00
		150	65.83	48.66	45.17	558.61	507.06	366.17	1.00	1.00	1.00
		160	69.05	67.18	57.92	452.14	248.59	208.38	0.99	0.97	0.94
		170	63.18	58.95	56.68	347.75	306.64	212.36	1.00	1.00	0.91
		180	85.56	70.24	68.83	983.16	411.94	409.69	1.00	1.00	1.00
		190	81.53	65.67	57.69	555.80	397.48	176.75	1.00	1.00	0.96
		200	80.68	69.32	65.68	3839.28	755.84	312.19	1.00	0.96	0.91
20	5	110	28.87	26.23	24.38	2489.89	14.86	6.26	1.00	1.00	1.00

$\|T\|$	$\|J\|$	$\|I\|$	Profit (10^6 Units) Full Model (36000 sec)			Execution Time (sec) Full Model (36000 sec)			Relative Profit Limited Model (180 sec)		
			Max	Med	Min	Max	Med	Min	Max	Med	Min
		120	32.59	31.69	19.96	21.66	18.84	4.56	1.00	1.00	1.00
		130	36.49	34.86	16.42	1221.66	144.77	5.41	1.00	1.00	1.00
		140	32.62	31.25	18.69	36000.00	960.41	945.41	1.00	1.00	1.00
		150	40.53	33.21	19.37	36000.00	588.75	16.95	1.00	1.00	1.00
		160	36.97	29.88	24.01	36000.00	36000.00	40.19	1.00	1.00	1.00
		170	36.71	34.67	27.86	36000.00	23350.19	515.03	1.00	1.00	1.00
		180	40.82	34.06	27.12	36000.00	36000.00	1450.20	1.00	1.00	1.00
		190	44.90	43.49	25.78	26192.02	997.25	26.06	1.00	1.00	1.00
		200	38.76	31.11	25.22	28948.23	595.94	216.16	1.00	1.00	1.00
	10	110	41.71	35.15	25.32	526.73	147.73	50.03	1.00	1.00	1.00
		120	56.54	45.95	40.86	199.86	198.99	107.75	1.00	1.00	1.00
		130	43.18	40.24	25.24	576.11	347.91	144.30	1.00	1.00	1.00
		140	48.18	43.15	41.08	4261.67	149.08	111.27	1.00	1.00	1.00
		150	53.78	43.87	42.78	1255.06	733.75	58.95	1.00	1.00	1.00
		160	53.24	53.16	36.61	868.80	247.47	114.83	1.00	1.00	1.00
		170	54.15	53.48	45.23	36000.00	1470.17	560.51	1.00	1.00	1.00
		180	63.57	58.85	54.93	3156.95	550.75	247.39	1.00	1.00	1.00
		190	60.34	47.43	38.23	16113.38	491.16	185.58	1.00	1.00	1.00
		200	68.60	53.64	48.57	27262.50	12598.45	288.58	1.00	1.00	1.00
	15	110	47.04	35.24	31.85	441.33	308.59	285.28	1.00	1.00	1.00
		120	58.22	46.61	34.63	505.02	308.81	194.11	1.00	1.00	1.00
		130	54.98	50.76	45.51	294.67	225.95	175.94	1.00	1.00	1.00
		140	57.64	55.35	36.73	701.28	559.33	460.97	1.00	1.00	1.00
		150	55.08	52.02	47.17	455.14	423.42	301.22	1.00	1.00	1.00
		160	64.54	62.13	59.88	737.03	476.13	428.98	1.00	1.00	1.00
		170	78.92	62.01	55.12	801.34	474.83	442.39	1.00	1.00	1.00
		180	71.25	64.99	60.16	5904.52	598.69	485.66	1.00	1.00	1.00
		190	65.60	63.62	62.76	856.55	847.97	418.94	1.00	1.00	1.00
		200	73.96	73.67	71.47	773.13	668.24	550.30	1.00	1.00	1.00
	20	110	49.68	42.74	41.41	1146.33	742.31	417.88	1.00	1.00	1.00
		120	51.55	49.58	47.37	875.73	835.84	495.51	1.00	1.00	1.00
		130	45.72	44.64	43.79	1067.70	249.63	129.51	1.00	1.00	1.00
		140	55.26	49.57	49.28	1125.39	870.47	823.50	1.00	1.00	1.00
		150	65.16	54.48	52.11	1454.63	1244.25	1094.05	1.00	1.00	1.00

$\|T\|$	$\|J\|$	$\|I\|$	Profit (10^6 Units) Full Model (36000 sec)			Execution Time (sec) Full Model (36000 sec)			Relative Profit Limited Model (180 sec)		
			Max	Med	Min	Max	Med	Min	Max	Med	Min
		160	70.36	68.36	57.82	1281.51	1001.77	951.06	1.00	1.00	1.00
		170	67.01	65.82	56.78	1269.94	1244.25	963.19	1.00	1.00	1.00
		180	79.81	78.16	63.25	1927.09	1679.55	1503.33	1.00	1.00	1.00
		190	75.50	73.61	56.38	1836.69	1836.30	1341.30	1.00	1.00	1.00
		200	78.84	75.33	75.00	4834.08	4413.94	1072.56	1.00	1.00	1.00
	25	110	44.77	38.51	37.05	658.91	236.55	116.78	1.00	1.00	1.00
		120	50.50	49.06	48.02	521.55	249.24	217.50	1.00	1.00	1.00
		130	51.20	49.03	48.00	1746.09	1489.44	1112.53	1.00	1.00	1.00
		140	60.29	58.42	55.67	1446.02	1341.74	512.59	1.00	1.00	1.00
		150	61.77	54.78	45.56	1938.61	1757.09	760.22	1.00	1.00	1.00
		160	72.75	57.48	57.24	2185.89	1521.80	1059.34	1.00	1.00	0.87
		170	74.48	62.42	53.15	5113.36	3032.06	2540.09	1.00	0.99	0.99
		180	79.55	69.03	64.52	2246.78	1896.09	1888.34	1.00	00.98	0.96
		190	72.57	70.81	62.16	5894.69	4351.88	2651.72	0.99	0.99	0.98
		200	95.20	94.74	83.49	9443.83	5364.39	2509.95	1.00	0.99	00.98

Appendix B TASLA-LB Evaluation Complete Test Results

This appendix presents the detailed test results used for the evaluation of TASLA-LB mathematical model.

B.1 TASLA-LB Computational Complexity Test Results

Table B-1 presents the complete computational complexity test results described in Section 4.3.2.1.

Table B-1: TASLA-LB Computational Complexity Test Results

$\|T\|$	$\|J\|$	$\|I\|$	Execution Time (sec) Full Model (36000 sec)		
			Max	Med	Min
10	5	10	0.16	0.09	0.08
		20	0.27	0.25	0.09
		30	2.22	0.44	0.33
		40	3.98	0.61	0.52

$\|T\|$	$\|J\|$	$\|I\|$	Execution Time (sec) Full Model (36000 sec)		
			Max	**Med**	**Min**
		50	1.58	1.11	0.73
		60	1.77	1.25	1.16
		70	14.56	1.89	1.25
		80	126.34	1.61	0.98
		90	4.91	3.55	3.05
		100	16.81	12.94	10.86
	10	10	0.31	0.19	0.17
		20	0.44	0.44	0.34
		30	0.94	0.94	0.72
		40	1.38	1.25	1.13
		50	11.34	2.83	0.48
		60	2.89	2.63	1.08
		70	35.83	7.52	2.22
		80	391.83	22.20	7.00
		90	1874.19	10.28	2.44
		100	64.91	38.69	4.09
	15	10	0.73	0.59	0.50
		20	1.25	1.08	0.83
		30	2.14	1.51	1.33
		40	9.11	3.00	2.78
		50	9.86	4.09	3.53
		60	9.31	4.49	4.06
		70	9.14	9.14	5.14
		80	29.58	14.38	13.64
		90	18.64	17.41	14.66
		100	30.33	21.64	8.44
	20	10	0.94	0.75	0.59
		20	2.69	1.81	1.56
		30	3.33	2.69	2.23
		40	17.08	6.91	5.72
		50	20.06	19.08	15.55
		60	24.77	11.42	5.70
		70	29.36	26.23	22.16
		80	55.05	27.78	12.81
		90	32.17	28.03	23.19

$\|T\|$	$\|J\|$	$\|I\|$	Execution Time (sec) Full Model (36000 sec)		
			Max	**Med**	**Min**
		100	93.14	51.27	35.39
	25	10	2.34	1.25	1.14
		20	13.00	2.97	2.88
		30	14.11	12.34	4.42
		40	26.16	21.28	14.41
		50	59.03	35.91	28.77
		60	63.16	57.59	42.88
		70	128.78	62.95	37.36
		80	130.25	83.48	47.88
		90	85.14	82.20	49.19
		100	533.53	203.14	74.19
20	5	10	0.42	0.14	0.13
		20	0.44	0.42	0.22
		30	0.48	0.47	0.33
		40	0.89	0.80	0.30
		50	0.84	0.63	0.52
		60	17.50	0.98	0.86
		70	1.52	1.41	0.98
		80	727.41	38.30	0.94
		90	156.41	49.14	0.86
		100	41.56	26.38	4.53
	10	10	0.38	0.33	0.25
		20	1.14	0.63	0.55
		30	1.89	1.52	1.28
		40	3.11	2.56	2.45
		50	4.17	3.63	0.17
		60	7.44	7.25	6.17
		70	9.55	6.61	5.47
		80	11.81	7.67	4.09
		90	14.61	9.14	6.25
		100	47.86	27.24	21.83
	15	10	1.17	1.02	0.58
		20	21.16	2.69	1.66
		30	12.16	4.56	3.45
		40	7.50	6.14	6.06

$\|T\|$	$\|J\|$	$\|I\|$	Execution Time (sec) Full Model (36000 sec)		
			Max	**Med**	**Min**
		50	11.69	7.11	6.77
		60	34.17	25.20	12.88
		70	37.81	28.03	13.98
		80	36.47	34.28	32.36
		90	53.00	33.23	22.03
		100	129.86	57.27	49.48
	20	10	1.63	1.52	1.41
		20	6.14	4.34	4.20
		30	10.58	10.22	6.03
		40	10.27	6.44	5.09
		50	62.22	31.91	30.63
		60	76.27	57.38	50.64
		70	79.06	57.47	43.47
		80	136.83	114.11	71.08
		90	161.97	86.47	94.56
		100	164.77	148.45	63.75
	25	10	5.63	2.52	2.16
		20	49.02	5.30	3.38
		30	43.64	28.45	22.64
		40	78.63	52.09	46.88
		50	98.94	58.89	52.58
		60	180.28	92.11	91.25
		70	133.64	107.33	73.42
		80	181.86	172.39	120.19
		90	592.55	158.45	131.70
		100	418.69	207.14	166.64

B.2 TASLA-LB Performance Test Results

Table B-2 presents the complete computational complexity results described in Section 4.3.2.2.

Table B-2: TASLA-GC Computational Complexity Test Results

$\|T\|$	$\|J\|$	$\|I\|$	Profit (10^6 Units) Full Model (36000 sec)			Execution Time (sec) Full Model (36000 sec)			Relative Profit Limited Model (180 sec)		
			Max	Med	Min	Max	Med	Min	Max	Med	Min
10	5	110	33.92	21.69	14.58	167.58	34.48	30.81	1.00	1.00	1.00
		120	31.04	25.86	20.71	407.73	192.08	121.86	1.00	1.00	1.00
		130	25.32	21.77	20.09	30.17	28.70	10.28	1.00	1.00	1.00
		140	27.10	22.24	21.11	176.95	31.59	3.38	1.00	1.00	1.00
		150	23.30	17.83	16.98	123.53	31.05	19.00	1.00	1.00	1.00
		160	34.66	25.40	18.48	1878.38	382.11	2.83	1.00	1.00	1.00
		170	32.69	28.36	20.70	35.53	18.09	2.61	1.00	1.00	1.00
		180	24.60	21.26	20.46	38.91	20.48	1.91	1.00	1.00	1.00
		190	31.58	29.46	17.29	537.05	209.66	1.52	1.00	1.00	1.00
		200	32.32	30.63	24.32	66.41	51.22	42.72	1.00	1.00	1.00
	10	110	40.62	39.60	32.69	36000.00	46.58	31.14	1.00	1.00	1.00
		120	44.60	41.31	35.45	36000.00	644.63	118.97	1.00	1.00	1.00
		130	46.47	44.54	35.04	36000.00	36000.00	36000.00	1.00	1.00	1.00
		140	41.93	41.81	39.99	36000.00	935.53	140.58	1.00	1.00	1.00
		150	49.66	47.60	40.54	36000.00	36000.00	33151.33	1.00	1.00	1.00
		160	49.17	48.56	45.12	36000.00	36000.00	36000.00	1.00	1.00	1.00
		170	44.23	42.88	38.16	36000.00	36000.00	13326.67	1.00	1.00	1.00
		180	48.54	48.23	43.68	36000.00	36000.00	27583.16	1.00	1.00	1.00
		190	48.28	43.66	37.91	36000.00	5617.84	1004.30	1.00	1.00	1.00
		200	46.23	45.64	43.63	3088.91	640.94	42.78	1.00	1.00	1.00
	15	110	39.69	39.00	30.61	36000.00	132.52	34.22	1.00	1.00	1.00
		120	52.49	44.65	41.17	603.06	487.89	19.92	1.00	1.00	0.87
		130	41.90	41.50	39.62	36000.00	574.73	216.58	1.00	1.00	0.99
		140	59.36	53.81	51.04	36000.00	453.86	222.52	1.00	1.00	1.00
		150	58.73	51.31	45.57	4854.13	604.17	236.03	0.99	0.99	0.99
		160	53.51	49.76	41.21	36000.00	36000.00	4448.69	1.00	1.00	0.99
		170	64.00	58.90	57.44	36000.00	36000.00	36000.00	1.00	0.99	0.99
		180	66.97	61.44	60.83	36000.00	2464.36	486.22	1.00	1.00	1.00
		190	69.97	60.30	56.84	36000.00	36000.00	1806.89	1.00	1.00	0.99
		200	72.31	67.96	65.51	11876.58	2764.38	156.97	1.00	0.99	0.92

|*T*|	|*J*|	|*I*|	Profit (10^6 Units) Full Model (36000 sec)			Execution Time (sec) Full Model (36000 sec)			Relative Profit Limited Model (180 sec)		
			Max	**Med**	**Min**	**Max**	**Med**	**Min**	**Max**	**Med**	**Min**
	20	110	52.43	51.08	36.16	50.45	45.20	43.06	1.00	1.00	1.00
		120	54.85	49.77	46.44	302.69	247.42	53.98	1.00	1.00	0.97
		130	49.78	49.54	47.03	392.17	155.48	96.98	1.00	1.00	0.99
		140	56.17	51.14	46.16	1082.30	323.06	77.08	1.00	00.99	0.97
		150	61.19	61.08	59.81	1513.53	1127.77	613.78	1.00	0.95	00.91
		160	59.56	57.10	42.90	1715.64	2426.95	3141.17	0.98	0.93	0.00
		170	61.43	61.06	58.65	36000.00	1825.00	916.38	1.00	0.96	0.96
		180	66.04	61.59	52.01	14655.58	5178.36	3049.25	0.85	0.73	0.47
		190	85.10	70.88	68.74	36000.00	18435.23	2607.81	0.99	0.88	0.87
		200	78.68	77.77	69.69	36000.00	1044.69	953.33	0.99	0.98	0.95
	25	110	41.37	39.52	35.29	453.31	145.02	117.39	1.00	1.00	0.86
		120	51.12	45.76	42.44	550.09	533.48	542.31	1.00	0.96	0.95
		130	56.57	52.17	52.02	1197.77	523.41	423.33	0.96	0.50	0.06
		140	68.58	66.81	57.29	1399.33	1242.53	1002.30	0.90	0.05	0.03
		150	65.83	48.66	45.17	1492.51	1275.27	378.06	0.78	0.47	0.00
		160	69.05	67.18	57.92	4747.02	3027.84	2813.84	0.74	0.19	0.00
		170	63.18	58.95	56.68	3140.72	2036.30	565.74	0.78	0.63	0.55
		180	85.56	70.24	68.83	6131.34	3164.70	1932.09	0.98	0.57	0.00
		190	81.53	65.67	57.69	4968.38	4877.63	3954.70	1.00	0.80	0.00
		200	80.68	69.32	65.68	7355.48	5939.98	33194.11	0.20	0.04	0.00
20	5	110	28.87	26.23	24.38	1920.27	6.44	2.47	1.00	1.00	0.93
		120	32.59	31.69	19.96	50.25	4.28	3.47	1.00	1.00	1.00
		130	36.49	34.86	16.42	662.33	388.25	1.39	1.00	1.00	1.00
		140	32.62	31.25	18.69	34264.63	5877.44	418.97	1.00	1.00	1.00
		150	40.53	33.21	19.37	36000.00	6993.28	36.88	1.00	1.00	1.00
		160	36.97	29.88	24.01	36000.00	837.53	41.17	1.00	1.00	1.00
		170	36.71	34.67	27.86	36000.00	36000.00	2175.84	1.00	1.00	1.00
		180	40.82	34.06	27.12	36000.00	36000.00	36000.00	1.00	1.00	1.00
		190	44.86	43.49	25.78	36000.00	16352.86	56.49	1.00	1.00	1.00
		200	38.68	31.11	25.22	36000.00	390.17	207.20	1.00	1.00	1.00
	10	110	41.71	35.15	25.32	27.58	22.89	21.00	1.00	1.00	1.00
		120	56.54	45.95	40.86	32.16	28.95	16.25	1.00	1.00	1.00
		130	43.18	40.24	25.24	39.63	31.91	20.78	1.00	1.00	1.00
		140	48.18	43.15	41.08	5542.72	43.36	12.75	1.00	1.00	0.99

$\|T\|$	$\|J\|$	$\|I\|$	Profit (10^6 Units) Full Model (36000 sec)			Execution Time (sec) Full Model (36000 sec)			Relative Profit Limited Model (180 sec)		
			Max	Med	Min	Max	Med	Min	Max	Med	Min
		150	53.78	43.87	42.78	75.73	53.95	30.39	1.00	1.00	1.00
		160	53.24	53.16	36.61	654.76	407.72	58.05	1.00	1.00	0.99
		170	54.15	53.48	45.23	16438.75	109.72	57.38	1.00	1.00	1.00
		180	63.57	58.85	54.93	1213.19	716.59	216.49	1.00	1.00	0.99
		190	60.34	47.43	38.23	8571.80	398.50	52.64	1.00	1.00	0.99
		200	68.60	53.60	48.57	36000.00	13868.73	2305.06	1.00	1.00	0.98
	15	110	47.04	35.24	31.85	72.81	61.08	55.70	1.00	1.00	1.00
		120	58.22	46.61	34.63	199.61	64.42	53.64	1.00	1.00	1.00
		130	54.98	50.76	45.51	398.70	80.27	46.45	1.00	1.00	1.00
		140	57.64	55.35	36.73	138.89	70.34	65.11	1.00	1.00	1.00
		150	55.08	52.02	47.17	125.27	69.44	53.03	1.00	1.00	1.00
		160	64.54	62.13	59.88	575.64	515.00	357.13	1.00	0.89	0.76
		170	78.92	62.01	55.12	1715.33	935.81	523.80	0.95	0.94	0.86
		180	71.25	64.99	60.16	2154.69	878.11	69.11	1.00	0.98	0.97
		190	65.60	63.62	62.76	1097.06	425.67	316.08	0.98	0.97	0.94
		200	73.96	73.67	71.47	11061.70	533.78	278.17	0.97	0.85	0.28
	20	110	49.68	42.74	41.41	509.53	102.56	54.63	1.00	1.00	0.95
		120	51.55	49.58	47.37	952.38	145.05	104.16	1.00	1.00	0.16
		130	45.72	44.64	43.79	511.84	272.48	191.81	1.00	0.96	0.04
		140	55.26	49.28	49.57	721.63	445.27	188.14	0.73	0.62	0.00
		150	65.16	54.48	52.11	977.78	168.05	106.59	1.00	1.00	00.11
		160	70.36	68.36	57.82	2862.08	2501.09	2135.14	0.96	0.90	0.79
		170	67.01	65.82	56.78	2473.23	1443.30	1030.17	0.99	0.89	0.11
		180	79.81	78.02	63.25	36000.00	3011.63	1569.34	0.94	0.37	0.00
		190	75.50	73.61	56.38	4193.75	3172.22	843.64	0.77	0.00	0.00
		200	78.84	75.33	75.00	2844.89	1940.23	1885.56	0.90	0.03	0.00
	25	110	44.77	38.51	37.05	308.50	286.88	122.55	1.00	0.55	0.11
		120	50.50	49.06	48.02	2209.72	1053.13	1035.27	0.91	0.62	0.08
		130	51.20	49.03	48.00	2683.09	2021.30	1517.48	1.00	0.90	0.14
		140	60.29	58.42	55.67	2046.67	1300.97	868.13	0.58	0.11	0.08
		150	61.77	54.78	45.56	2308.56	2225.98	1233.98	0.49	0.05	0.05
		160	72.75	57.48	57.24	6204.44	6060.75	316.53	0.16	0.04	0.04
		170	74.48	62.42	53.15	8025.63	2511.33	159.38	1.00	0.08	0.03
		180	79.55	69.03	64.52	9450.31	3981.59	2117.91	0.76	0.35	0.00

$\|T\|$	$\|J\|$	$\|I\|$	Profit (10^6 Units) Full Model (36000 sec)			Execution Time (sec) Full Model (36000 sec)			Relative Profit Limited Model (180 sec)		
			Max	**Med**	**Min**	**Max**	**Med**	**Min**	**Max**	**Med**	**Min**
		190	72.57	70.81	62.16	7060.61	5314.75	3653.73	0.49	0.04	0.00
		200	95.20	94.74	83.49	5853.70	2839.45	2246.44	0.23	0.00	0.00