Model and Algorithm for Real-Time Resource Allocation in Fog Computing Networks

by

Jonathan Daigneault, **B.Eng**

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

Masters of Applied Science in Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario
August, 2020

The undersigned hereby recommends to the
Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis

Model and Algorithm for Real-Time Resource Allocation in Fog Computing Networks

submitted by **Jonathan Daigneault**, **B.Eng**

in partial fulfillment of the requirements for the degree of

Masters of Applied Science in Electrical Engineering

Professor Marc St-Hilaire, Thesis Supervisor,
Carleton University

Professor Amiya Nayak, University of Ottawa

Professor Chung-Horng Lung, Carleton University

Professor Sreeraman Rajan, Carleton University

Professor James Green, Chair,
Department of Systems and Computer Engineering

August, 2020

Abstract

Fog networks are a proposed solution to allow the generalization of endpoint devices. Previous efforts have been dedicated to the optimization of fog resource initial installations, but no solution has been proposed to optimize the real-time resource allocation of a fog network in operation. An exact model was developed to compute the upper bound for profit generation with a processing time exponentially related to the network size. A real-time heuristic was also developed to allow for the network to perform operations. Its performance remained constant through a variety of tested networks at a profit result between 78 to 88% of the exact model and a far reduced processing time. The heuristic model uses a statistical approach to predict the requirements of future tasks. The results of this thesis demonstrate that the use of the heuristic model is essential to the efficient operation of a long term fog computing network.

To my wife and her essential support in enabling this journey.

Acknowledgments

I would like to acknowledge the contribution and patience Professor Marc St-Hilaire provided me during this journey. The development of this thesis has not been without challenge and his guidance was essential in fulfilling a life milestone.

I would like to thank my employer for allowing me time to pursue academic advancement. Unlike many, I chose to begin a Master's in Applied Sciences 10 years after the start of my professional career. This isn't without unique challenges, but their support was essential in maintaining a balanced life style to fulfill this milestone.

I would also like to thank my family for their unconditional support and assistance throughout this process. My wife for her enablement and patience and my son for showing me the importance of cherishing every moment.

Table of Contents

List of Tables

List of Figures

Nomenclature

Throughout this thesis, a mathematical nomenclature is used to refer to specific parameters used in the models. This page serves as a reference in alphabetical order to the reader.

Symbol	Definition
cd	Constant value indicating a baseline for up-front costs of changing the state of a fog node from up to down
cu	Constant value indicating a baseline for up-front costs of changing the state of a fog node from down to up
I	Set of tasks
J	Set of fog nodes
mn	Maintenance cost of a fog node to remain active in dollars per time window
r	Relative revenue (in currency units) per time window of assigning a task to a fog node rather than the cloud
T	Set of time windows
x_{ijt}	Binary variable indicating if task i is connected to a fog node j at time window t
α_i	Maximum distance, in meters to its fog node (server), demanded by i

χ_i	Start time of task i
δ_j	Two-tuple value corresponding to the latitude and longitude coordinates of a fog node
ϵ_j	Maximum memory (in GB) that can be allocated to tasks
η_i	Minimum number of vCPU cores requested by task i
γ_i	Two-tuple value corresponding to the latitude and longitude coordinates of the task request i
μ_{jt}	Binary variable used to capture a fog node startup at time window t (1 is starting up and 0 is not)
ν_j	Maximum number of parallel tasks a fog node can handle in units
ϕ_{jt}	Binary variable used to capture the fog node shutdown at time window t (1 is shutting down and 0 is not)
ψ_i	Duration in number of time windows required to complete task i
θ_{jt}	Binary variable used to capture the state of a fog node j at time window t (1 is on and 0 is off)
υ_j	Maximum number of vCPU cores that can be allocated to tasks by fog node j
ζ_i	Minimum memory, in Gigabytes (GB), demanded by i

List of Acronyms

BBU Baseband Unit

CLI Command Line Interface

CPLEX IBM ILOG CPLEX Optimization Studio

cRAN Cloud Radio Access Network

DDR3 Double Data Rate 3

EWMA Exponentially Weighted Moving Average

FN Fog Node

fRAN Fog Radio Access Network

FUSD Fast Up Slow Down

GB Gigabyte

GHz Gigahertz

GPU Graphics Processing Unit

GUI Graphical User Interface

MHz Megahertz

NS No Solution

OoM Out of Memory

OPL Optimization Studio

QoS Quality of Service

RAN Radio Access Network

RRU Remote Radio Unit

SLA Service Level Agreement

SSD Solid State Drive

tpw Tasks per Time Window

VANET Vehicular Ad Hoc Network

vCPU Virtual Central Processing Unit

VSVBP Variable Sized Vector Bin Packing

Chapter 1

Introduction

Information technology data volumes are increasing at an exponential rate. Cisco reports in a 2021 forecast that Internet Protocol traffic in Canada will grow two fold between 2016 and 2021 to 178 petabytes per day [3]. Notably, 20% of Internet traffic in Canada will be generated on mobile networks further increasing network load in these environments.

According to Cisco, the number of user devices per capita is also forecasted to continue its adoption increase. Interestingly, the number of desktop and laptop devices are expected to decrease with lower processing capacity Internet of Things devices replacing them. This emergence will rely on remote processing capabilities to enable low capacity devices to meet user requirements. Interactive environments such as virtual machines operating in cloud and fog networks will increase reliance on maintaining a low latency service to users [4]. Fog computing networks are predicted to be a technical solution to service latency sensitive user requests such as the remote operation of virtual machines. However, several considerations arise in the long term operation of a fog computing network with fixed location fog nodes.

- **Service Coverage:** The level of service of a fog network is heavily dependent on its configuration. Mobile network technologies such as the emergence of 5G will continue to improve network availability and stability in populated areas.
- **Network Efficiency:** As networks grow to accommodate the growing demand, network providers will look to green computing solutions to reduce their operating costs and increase their profit. Notably, correctly predicting network load and areas of over subscription will greatly benefit network efficiency in these applications.

- **Network Saturation:** Service providers will look to advanced network planning solutions to avoid saturation while maintaining an efficient network. Networks will be required to be dynamic to accommodate burst customer demands.
- **Network Backbone:** The core network capacity required to support these networks would continue to increase exponentially. To reduce the reliance and load on back-end networks, fog computing networks are the proposed technology to reduce the reliance on the core network and allow networks to support demand primarily at the edge. This provides service providers with a lower cost and simpler architecture to lower latency.

1.1 Problem Statement

In 2020, documented fog computing networks are being designed initially to be efficient in their environments. This provides short term benefits to service providers, but fails to scale as demand increases or shifts within their coverage area [5]. This static development of fog networks is inefficient and lacks the ability to adapt to inherently dynamic activities such as allocating tasks to fog node resources. As the demand frequency and patterns shift over time, the fog computing network becomes less efficient at supporting these requests and will relegate additional tasks to the cloud when it can't support the low latency resources demanded by the applications.

To address this problem, network operators must use resource assignment models to assign fog network assets to task requests while maximizing profit. The problem is further complicated when combining the objectives of service coverage (assigning as many incoming tasks to a fog node) and increasing network efficiency (green computing) by shutting down unused fog nodes and only starting them up when required.

An exact model can be used to optimally solve this problem. This thesis stipulates that optimally assigning a set of tasks to a fog computing network is a NP-hard problem. It predicts that the size and complexity of the optimization problems will grow exponentially with the size of the network and the number of task requests in the evaluation time period. Service providers would receive a stream of task requests rather than a complete set of tasks demanded of the network. Any exact model must rely on complete knowledge of network requests to make its decisions on task allocation and fog node status changes. These restrictions prevent an exact model

from being used for real-time resource allocation applications. However, it can be used to determine the upper bound values for a network assignment and should be used as a benchmark for the evaluation of real-time resource allocation models.

A real-time allocation model must make allocation decisions without exact knowledge on the frequency, location and type of requests that will be demanded of the network in the future. The development of such a model would rely on assignment heuristics and a probabilistic approach using historic data to fog node status changes. Furthermore, the assignment operations for a given time window must be less than the time window itself to ensure that the buffer of incoming tasks remains empty and that operations can continue in real-time. This thesis will propose ideas to assist in prioritizing tasks and manipulating the state of fog nodes to approach the upper bound.

1.2 Research Objectives and Contributions

The main objective of this thesis is to develop models and algorithms to maximize the profitability of a 2-tier fog-cloud network environment. Specifically, it can be divided into the following sub-objectives:

- Propose a mathematical model to calculate the optimal task allocation of a given fog computing network using complete knowledge of all tasks demanded. This model is used to determine the theoretical maximum profit for past task assignments.
- Develop a heuristic model that can perform real-time task assignment and scale to large and complex networks. The model must account for the incomplete knowledge of future demanded tasks and the duration of active tasks at a given time. Furthermore, the model needs to operate efficiently to ensure that the buffer of tasks to assign does not overflow.
- Implement the mathematical model in a linear programming software solver and validate its design.
- Implement the heuristic model in a computing software and validate its design. While implementing the heuristic, this thesis aims to use a modular approach so that adjustments could be made to a single module without affecting the rest of the model.

- Evaluate the performance of multiple configurations of the real-time heuristic model by comparing its profit generation to the mathematical model upper bound.

By reaching the above objectives, the following contributions can be made to this research area:

- The development of a mathematical model to solve an input set of tasks with a fog computing network resource fog nodes into an optimal profit result. It is not designed to solve the real-time resource allocation problem, but can be used as a benchmark to compare various approximate algorithms. For this reason, it is an essential contribution to real-time resource allocation in fog computing networks.

- The development of a heuristic model to provide a real-time solution to optimize profit. This model can be used by network operators to improve profit generation and user experience when compared with the current state of resource allocation in fog computing networks.

1.3 Methodology

This section addresses the methodology that will be used to meet the research objectives enumerated in Section 1.2. The first 3 steps are linked to the development of the mathematical model, the following 2 steps to the development of the heuristic model and the final 2 steps to the performance evaluation objective.

- **Develop the theoretical model:** Initially, this thesis will study the components of fog and cloud computing networks in general such as ones presented in the OpenFog consortium reference architecture [6] to develop a mathematical formulation that simulates the operation of a fog computing network. Its design aims to include flexibility in its parameters to allow it to adapt to numerous fog computing applications. Afterwards, we will define and detail the complete input data sets that are required to allocate tasks in a fog computing network. Finally, the mathematical model can be finalized to perform the linear programming calculations.

- **Implement the theoretical model:** Any non-trivial test network will require a software solver (CPLEX) for the mathematical model. It is expected that the complexity of the linear programming model is correlated to the number of tasks, fog nodes and their parameters.

- **Validate the model:** The mathematical model must be validated for accuracy since it will be used as a primary performance evaluation tool. This is achieved by individually testing each constraint in trivial networks and ensuring that the behaviour matches what is expected. Then, the model will solve simple networks to compare with a manually calculated upper bound. Finally, the model will scale in size and complexity to ensure that the solutions proposed are feasible and that no better solution exists (at least intuitively for more complex simulations). This thesis will assume that the mathematical model will remain accurate as it scales to larger networks that can not be trivially solved without the mathematical model solver.

- **Develop the heuristic model:** This thesis will transition to the development of a real-time heuristic model by truncating the information available to the model and preventing changes to task allocation outside of the current time. Methods will be developed to allocate tasks, de-allocate tasks and modify the state of fog nodes to best simulate a realistic application of the technology. The model will initially develop its framework using a modular approach and simple algorithms. Then, it will attempt to improve its performance by developing tailored algorithms for each of the main modules that provide results more closely resembling the theoretical model.

- **Implement the heuristic model:** Once again, the heuristic model will require the assistance of a software (MatLab) to make a determination on a chosen task to resource mapping that most closely approaches the theoretic maximum determined in the mathematical model.

- **Generate test networks:** We will develop and follow a test plan to generate networks for performance evaluation of the heuristic model. Specifically, the test plan must represent the different types of task demand (frequency), the duration of tasks (level of variance) and their resources demanded (quantity). The plan must also vary in terms of network coverage and resources per node.

- **Evaluate the performance:** This thesis will configure the heuristic model parameters for different types of test environments and evaluate the impact of task predictability to that configuration. It will also compare its performance in profit generation and computing complexity with the mathematical model using various test networks.

1.4 Thesis Outline

This thesis will be divided as follows:

- Chapter 2: Background and Related Work will introduce the concept of fog computing, its applications and current research ideas in resource allocation in fog and cloud computing applications.
- Chapter 3: Formulation of the Task Allocation Problem will outline the assumptions, formulate the problem and its mathematical representation. It will also provide a framework for the heuristic algorithm.
- Chapter 4: Results and Analysis will provide details to the validation of the algorithm development, the test plan and its performance.
- Chapter 5: Conclusion and Future Work will summarize the thesis results and propose directions for possible future research.

Chapter 2

Background and Related Work

In this chapter, we will provide an overview of the research landscape related to fog computing. It will encompass preceding research and current efforts in the field. Specifically, Section 2.1 will provide background information on the topic, Section 2.2 will introduce complementary research in this field and Section 2.3 will summarize the chapter.

2.1 Background

In this section, we will introduce the concept of fog computing and its evolution in Section 2.1.1. Section 2.1.2 will introduce various applications that benefit from fog computing.

2.1.1 Fog Computing

Fog computing was introduced in 2012 by Cisco Systems Inc. It was designed as an evolution from cloud computing concepts to better support ubiquitous networks by positioning computing resources at the edge of the network. This enables remote computing solutions to perform at lower latency than an equivalently designed cloud. Several latency sensitive applications, categorized as infrastructure as a service, platform as a service and software as a service, are well suited for fog computing in the current market [7] as a preferred method to reduce data round trip time.

Fog computing is composed of n tiers allowing for proper prioritization of tasks within a population base. It is designed to operate in a heterogeneous environment and supports a cooperation between similar devices on the network. It is viewed as a cloud

computing solution that is closer to the end user in order to improve performance. It is poorly tailored to operate in a standalone mode due to its requirement to position its processing resources close to its users. Legacy standalone networks are, by definition, antonymous to this concept, but are plentiful in the current network landscape. Software defined networking is a proposed solution that virtually combines standalone network architectures within a common heterogeneous fog computing network architecture [8]. Figure 2.1 shows the hierarchical architecture of a fog computing network as illustrated by Hu et al. [1].

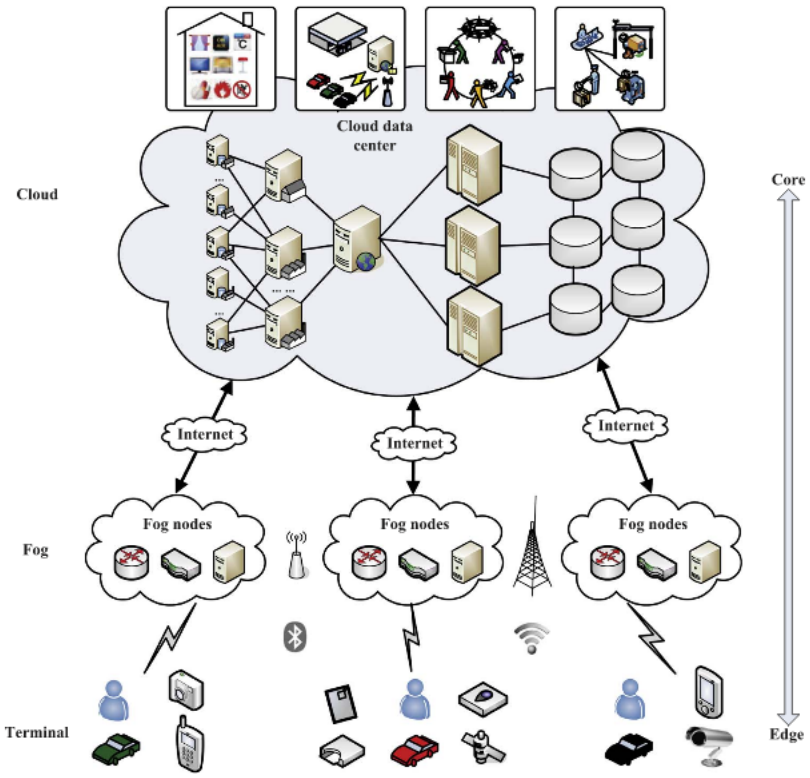

Figure 2.1: Fog Computing Network Hierachical Architecture [1]

Fog architectures are "either application agnostic or application specific" [9]. Regardless, fog computing network architectures are planned with possible applications in mind to evaluate the possible customer requirements and generate the test data. Applications will diverge in processing, network bandwidth and latency requirements. Following are a few examples of common applications that highly benefit from fog computing networks.

2.1.1.1 Cyber Foraging

The founding concepts behind edge computing were introduced in a 2001 paper by Satyanarayanan et al. [10]. It discusses pervasive computing as an evolution of distributed systems and mobile computing and introduces cyber foraging as a method to offload computing intensive tasks from mobile devices to remote computing resources [11]. According to Balan et al. [12], computing surrogates are used to improve performance in data staging by caching and pre-fetching useful content closer to the edge device. It can also be beneficial for remote code execution, as long as application developers consider the cost versus benefits of doing so.

2.1.1.2 Cloudlet

In 2009, Cloudlets were introduced as a virtual cloud infrastructure evolution to Cyber Foraging. Similarly, it uses virtual machines to offload intensive computing resources from Internet of Things devices to the cloud [13]. It is similar to traditional cloud computing, but with cloudlet resources strategically positioned near edge resources to improve latency for beneficial applications. Cloudlet computing resources scale dynamically to the processing requirements of devices either taking or providing resources to the larger cloud.

2.1.1.3 Multi-access Edge Computing

Multi-access edge computing was introduced in 2014 by the European Telecommunications Standards Institute as a purely standalone single purpose cloud architecture [9]. It is often configured to be optimized for that single purpose with its standard operating procedures well defined.

2.1.1.4 Other Fog Computing Concepts

Other similar concepts were designed specifically for a given application. They are viewed as part of the evolution of fog computing, but lacks support for heterogeneous clients [14]. Mobile cloud computing and mobile edge computing are examples of such concepts.

Mobile cloud computing is a cloud computing infrastructure specific for mobile applications. This enables applications to standardize computing power and data storage

for their requirements in the cloud while running across a heterogeneous set of end user devices in a mobile network (along with varying computing specifications) [15].

Mobile edge computing is a remote processing capability at the edge of the mobile network to complete tasks that could not be completed in the core (ex: user device cell allocation). [16]. Fog computing applied to mobile networks is a blend of mobile edge and cloud computing concepts.

2.1.1.5 Task Allocation in Fog Computing

Allocating tasks to a fog computing network is an essential element to its ongoing operation. In a centralized computing network architecture such as the cloud, service providers ensure that the processing capacity of the network can support the resource demand [17]. In fog computing, resource allocation is used to support the remote processing of latency sensitive tasks. It must allocate tasks to nearby fog resources. Some task allocation models propose that users make a determination of whether to process tasks locally, at the network edge (fog) or in the cloud [18]. This model assumes that service providers have planned their networks appropriately to support the majority of their user base in each category. Other research proposes a decision making process made by the network provider. It uses a load balanced approach to task allocation the appropriate supporting resources for each task [19] [20] to benefit the overall operation of the fog network.

2.1.2 Applications of Fog Computing

2.1.2.1 Fog Computing in Radio Access Networks

Wireless technologies and specifically Radio Access Network (RAN) is a research area that is improved using pooled computing resources. Many complex problems in the network planning of RAN is trivialized based on the application of these concepts.

The evolution of cellular technology from 3G to 4G and even going forward to 5G introduced the concept of a logical processing node at the RAN layer used to handle all radio related processing [16]. In a hardware deployment of RAN equipment, the processing is done at each cell node (eNode B). Network planners must predict network usage across their area of coverage to allocate sufficient processing at each edge

node (eNode B) while limiting over-allocation to manage operational costs. As wireless networks increase in size, these costs are quickly overtaking revenues in many deployments [2]. Figure 2.2 illustrates the functions of a 4G mobile edge node as illustrated by Checko et al. [2].

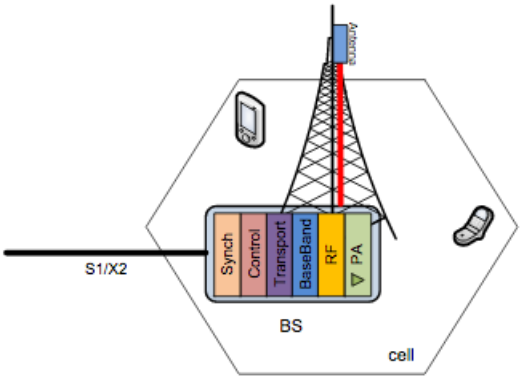

Figure 2.2: Cloud RAN Mobile Deployment [2]

The introduction of cloud RAN (cRAN) was designed to reduce the load at the mobile core network by processing data at the RAN core (distinct from the LTE core network). The eNode B is divided in three parts: the remote radio unit (RRU), baseband unit (BBU) and a high bandwidth, low latency fronthaul connection between the RRU and BBU. The mobile core is then connected to the BBU using a backhaul connection.

cRAN centralizes BBU resources into a single pool. This provides network planners with the ability to dynamically allocate and manage processing resources [21] rather than dedicate their allocation to a single eNode B. This simplifies the problem of predicting and adjusting processing resources for the edge network. However, radio resource management is infeasible due to its high computational complexity and latency requirements. As networks evolve to heterogeneous device support, the scalability issues of the fronthaul connection is a notable issue with the cRAN architecture [22]. Figure 2.3 illustrates the functions of a 4G cloud mobile edge node as illustrated by Checko et al. [2].

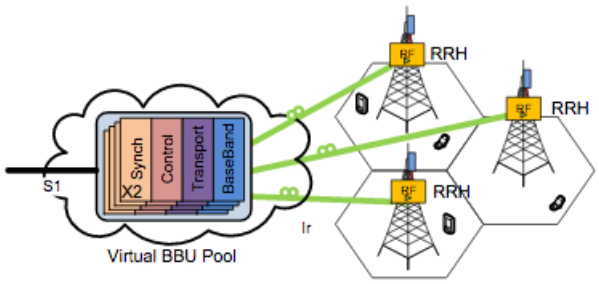

Figure 2.3: Cloud RAN Mobile Deployment [2]

Fog RAN (fRAN) divides the cloud into processing components to better distribute load based on the dominant performance criterion. The cRAN cloud is divided into 4 function groups. The distributed storage cloud, distributed communication clouds and control cloud are moved from the RAN core layer to the fog layer while the centralized communication and storage cloud remains at the RAN core. This allows for network planners to distribute the processes between the four clouds to meet latency and resource requirements for each task, thus improving efficiency in the network [23].

Similarly to cRAN, resource allocation is pooled and can easily scale. However, the dependency on the performance of the fronthaul connection is reduced from the cRAN model further improving computing performance of wireless RAN. In this model, resource allocation to a fog node is based on service requests rather than client allocation. To further improve performance, network planners use predictable content relevant to user geographic location [24]. In a wireless network application, this prediction is relatively simple since cells are inherently geographically bounded. When considering a static model, the allocation of requests to an optimal fog node or cloud is relatively simple. However, the simplistic model described previously is inherently sub-optimal and dependent on the users on the network. Once a fog node (distributed cloud) is at capacity, the users are relegated to the centralized cloud rather than being serviced by another fog node. Contrarily, latency sensitive applications can be forced from a cloud server to a fog node given that sufficient processing on the fog is available [25]. Considering this complete mobile network for fog node and cloud server allocation is a hard problem to optimize.

2.1.2.2 Vehicular Ad-Hoc Networks

Vehicular ad hoc networks (VANET) were introduced as a solution to reduce road congestion and accidents by using computing to synchronize traffic lights, optimize path selection, manage emergencies and regulate vehicle speed. Cloud computing was initially introduced, however the static deployment of clouds made it difficult to achieve latency, performance, scalability and reliability thresholds.

Fog computing is a paradigm that is applicable as an extension of cloud-based VANETs [26] to improve performance. This enables networks to expand its cloud infrastructure to include a localized computing concentration with its users. However, many VANET applications leverage fog computing nodes that are either static or dynamic. Static fog computing solutions include using stationary public and / or private road side units as fog nodes [27] [28]. Dynamic fog computing solutions include using underutilized vehicle resources as ad-hoc fog nodes [29] enabling vehicle to vehicle communication as a low latency solution to many vehicular applications.

However, a true fog computing solution combines both static and dynamic computing solutions as cloud and fog nodes respectively. All these resources are available to any end user in a cross-layer model [30]. This VANET architecture leverages the advantages of a fog computing network and allows for fog node mobility within the architecture. However, the problem of resource allocation is left unresolved in this field.

2.2 Related work

Fog computing research is being described as a significant improvement to cloud computing in mobile and heterogeneous environments [26]. Applications such as connected vehicles [31], content delivery [32], wireless edge computing [14], smart grid design [14] and health care connectivity [33] are some applications that generally benefit from low latency fog computing architectures. However, research related to the optimal planning and design of fog networks is starting to emerge, notably with contributions related to exact and approximate algorithms for the planning and design of fog nodes [34] [35].

Once a network is in place, the optimal allocation of resources is a difficult problem to

solve. This section will review notable contributions related to resource allocation in both fog and cloud computing applications. Subsection 2.2.1 will explore the general concept of task uncertainty and its application in cloud computing. Subsection 2.2.2 and 2.2.3 will explore resource allocation using predictive factors. Subsection 2.2.4 introduces a game theoretic approach to resource allocation. Finally, Subsection 2.2.5 introduces fog computing allocation using quality of service (QoS) as a main parameter.

2.2.1 Understanding Uncertainty in Cloud Computing

Tchernykh et al. [36] discuss the concept of uncertainty programming in cloud computing. It is based on the consideration of incomplete knowledge within its decision parameters. Specifically in cloud computing, this can be summarized as users (or tasks) predicting their resource requirements while accounting for the dynamic performance of the communications link to the cloud. For example, the network routing latency between a user and the cloud changes based on many unpredictable factors such as network load and QoS configurations.

In cloud computing, uncertainty can be handled either by the client or by the cloud itself. Although some client programs can optimize their decisions based on uncertainty, the cloud ensures that this parameter is considered in permanence. This allows network planners to ensure that uncertainty is considered in every client (task) requesting resources from the cloud.

Kliazovich et al. [37] propose a model by introducing communication awareness within its procedure. Specifically, it separates communication requirements (uncertainty) with the other certain decision parameters (latency requirements, computing resources requested, memory requested, etc.) and is optimized for scheduling separately based on both sets of parameters. The approach allocates resources using certain parameters dependent on the results of the communications (uncertain) parameters. This artificially changes the priority of a given task based on the present communication performance. This introduces the concept of dynamic resource allocation which is a founding idea for many subsequent efforts.

2.2.1.1 Resource Provisioning

When scheduling tasks between clients and the cloud, there are 2 methods: static and dynamic scheduling. Static scheduling is set for a given period and is effective in applications where network planners have complete knowledge of resource requirements and client to cloud resource under allocation. Dynamic scheduling is a much more common solution since resource demands are difficult to predict and resource usage will naturally fluctuate over time.

Service Level Agreements (SLA) between a client and a cloud provider to better document the minimum QoS and task delivery time for a cloud to shift the cloud computing design approach from a static model to allow for dynamic resource allocation. Schwiegelshohn et al. [38] designed a resource scheduling model where customers (serviced in an SLA) are chosen based on a cost and a slack per requested task. The slack is the allowable delay that a task can undertake to maintain the SLA and the smallest slack (earliest due date) were prioritized in their algorithm. The authors performed an analysis of task allocation over single and parallel cloud machines against the income generated from their scheduling algorithm against the optimal income.

2.2.1.2 Load Balancing

Load balancing is a method to optimize resource allocation when considering computing and communication imbalance created by uncertainty. Elasticity is a solution to handle this unpredictability. Herbst et al. [39] define elasticity as a systems ability to adapt itself by provisioning and de-provisioning its resources autonomously. Its concepts are naturally useful in fog computing notably when including green computing considerations in many research publications [40] [41].

2.2.1.3 Adaptive Scheduling

Adaptive scheduling of tasks to multiprocessing units is a hard problem and has not been resolved in large scale installations to date. Some methods include adaptive scheduling in their algorithms, but with an overflow of processing related to demand. [42]. A common problem with adaptive scheduling to cloud processing units is that cloud computing nodes resources are reserved, but not utilized by a given task. This creates an unnecessary queue in task scheduling leading to many inefficiencies.

Tchernykh et al. [43] introduce a two stage heuristic to improve adaptive scheduling. Firstly, tasks are assigned to the cloud computing node with the smallest resources to complete the task. Then, a local scheduling heuristic is applied on each node to maximize resource utilization and minimize overall run time. This approach is interesting, but may need to be adapted in cases where the set of tasks all require similar resources. Perhaps a weight parameter could be added to distribute tasks across cloud computing nodes.

2.2.1.4 Knowledge-free Approach

Tchernykh et al., [44] discuss resource allocation to computing nodes without considering task specific parameters. This is relevant in applications with dynamic client and server resources. They suggest over-allocating tasks to computing resources along a dynamic replication threshold to account for uncertainty in a given application. The replication threshold is adapted over time based on historic performance to improve energy efficiency. Some tasks are historically static, thus requiring less replication while others require more replication to ensure processing. The performance of multiple algorithms is evaluated using a Pareto optimal set.

2.2.1.5 Scheduling with Uncertainty

Probability theory and stochastic processes are discussed methods to consider uncertainties in an optimization problem. In cloud computing, two main frameworks are used: stochastic scheduling and online scheduling.

Stochastic scheduling is used to consider the characteristics of tasks while online scheduling is used to consider the unknown requirements related to future task requests. A decision must be made every time a task is requested to the cloud. Both of these concepts are always relevant in cloud computing resource allocation problems.

Megow [45], Megow et al. [46], and Vredeveld [47], present task scheduling models that are based on stochastic properties and a knowledge of the probabilistic distribution of task processing time. “They minimize expected value of the weighted completion times of jobs” to achieve a favorable performance ratio (2 in this case). Cai et al. [48] expanded the work to include due dates, machine breakdowns in addition to the processing time probability distribution while maintaining the same performance ratio.

In both models described above, the model performance is dependent on an accurate representation of processing time. Various research efforts have been made to improve this dependency. Kianpisheh et al. [49] apply various machine learning algorithms to attempt to better predict this parameter. Smith et al. [50] and Ramirez et al. [51] use historical processing statistics to generate the estimate. They "apply self-similarity and heavy tails characteristics to create scalability models". They follow a two step process to the model. The first step models the queueing process using historical similarities in each parameter considered. The second step predicts remaining execution time using conditional probabilities.

2.2.2 Resource Allocation in the Fog Using Mobility as the Decision Factor

Aazam et al. [52] propose a resource allocation model based on the probability of mobility of users. Resource requests made to the fog network are evaluated based on the historical user mobility. The model states that the more mobility observed within a group of users, the lower the amount of resources the network is willing to allocate to them.

This concept aims to reduce fluctuation in the amount of resources being allocated on a fog computing network by normalizing the quantity of resources to allocate over time. Dynamic fog nodes will use smaller resource allocation blocks per client while more static nodes will allocate larger blocks of resources to a given user. This ensures that the resources are used on a fog most efficiently.

Do et al. [53] propose a proximal algorithm to optimize resource allocation in a fog computing network. Geo-distributed cloud networks are often energy inefficient and have a large carbon footprint. As such, network planners not only wish to optimize processing resource allocation in a fog network, but also consider the start up costs of a fog node. The paper proposes that the optimal dynamic resource allocation of tasks across a fog computing network is often inaccurate due to the optimization function being sometimes convex, especially in large fog networks where the optimal client to fog node mapping changes constantly. In these cases, traditional optimization resolution methods such as the steepest gradient [54] do not always provide an optimal solution to the problem. The paper proposes using proximal functions to transform

the optimization algorithms to eliminate the possibility of dealing with convex functions, thus improving resource allocation in fog computing networks overall.

2.2.3 Resource Allocation in the Cloud Using a Predictive Model

Xiao et al. [55] suggest a model to address the problem of over-provisioning of physical servers in cloud computing applications. The model is based on two goals: avoiding server overload and green computing. Both goals are somewhat contradictory, but they claim to develop an automated system for a balanced resource provisioning.

Firstly, a predictive model for resource requirements is developed. It is based on an exponentially weighted moving average (EWMA) to achieve a short term prediction of load. The algorithm is designed to reflect resource acceleration and develop a fast up slow down (FUSD) model. The authors use this approach to more often predict higher values than measured in testing. The impact of this design decision is that server overload is further minimized at the expense of green computing.

Secondly, a skewness algorithm is developed to measure the distance of resource allocation on physical servers from a desired state. Cloud computing physical servers are categorized as hot, cold or warm. In all states, temperature is measured based on the square sum distance from its threshold.

Hot servers are simply servers that have resources allocated over a desired threshold. These servers are candidates for having virtual machines migrated to another physical server. Cold servers are mostly idle servers and are candidates to decommission or to assign additional virtual machines. Warm servers are servers that are operating in a desired window. An ideal network would only contain warm servers.

In the model, the authors introduce a consolidation limit which is designed to limit the amount of changes in one optimization iteration to limit the probability of oscillation within resource allocation.

The model combines both the predictive and skewness algorithms to optimize resource allocation in cloud computing applications. The model that uses load prediction in its skewness algorithm has shown better performance, notably to prevent some hot and cold zones from appearing in the cloud computing architecture.

2.2.4 Game Theoretic Method for Resource Allocation in Cloud Computing

Wei et al. in [56] propose a cost-time optimized algorithm for client to cloud resource allocation. Firstly, it considers dependencies, a NP-hard problem at large scale between certain tasks and accounts for it within the heuristic. They solve this issue by adding constraints in a binary integer programming model where each variable is a resource requested to the cloud (where a true value is the resource assigned to the cloud).

Secondly, the model considers computing resources (cost) requested by a user to the cloud for a given task (users can have multiple tasks). Tasks can also benefit multiple users which is handled in the model by dividing the cost between users. However, the model adds a constraint to ensure that the optimization assigns all the users or none of them.

The goal of the optimization problem is to assign the tasks to the cloud while minimizing cost. However, the model can only be applied to a single cloud instance with immobile users. It would need to be adapted to consider a joining cost to each cloud in order to be adapted to a geographically distributed cloud or fog computing model. It also assigns resources based on a assumed perfect budget and does not adapt its mapping when tasks are either completed or are added to the request queue.

2.2.5 Fog Computing Allocation Using Dynamic Quality of Service

Lai et al. [57] propose a model that allocates users to a fog node using a dynamic set of resource demands. Specifically, they propose that the application scenario, such as a remotely hosted video game application, can be used to tailor which demanded parameters can be reduced without significantly impacting user experience. An example proposed is that the difference between streaming the graphics at 360p to 1080p is of much larger than the user impact of streaming graphics from 1080p to 1440p. The authors propose that maximizing allocated users while diminishing QoS can significantly improve user experience across the network. The objective function aims to maximize the QoS value (user factored to the allocation value of the QoS parameters) allocated to a fog node.

The model compares three allocation schemes: theoretic, random and variable sized vector bin packing (VSVBP). The VSVBP "proposes an approach that maximizes the number of allocated users". [57] The theoretic allocation uses a linear programming function and simulates the perfect allocation, but poorly scales to large networks. As expected, the random allocation performs significantly lower than the VSVBP allocation. In tested experiments, the VSVBP allocation performed close to the theoretical bound for small or large networks (depending on the tuning) while the random allocation was consistently significantly lower than the upper bound.

2.3 Chapter Summary

This chapter provided an overview of existing background and related literature pertinent to the research of this thesis. It presented past and current concepts related to cloud, fog computing and their evolution towards additional devices. Supporting literature presented the concepts of cloud, fog and task allocation. Many efforts are centered around predicting and modeling demanded resources to a cloud or fog network. Some propose using historic task request parameters as factors to influence network allocation. Others modify the demanded resources to reduce the impact of network service bottlenecks to improve network allocation.

No related work was found that proposes a real-time resource allocation model while considering the green computing requirements of a fog computing network. Furthermore, no related work was found that proposes an exact solution to determine the resource allocation upper bound to evaluate the performance of an established fog computing network. At the best of this thesis' research, there is no work that effectively proposes a solution to these problems.

Chapter 3

Formulation of the Task Allocation Problem

In this chapter, the dynamic task allocation theoretic and heuristic models are formulated. Section 3.1 describes the basic concepts used in this thesis. Section 3.2 describes the theoretic model, notation and mathematical formulation. Section 3.3 describes the heuristic model and its architecture. Finally, Section 3.4 summarizes the chapter.

3.1 Basic Concepts

The task scheduling problem is presented when resources are requested from a fog computing network by a series of tasks. Each task can be assigned to a fog node or the cloud. The model calculates the increase in profit that a service provider would make from operating a fog computing network with a cloud assignment backup rather than a cloud only architecture. The inherent value of operating a fog computing network can also include additional capabilities that a network provider can offer its clients over a simple cloud assignment. This thesis does not provide a decision point to network providers on whether they should operate a fog computing network. It instead assumes that a 2-tier fog-cloud infrastructure is already deployed and provides optimal operating configuration for fog computing networks in regards to task allocation.

Firstly, in most practical scenarios, fog networks are statically allocating their nodes across a geographical area. As discussed in Chapter 2, previous work was made to optimize the initial location of fog nodes [58] [59]. They largely rely on predictions

and historical data related to tasks requested [60] or use a population density coverage model [61]. Regardless of the performance of a fog network at a given start point, its performance in time varies since the static allocation of the network must serve a dynamic set of tasks in real-time. The task scheduling problem assumes that the fog network is established and functional and can not directly impact the location of the fog network hardware. It must optimally assign tasks to the network as it stands.

The task scheduling problem aims to optimally allocate tasks over time in order to optimize its objective function parameter. In this case, profit is optimized as it can both evaluate performance in task allocation while considering green computing requirements. In this application, the over allocation of resources could lead to a poor performance even in cases where all tasks are assigned to the fog due to the consideration of operating costs. Tasks that are not assigned to a fog node are relegated to the cloud so the profit variable in this case indicates the improvement in profit between both methods of allocation.

3.1.1 Overview

Figure 3.1 shows an overview of the problem being solved in this thesis. Tasks request resources to the fog computing network and the network controller determines how they are assigned: either by assigning it to a fog node or the cloud. Assignment to a fog node leads to an improved experience to the user (i.e. significant improvement in latency). Tasks can also be relegated to the cloud for processing, taking no fog computing resources, but eliminating any performance benefits of operating in the fog. Each fog assignment is also not equivalent. Notably, an allocation to a distant fog may not provide any performance benefits to the end user while still taking away resources in the fog for other tasks that could benefit from a fog assignment. Moreover, the operational status of the fog nodes must also be considered before assigning tasks. As can be seen, an optimal management of available resources is paramount to operating a fog computing network efficiently.

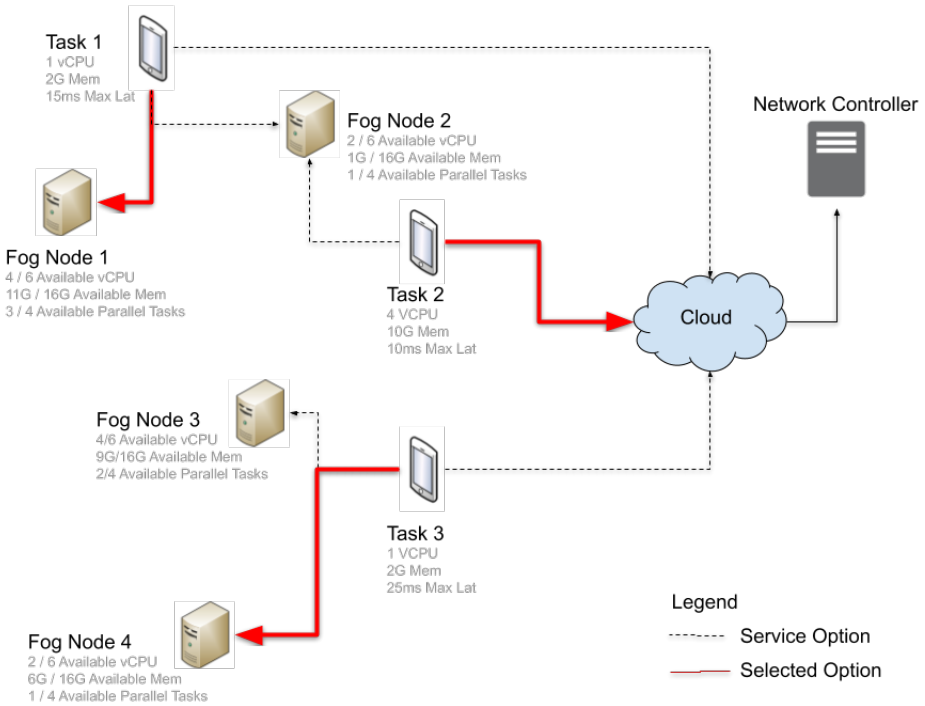

Figure 3.1: Fog Computing Network Overview.

In this thesis, we assume that a network controller, located in the cloud, is the decision making authority for task assignment and fog node status change decisions. It is not a latency sensitive function since it relies on internal processing for its decision making and sends short messages to the various network elements (clients and fog nodes) to communicate its decisions. For that reason, it can be located in the service provider's core cloud network to leverage its significant processing capability and support large scale task allocation applications.

The network controller is the only element in the fog computing network with complete state information of all active tasks and fog nodes. Therefore, the constant availability of the network controller is critical to the operation of a fog computing network. As with other service provider core network functions, the stability, scale and quantity of computing resources must be carefully designed to avoid an outage. The following considerations should be made by a service provider when provisioning a network controller within their core network.

Communication setup delay: The location and range of the network controller can

introduce extra delays in the setup of latency sensitive communications, such as cases where the network controller is far away from some fog nodes or tasks. The impact of such delays shouldn't impact the overall operation of the latency sensitive communications since the controller is not required past the communication setup stage. However, some applications of fog computing network rely on a rapid setup time and have stricter performance requirements.

Network controller service overload: The processing requirements of the network controller are relative to the size of the network it supports. Service providers should consider the expected traffic volumes in their environment when provisioning computing resources to a network controller. The impact of an overloaded controller will affect service availability and overall operation of the fog computing network. A cluster of network controllers can also be considered to improve service resilience and limit the impact of a hardware outage by only affecting a subset of the computing resources allocated to the controller function.

The client, generating one or more tasks, has no control over its assignment in the fog computing network, similarly to the association of an endpoint device to a cell in a mobile network. It is the network's role to optimize the assignment of demanded tasks to benefit the operation of the network as a complete entity. The details of the communications protocol between the network controller and network elements deemed outside of the scope of this thesis. They are assumed to already be in place.

Resource demand: Three fog node computing resources are considered at each fog node: vCPU cores, memory and parallel tasks. Other resources can be considered such as Graphics Processing Unit (GPU) resources, availability of requested content and storage, but for simplicity and without loss of generality will be excluded from this thesis. They could be included by increasing the demanded task parameters, but are of little consequence to the problem described in this thesis. Any task allocated or scheduled to a fog node must supply sufficient resources. Otherwise, the demand will be sent to the cloud.

Network demand: A single network computing resource is considered: network latency. The network latency of a task is related to the distance between its user and the allocated fog node. In some cases, a given fog node may not be able to service tasks within the maximum allowable latency. Figure 3.1 presents the maximum latency value in milliseconds (ms) that must be supplied by a fog node.

Edge-cluster: In this thesis, the notion of users, task edge-clusters and single request will be considered equally. The decision factors will be purely based on network and resource requirements. In Figure 3.1, an edge-cluster is represented by a task element (which can represent 1 or more independent tasks).

3.1.2 Assumptions

This section describes the assumptions used in this thesis.

- The locations of fog nodes are static and predetermined. In other words, this is an input to the model. Typical planning model/algorithms like the ones proposed in [58] and [59] could be used to determine the initial location of fog nodes.
- The communication protocol between the network controller and elements is assumed to be efficient and already in place.
- Each task can be assigned to a fog node or the cloud.
- The task processing handover transition time between network supply elements (i.e. fog nodes and the cloud) is significantly smaller than the time window size. Therefore, it is treated as negligible.
- Tasks and fog nodes locations are assumed to be static.
- All fog nodes and tasks use a common metric for computing power per core.
- All types of tasks can be supported by any fog node if demanded resource requirements can be supplied.
- All exact computing and network requirements are known for every task. In the theoretical model, the task start time and duration are also known.
- The variance interval of the actual latency between a task and a fog node over the duration of a task is negligible.
- In a practical network, the measured latency values would be supplied to the network controller using exact latency measurement between a given task and every fog node in the network within its communication protocol. These values

are based on the quality of a wired or wireless signal, number of network hops between the task and each fog node, types of network appliances and endpoint user-agent performance. These are all challenging to simulate without real network appliances. For this reason, this thesis uses the Cartesian distance between the task and fog nodes to simulate a latency value obtained by the network controller.

- For simplicity, all tasks use only memory for storage. The models can be easily extended to consider other storage mechanisms.
- The bandwidth between a user and its edge fog node is stable and amply sufficient to accomplish the requested tasks.
- Fog nodes have finite and known capacity and are never subject to an unplanned outage. However, they can only be allocated tasks when they are powered.
- The startup and shutdown processing time of fog nodes is equal to one time window. Of note, increasing the startup and shutdown time would not increase the complexity of the algorithm, but would make results more difficult to visualize. This was the reason for making this design choice.
- The cloud has unlimited capacity. If a task cannot be processed by the fog, it will be serviced by the cloud.
- In comparison to the fog nodes, the cloud is located far away from the users.
- For simplicity in performance comparison, profit is generated only when tasks are assigned to a fog resource. Assignment of tasks to the cloud provides no profit in this thesis. Assigning a null profit value to cloud assignment is not a practically realistic decision since assigning tasks to cloud should also generate revenue. However, it provides a clear indicator to cloud service providers on the potential value of migrating to a fog network. The performance metric used in this thesis is the relative difference in profit.
- The relationship between the controller, tasks and fog nodes are not considered latency sensitive communications. These controller messages are out of band from the latency sensitive communications of the fog computing network. The messages to assign tasks to a fog node or cloud and fog node state change messages are considered small and of no significant impact to network resources.

3.2 Theoretical Model Description

The following notation used to model the problem formulation described in this section. The notation is divided into 3 sections: input data sets, mathematical model and output data sets. They are linked together as demonstrated in Figure 3.2.

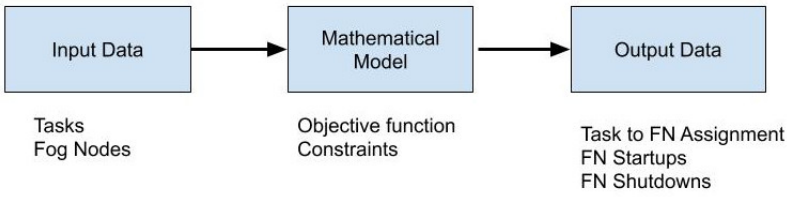

Figure 3.2: Notation Flow.

3.2.1 Input Data

3.2.1.1 Sets

$I = (i_0(\eta_0, \zeta_0, \gamma_0, \alpha_0, \psi_0, \chi_0), \ldots, i_n(\eta_n, \zeta_n, \gamma_n, \alpha_n, \psi_n, \chi_n))$ is the set of tasks requesting service from the fog computing network. Each task requests vCPU, memory and link bandwidth resources from the fog network and sets an upper bound limit on latency. Each task $i \in I$ is defined with the following parameters.

- η_i is the minimum number of vCPU cores requested by task i.
- ζ_i is the minimum memory, in Gigabytes (GB), demanded by task i.
- γ_i is a two-tuple value corresponding to the latitude and longitude coordinates of the task request i.
- α_i is the maximum distance, in meters to its fog node (server), demanded by i.
- ψ_i is the duration in number of time windows required to complete task i.
- χ_i is the start time of task i.

$J = (j_0(\upsilon_0, \epsilon_0, \nu_0, \delta_0), ..., j_m(\upsilon_m, \epsilon_m, \nu_m, \delta_m))$ is the set of fog nodes located in the fog network. Each fog node can supply vCPU and memory to accomplish demanded tasks. Each fog node $j \in J$ is defined with the following parameters.

- υ_j is the maximum number of vCPU cores that can be allocated to tasks by fog node j.
- ϵ_j is the maximum memory (in GB) that can be allocated to tasks by fog node j..
- ν_j is the maximum number of parallel tasks that fog node j can handle (in units).
- δ_j is a two-tuple value corresponding to the latitude and longitude coordinates of fog node j.

$T \in (t_0, t_1, t_2, ..t_k)$ is the set of time windows. It contains a set of iterative values between 0 and k where k is the number of time windows required to complete all the processing in a simulation. It is used to simulate billing from a network provider. Tasks that are started and stopped within a time window are being billed for the complete time unit.

3.2.1.2 Constants

The following values are constants set by the fog computing network owner to highlight their costs and relative revenue in the model.

- cu, constant value indicating a baseline for up-front costs (in currency units) of changing the state of a fog node from down to up.
- cd, constant value indicating a baseline for up-front costs (in currency units) of changing the state of a fog node from up to down.
- mn, maintenance cost (in currency units) of a fog node to remain active in dollars per time window.
- r, relative revenue (in currency units) per time window of assigning a task to a fog node rather than the cloud.

3.2.1.3 Decision variables

The following variable values are determined in the solution for the linear programming model defined in Section 3.2.2 for a given the sets of tasks I and fog nodes J.

- x_{ijt}, a binary variable indicating if task i is connected to a fog node j at time window t.
- θ_{jt}, a binary variable used to capture the state of fog node j at time window t (1 is on and 0 is off).
- μ_{jt}, a binary variable used to capture a fog node startup at time window t (1 is starting up and 0 is not).
- ϕ_{jt}, a binary variable used to capture the fog node shutdown at time window t (1 is shutting down and 0 is not).

The fog node state variables (θ_{jt}, μ_{jt} and ϕ_{jt}) were divided into 3 variables to allow for the objective function to apply maintenance costs to each state change operation in the network. These costs can be individually configured to mimic the infrastructure costs of a service provider. It also allows the model to easily increase or decrease the number of time windows required for powering up or down fog nodes. They also serve to simplify the formulation of the constraints, reduces the complexity in computation by only manipulating binary variables and facilitates result visualization.

3.2.2 Mathematical Model

As a general statement, iterations within the mathematical model will use the I variable for the set of tasks, J for the set of fog nodes and T for the set of time windows. The set of tasks, fog nodes and time window sizes are pre-processed during the input data sets generation module of the model (see Figure 3.2).

Time windows are used to simulate billing from a network provider. Tasks that are started and stopped within a time window are being billed for the complete time unit. This is used to simplify calculations and will later allow the heuristic model to group the allocation and de-allocation of tasks without impact to the profit bottom line. This approximation should be negligible in cases that the time window intervals are short relative to the average duration of tasks. The time window size values in this thesis are set to 2 seconds, but they can be configured to an advantageous value for a given application.

3.2.2.1 Objective function

The objective function, shown in Equation 3.1, tries to maximize the operating profit of the fog network relative to a cloud only architecture. It must operate on historical data sets (i.e. all requests are known in advance) requested for assignment to the fog computing network. Therefore, the resolution of the mathematical objective function serves to determine the profit upper bound to evaluate the performance of heuristic models that executes its network operations in real-time (i.e. some information of the task set I is hidden from the heuristic model).

Equation 3.1 provides the maximization of profit by calculating the quantity of time windows where tasks are assigned to a fog and minimizing operating costs. In the equation below, the first term $\left(r \sum_{j \in J} \sum_{i \in I} \sum_{t \in T} x_{ijt}\right)$ represents the relative revenue of assigning each task to a fog node for a time window. The second term $\left(mn \sum_{j \in J} \sum_{t \in T} \theta_{jt}\right)$ represents the costs of operating all active fog nodes. The third term $\left(cu \sum_{j \in J} \sum_{t \in T} \mu_{jt}\right)$ represents the startup costs of starting up fog nodes at each time window. Finally, the fourth term $\left(cd \sum_{j \in J} \sum_{t \in T} \phi_{jt}\right)$ represents the costs of shutting down fog nodes.

$$max \left(r \sum_{j \in J} \sum_{i \in I} \sum_{t \in T} x_{ijt} - mn \sum_{j \in J} \sum_{t \in T} \theta_{jt} - cu \sum_{j \in J} \sum_{t \in T} \mu_{jt} - cd \sum_{j \in J} \sum_{t \in T} \phi_{jt} \right) \tag{3.1}$$

Tasks that are not assigned to a fog (i.e. $\left(\sum_{j \in J} x_{ijt} = 0\right)$) are sent to the cloud. In cases where a task is sent to the cloud, the improvement in profit is 0 for task i at time window t. This allows for any comparison tests to use 0 as the threshold to determine whether it is profitable to operate a fog network rather than a cloud only network.

3.2.2.2 Constraints

Following are the 12 categories of constraints used in the optimization problem. The specific number of constraints depends on the sizes for the set of tasks I, fog nodes J and time windows T.

Fog node status loop prevention constraints: Equation 3.2 ensures that each fog node can only be in one of 4 valid states:

- Powered off: $\theta_{jt} = 0$, $\mu_{jt} = 0$, $\phi_{jt} = 0$
- Powered on: $\theta_{jt} = 1$, $\mu_{jt} = 0$, $\phi_{jt} = 0$
- Starting up: $\theta_{jt} = 0$, $\mu_{jt} = 1$, $\phi_{jt} = 0$
- Shutting down: $\theta_{jt} = 0$, $\mu_{jt} = 0$, $\phi_{jt} = 1$

$$\mu_{jt} + \phi_{jt} + \theta_{jt} \leq 1 \quad (j \in J, t \in T) \tag{3.2}$$

Fog node status verification constraints: Equation 3.3 ensures that tasks can not be assigned to a fog node that is powered off.

$$\sum_{i \in I} x_{ijt} \leq \sum_{i \in I} x_{ijt}\theta_{jt} \quad (j \in J, t \in T) \tag{3.3}$$

vCPU capacity constraints: Equation 3.4 ensures that each fog node has sufficient vCPU capacity for its assigned tasks.

$$\sum_{i \in I} x_{ijt}\eta_i \leq \upsilon_j \quad (j \in J, t \in T) \tag{3.4}$$

Memory capacity constraints: Equation 3.5 ensures that each fog node has sufficient memory capacity for its assigned tasks.

$$\sum_{i \in I} x_{ijt}\zeta_i \leq \epsilon_j \quad (j \in J, t \in T) \tag{3.5}$$

Parallel tasks capacity constraints: Equation 3.6 ensures that each fog node has sufficient parallel task capacity for its assigned tasks.

$$\sum_{i \in I} x_{ijt} \leq \phi_j \quad (j \in J, t \in T) \tag{3.6}$$

Task single assignment constraints: Equation 3.7 ensures that a task can be assigned to a maximum of 1 fog node.

$$\sum_{j \in J} x_{ijt} \leq 1 \quad (i \in I,\, t \in T) \tag{3.7}$$

Latency verification constraints: Equation 3.8 ensures that the distance between task and fog node does not exceed its maximum allowable distance.

$$x_{ijt}[(\gamma_i(x) - \delta_j(x))^2 + (\gamma_i(y) - \delta_j(y))^2] \leq \alpha^2 \quad (i \in I,\, j \in J,\, t \in T) \tag{3.8}$$

Task start time check constraints: Equation 3.9 ensures that no task is mapped to a fog node before its start time.

$$\sum_{j \in J} x_{ijt} \leq 0 \quad (i \in I,\, t \leq \chi_i \in T) \tag{3.9}$$

Task stop time constraints: Equation 3.10 ensures that no task is mapped to a fog node after it is finished.

$$\sum_{j \in J} x_{ijt} \leq 0 \quad (i \in I,\, t \geq (\chi_i + \psi_i) \in T) \tag{3.10}$$

Starting up transition constraints: Equation 3.11 calculates the required values for all startup variables μ_{jt}.

$$\mu_{jt} + \theta_{jt} \geq \theta_{j(t+1)} \quad (j \in J,\, t \in T) \tag{3.11}$$

Powered on transition constraints: Equation 3.12 calculates the required values for all the shutdown variables value ϕ_{jt}.

$$\phi_{jt} + \theta_{jt} \geq \theta_{j(t-1)} \quad (j \in J,\, t \in T) \tag{3.12}$$

Variable bound constraints: Equations 3.13, 3.14, 3.15 and 3.16 determine the variable bounds for all x_{ijt}, θ_{jt}, μ_{jt}, and ϕ_{jt}.

$$x_{ijt} \in \{0, 1\} \quad (i \in I,\, j \in J,\, t \in T) \tag{3.13}$$

$$\theta_{jt} \in \{0, 1\} \quad (j \in J, t \in T) \tag{3.14}$$

$$\mu_{jt} \in \{0, 1\} \quad (j \in J, t \in T) \tag{3.15}$$

$$\phi_{jt} \in \{0, 1\} \quad (j \in J, t \in T) \tag{3.16}$$

3.2.3 Output Data

The output data is the calculated optimal values for the decision variable and objective function (profit) which represent the various decisions made by the theoretical model in its optimization.

3.3 Heuristic Model Description

The theoretical model presented in Section 3.2 can only be applied in historical task allocation applications where all values of the task set I are known. The development of a heuristic model is a necessary step in any real-time application of the concepts explored previously if only for the lack of knowledge of future input set parameter values. There are also many applications that are too complex for the theoretic model to reach a solution in polynomial time due to the number of decision variables and constraints. This further justifies the requirement for a heuristic model to solve problems with a large number of tasks and / or fog nodes.

Section 3.3.1 will detail the heuristic model differences from the theoretic model. Section 3.3.2 will provide the general heuristic framework of the model.

3.3.1 Changes from Theoretical Model

3.3.1.1 Changes to Input Data

The input parameter values described in Section 3.2.1 can remain unchanged for a real-time heuristic model. However, two input handling limitations must be added to ensure that the set of tasks I is modified for real-time assignment.

Task Start Time Limitation: It ensures that each new task i is revealed to the model as a new request based on its start time.

Task Duration Time Limitation: It ensures that the duration parameter of each task i is revealed once a task makes a stop request to the model.

3.3.1.2 Task Start and End Time

The theoretical model constraints 3.10 and 3.11 for task start and stop time are no longer required in the heuristic model. Tasks that have ended can safely be removed from the input data set. The task input data set I will behave as a rolling window containing the number of active tasks at time window t. This should assist the model in maintaining a similar processing complexity independently of the total run time of the model. This is considered a requirement of the heuristic model since it is designed to provide real-time task assignment.

3.3.1.3 Fog Node State Changes

Unlike the theoretical model, the heuristic model must make a determination on active task assignment at every $t \in T$. When the theoretical model makes an assignment decision, it can not be modified at a later time window. The heuristic model only impacts the current time window in its operations. This is of significant importance to achieving good results since the fog node start-up and shut-down decisions are only made from the state of the fog computing network at time window t and its past trends. This creates a larger duality between the green computing and service provision requirements. A significant consideration must be made to determine task request trends within the heuristic to account to configure the decision point parameters for fog node state changes to optimize profit values.

3.3.2 Implementation Framework

A general overview of the heuristic model framework is shown in Figure 3.3. It is designed to perform task allocation and de-allocation determinations at every time window as required in a real-time allocation application. During operation, the framework accounts for it by looping the allocation group and de-allocation group processes over the total number of time windows in its operating environment. These groups

include all tasks that must be processed within a time window t. All processes within the loop must be completed under the time window size to ensure proper operation.

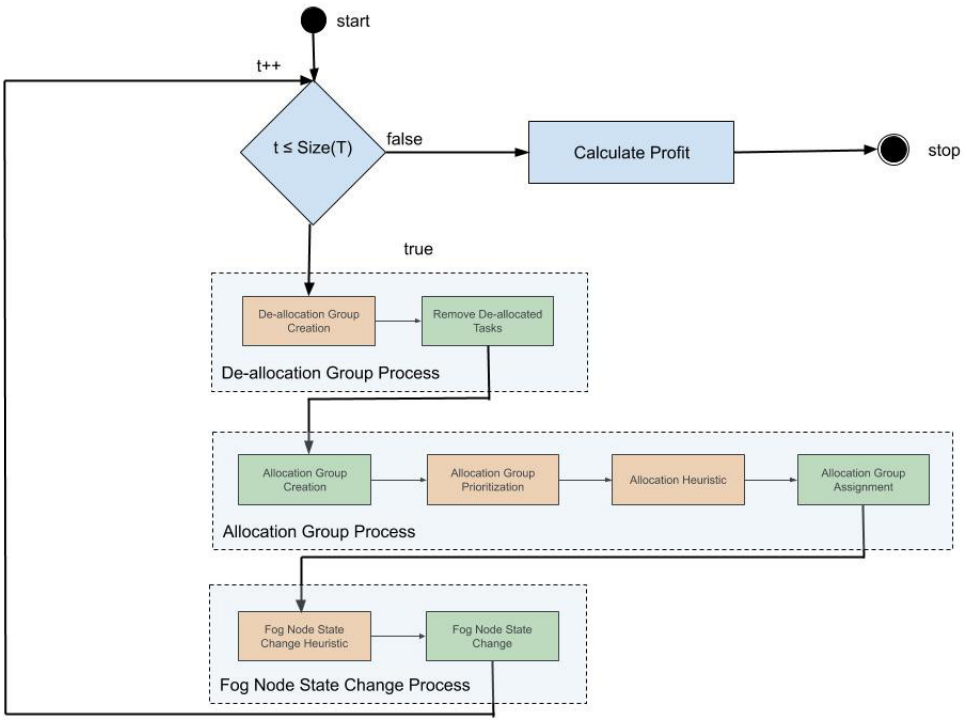

Figure 3.3: Heuristic Model General Framework Flow

The heuristic algorithm life-cycle must be finite (either actual or artificial) to enable a comparison with the theoretic model to evaluate performance. In a real world application of this model, the run time of the algorithm would be very long. In those cases, the benefits of having a larger historical knowledge of the operating environment should provide a performance improvement to the heuristic model. The heuristic model will use various parameter configuration values to improve its decision making.

The following paragraphs will detail the process groups shown in the heuristic framework. The figures in this section (3.3, 3.4, 3.5 and 3.6) are categorized in 2 groups: the fixed functions (coloured in green) do not contain configurable values while the tailored functions (coloured in orange) contain values that can be customized to improve performance. All algorithms implemented for tailored functions will be detailed in their respective sections. They use the same mathematical notation as described in

Section 3.2 including the modifications explained in Section 3.3.1. Any local variables used in the algorithm will be defined in their respective sections.

3.3.2.1 De-allocation Group Process

The de-allocation of tasks is the simplest process in the heuristic model. As shown in Figure 3.4, there are 2 primary functions in the process: the creation of the de-allocation group and its removal from its assigned allocation.

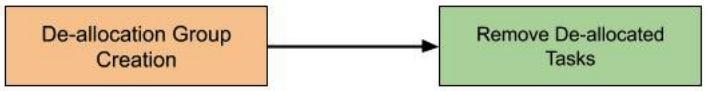

Figure 3.4: Heuristic De-allocation Group Process Framework

De-allocation Group Creation
The de-allocation group creation process lists the tasks that are tagged for de-allocation in the current time window. Algorithm 1 adds any completed tasks to the de-allocation data set D. The heuristic model will only populate the duration parameter ψ_i if a completed request is made to the fog computing network.

Algorithm 1: Add Completed Tasks to the De-Allocation Group

```
for i ∈ I do
    if ψ_i  ∃  /* When task i is completed */
    then
        D ← i  /* Add task i to the de-allocation set */
    end
end
```

Tasks that have completed previously are obviously included in the de-allocation group since they offer no benefit to being assigned to the fog computing network. However, task completion can be predicted and used for advance de-allocation. This can be notably beneficial in scenarios where operating costs could be reduced to outweigh the revenue generation of having it mapped to the fog node. The performance of advance de-allocation is dependent on the predictability of task duration and its benefits are also based on available resources in the fog computing network.

Algorithm 2 will add a negative weight to its assignment prioritization for tasks that are predicted to complete. The following local variables are used:

- $PredDeallocWeight$ is the weight that the de-allocation algorithm will have on the allocation prioritization as configured. It is likely a high value since it is almost always valuable to assign nearly completed tasks to the cloud if there is a shortage of fog resources.
- $PredDur$ is the duration threshold of tasks to trigger the preemptive de-prioritization of tasks as configured. This thesis configured the $PredDur$ variable to be equal to the average of all completed tasks in a simulation. This decision is based on the network generation methodology used in Chapter 4 where tasks are divided into n categories all averaging the same duration. However, there are many other methods to determine the value for this variable and all depend on the statistical patterns of the task allocation used in network simulations.
- w_i is the prioritization weight vector with its indices corresponding to each task $i \in I$. Once the prioritization processes complete, the vector will be ordered in descending order (i.e. higher weight tasks will be processed first).

Algorithm 2: Preemptive De-Prioritization of Tasks

```
for i ∈ I do
    w_i ← 0
    /* Comparison of the current duration of task i with its
       expected duration */
    if PredDur − (t − χ_i) ≥ 0 then
        /* Adjust the prioritization weight value */
        w_i ← w_i − PredDeallocWeight
    end
end
```

Remove De-Allocated Tasks

The removal from its assigned fog node is a process that simply removes tasks from the active task set to the completed task set.

3.3.2.2 Allocation Group Process

The allocation of tasks is a complex process in the heuristic model. As shown in Figure 3.5, there are 4 sequential functions in the process: allocation group creation, allocation group prioritization, allocation heuristics and their assignment to the network.

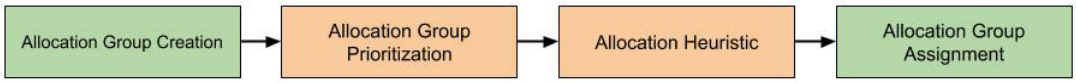

Figure 3.5: Heuristic Allocation Group Process Framework

Allocation Group Creation
Contrary to the de-allocation group process, the creation of the allocation group is a trivial process. Any task in the input data set I with a $\chi_i \leq t$ is added to the allocation group. Fortunately, the de-allocation group process removes any completed task from the input data set I removing any requirement for the allocation group process to track task completion.

The heuristic model generates a new allocation group at each time window t. The heuristic model will reassign all tasks at every time window and uses its prioritization mechanic to influence the network to reassign tasks where appropriate.

Allocation Group Prioritization
The allocation group prioritization function is used to establish the processing order within the allocation group. This is an important preliminary step to the heuristic since task processing order can impact allocation performance. The allocation group tasks include any tasks that have a $\chi_i \leq t$ and is built by including any task $i \in I$ that satisfy the constraint. A complementary weight data set W is added with values for every w_i for all $i \in I$.

The allocation group is ordered based on a weight parameter included in each of its member. Algorithm 2 already adds a negative weight value to tasks that are nearly completed and should first be relegated to the cloud in cases of insufficient resources within the fog computing network. Algorithm 3 adds additional verification mechanics to better prioritize the allocation group.

Firstly, lines 2 to 11 are used as a latency verification to determine how many active fog nodes can service each task. They are ordered and weighted in ascending order with the tasks with fewer options prioritized higher. The number of available fog nodes weight parameter is a configurable value for the heuristic model.

Secondly, line 12 weighs the tasks based on their age, prioritizing tasks that are newer for assignment to the network. The age weight parameter is a configurable value for the heuristic model.

Algorithm 3: Allocation Group Prioritization

```
1  for i ∈ I  /* Loop for the whole active task set */
2  do
3      NbFnAvail ← 0
4      for j ∈ J  /* Loop for all powered on fog nodes */
5      do
6          if θ_j ≥ 1 then
               /* Latency calculation between task i and fog node j
               */
7              Latency = √((γ_i(x) − δ_j(x))² + (γ_i(y) − δ_j(y))²)
               /* Increment the candidate fog node counter if latency
                  is below the task demand */
8              if Latency ≤ α_i then
9                  NbFnAvail++
10             end
11         end
12     end
       /* Adjust the weight based on number of candidate fog nodes
          */
13     w_i ← w_i − NbFnAvail.NbFnWeight
       /* Adjust the weight based on the age of task i */
14     w_i ← w_i − TaskAgeWeight(t − χ_i)
15 end
```

The weight factors used in Algorithms 2 and 3 are configured to optimize performance based on how close the heuristic model performs compared to the upper bound for various values. This is done by measuring past time window assignments to the

theoretical model (exact upper bound) or by using a heuristic algorithm to determine optimal weight factor values of past task assignments (estimated upper bound). The heuristic algorithms tested are identical to the task assignment heuristics that will be described in the following paragraphs.

Allocation Heuristic
The allocation heuristic algorithm is used to assign tasks to the fog computing network. Each task in the allocation group is selected in its prioritized order and allocated to the fog computing network using a best fit heuristic where the fog node in range with the least available resources is selected. This algorithm was chosen because it required little processing time and provided identical results to the theoretic model in nearly all tests. The best fit heuristic matched the theoretical upper bound in most tests when removing other factors from the model such as fog node status changes. Chapter 4 will provide details of those results.

Numerous other methods such as: Tabu Search [62], Swarm Intelligence [63], Simulated Annealing [64], Genetic Algorithms [65], Artificial Neural Networks [66] and Support Vector Machines [67] were considered. However, the design decision of having a task prioritization operation simplified the type of operations needed and had no impact to the resulting profit. Essentially, the task prioritization algorithm ensures that tasks are ordered optimally trivializing the actual allocation heuristic. For the above reasons, these methods were not chosen.

Allocation Group Assignment
Finally, the allocation group assignment will use the allocation heuristic output and perform the task assignment on the fog computing network and update its remaining resources.

3.3.2.3 Fog Node State Change Process

The fog node state change process is the set of functions used to modify the state of fog nodes in the heuristic model. It is executed as the final process in each time window to allow the model to include the current time window in its decision process. Fog nodes that are starting up in time window t will only be available to the fog network in time window $t + 1$. Any status changes that could benefit the current time window are unfortunately not possible based on that constraint. Figure 3.6 shows its 2 functions: the fog node state change identification heuristic and applying fog node state changes.

Figure 3.6: Heuristic Fog Node State Change Process Framework

Fog Node State Change Heuristic
The fog node state change heuristic is a complex process and relies to varying levels on the correlation of historic to future tasks based on its configuration. The network status change frequency is determined by the level of fog node coverage in a geographical area coupled with predicted network demand.

Every status change algorithm relies on the heuristic models ability to predict task demand and detecting where network coverage is not optimal. To simulate future tasks, the status change heuristics generate a set of simulated tasks to artificially assign to the fog network using the heuristic assignment group process. The number of tasks in this set is configured in the model. The higher the number of simulated tasks, the higher the processing time, but the closer its statistical distribution will match tasks seen by the heuristic model. Simulated tasks are generated based on a bootstrap of historic tasks observed by the network. The selected task is slightly modified using a normally distributed random variable to skew the parameter values. The random variable uses μ as the bootstrap parameter value and σ as the standard deviation of each parameter based on all historic tasks.

Algorithm 4 provides an overview of the operations needed to initiate the shutdown of a fog node. In summary, it aims to determine situations where the operating cost is higher than the revenue of its critical tasks and predicted demand for future tasks that could only be serviced by the candidate fog node. Tasks are generated from the active and completed task sets to simulate future tasks.

The *ShutThreshold* is a configured value to determine the minimum probability of a task being assigned to the fog node j before it is believed to be valuable to power it off. It also requires the following local variables that are calculated based on past task demand on the fog computing network.

- $\overline{\alpha}$: Average minimum latency demand of historic tasks.

- $\overline{\eta}$: Average vCPU demand of historic tasks.
- $\overline{\zeta}$: Average memory demand of historic tasks.
- $\overline{i}$: Average tasks per time window based demand of historic tasks.

Algorithm 4: Fog Node Shutdown Algorithm

```
/* Calculate the quantity of time windows to simulate */
N ≥ (cu + cd) / mn
for j ∈ J where θ_j ≥ 1 do
    Perform the artificial reassignment of any task assigned to j to another
     fog node using the temporary variable x*_ijt.
    /* Verify if the artificial reassignment is profitable */
    if (Profit − Profit*) ≤ mn then
        for j* ∈ J where (j* ≠ j) do
            if θ_j* ≥ 1 then
                /* Check the fog node j* available resource capacity
                   */
                AvailableCPU = υ_jt − Σ_{i=0}^{I} x_ijt.η_i
                AvailableMem = ϵ_jt − Σ_{i=0}^{I} x_ijt.ζ_i
                AvailableTasks = ν_jt − Σ_{i=0}^{I} x_ijt
                /* Determine the bottleneck resource for each fog
                   node j* expressed as the capacity in number of
                   supported simulated tasks */
                AvailableRes_j =
                  min(AvailableCPU / η̄, AvailableMem / ζ̄, AvailableTasks)
            end
            Assign all simulated tasks for N future time windows to x_ijt
            Calculate the probability (P_shut) where keeping fog node j powered
             on will generate more profit than with it shutdown for N time
             windows using simulated tasks and the AvailableRes values.
        end
    end
    // Verify the shutdown probability against the threshold
    if P_shut ≥ ShutThreshold then
        φ_jt = 1
    end
end
```

The algorithm begins by attempting to temporarily reassign any real and simulated tasks assigned to each fog node by running the allocation algorithm to determine its critical tasks. Situations where the revenue generated by a fog node critical tasks is smaller than the maintenance cost is the trigger used to determine candidate fog nodes for shutdown. The quantity of future time windows to consider is determined by making a calculation point of equality between the number of time windows and its operating cost mn and the shutdown cd and start-up costs cu.

For every candidate fog node, the algorithm iterates over neighboring fog nodes with an overlap in geographic coverage area based on $\overline{\alpha}$. It determines their limiting resource and translates it into a number of tasks that could be serviced by the other fog nodes. The algorithm also calculates the probability of those neighboring fog nodes to become above capacity in future time windows using simulated tasks. This design is made to ensure that a minimum level of available resources are kept in the fog network before shutting down a candidate.

The final probability value is compared with the $ShutThreshold$ to determine whether it should be powered off.

Algorithm 5 shows the opposing process to determine if powered off fog nodes should be started to enhance the network. Similarly to the previous algorithm, it seeks to determine the probability that a task is requested in future time window that could not be serviced by the current fog node coverage. The resulting probability is compared with a configured threshold to determine whether a fog node should be started.

Algorithm 5: Fog Node Startup Algorithm

```
/* Calculate the quantity of time windows to simulate */
```
1 $N \geq \frac{cu - r}{mn}$

2 **for** $j \in J$ *where* $\theta_{jt} \leq 0$ **do**

3 Perform the artificial reassignment of any task assigned to j to another fog node using the temporary variable x^*_{ijt}.

4 **for** $j^* \in J$ *where* $\theta^*_{jt} \geq 1$ **do**

```
/* Check the fog node j* available resource capacity */
```
5 $AvailableCPU = \upsilon_{jt} - \sum_{i=0}^{I} x_{ijt}.\eta_i$

6 $AvailableMem = \epsilon_{jt} - \sum_{i=0}^{I} x_{ijt}.\zeta_i$

7 $AvailableTasks = \nu_{jt} - \sum_{i=0}^{I} x_{ijt}$

```
/* Determine the bottleneck resource for each fog node j*
   expressed as the capacity in number of supported
   simulated tasks */
```
8 $AvailableRes_j = min(\frac{AvailableCPU}{\overline{\eta}}, \frac{AvailableMem}{\overline{\zeta}}, AvailableTasks)$

9 **end**

10 Assign all simulated tasks for N future time windows to x_{ijt}

11 Calculate the probability (P_{start}) where keeping fog node j powered off will generate more profit than with it powered on for N time windows using simulated tasks and the *AvailableRes* values.

```
/* Verify the shutdown probability against the threshold */
```
12 **if** $P_{start} \geq StartThreshold$ **then**

13 $\mu_{jt} = 1$

14 **end**

15 **end**

Fog Node State Change

The application of the status change uses the results ϕ_{jt} and μ_{jt} of the previous functions to determine the status change operations to complete.

3.3.2.4 Calculate Profit Process

The calculate profit process implements the same objective function as the theoretic model. The heuristic model simply tracks its task assignment and fog node status operations during each time window and uses this information to calculate the profit using Equation 3.1.

3.4 Chapter Summary

In this chapter, we first presented a mathematical formulation of the task allocation problem in Section 3.2. As discussed, the mathematical model can only be applied to task allocation problems with complete information (i.e. historic problems) and uses static input sets to ensure the maximization of the objective function. Then, a heuristic model was presented in Section 3.3 for the more realistic use case of real-time task allocation using dynamic input sets notably for the task demand and duration evolving over time windows. In future experiments, the theoretical model will serve as a benchmark to determine the upper bound of a task allocation problem. It will be used to evaluate the performance of the heuristic model and its tuning parameters.

Chapter 4

Results and Analysis

This chapter contains the results and analysis of each task assignment model described in Chapter 3. Notably, this chapter will present the results obtained in the testing of each model using various sets of tasks and fog nodes.

Firstly, Section 4.1 will present the simulated environment used to generate the various results of this chapter. Secondly, Section 4.2 validates each constraint of the theoretical model implementation described in Section 3.2.2.2 and characterizes the computing time required to solve networks of various sizes. Thirdly, Section 4.3 provides the results and testing used to determine the optimal values for the heuristic model parameters described in Chapter 3. Fourthly, Section 4.4 provides results and analysis of the performance of the heuristic model. Finally, Section 4.5 summarizes the chapter.

4.1 Simulation Environment

The simulation environment used in all generation of the results in this chapter use a Unix Operating System with the following processing specifications:

- Processor: 2.66 GHz Intel Core i7
- Memory: 8 GB 1067 MHz DDR3
- Storage: 256 GB SSD

The theoretical model was implemented in CPLEX and run using the IBM iLog CPLEX Optimization Studio version 12.10 (OPL) command line interface (CLI).

The CLI performed notably better than the graphical user interface (GUI) in terms of processing speed and ability to automate the execution of the theoretical model.

The heuristic model was developed using Matlab 2019b. The results from the algorithm populate text files that are written to disk as they are generated in the algorithm. Using this environment, service providers could use the text file outputs of the heuristic model as inputs to a fog computing network controller server.

Figure 4.1 shows an overview of the main modules implemented that can directly impact the results presented in this chapter.

- The network generation module is used to create the sets of tasks and fog nodes to process the theoretic and heuristic models to achieve results. The methods used to create the various networks are detailed in their respective subsections.
- The implementation of the theoretic model presented in Section 3.2 solved with CPLEX. It provides the theoretical upper bound profit to any generated network. Larger networks will require additional processing time since the resulting optimization problem size grows exponentially with the network size.
- The heuristic model has 2 main modules that can impact results. The complete design details of the heuristic model are presented in Section 3.3.
  - The task order manipulation module is composed of the allocation and deallocation group processes. It determines the priority of each active task to submit them in sequence to a best fit assignment heuristic.
  - The fog node status manipulation module uses various techniques to modify the states of each fog node in the network to best service the current task load.

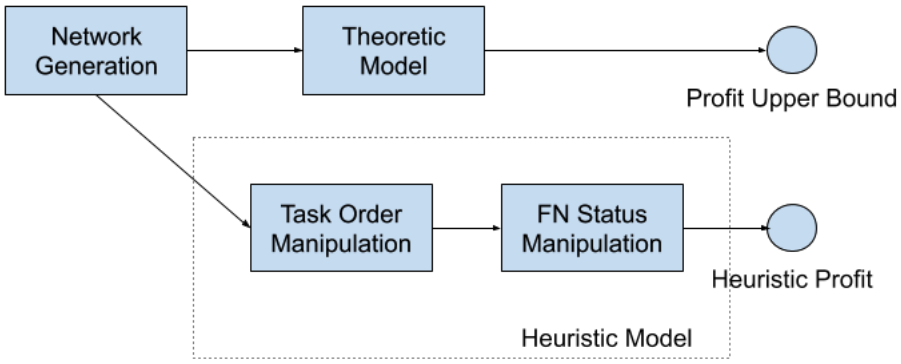

Figure 4.1: Overview of the Theoretic and Heuristic Modules

4.2 Validation of the Theoretical Model

The validation of the theoretic model presented in Section 3.2 was accomplished by creating a test environment that trivialized all constraints except for one. This allows for the theoretical model to artificially remove all but one type of constraints in the problem, making it trivial to determine if the model operates as expected.

Table 4.1 below provides initial values for the revenue and maintenance costs of the optimization function. These will be used for each validation described below unless explicitly specified otherwise. They are used to calculate the solution to the optimization function as described in Section 3.2. All other values needed to build the network are described in their respective subsection. The time window size is not a necessary parameter to generate a result in this model because only task density per time window will impact task assignment. The size of a time window is useful in a practical scenario since it determines the maximum processing time of the algorithm for each time window. However, this will only be a factor in Sections 4.3 and 4.4 when the heuristic model is considered. Therefore, for simplicity, time windows were normalized to be incremental in this section.

Table 4.1: Fixed Cost and Revenue Values (in currency units)

Parameter	**Value**
Maintenance costs (mn)	0.1
Startup costs (cu)	0.5
Shutdown costs (cd)	0.2
Revenue (r)	1

The theoretical model runs until all elements of the task set I have completed. As illustrated in Table 4.1, the cost of shutting down a fog node is double its maintenance cost. Therefore, a fog node should only be shutdown in cases where it is not in use for at least 2 time windows. Many validations presented in the following sections will present that fog nodes remain powered on during the last time window even when no tasks are assigned to it. This behaviour is consistent with the theoretically optimal solution to the optimization problem.

The following subsections detail the series of tests completed to validate each of the theoretical model constraint groups. The tests were designed to trivialize many constraints within the model to simplify the visualization of its behaviour across different input types. The combination of these tests are designed to ensure that the model reliably operates to provide an optimal solution to the problem described in Chapter 3. The final subsection offers an overview of the theoretical model complexity, notably in terms of its processing time. For this section, all detailed results are included in Appendix A.

4.2.1 Fog Nodes Status Change Validation

Equation 3.2 from Section 3.2 ensures that fog nodes can only be in a valid state as described in that section. The objective of this validation is to ensure that fog nodes can be set in each of the fog node states and that there are no scenarios where a fog node is in an invalid state.

For this validation, elements in the set of tasks I are co-located and the start time and duration parameters ensure that the tasks do not overlap. It would be possible to have some tasks overlap, but it would complicate the test case without providing

any advantages to better validate the constraints. Table 4.2 presents the set of tasks used in this validation. Of note, a task duration of 1 as shown in Table 4.2 indicates that the task starts and ends in the same time window.

Table 4.2: Constraint 1 Validation Task Table

i	Start time χ_i (windows)	Location γ_i	Max latency α_i (meters)	vCPU η_i (cores)	Memory ζ_i (GB)	Duration ψ_i (windows)
1	0	(0, 0)	40	1	1	1
2	1	(0, 0)	15	1	1	1
3	2	(0, 0)	15	1	1	1
4	3	(0, 0)	40	1	1	1
5	4	(0, 0)	30	1	1	1

The fog node set J for this validation is a single element that is geographically within range of all tasks and with sufficient resource supply to support any of the tasks. The starting state of the fog node is set to 0 (powered off) in order to validate the transition from a powered off fog node to a powered on fog node. Table 4.3 shows the fog node set J used in this validation.

Table 4.3: Constraint 1 Validation Fog Node Table

j	Location δ_j	vCPU ν_j (cores)	Memory υ_j (GB)	Parallel tasks ϵ_j (units)	State
1	(0, 0)	3	2	4	0

The result of the testing show that time window 0 is used to set the startup variable μ_{10} to 1 for the only fog node in the network. Since the fog node is in a startup state, no tasks can be assigned to it. Task 1 variable values (x_{1jt}) are never assigned to a fog node. Therefore, it is assigned to the cloud by having a value of 0 for all $j \in x_{1jt}$. All other tasks are assigned to fog node 1 since it is in a powered on state for all other time windows.

If task 1 was requested at another time window, it would have been assigned to a fog node rather than the cloud. In fact, the theoretical model can perfectly predict whether a fog node is required at a later time window and run its startup state in anticipation. The only exception is for tasks that are requested at time window 0

where the model uses 0 as an absolute starting point (i.e. there are no time windows considered before 0).

4.2.2 Tasks are Only Assigned to Powered On Fog Nodes

Equation 3.3 ensures that tasks can only be assigned to a fog node that is powered on. The objective of this validation is to ensure that a fog node is first powered on before being assigned one or more tasks.

For this validation, elements in the set of tasks I are all located at a large distance from on another with an identical start time and duration. The duration of 2 time windows for each task should allow for tasks to be assigned to a fog node (if possible) in time window 1, but not 0. Table 4.4 shows the set of tasks I used for the validation.

Table 4.4: Constraint 2 Validation Task Table

i	Start time χ_i (windows)	Location γ_i	Max latency α_i (meters)	vCPU η_i (cores)	Memory ζ_i (GB)	Duration ψ_i (windows)
1	0	(1000, 0)	20	1	1	2
2	0	(2000, 0)	20	1	1	2
3	0	(3000, 0)	15	1	1	2
4	0	(4000, 0)	20	1	1	2
5	0	(5000, 0)	30	1	1	2

The fog node set J for this validation contains the same number of elements as the task set I. Each fog node has exactly 1 element that is located within range of all tasks and with sufficient resource supply to support any of the tasks. The starting state of the fog node is set to 0 (powered off) in order to validate the transition from a powered off fog node to a powered on fog node. Table 4.5 shows the fog node set J used in this validation.

Table 4.5: Constraint 2 Validation Fog Node Table

j	Location δ_j	vCPU ν_j (cores)	Memory υ_j (GB)	Parallel tasks ϵ_j (units)	State
1	(1000, 0)	3	2	4	0
2	(2000, 0)	3	2	4	0
3	(3000, 0)	3	2	4	0
4	(4000, 0)	3	2	4	0
5	(5000, 0)	3	2	4	0

As designed, time window 0 is used to set each startup variable μ_{j0} to 1 for every fog node element in the set J. Although each fog node has sufficient resources to support up to 2 tasks, the task maximum latency parameter ensures that each task is paired with a different fog node. For this reason, the theoretic model powers up all fog nodes to support the demand. Since the fog node is in a startup state, no tasks can be assigned to it and they are sent serviced by the cloud. However, every task element from the set I is assigned to a fog nodes for time window 1 since every fog node is in a powered on state.

4.2.3 Fog Nodes Capacity Validation

This section will cover the testing used to validate the fog node resource capacity constraints: vCPU, memory and parallel tasks. The validation of these constraints behave along the same logic and don't require individual validations in this thesis. Each resource capacity constraint was validated in the testing to ensure that the implementation functioned as intended.

Equations 3.4, 3.5 and 3.6 ensure that tasks can only be assigned to a fog node with sufficient supply of each resource supplied by a fog node. The objective of this validation is to ensure that fog node resource capacity is sufficient to provide service to a maximum number of tasks.

For this validation, elements in the set of tasks I are all co-located to simulate multiple parallel task requests. Similarly to previous tests, the duration is set to 2 time windows where we expect the first time slot to be used to power on the fog node(s)

and the second time slot to assign the tasks where optimal. Table 4.6 shows the set of tasks used for this validation.

Table 4.6: Constraints 3, 4 and 5 Validation Task Table

i	Start time χ_i (windows)	Location γ_i	Max latency α_i (meters)	vCPU η_i (cores)	Memory ζ_i (GB)	Duration ψ_i (windows)
1	0	(0, 0)	30	1	1	2
2	0	(0, 0)	40	1	1	2
3	0	(0, 0)	30	1	1	2
4	0	(0, 0)	30	1	1	2
5	0	(0, 0)	30	1	1	2

The fog node set J for this validation contains only one fog node that is co-located with all the tasks shown in Table 4.6. The supply resources for the validated resources to a bottleneck value while setting large relative values (99,999) to the other resources. The quantity of the limited resource is less than would be required to supply all the elements in set of tasks shown in Table 4.6. Table 4.7 shows the set of fog nodes used for this validation the vCPU resource. A simple permutation of values can be used to validate the memory and parallel task resources.

Table 4.7: Constraint 3 Validation Fog Node Table

j	Location δ_j	vCPU ν_j (cores)	Memory υ_j (GB)	Parallel tasks ϵ_j (units)	State
1	(0, 0)	99,999	2	99,999	0

The results show that time window 0 is used to set the startup variable μ_{10} to 1 for the only fog node in the set J. No tasks can be assigned to it since it is in the startup state and is serviced by the cloud. In time window 1, only 2 tasks are assigned to the fog node since its vCPU supply can not support additional tasks. The algorithm arbitrarily chooses tasks 1 and 2 because all tasks have the same resource demand and profit. However, any task combination could be chosen in this example to generate the same solution to the optimization function.

4.2.4 Tasks Assigned to a Single Fog Node or Cloud

Equation 3.7 ensures that tasks can only be assigned to a single fog node or the cloud. The objective of the test is to ensure that there are multiple candidate fog nodes for each task and ensures that the algorithm only chooses a single fog node.

For this validation, the set of tasks I contains only a single element. Table 4.8 shows the set of tasks used for this validation.

Table 4.8: Constraint 6 Validation Task Table

i	Start time χ_i (windows)	Location γ_i	Max latency α_i (meters)	vCPU η_i (cores)	Memory ζ_i (GB)	Duration ψ_i (windows)
1	0	(0, 0)	20	1	1	2

The fog node set J for this validation contains multiple identical fog nodes that can support the single task contained in the set I. Table 4.9 shows the fog node set J used in this validation.

Table 4.9: Constraint 6 Validation Fog Node Table

j	Location δ_j	vCPU ν_j (cores)	Memory υ_j (GB)	Parallel tasks ϵ_j (units)	State
1	(0, 0)	3	2	4	0
2	(0, 0)	3	2	4	0
3	(0, 0)	3	2	4	0
4	(0, 0)	3	2	4	0
5	(0, 0)	3	2	4	0

The results show that time window 0 is used to set the startup variable μ_{j0} to 1 for only a single fog node element in the set J since only one will be used to assign the task. The task is then assigned to that fog node at time window 1. All other fog nodes can remain powered off since moving them to another state would not be optimal. The theoretic model chose fog node 1 from the set J, but any other element would provide the same result to the optimization function.

4.2.5 Task Maximum Latency Validation

Equation 3.8 ensures that tasks can only be assigned to a fog node that can provide a latency (distance) under the maximum demanded value. The objective of this test is to set a fog node just outside of the maximum latency requirement of a task and ensure that the algorithm assigns it to the cloud.

For this validation, the set of tasks I contains only a single element. Table 4.10 shows the task set used for the validation. The location of the task is set to values than can be trivially derived when calculating distance from elements in the fog node set J.

Table 4.10: Constraint 7 Validation Task Table

i	Start time χ_i (windows)	Location γ_i	Max latency α_i (meters)	vCPU η_i (cores)	Memory ζ_i (GB)	Duration ψ_i (windows)
1	0	(1.1, 0)	1	1	1	2

The fog node set J for this validation contains a single element located at the origin. As expected, the distance between it and the task shown in Table 4.10 will be larger than the maximum latency value of the task in Table 4.10. Table 4.11 shows the fog node set J used in this validation.

Table 4.11: Constraint 7 Validation Fog Node Table

j	Location δ_j	vCPU ν_j (cores)	Memory υ_j (GB)	Parallel tasks ϵ_j (units)	State
1	(0, 0)	3	2	4	0

The results show that all variables remain set to 0 since assigning the task to the cloud is the only valid option.

4.2.6 Task Start and Stop Time Constraints Validation

Equation 3.9 ensures that tasks can't be assigned to a fog node before its start time window. The objective of this test is to start a task at a specific time window and ensure that all values of the variable x_{ijt} where t is smaller than the start time window of each task.

For this validation, the set of tasks I contains only a single element. Table 4.12 shows the task set used for the validation. The start time window of the task is a value of 3 to ensure that the algorithm doesn't begin operations prematurely.

Table 4.12: Constraint 8 Validation Task Table

i	Start time χ_i (windows)	Location γ_i	Max latency α_i (meters)	vCPU η_i (cores)	Memory ζ_i (GB)	Duration ψ_i (windows)
1	3	(0, 0)	1	1	1	2

The fog node set J is identical to Section 4.2.5. Its only requirement is that at least one element of the fog node set J must be able to serve the task element shown in Table 4.12.

The results show that time windows 0 and 1 have no operations on the fog node. Time window 2 starts the fog node and is assigned for time windows 3 and 4. The values of x_{11t} are only set to 1 for $t = 3, 4$ and are 0 for all other values.

Equation 3.10 ensures that tasks can't be assigned to a fog node after its start time window plus its duration. This use case was proven in every previous validation since the final time window in every case does not have a task assigned because it has ended. Notably, Section 4.2.1 shows a clear example with 10 tasks.

4.2.7 Start-up Cost Constraint Validation

Equation 3.11 ensures that start-up costs are considered before powering on a fog node. The objective of the validation is to increase the startup cost value cu to make it not optimal to use a fog node over the cloud for specific tasks.

For this validation, the set of tasks I only have a single element with no notable special parameters. Table 4.8 shows the task set used in this validation. However, the startup cost value cu was increased to 2 from 0.5.

The fog node set J for this validation contains a single element with no notable special parameters. Table 4.11 shows the fog node set J used in this validation.

The results show that the algorithm detects that the startup costs are higher than the generated revenue and does not startup the fog node and assigns the task to the cloud.

4.2.8 Shutdown Cost Constraint Validation

Similarly to Section 4.2.7, this validation uses the same values for the task set I (Table 4.8) and the fog node set J (Table 4.11). However, the startup cost value is set to the default value (0.5), but the maintenance cost is increased to 0.3 from 0.1. This change ensures that a fog node will be turned off once the task is completed since the shutdown cost cd is smaller than the maintenance cost.

The results show that time window 0 is used to set the startup variable μ_{10} to 1 for the only fog node element in the set J. Since the fog node is in a startup state, no tasks can be assigned to it. However, the only task element from the task set I is assigned to a fog node in time window 1 since every fog node is in a powered on state. Finally, in time window 2, the fog node is shutdown due to the cost value changes proving that the constraint works as intended in the algorithm.

4.2.9 Theoretical Model Complexity

Table 4.13 shows the profit and processing time statistics in seconds of 5 different tests for each network size. Two types of theoretical models were tested: a full model with a 5,000 second time limit and a time limited (TL) theoretical model with a 180 second time cap. Some theoretical model testing initially ran for over 50,000 seconds for some tests. These tests all iterated through branches of the optimization tree with identical profit values after approximately 1 hour of computation. Every test with a processing time over 5,000 seconds showed identical behaviour and is believed to present the upper bound profit with a high degree of confidence.

The full model's primary objective is to try and obtain an optimal solution by letting the solver process for a significant time period if needed. The TL model is used to illustrate that most of the processing time is used to prove that the solution is optimal well before CPLEX completes its process. If the time limit is reached, CPLEX will simply return the best solution found so far. The profit values for the TL model are relative to the full theoretical model for simplicity ($Profit(TL)/Profit(Full)$). As

a result, a relative profit of 1 means that the optimal solution was also found by the TL model. A relative profit of 0 indicates that a null solution was found by the solver when processing the TL model. This is typically indicative that the solver reached the time limit before an initial solution was found.

The minimum, typical (median) and maximum values are provided for each network size. The typical statistic uses the median rather than the average because the span of values is large and the median more clearly represents that behaviour. The full and TL models were processed in series for all tests using identical input data. Some simulations (e.g. $|I| = 46$ and $|J| = 5$) show slight fluctuations between the processing times in both models. These are tied to the rounding errors and processing performance incertitude of the simulation computer. An overview of the results are shown in Table 4.13. The full results are presented in Table A.1 from Appendix A.

Table 4.13: Theoretic Model Average Processing Time Results (in seconds)

$\|I\|$	$\|J\|$	Theoretical Model						TL Theoretical Model (180s)					
		Profit (units)			Time (s)			Relative Profit			Time (s)		
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
6	1	3.0	7.2	9.0	0.1	0.1	0.1	1.0	1.0	1.0	0.1	0.1	0.1
	5	10.2	12.1	14.0	0.4	1.2	3.8	1.0	1.0	1.0	0.4	1.2	3.8
	9	10.5	12.2	14.1	0.7	1.5	2.7	1.0	1.0	1.0	0.6	1.5	2.7
	13	12.1	13.4	14.0	2.4	2.7	3.4	1.0	1.0	1.0	1.9	2.5	3.2
	17	12.1	13.3	14.0	2.7	5.1	7.9	1.0	1.0	1.0	2.7	5.0	7.8
	21	12.1	13.4	15.9	4.4	6.6	11.1	1.0	1.0	1.0	4.5	6.5	10.4
16	1	7.5	12.7	22.6	0.1	0.1	0.2	1.0	1.0	1.0	0.1	0.1	0.2
	5	34.4	38.0	42.0	2.0	4.0	7.1	1.0	1.0	1.0	2.0	4.0	7.1
	9	39.1	40.9	42.2	7.9	12.5	20.6	1.0	1.0	1.0	8.0	12.5	20.4
	13	39.8	41.5	44.1	14.6	39.0	79.1	1.0	1.0	1.0	16.3	40.2	83.3
	17	39.0	41.2	43.1	21.1	59.0	185.6	1.0	1.0	1.0	21.2	58.5	180.0
	21	38.0	41.5	44.1	27.8	1486.4	5000.1	1.0	1.0	1.0	27.8	107.5	180.0
26	1	12.0	20.7	39.0	0.2	0.2	0.3	1.0	1.0	1.0	0.2	0.2	0.3
	5	59.6	65.2	71.0	5.6	8.2	10.1	1.0	1.0	1.0	5.7	8.3	10.3
	9	66.4	69.1	70.5	10.3	19.8	26.3	1.0	1.0	1.0	10.4	19.8	26.0
	13	66.2	70.1	74.7	42.7	1360.8	5000.1	1.0	1.0	1.0	45.5	116.9	180.0
	17	67.9	69.7	71.9	101.7	3071.5	5005.1	1.0	1.0	1.0	102.3	164.5	180.1
	21	66.9	70.3	73.9	415.0	3286.5	5000.1	0.0	0.8	1.0	180.0	180.1	180.2
36	1	15.6	28.4	52.5	0.3	0.4	0.4	1.0	1.0	1.0	0.3	0.3	0.4
	5	95.4	96.6	98.1	5.0	7.8	11.1	1.0	1.0	1.0	5.0	7.8	11.1
	9	96.1	97.5	99.8	69.6	1104.8	5000.1	1.0	1.0	1.0	63.4	121.7	180.1
	13	92.4	98.2	102.7	99.3	886.5	2183.5	1.0	1.0	1.0	99.0	145.6	180.0
	17	95.0	96.7	98.3	163.8	2161.4	5009.3	0.0	0.8	1.0	165.9	177.2	180.1
	21	94.1	99.1	104.1	334.6	4067.0	5000.2	0.0	0.0	0.0	180.0	180.0	180.0
46	1	21.0	32.1	73.8	0.3	0.3	0.5	1.0	1.0	1.0	0.3	0.4	0.5
	5	109.7	121.3	124.7	7.0	11.7	18.7	1.0	1.0	1.0	7.1	11.3	16.8
	9	117.9	125.6	129.7	18.4	1444.4	5000.0	1.0	1.0	1.0	20.1	129.4	180.1
	13	121.2	126.7	132.1	101.8	2098.4	5002.3	1.0	1.0	1.0	101.5	149.3	180.1
	17	122.9	128.3	131.0	198.8	2495.4	5000.1	0.0	0.4	1.0	180.0	180.0	180.0
	21	126.9	128.7	131.4	3519.3	4704.7	5003.7	0.0	0.0	0.0	180.0	180.0	180.1

Processing Time Simulation Analysis

The profit results found in Table 4.13 show that the TL model performs with near perfection when compared with the full model. Once the networks grow to where the TL model can not obtain a solution (such as $|I| = 26$ and $|J| = 21$), the TL model is consistently unable to find results for larger network simulations. This break point is predictable since this behaviour is consistently repeated for optimization problems of identical and larger sizes. In every simulation, the full model found an optimal solution to the problem at approximately 15% of the total processing time and used the rest of its processing to iterate through the rest of the optimization tree as a thorough verification mechanism. Therefore, it may be possible to predict the minimum time limit needed to obtain the optimal profit value with a high degree of confidence.

The processing time values shown in Table 4.13 demonstrate that the values are exponentially related to the sizes of its input sets with a plateau towards 5,000 seconds (which was the maximum allowed) for larger network sizes. If the simulations were not automatically stopped after 5,000 seconds, it is expected that the values would continue to grow exponentially. Of note, some processing time slightly exceeds the maximum value due to the solver completing its current branch processing after the time limit is reached.

Simulated networks are expected to have a large number of tasks to simulate an environment running over weeks with several hundred tasks. As expected, the theoretical model can only be used to solve smaller simulations and is not suited to assign tasks in scenarios of practical scale. However, a time limited execution of the model also provides the profit upper bound with a high degree of confidence as long as the time limit is scaled to the size of the simulation.

Computational Complexity Analysis

The computational complexity is correlated to the size of the optimization problem. This is largely composed of decision variables and constraints in terms of the number of lines in the problem to solve.

The number of decision variables increases the number of computations needed in each constraint, but is a relatively minor contributor to the overall computational complexity of the theoretical model. It can be calculated using Equation 4.1. It

simply computes the size of each decision variable described in Section 3.2.1.3 which is determined from the size of the sets of tasks I, fog nodes J and time windows T.

$$|DecisionVariables| = |I||J||T| + 3(|J||T|) \tag{4.1}$$

The number of fog nodes, tasks and total time windows also increases the number of constraints in the problem which is the major contributor to the computational complexity of the model. Equation 4.2 approximately calculates the number of constraints to solve by adding size of the groups described in Section 3.2.2.2. It is an approximation because the *task start time check* and *task stop time check* constraints do not generate values for all $t \in T$, but only for values of t that are relevant to each task. This design decision was made to improve computational performance.

$$|Constraints| \leq 7(|J||T|) + 4(|I||T|) \tag{4.2}$$

The above equations clearly indicate that the size of the time window set T contributes the most to the computational complexity of the optimization problem. The size of T is correlated to the number of tasks in the problem. Also, the network generation algorithm adds variance in the size of T. It uses a normally generated normal variable to determine the duration of a task around an average $\mu = 4$ and standard deviation $\sigma = \mu/6 = 2/3$. It also ensures that it has an average of 2 tasks per time window. These variables can sometimes cause simulations with a higher number of tasks and fog nodes to have less constraints to compute. This occurred in some results in Table 4.13 notably for $|I| = 36, |J| = 13$ where the processing time was higher than expected. The location of tasks and fog nodes also impact the complexity of the problem to solve. In cases where there are many possible solution, the optimization tree is larger and increases the computation required to obtain a solution.

The theoretical model relies on complete knowledge of the task set which is not a realistic scenario when conducting task assignment and fog node changes in real-time. However, it allows us to evaluate the performance of the heuristic models applied. The extensive validation completed in this section is aimed at ensuring that the theoretic model is functioning as designed since it will be used as the main validation method for the following sections of this chapter.

4.3 Heuristic Model Parameters

This section provides an overview of each configuration variable described in Section 3.3 for the heuristic model. Each variable has an effect on others, but the test plan isolated their effects by using specific test networks to determine optimal values for each of these parameters. As described in Section 3.3, configurable parameters are divided in two categories: task order manipulation parameters and fog node status change parameters. The heuristic model also has processing time requirements to ensure that the processing time for every time window is under the time window size. For this reason, the task allocation tables will provide values in seconds and be converted to time windows within the heuristic model. The time window size for all tests in this section is 2 seconds.

4.3.1 Task Order Manipulation Parameters

Section 3.3 describes the allocation and de-allocation group processes of the heuristic model which work together to manipulate the active task prioritization. Therefore, they are grouped in this chapter as task order manipulation parameters. The value of these parameters are used to optimize current and new tasks to determine the optimal order to assign them to the fog network. Once tasks are prioritized, they are assigned to a fog node or cloud in that order using a heuristic algorithm. The task assignment heuristic used to assign tasks from the allocation group to a fog or cloud is to select a powered on fog node with the fewest available resources (best fit). This favours a green computing approach to fog computing by reducing the number of fog nodes required and reducing network maintenance costs. The following paragraphs detail the parameters used to weight active tasks at each time window to determine their assignment priority.

Task predicted de-allocation: This parameter is described in Algorithm 2 in which it uses the duration of all past tasks to predict probability of a task ending in the current time window. Tasks that are predicted to end are de-prioritized based on the predicted de-allocation weight.

Number of powered on fog nodes in range: Tasks with fewer available fog nodes are prioritized higher than ones with more options. This is used to assign tasks that have more restrictive assignment requirements first to reduce the number

of assignment decisions that could reduce the total number of tasks being assigned to a fog node.

Task Age: A weight is applied to tasks to prioritize newer tasks over older ones. Simply put, assigning newer tasks first will favour them being sent to a fog node with more traffic, thus require it to be powered on for longer. Furthermore, assigning older tasks to lesser used fog nodes will favour those resources being used for less time windows on average and assisting in the fog node status change processes of the heuristic model.

4.3.1.1 Task Order Manipulation Weight Optimization

The task order manipulation weight parameters usually only optimize the solution in scenarios where at least one fog node has insufficient resources to support its possible set of tasks. Scenarios where a fog node can support all possible tasks in parallel won't benefit from modifying the task order since all of these tasks can be optimally assigned to a fog network in any order. However, most scenarios will have some bottlenecks that will benefit from assigning tasks in a correct order.

In order to determine the correct parameter values in testing, all fog node status changes are removed from the problem by turning all fog nodes on at the start and having 0 maintenance costs to remove any fog node status change operations from the test.

Table 4.14 enumerates the global parameters used in testing for this model.

Table 4.14: Task Order Manipulation Testing Global Parameters (in currency units)

Parameter	**Value**
maintenance costs (mn)	0
startup costs (cu)	0
shutdown costs (cd)	1000
revenue (r)	1

Detailed Example

The following example will be presented and analyzed to demonstrate the importance of manipulating the order of allocating tasks. The scenario is simply to present a trivial model where the order in which we allocate tasks makes the difference between replicating the optimal result and not.

Firstly, the Table 4.15 shows the task set I used to execute the heuristic and theoretic models. The duration of the tasks is shown in the table, but is obviously not used in the heuristic model. It generates a subset of active tasks from I at each time window to assign to the fog or cloud without knowledge of when it would be completed.

Table 4.15: Task Manipulation Weight Testing Detailed Example Tasks

i	Start time χ_i (windows)	Location γ_i	Max latency α_i (meters)	vCPU η_i (cores)	Memory ζ_i (GB)	Duration ψ_i (windows)
1	0.1	(0, 0)	1	1	1	4.8
2	0.1	(2, 0)	1	1	1	3.9
3	2.1	(1, 0)	1.1	1	1	4.6

Secondly, Table 4.16 shows the set of fog nodes used in the example.

Table 4.16: Task Manipulation Weight Testing Detailed Example Fog Nodes

j	Location δ_j	vCPU ν_j (cores)	Memory υ_j (GB)	Parallel tasks ϵ_j (units)	State
1	(0.5, 0)	1	2	4	0
2	(2, 0)	3	2	4	0

This simple example is illustrated in Figure 4.2 and was specifically engineered to demonstrate the importance of assigning tasks in the correct order to simplify the actual task assignment heuristic. One of the tasks (task 3) was designed to be assignable to either fog node 1, 2 or the cloud. However, tasks 1 and 2 only have the option of fog nodes 1 and 2 respectively in addition to the cloud. If task 3 is assigned before the others, the best fit algorithm will allocate it to fog node 1, utilizing all its resources and provide a sub-optimal heuristic solution to the problem. However, if task 1 is prioritized ahead of 3, it will provide the optimal solution at every simulation. Table 4.17 shows the results based on each of the task order manipulation parameters. The theoretic model calculates the optimal profit at 5.

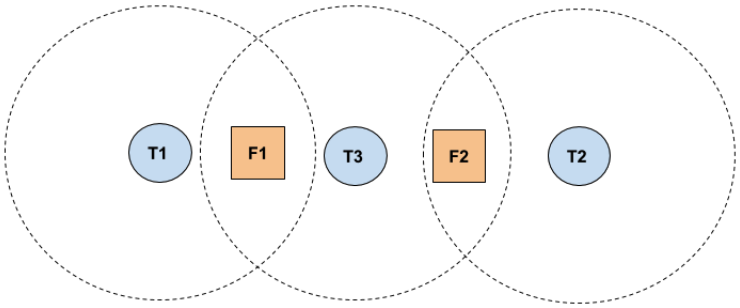

Figure 4.2: Possible Impact of Number of Powered Fog Nodes in Range

As designed, the heuristic model assigns profit to tasks assigned to a fog node at each time window regardless of how those tasks were assigned in the past or in the future. Table 4.17 demonstrates it is less important to order tasks based on their age or provide a predictive de-allocation of tasks in this example. The predictive de-allocation and age parameters have little impact on profit improvement. This is largely based on how task profits were allocated at each time window rather than with another method such as using service level agreements. Chapter 5 provides additional discussion on modifying the model design to increase the impact of the predictive de-allocation script.

Table 4.17: Task Manipulation Weight Testing Detailed Example Results

Test Case	Input			Output
	Dealloc	FN Opt	Age	Profit (units)
1	0	0	0	4
2	0	0	5	4
3	0	0	10	4
4	0	5	0	5
5	0	5	5	5
6	0	5	10	5
7	0	10	0	5
8	0	10	5	5
9	0	10	10	5
10	5	0	0	4
11	5	0	5	4
12	5	0	10	4
13	5	5	0	5
14	5	5	5	5
15	5	5	10	5
16	5	10	0	5
17	5	10	5	5
18	5	10	10	5
19	10	0	0	5
20	10	0	5	5
21	10	0	10	4
22	10	5	0	4
23	10	5	5	4
24	10	5	10	5
25	10	10	0	5
26	10	10	5	5
27	10	10	10	5

Results show that predicted de-allocation and age have no deterministic impact when uniquely considering task assignment and removing fog node state changes. There are also some examples that can be generated where age and predictive de-allocation can be used to facilitate modifying the state of a fog node. Notably, models where there are high maintenance costs would rely on those parameters more heavily to assign the correct tasks to the cloud in order to shutdown the correct fog nodes.

Table 4.17 shows that the number of fog nodes available to a task is the most important parameter to impact the heuristic model as it was designed. In this example, ordering in ascending order based on the number of fog nodes available ensures that they will be assigned in either the order 1, 2 and 3 or 2, 1 and 3 (no impact to optimization). The result of an optimal assignment is shown in Result 1.

Result 1: Heuristic Model Task Order Prioritization Detailed Example Results

Time Window 0 Running fog node: None
    Start of tasks 1, 2 and 3
    $\mu_{10} \leftarrow 1$, $\mu_{20} \leftarrow 1$ `/* Startup fog nodes 1 and 2 */`
    Tasks 1, 2 and 3 assigned to the cloud

Time Window 1 Running fog nodes: 1 and 2
    $x_{111} \leftarrow 1$ `/* Task 1 assigned to fog node 1 */`
    $x_{221} \leftarrow 1$, $x_{321} \leftarrow 1$ `/* Tasks 2 and 3 assigned to fog node 2 */`
    $\theta_{11} \leftarrow 1$, $\theta_{21} \leftarrow 1$

Time Window 2 Running fog nodes: 1 and 2
    End of tasks 1 and 2
    $x_{312} \leftarrow 1$ `/* Task 3 assigned to fog node 1 */`
    $\theta_{12} \leftarrow 1$, $\theta_{22} \leftarrow 1$

Time Window 3 Running fog nodes: 1 and 2
    $x_{313} \leftarrow 1$ `/* Task 3 assigned to fog node 1 */`
    $\theta_{13} \leftarrow 1$, $\theta_{23} \leftarrow 1$

Time Window 3 Running fog nodes: 1 and 2
    End of task 3
    $\theta_{13} \leftarrow 1$, $\theta_{23} \leftarrow 1$

$$\boldsymbol{Profit = r(x_{111} + x_{221} + x_{321} + x_{312} + x_{313}) - cu(\mu_{10} + \mu_{20}) - mn(\theta_{11} + \theta_{12} + \theta_{12} + \theta_{22} + \theta_{13} + \theta_{23} + \theta_{14} + \theta_{24}) = 5}$$

Bulk Testing

The bulk testing parameters are similar to the previous example where the maintenance costs are as shown in Table 4.14 and the fog nodes have fixed parameters as shown in Table 4.18.

Table 4.18: Task Manipulation Weight Testing Detailed Example Fog Nodes

j	Location δ_j	vCPU ν_j (cores)	Memory υ_j (GB)	Parallel tasks ϵ_j (units)	State
1	(0, 0)	3	2	4	0
2	(1, 0)	3	2	4	0
3	(2, 0)	3	2	4	0

The task generation mechanism is shown in the bullet points below. It uses values designed to engage each of the parameters used to manipulate the task order.

- i: Incremental value between 1 and n where n is the number of tasks.
- χ_i: Randomized value based on the previous task start time and task density.
- γ_i: Randomized value between 0 and 3 for the x coordinates and 0 for the y coordinates.
- α_i: Value equal to 1.
- η_i: Value equal to 1.
- ζ_i: Value equal to 1.
- ψ_i: Value equal to 4.8

The detailed example showed us that the fog node options weight is the most important factor to consider when removing fog node changes. A total of 100 randomized networks were generated: 10 networks with 15 and 20 total tasks with a task density of 4, 6, 8, 10 and 12 tasks per time window. As such, Result 2 shows the hypothesized optimal weight values used in testing. Although the predicted de-allocation and age showed no impact to the test environment, the benefit to these factors may impact the performance of the full heuristic model once fog node status changes are introduced. At the very least, their inclusion does not negatively impact the performance of the heuristic model.

Result 2: Task Order Manipulation Testing Global Parameters

1 $preddeal \leftarrow 5$
2 $fnoptions \leftarrow 10$
3 $age \leftarrow 5$

The testing showed that those parameters provided the optimal solution for 93 out of 100 networks. Three other networks yielded an optimal solution with other weight parameter values. Each of those examples showed an optimal solution when age was the dominant weight applied to the heuristic model. However, most of the other networks would have provided a sub-optimal solution if the number of fog node options was not used as a predominant weight. The other four instances showed no weight parameters that would have provided an optimal solution. In each of these cases, the heuristic was one (2 cases) or two (2 cases) assignment decisions away from the optimal profit.

Approximately 70% of networks showed no difference when iterating weight values. This is logical since many networks don't necessarily have a resource bottleneck on fog nodes that directly lead to a reduction in the overall number of tasks assigned to fog nodes. It depends on the timing and placement of tasks which were randomized in the bulk testing.

Overall, the task order manipulation provides a near optimal solution in the majority of tested networks when using the task order manipulation weights shown in Result 2. The complete results are included in Appendix B.

4.3.2 Fog Node Status Manipulation Parameters

Fog node status manipulation parameters rely on the heuristic model correctly predicting future task demands for the network. As such, the performance of the heuristic model will differ greatly based on the predictability of a network, notably on the placement of tasks. Since the heuristic model does not have knowledge of future events, it must use statistics from previous events to evaluate an approximate probability of requiring each fog node to determine in which state is should be at each time window.

In the absence of the heuristic model correctly predicting powering on a fog node, the algorithm will always begin the startup process of a fog node when a task appears that could have been assigned to a powered off fog node but was assigned to the cloud. This ensures that a critical fog node is ready to be used for the next time window. However, the utilization of this procedure will always yield a sub-optimal mapping when compared to the theoretical model because there is at least one time window where a task was sent to the cloud when it could have been assigned to a fog node.

The only exception to this rule is at the very first time window where the theoretic and heuristic models both can not proactively power on a fog node. Furthermore, the heuristic model determines the average task duration of historically assigned tasks to ensure that the task is likely to be required at the next fog node. A situation where this routine is not advantageous is for tasks with a duration of a single time window.

Thirdly, the powering off of a fog node uses a strictly probabilistic process in the heuristic model since a fog node must only be shut down if the fog computing network believes it will not be used for N time windows where $N \geq (cu + cd)/mn$. In the absence of knowledge of future events, the heuristic model must use a probabilistic algorithm to predict the value of N in this example.

The following subsections will provide a detailed overview of the fog node status change parameter values used to optimize networks using parameters generated along the following set of guidelines.

Tasks (Quantity: 20)

- i: Iterating value between 1 and 20.
- χ_i: Ascending start time based on the previous task start time and a pseudo random value generated from a normal distribution centered on the average frequency of tasks.
- γ_i: Randomly generated value within the x and y grid boundaries (100 in this case).
- α_i: Value always equal to 1.
- η_i: Value always equal to 1.
- ζ_i: Normally distributed random value based on $\mu = 4$ and $\sigma = \frac{2}{3}$.
- ψ_i: Average value of 8 seconds.

Fog Nodes (Quantity: 5)

- j: Iterating value between 1 and 5.

- δ_j: Randomly generated value within the x and y grid boundaries (100 in this case).
- ν_j: Value always equal to 1.
- υ_j: Value always equal to 2.
- ϵ_j: Value always equal to 4.
- State: Value always equal to 0 (powered off).

4.3.2.1 Fog Node Memory Threshold

In initial testing, the heuristic model fog node status changes were significantly more frequent than the theoretic model. As such, the heuristic model suffered status change iteration issues where a fog node would turn on at time window t, turn off at $t+1$ and turn on again at $t+2$, etc. To prevent this issue, a fog node memory threshold was implemented where a fog node status would be reserved for a configurable number of time windows to either remain powered on or off. The only exception built into the heuristic model is the non-predictive powering on routine since the model knows with certainty that the fog node will be required in the next time window given that revenue is higher than the start-up cost of a fog node.

- The start memory is used to track when a fog node was turned on and prevents a shutdown before the memory parameter is reached.
- The stop memory is used to track when a fog node was turned on and prevents a predictive startup before the memory parameter is reached.

The optimal value of the parameters is dependent on the task demand patterns and statistics. There are some edge cases that perform well in a low memory threshold such as cases with negligible operating costs relative to the revenue. However, outside of very specific cases, implementing a fog node status memory threshold consistently produced higher relative profit when compared with the theoretic model.

Figure 4.3 shows an average relative profit for 10 tests and clearly demonstrates that the benefit of a higher threshold plateaus at the following values:

- Start Memory: 7

- Stop Memory: 5

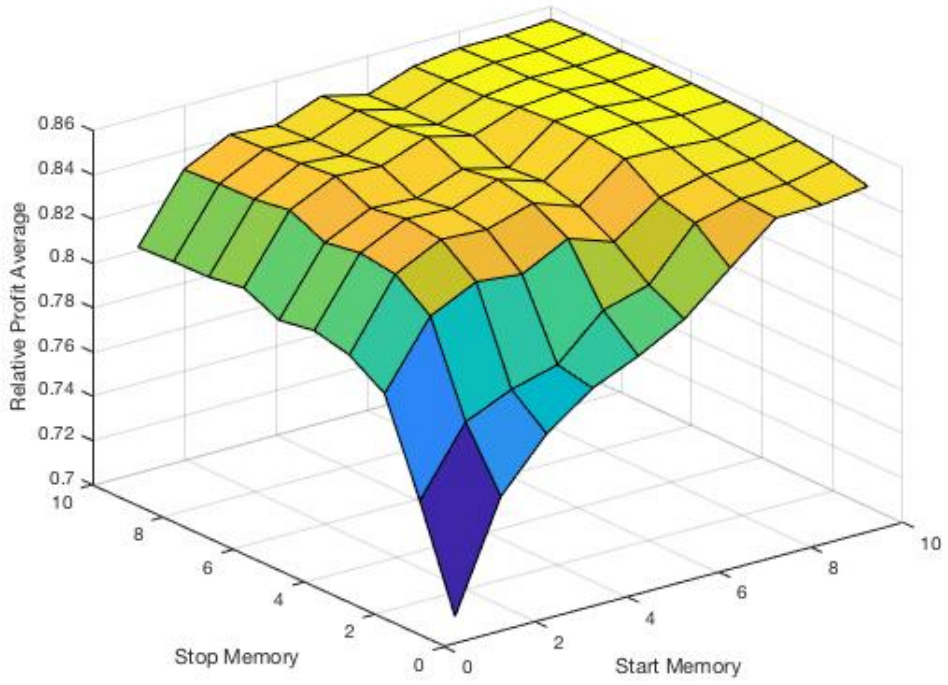

Figure 4.3: Heuristic Model Start Stop Memory Results Summary (10 tests).

There are obviously other values that produce similar results on average and depend on the statistical properties of the tasks. The testing showed that a good understanding of historical tasks should be used when possible to determine the optimal values for a given task layout. The complete results from each individual tests are included in Appendix B.

4.3.2.2 Fog node Simulated Tasks

Simulated tasks are used to determine the probability that certain fog nodes should be powered on or off based on historic task allocation during the fog node state change heuristic. Each simulated task is artificially assigned to the current state of the fog network and records which fog node it could be assigned to including powered off fog nodes. This provides a measure of probability to change the fog node status in the network's current state. These values are used when calculating the probabilities to

determine whether a fog node should be powered on or off. The higher the number of simulated tasks produces more statistically consistent results (averaged over 10 tests) as demonstrated in Table 4.19, but the processing time per time window of the heuristic model also increases. As such, a compromise must be made between processing time (which must be under the time window size) and reliability of the results.

This value depends on the size of time window. In this test, the maximum number of simulated tasks is 21. However, to account for fluctuation, we will select a value of 13. As demonstrated in the results, the impact of this parameter plateaus once the number of simulated tasks reaches 13 since there is little statistical fluctuation in the relative profit between the heuristic and theoretical models. This value is approximately 6 times the number of tasks per time window (2). The allowable tolerance a model can have on the number of simulated tasks fluctuates based on the time window size. A network with small time windows is required to reduce the number of simulated tasks in order to process the data in time for the next time window.

Table 4.19: Fog Node State Manipulation Simulated Tasks Average Results (with a 95% Confidence Interval)

Input	Output	
Number of Simulated Tasks	Relative Profit	Processing Time (s)
1	0.88 ± 0.02	0.24 ± 0.03
3	0.93 ± 0.02	0.44 ± 0.03
5	0.92 ± 0.02	0.51 ± 0.04
7	0.90 ± 0.02	0.56 ± 0.04
9	0.89 ± 0.02	0.65 ± 0.04
11	0.88 ± 0.02	0.74 ± 0.06
13	0.94 ± 0.02	0.98 ± 0.07
15	0.92 ± 0.02	1.06 ± 0.08
17	0.94 ± 0.01	1.23 ± 0.05
19	0.94 ± 0.02	1.34 ± 0.06
21	0.94 ± 0.01	1.48 ± 0.07

The detailed results for each test are documented in Appendix B.

4.3.2.3 Fog node Status Change Sensitivity Threshold

The last remaining parameter is the probability threshold to start up and shut down a fog node. This value compares the probability that a positive impact to the network will occur if a status change to the network is made. In all the simulated tasks, the number of tasks that will benefit from powering on fog node j out of the total number of simulated tasks provide a probability ratio. If it is over the configured threshold, then the status change operation occurs.

In cases where there are multiple candidate status changes, the algorithm will proceed one at a time in order of probability (largest first) and re-evaluate the probabilities for the remaining fog nodes until the list is completed. This avoids a situation where there are multiple fog nodes with overlapping coverage areas being powered on all at once to have more fog node supply than required at that time. However, the heuristic model runs the simulated assignment multiple times to account for scenarios where multiple overlapping fog nodes should be powered on.

Tables 4.20 and 4.21 show an average of the relative profit for various probability thresholds and task density in average tasks per time window (tpw) over 10 tests with a 95% confidence interval. The results indicate that the value of the start probability doesn't greatly impact the relative profit. However, the stop probability benefits from having a lower threshold of 0.2 or lower.

Table 4.20: Simulated Start Probability Threshold Average Relative Profit

P(Start)	6 tpw	8 tpw	10 tpw	12 tpw
0.0	0.751 ± 0.040	0.799 ± 0.042	0.791 ± 0.040	0.824 ± 0.046
0.2	0.748 ± 0.042	0.802 ± 0.041	0.793 ± 0.040	0.820 ± 0.047
0.4	0.749 ± 0.042	0.800 ± 0.039	0.794 ± 0.040	0.821 ± 0.046
0.6	0.750 ± 0.045	0.802 ± 0.039	0.791 ± 0.042	0.819 ± 0.047
0.8	0.747 ± 0.043	0.798 ± 0.040	0.796 ± 0.041	0.820 ± 0.048
1.0	0.751 ± 0.045	0.802 ± 0.038	0.793 ± 0.040	0.822 ± 0.047

Table 4.21: Simulated Stop Probability Threshold Average Relative Profit

P(Stop)	6 tpw	8 tpw	10 tpw	12 tpw
0.0	0.953 ± 0.025	0.966 ± 0.034	0.935 ± 0.047	0.932 ± 0.041
0.2	0.861 ± 0.054	0.917 ± 0.041	0.955 ± 0.041	0.954 ± 0.048
0.4	0.812 ± 0.057	0.885 ± 0.053	0.768 ± 0.057	0.848 ± 0.058
0.6	0.646 ± 0.064	0.726 ± 0.058	0.748 ± 0.065	0.764 ± 0.078
0.8	0.621 ± 0.063	0.656 ± 0.071	0.670 ± 0.068	0.715 ± 0.076
1.0	0.604 ± 0.063	0.653 ± 0.069	0.681 ± 0.071	0.712 ± 0.073

The complete results are included in Appendix B.

4.3.3 Impact of Lower Entropy Task Location

The fog node status manipulation parameters are much less static in nature and depend on a knowledge of the environment in order to perform well. When possible, the ideal practical scenario would be to run the same iterative validation testing demonstrated in Sections 4.3.1 and 4.3.2 to determine the best values for a given network. However, this activity requires a significant amount of time to complete and would not be adequate for a heavily dynamic environment. This idea assumes that historical tasks (notably the location of tasks) are representative of future tasks. In a practical scenario, many environments will note statistical patterns in long term data to assist in this process. Scenarios where the task location is not fully random may benefit from training the parameters described in Section 4.3.2.

Using that caveat, the fog node status change parameter values determined from the testing of networks without a pattern in their location are as follows.

- Fog node start memory threshold: 7.
- Fog node stop memory threshold: 5.
- Fog node number of simulated tasks: $6 * tpw$
- Fog node start probability threshold: 0.4.
- Fog node stop probability threshold: 0.2.

The parameters that will most fluctuate based on the network are the start and stop probability thresholds. As the networks become more predictable, the threshold values will increase since the simulated data will more closely represent future data. The validations discussed in Section 4.3.2 were made for network with completely random task demands.

To validate this hypothesis, 40 networks were generated where tasks were centralized around 9 areas on a 20x20 grid (Coordinates (5, 5), (5, 10) (5, 15), (10, 5), (10, 10), (10, 15), (15, 5), (15, 10), (15, 15)). Tasks were generated using a normally distributed random variable with μ as the coordinates and σ as one sixth of the grid size (20/6). The hypothesis presumes that the heuristic algorithm will perform better in this type of network and may modify the threshold parameters for the fog node stop and start probability. Tables 4.22 and 4.23 show the average results for a 95% confidence interval for 10 tests with a task density of 3, 4, 5 and 6 tasks per second.

Table 4.22: Simulated Start Probability Threshold Results Summary with Predictable Tasks

P(Start)	6 tpw	8 tpw	10 tpw	12 tpw
0.0	0.970 ± 0.019	0.935 ± 0.029	0.932 ± 0.021	0.961 ± 0.026
0.2	0.974 ± 0.021	0.934 ± 0.031	0.933 ± 0.020	0.963 ± 0.027
0.4	0.974 ± 0.021	0.934 ± 0.031	0.933 ± 0.020	0.963 ± 0.027
0.6	0.974 ± 0.021	0.934 ± 0.031	0.933 ± 0.020	0.963 ± 0.027
0.8	0.974 ± 0.021	0.934 ± 0.031	0.933 ± 0.020	0.963 ± 0.027
1.0	0.974 ± 0.021	0.934 ± 0.031	0.933 ± 0.020	0.963 ± 0.027

Table 4.23: Simulated Stop Probability Threshold Results Summary with Predictable Tasks

P(Stop)	6 tpw	8 tpw	10 tpw	12 tpw
0.0	0.957 ± 0.037	0.959 ± 0.038	0.911 ± 0.056	0.936 ± 0.056
0.2	0.987 ± 0.026	0.911 ± 0.059	0.954 ± 0.041	0.988 ± 0.018
0.4	0.961 ± 0.039	0.958 ± 0.038	0.912 ± 0.056	0.938 ± 0.057
0.6	0.987 ± 0.026	0.911 ± 0.059	0.954 ± 0.041	0.988 ± 0.018
0.8	0.961 ± 0.039	0.958 ± 0.038	0.912 ± 0.056	0.938 ± 0.057
1.0	0.987 ± 0.026	0.911 ± 0.059	0.954 ± 0.041	0.988 ± 0.018

The results observed in this testing are notably higher relative profit in both the start and stop probability values than networks without any statistical predictability (from Table 4.20). In fact, as the predictability of tasks increase, the predicted fog node state change parameters have less impact on the overall performance of the heuristic model and the overall profit is over 90% of the theoretical profit in all settings. As such, the hypothesis that a more predictable set of tasks will perform better with a higher probability threshold is partially correct. The true discovery is that the value of the probability threshold has less impact on the relative profit. In fact, it is not possible to predict the entropy of tasks in a practical environment so it is more beneficial to maintain them at the values determined in Section 4.3.2.

4.4 Large Scale Testing

This section will provide a few detailed examples combining the parameters set in Section 4.3 and provide an analysis of these results. The main objective is to ensure that the heuristic model performs close to the theoretical upper bound. Any differences between the theoretic and heuristic models should be due to decisions that could not be better made without complete knowledge of the set of tasks. This section will provide a performance analysis of the heuristic model in a large scale of testing and provides a detailed analysis of deficiencies in the heuristic model. As with other tests in this chapter, the time window size is set to 2 seconds.

Tasks (Quantity: n)

- i: Iterating value between 1 and n.
- χ_i: Ascending start time based on the previous task start time and a pseudo random value generated from a normal distribution centered on the average frequency of tasks.
- γ_i: Randomly generated value within the x and y grid boundaries (100 in this case).
- Task resources are randomly selected based on 5 sets of values
  - Resources set 1: α_i: 10, η_i:: 1, ζ_i: 1. ψ_i: 8 seconds.
  - Resources set 2: α_i: 15, η_i:: 2, ζ_i: 1. ψ_i: 6 seconds.
  - Resources set 3: α_i: 30, η_i:: 2, ζ_i: 2. ψ_i: 12 seconds.
  - Resources set 4: α_i: 20, η_i:: 1, ζ_i: 2. ψ_i: 10 seconds.
  - Resources set 5: α_i: 40, η_i:: 1, ζ_i: 1. ψ_i: 4 seconds.

The task duration value is generated for each task using a normally distributed random variable with μ as the average task duration and the standard deviation σ as one sixth of the average task duration.

Fog Nodes (Quantity: m)

- j: Iterating value between 1 and m.
- δ_j: Randomly generated value within the x and y grid boundaries (100 in this case).
  - Resources set 1: ν_j: 3, υ_j: 6, ϵ_j: 6.
  - Resources set 2: ν_j: 2, υ_j: 2, ϵ_j: 2.
  - Resources set 3: ν_j: 4, υ_j: 5, ϵ_j: 5.
  - Resources set 4: ν_j: 1, υ_j: 2, ϵ_j: 2.
  - Resources set 5: ν_j: 2, υ_j: 3, ϵ_j: 4.
- State: Initial value always equal to 0 (powered off).

4.4.1 Comparison Between Theoretic and Heuristic Models

The following series of simulations focus on comparing the theoretic and heuristic model performances, both in terms of profit and processing time. The simulations are made for a variety of sizes for the tasks and fog node sets to evaluate the ability for the models to scale as the simulation sizes increase. If this thesis can demonstrate that the performance of the heuristic model is predictable, it can assume that it will perform the same for larger networks that can't be compared with the theoretical model due to the large processing time.

The network generation parameters used are similar to other validations in this section. For consistency, the number of concentration areas for tasks are equal to 5 for all simulations. The only important element to these values is that they remain consistent for all the tests used to compare the models. The results provided in Table 4.24 are an average value for 5 tests at each network size with at a 95% confidence interval. However, as in the results presented in Section 4.2.9, the theoretic model run time is presented as minimum (min), typical (typ) which represents the median and maximum (values) across the tests to better illustrate the large variance in processing time across those simulations. The processing time was capped at 8,000 seconds to prevent some simulations to run over several days without improvement to the profit value (as analyzed in Section 4.2.9). An 8,000 second time limit was selected to ensure that the values obtained in this section were optimal with a very high degree of confidence. Although many simulations reached the processing time limit, the theoretic model tree results had not improved several thousand seconds before the cap was reached. Furthermore, the relative profit was consistent with smaller networks. For those reasons, this thesis has a high confidence that the results presented in the table represent the theoretical upper bound. There are a few exceptional notations in the table:

- No solution (NS) indicates that the theoretical model ran to its capped duration without finding a solution to the optimization problem.
- Out of memory (OoM) indicates that the theoretical model was unable to complete due to a lack of memory in the simulation environment.

Table 4.24: Performance Comparison Between Theoretic and Heuristic Models (time in seconds)

$\|I\|$	$\|J\|$	Average Relative Profit	Theoretic Model			Heuristic Model	
			Processing Time (s)			Average Processing Time (s)	Average Processing Time per Window (s)
			Min	Typ	Max		
5	5	0.807 ± 0.005	0.8	3.7	6.1	2.0 ± 0.1	0.2 ± 0.0
	10	0.799 ± 0.007	2.8	2.8	18.6	2.7 ± 0.5	0.3 ± 0.0
	15	0.800 ± 0.012	2.3	14.1	22.9	3.5 ± 0.5	0.4 ± 0.0
	20	0.809 ± 0.012	12.7	19.9	32.3	4.2 ± 0.7	0.5 ± 0.0
	25	0.673 ± 0.012	14.3	32.8	55.6	5.5 ± 0.5	0.6 ± 0.0
	30	0.696 ± 0.014	5.3	154.1	412.4	6.6 ± 1.1	0.7 ± 0.0
15	5	0.866 ± 0.016	2.3	7.2	19.7	4.7 ± 0.7	0.3 ± 0.0
	10	0.843 ± 0.030	32.5	50.1	102.7	6.7 ± 0.8	0.6 ± 0.0
	15	0.803 ± 0.030	70.2	988.6	8000.2	8.9 ± 0.6	0.7 ± 0.0
	20	0.761 ± 0.045	229.8	8000.2	8012.9	11.6 ± 1.2	0.8 ± 0.1
	25	0.815 ± 0.048	1736.6	8000.1	8008.9	13.1 ± 1.9	0.9 ± 0.1
	30	0.679 ± 0.045	8000.1	8000.1	8012.1	15.3 ± 1.6	1.1 ± 0.1
25	5	0.806 ± 0.032	8.1	25.6	45.2	7.5 ± 0.3	0.42 ± 0.0
	10	0.834 ± 0.067	115.5	196.6	2249.8	11.4 ± 0.5	0.7 ± 0.1
	15	0.848 ± 0.090	545.3	8000.1	8006.1	15.3 ± 1.9	0.8 ± 0.1
	20	0.809 ± 0.087	8000.1	8000.1	8008.6	19.5 ± 0.6	1.1 ± 0.1
	25	0.789 ± 0.111	8000.1	8000.2	8010.0	21.3 ± 2.1	1.3 ± 0.1
	30	0.784 ± 0.106	8000.1	8000.2	8009.6	22.7 ± 3.4	1.4 ± 0.1
35	5	0.849 ± 0.031	8.1	15.1	64.6	9.2 ± 0.9	0.4 ± 0.0
	10	0.832 ± 0.049	290.6	564.0	8000.3	17.2 ± 2.2	0.6 ± 0.1
	15	0.816 ± 0.076	8000.1	8000.5	8004.8	21.1 ± 2.4	0.9 ± 0.1
	20	0.807 ± 0.115	8000.2	8000.2	8000.2	24.9 ± 1.8	1.1 ± 0.1
	25	0.772 ± 0.142	8000.1	8000.2	8000.2	33.6 ± 3.3	1.4 ± 0.1
	30	0.795 ± 0.131	8000.1	8000.3	8000.4	38.4 ± 4.2	1.5 ± 0.1
45	5	0.828 ± 0.031	20.7	62.1	543.7	10.8 ± 0.2	0.4 ± 0.0
	10	0.820 ± 0.062	689.6	8000.1	8000.6	21.1 ± 2.3	0.8 ± 0.1
	15	0.815 ± 0.080	6630.1	8000.1	8000.2	29.2 ± 2.2	0.9 ± 0.1
	20	0.795 ± 0.100	8000.1	8000.2	8000.3	36.3 ± 4.1	1.1 ± 0.1
	25	0.787 ± 0.123	8000.3	8000.4	8010.3	43.7 ± 2.8	1.4 ± 0.1
	30	0.811 ± 0.162	8000.2	8000.4	8000.4	47.2 ± 4.0	1.7 ± 0.2
55	5	0.864 ± 0.038	66.1	111.6	262.4	15.0 ± 2.1	0.5 ± 0.0

	10	0.840 ± 0.064	38.0	8000.1	8000.2	27.3 ± 2.6	0.8 ± 0.1
	15	0.807 ± 0.088	8000.1	8009.2	8013.6	33.5 ± 2.7	1.0 ± 0.1
	20	0.808 ± 0.096	8000.3	8000.3	8010.1	42.3 ± 3.3	1.2 ± 0.1
	25	0.789 ± 0.000[1]	8000.3	8000.4	NS	49.57 ± 4.04	1.6 ± 0.1
	30	NS	OoM	OoM	OoM	60.4 ± 4.0	1.7 ± 0.2
65	5	0.899 ± 0.062	23.6	65.3	8000.3	31.5 ± 5.0	0.8 ± 0.1
	10	0.896 ± 0.124	1477.3	8000.2	8010.4	57.0 ± 8.2	1.5 ± 0.1
	15	0.882 ± 0.174	8000.2	8000.2	8000.3	72.8 ± 5.6	1.9 ± 0.2
	20	NS	OoM	OoM	OoM	98.6 ± 7.3	2.6 ± 0.2
	25	NS	OoM	OoM	OoM	108.8 ± 9.7	2.8 ± 0.2
	30	NS	OoM	OoM	OoM	132.1 ± 17.8	3.5 ± 0.3

Processing Time

The table clearly shows that the theoretic model processing time reaches the processing time limit sooner as the number of tasks increase. This would represent an exponential curve if there were no time limit. The quantity of variables and constraints to solve are directly related to the number of tasks and fog nodes which increases the complexity of the optimization problem. CPLEX was consistently unable to find a solution in larger simulations. In fact, several dozen tests were made for larger networks, each resulting in a lack of memory error.

The heuristic model processing time was divided in the full processing time and the processing time per time window. The most important value to consider for the heuristic model is the processing time per time window since it must remain under the time window size (2 seconds in this case) for the model to complete its decisions before the next time window begins. The processing time increases primarily based on the number of fog nodes for the network to manage and does not increase significantly based on the task set size. The task set size is never used by the heuristic model since it only considers active tasks in its assignment. The set of active task size is dependent on the task arrival frequency and duration. In all these simulations, both factors were constant across the test series. However, the simulated task generation routine used in the fog node status change algorithm requires slightly more processing time since it considers all completed tasks during the simulation. A design change to the heuristic model could be made to only consider the completed tasks in the past n time windows to reduce the processing time increase rate.

[1] Only 1 out of 5 simulations was able to find an optimal solution

The processing time values are highly dependent on the test environment. A practical utilization of these models would likely have access to significantly higher processing capacity for both models.

Relative Profit

The values for relative profit fluctuations were inversely correlated to the number of fog nodes in a simulation. Notably, networks with small number of tasks and large number of fog nodes (ex: 5 tasks, 30 fog nodes) showed the most significant impact. The reasons for this behaviour is based on the difficulty of the heuristic model to predict the correct fog node state and will be analyzed in Section 4.4.3. Figure 4.4 shows the distribution of the relative profit values from Table 4.24. The relative profit value interval are mostly contained within the 0.78 to 0.88 range when removing simulations where the number of fog nodes far exceeds the number of tasks (0.67 to 0.68 range). The relative profit values showed no correlation with the size of the network other than the specific case described in this paragraph.

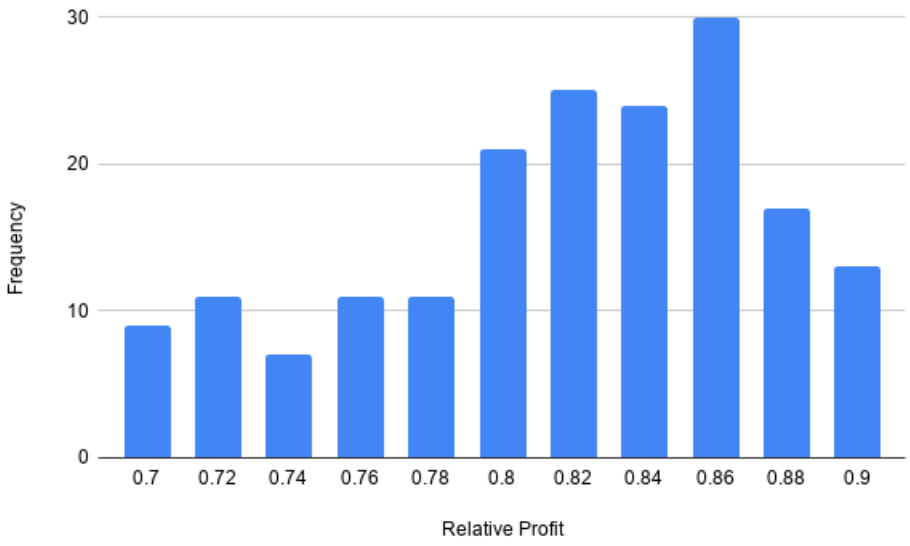

Figure 4.4: Normal Relative Profit Distribution of 40 Tasks and 20 Fog Nodes

4.4.2 Performance Analysis

This thesis presented a heuristic model that performed well in validation and parameter configurations. This section will provide large networks to test and evaluate their performance relative to the theoretical model upper bound when possible. This

section will generate different types of networks where the location of the tasks are concentrated in z areas and can be more easily predicted. The heuristic model should perform better as the value of z decreases. Tasks are simulated around infinity (random), 10, 5 and 3 concentration areas within the network service area. This was used to simulate concentration areas in a typical urban scenario. For example, each centralized area represents densely populated areas with a majority of tasks, while other areas represent rural or sparsely populated areas. These tests were run for 30 simulations in each category.

The location of tasks for the scenarios were calculated using a normal random variable with the central area of the concentration area as μ and one sixth of the size of the grid area along each axis as σ. The algorithm then iterates the value of the random variable until it generates a valid location coordinate (i.e. positive number and under the maximum grid size). Figure 4.5 shows an example of the calculation for a $\mu = 50$ and a grid size of 100. The concentration areas still allow for a large span of randomness in the model. This was designed to represent population drops within a city as described in [68]. As such, although we predict an improvement in performance with fewer concentration areas, this activity aims to not trivialize the heuristic model by setting grid values that are evident as it wouldn't be representative of a practical scenario.

.

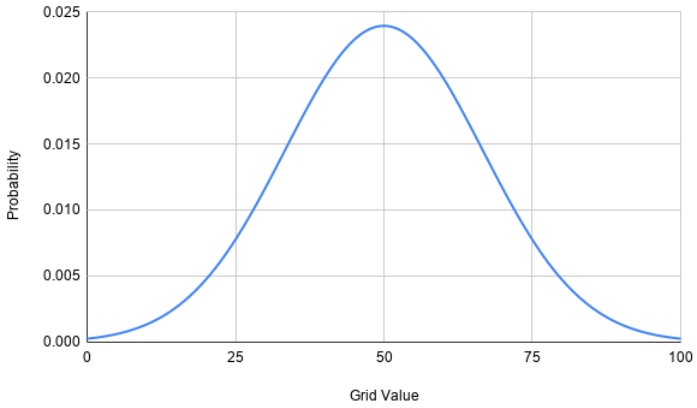

Figure 4.5: Centralized Grid Value Probability Density Function

The first set of tests was run with $n = 40$ tasks and $m = 20$ fog nodes. This was a sufficiently small network to run the theoretic model with a relatively small computing

time as a comparison point to the results obtained in the heuristic model.

The distribution of the relative profit between the heuristic and theoretic model clearly shows an improvement in performance as the number of concentration areas diminish. The notion of relative profit is calculated by dividing the heuristic profit with the upper bound calculated with the theoretic model. The improvement is not significant since the task locations were distributed somewhat widely, but it clearly shows that correlating past and future events benefit the heuristic model. Figure 4.6 shows the histogram distribution of tested events and Figure 4.7 shows the normal distribution generation of tested events.

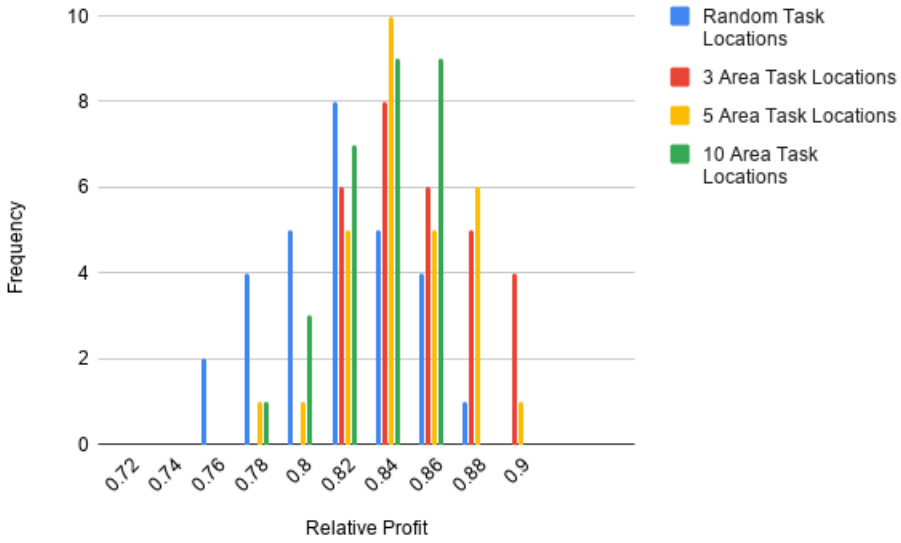

Figure 4.6: Histogram Relative Profit Distribution of 40 Tasks and 20 Fog Nodes

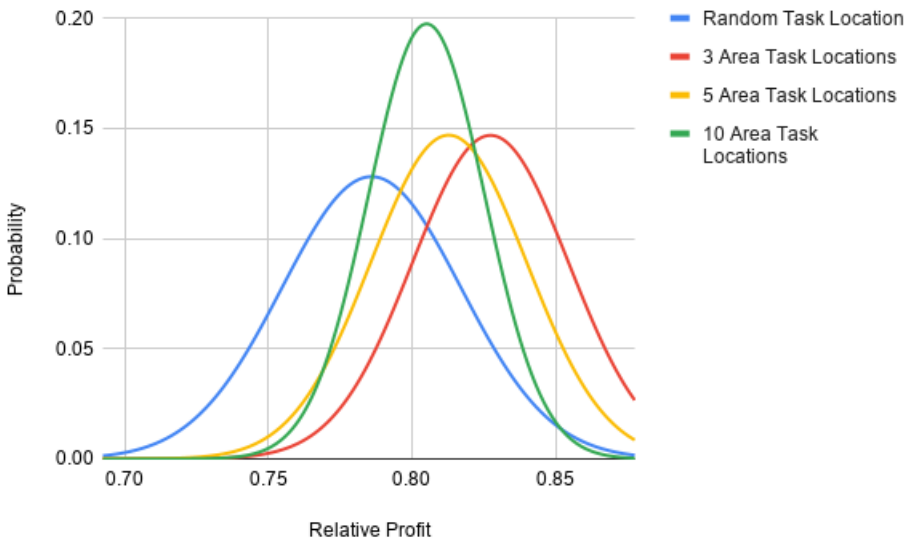

Figure 4.7: Normal Relative Profit Distribution of 40 Tasks and 20 Fog Nodes

The second set of tests was run with $n = 200$ tasks and $m = 20$ fog nodes. These tests were too large to compute results using the theoretical model, but the profit trends to a slight increase as the overall randomness of the task locations diminishes. Figures 4.8 and 4.9 show the histogram and normal distribution of tested events and clearly show an improvement in performance as the number of concentration areas diminish.

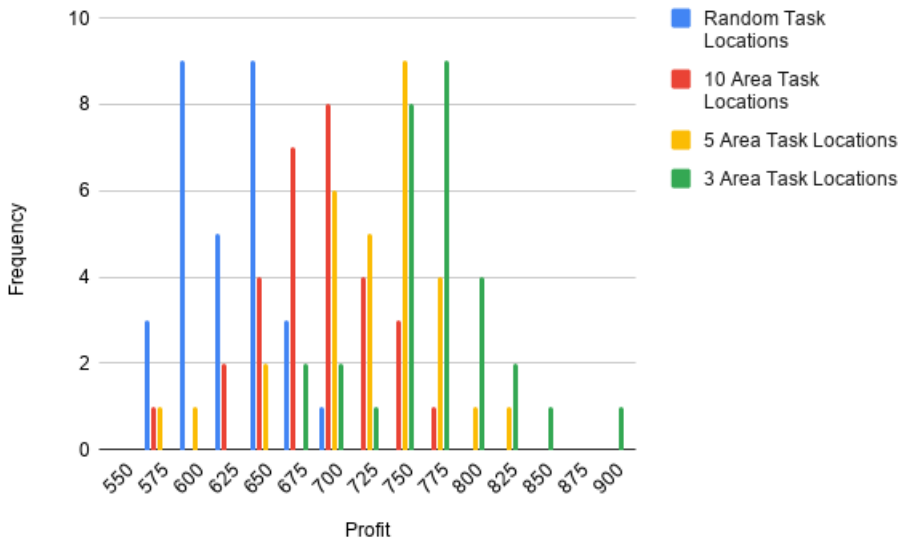

Figure 4.8: Histogram Profit Distribution of 200 Tasks and 20 Fog Nodes

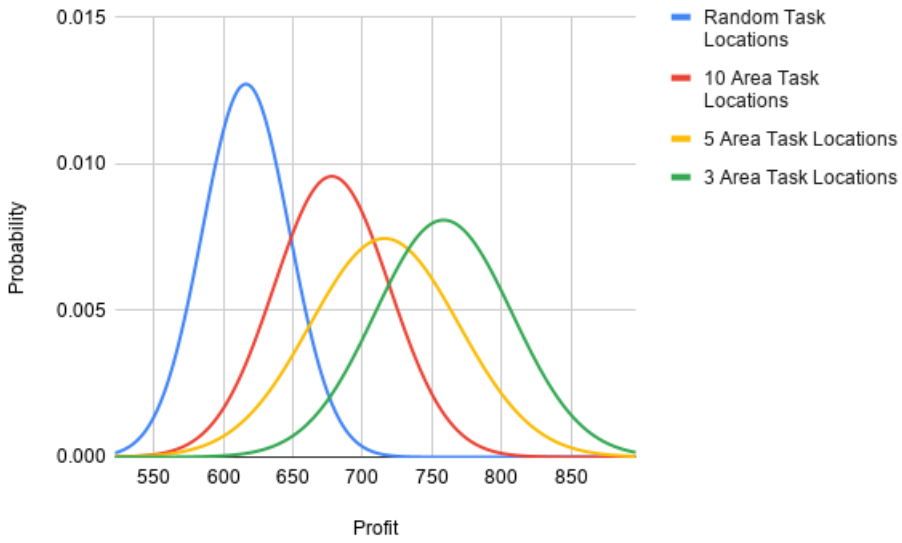

Figure 4.9: Normal Profit Distribution of 200 Tasks and 20 Fog Nodes

These tests justify that a slight correlation between past and future events increase the performance of the heuristic model. Detailed results from this section are included in Appendix C.

4.4.3 Analysis of Differences between the Heuristic and Theoretic Models

The heuristic model inherently behaves differently than the theoretic model in some aspects. This divergence is caused by the lack of information in the heuristic model due to its intent on running in real-time rather than on historical data as with the theoretical model. This change is the largest contributor in the heuristic model making sub-optimal decisions once the task set I is known. Chapter 3 proposes algorithms to circumvent this issue, but it would be unrealistic to expect that both models could achieve the same results in all but trivial networks. In testing, many common sub-optimal decisions were made based on a few categories when compared to the theoretic model. These scenarios will be discussed below using samples of detailed mapping from a fog computing network with 40 tasks and 20 fog nodes. The complete detailed mapping, fog node and task details for this simulation are included in Appendix C. The objective in this section is to show that although these decisions are not optimal, they are actually the best that could be made using the information available to the model.

4.4.3.1 Incorrect Fog Node Status Changes in Early Time Windows

The early time windows inherently have less historical knowledge of tasks than later time windows. As such, the list of simulated tasks will be less representative than later time windows when the model has been exposed to more task elements. Therefore, there are often unnecessary fog nodes that are turned on due to this issue. The heuristic model detects that the fog computing network contains some areas without fog node coverage and attempts to start as few fog nodes as possible to improve the coverage area. The issue during the early time windows is that there isn't a set of simulated tasks that can be used for the model to understand where tasks are likely to appear next and can unnecessarily turn on fog nodes in a location that is unlikely to produce a task.

Most networks should use all fog nodes if designed properly and not have fog nodes positioned in areas without a task demand. It is therefore likely that the fog node will be used in a future time window. However, in cases that another task is not immediately demanded of the network in the next time window, this becomes a sub-optimal decision. Result 3 shows an example of this.

Result 3: Incorrect Fog Node Status Changes in Early Time Windows

1 **Time Window 0** Running fog node: None
2 Start of tasks 1, 2 and 3
3 Tasks 1, 2 and 3 assigned to the cloud
4
5 $\mu_{4-0} \leftarrow 1$, $\mu_{5-0} \leftarrow 1$, $\mu_{11-0} \leftarrow 1$, $\mu_{12-0} \leftarrow 1$ `/* Startup fog nodes 4, 5, 11 and 12 */`
6
7 **Time Window 1** Running fog nodes: 4, 5, 11 and 12
8 Start of tasks 4 and 5
9 $x_{2-4-1} \leftarrow 1$ `/* Task 2 assigned to fog node 4 */`
10 $x_{3-11-1} \leftarrow 1$ `/* Task 3 assigned to fog node 11 */`
11 $x_{5-12-1} \leftarrow 1$ `/* Task 5 assigned to fog node 12 */`
12 Tasks 1 and 4 assigned to the cloud
13
14 $\mu_{7-1} \leftarrow 1$ `/* Startup fog node 5 */`
15 $\theta_{4-1} \leftarrow 1$, $\theta_{5-1} \leftarrow 1$, $\theta_{11-1} \leftarrow 1$, $\theta_{12-1} \leftarrow 1$

l

In the example shown in Result 3, the heuristic model makes 2 decisions that are not prompted from the initial 3 tasks in time window 0: turning on fog nodes 5 and 12 ($\mu_{5-0} = 1$ and $\mu_{12-0} = 1$). These decisions are made by using Algorithm 5, notably the section that calculates the unique coverage surface of a fog node.

In the case of fog node 12, it is used in time window 1 by task 5, but fog node 4 is not needed. Therefore, fog node 4 was not required to be turned on at time window 0 and produced a sub-optimal solution when compared to the theoretic model. However, there isn't a deterministic method to make that decision without knowledge of future events.

In this situation, the heuristic model has no way to make a better decision and should the predictive fog node startup algorithm not run, it would produce a lesser solution in this case since task 5 would not have a fog node match at time window 1. In a realistic scenario, the heuristic algorithm would stream constantly. Therefore, the initial startup inaccuracies wouldn't have a significant impact to a practical scenario where the model would run over long periods of time.

4.4.3.2 Differences for Fog Node Status Change Decisions at the End of a Simulation

The heuristic model does not consider the number of remaining tasks within a simulation. The primary reason is that a practical utilization of this application wouldn't technically have an end. Tasks are expected to be constantly requested from the network although their frequency are likely to change in time. For this reason, the heuristic model was not designed to consider the end of a simulation and makes decisions accordingly although the theoretic model does so. Result 4 shows an example network for 40 tasks and 20 fog nodes at a late time window (no new tasks will start). Powered on fog nodes with a task assigned are in bold. In this scenario, the theoretic model should shut down all fog nodes without an associated task (1, 5 and 15 in this case), but does not do so. The primary reason is that those fog nodes offer a beneficial coverage area of the network and were recently needed in previous time windows. If the simulation was to continue, it is likely that they would be needed and should not be turned off. However, when doing a direct comparison to the theoretic model, it is not an optimal solution.

Result 4: Incorrect Fog Node Status Change Decisions at the End of a Simulation

1 **Time Window 20** Running fog nodes: 1, 2, 3, 4, 5, 6, 9, 10, 11, 14 and 15
2 $x_{31-11-20} \leftarrow 1$ `/* Task 31 assigned to fog node 11 */`
3 $x_{32-10-20} \leftarrow 1$ `/* Task 32 assigned to fog node 10 */`
4 $x_{34-14-20} \leftarrow 1$ `/* Task 34 assigned to fog node 14 */`
5 $x_{35-2-20} \leftarrow 1$ `/* Task 35 assigned to fog node 2 */`
6 $x_{36-9-20} \leftarrow 1$ `/* Task 36 assigned to fog node 9 */`
7 $x_{37-6-20} \leftarrow 1$ `/* Task 37 assigned to fog node 6 */`
8 $x_{38-4-20} \leftarrow 1$ `/* Task 38 assigned to fog node 4 */`
9 $x_{39-3-20} \leftarrow 1$ `/* Task 39 assigned to fog node 3 */`
10 Task 40 assigned to the cloud
11
`/* Operating fog nodes not required */`
12 $\theta_{1-20} \leftarrow 1$ $\theta_{5-20} \leftarrow 1$ $\theta_{15-20} \leftarrow 1$
13
`/* Operating fog nodes required */`
14 $\boldsymbol{\theta_{2-20} \leftarrow 1}$ $\boldsymbol{\theta_{3-20} \leftarrow 1}$ $\boldsymbol{\theta_{4-20} \leftarrow 1}$ $\boldsymbol{\theta_{6-20} \leftarrow 1}$
$\boldsymbol{\theta_{9-20} \leftarrow 1}$ $\boldsymbol{\theta_{10-20} \leftarrow 1}$ $\boldsymbol{\theta_{11-20} \leftarrow 1}$ $\boldsymbol{\theta_{14-20} \leftarrow 1}$

Similarly, the heuristic model will start fog nodes at the last time window based on the results of Algorithm 5. Again, a similar reasoning can be applied where it would be beneficial to the optimization function should the simulation continue.

4.4.3.3 Challenges to shut down fog nodes

Shutting down fog nodes are more challenging to replicate between the theoretic and heuristic models since they heavily rely on knowledge of future events to configure optimally. The heuristic model relies on a given fog node to not have tasks assigned to it (or tasks that can be transferred to another fog node) in order to turn it off. It also uses a series of simulated tasks to predict future tasks and proceed to a quicker shutdown. However, there are differences between both of these routines and the theoretic solution.

Firstly, turning off a fog node after inactivity is inherently not an optimal decision. The fog node should have been turned off previously, but the model was not able to

detect it in time. However, depending on how the maintenance costs are distributed, there is a high probability that a fog node should not be turned off in order to be available for a future task. For example, in the simulated models, the network revenue and costs are the same as most simulations from this chapter (ex: Table 4.1).

A task requiring a inactive, but powered on fog node must be demanded of the network within 7 time windows ($r(cu+cd)/mn$). The probability of occurrence of such an event is likely in all scenarios except where there is a heavy overlap in coverage between fog nodes. Therefore, it is rarely correct in the theoretical model to shut down a fog node immediately when it serves no task. This makes the decision process in the heuristic model very challenging to achieve optimally.

Secondly, the simulation routines are used to predict the next 7 time windows and make a probabilistic decision on turning off a given fog node. The effectiveness of this routing relies on the relevance between past events and future ones which is not always the case. The simulated networks were specifically designed to keep some randomness in the location of tasks to simulate an organic environment. Testing has shown that setting the probability threshold too high leads to fog nodes being turned off too aggressively and requires being turned on preemptively. This is one of the main reasons that the probability threshold for this routine was set to 0.2.

When comparing to the theoretic model, the decisions made in these routines are always less optimal, but there is no deterministic method to reliably make these decisions mode quickly.

4.5 Chapter Summary

This chapter provided a detailed analysis of the theoretic and heuristic models and evaluated the performance of various example networks.

Each constraint of the theoretical model was validated in turn by using specific test networks that trivialized all but one set of constraints to ensure the implementation of the theoretical model performed as expected. The theoretical model processing time was also measured to provide some boundaries on which test networks could be compared with a theoretical model in a reasonable time frame.

The heuristic model parameters were tested and selected over a large number of test networks. The task order manipulation parameters showed a perfect representation of the theoretical model in 93% of tests with the optimal parameters when removing the fog node status change operations. The remaining 7% of networks were close to the theoretical solution with only 1 or 2 operations within the network that differed from the theoretical model. As expected, the fog node status change operations were more difficult to adjust and performance depended on the type of network. The heuristic model depends on a correlation between past and future tasks to perform well. Networks without correlation between these tasks performed more poorly than ones with a strong correlation. Regardless, there were clear parameter values that showed optimal performance in all network types and can be fixed in the heuristic model.

Large scale testing showed that the performance of the heuristic model is inversely proportional to the entropy of tasks, even in networks where a large element of randomness remains. Much of the divergence between the heuristic and theoretical models are contained within specific events that are challenging to predict in an algorithm such as anomalous tasks appearing without a past indicator. In a practical scenario, a service provider would run the heuristic model over a long period and it is expected that the frequency of these anomalies would decrease as the model builds its knowledge and improves its decision making.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

There are many parallels that could be made between fog and cloud computing networks such as their ability to offload processing from the device to a remote processing entity. However, the advantage of fog computing networks lies in the low latency processing that can enable devices to be reduced to simple peripherals (audio, video and Input/Output) and offload all essential and application processing to the fog. These concepts introduce the virtualization of these functions as an essential element to their functionality. To achieve success, the allocation of all functions within a service network must largely be sent to the fog and rarely to the cloud. Advantageous network planning of fog nodes and a smart dynamic allocation of tasks are essential to an evolution to this idea. Concepts such as green computing also play a large role to reduce operating costs to service providers.

In this thesis, a theoretical model was developed and implemented which provided an upper bound limit for a simulated period of the life of a fog computing network. It used a mathematical approach to assigning resources and ensured that profit was maximized while considering individual task revenue and operating costs. Unlike a practical fog computing network, the theoretical model relies on knowledge of the complete task set to make optimal decisions and finally provide a perfect mapping of the input tasks to fog nodes. The computing time of this model was also significant since the size of the optimization problem is directly related to the number of tasks and fog nodes. For example, the model typically took more than 1,000 seconds to solve a network with 35 to 40 network elements (sum of the number of tasks and fog

nodes). Therefore, simulated networks over a few dozen tasks and fog nodes require significant processing time. A time limited model can be used to generate a solution believed to be the optimal profit with a high degree of confidence with only 15% of the processing time. This thesis demonstrated that a 180 second time limited model can be used to reliably generate the optimal profit in networks with under 43 network elements. The full processing model ran for the maximum allowed 5,000 seconds for many of those simulations. For those reasons, the theoretical model is useful for identifying the upper bound of many simulations used in this thesis, but would not provide a value to a service provider seeking a real-time mapping of their network demand.

The heuristic model shifts the role to directly support service providers by enabling them to run in real-time. The processing time of the model was consistently under 2 seconds of processing per time window for a total run time of under 60 seconds for the largest simulations. Comparatively, the theoretic model took over 8,000 seconds for the same simulations. The main contributor of the heuristic model run time is the number of tasks per time window which allows for the computing per time window to remain fairly constant as the total number of tasks increase. For example, a simulation with $|I| = 5$ and $|J| = 15$ completed in an average of 0.4 seconds per time window. The simulations with $|I| = 65$ and $|J| = 15$ completed in an average of 1.9 seconds per time window with a relatively linear relation in processing time for $5 < |I| < 65$.

This thesis used simulated demand, but the design can be easily adapted to a streamed flow of input tasks. Manipulating the task order at each time window proved an essential step to simplifying the best fit assignment heuristic and provided optimal results 93% of simulated cases when removing other factors from the simulations. As expected, modifying the status of fog nodes proved a more complex operation since it can not rely on knowledge of the future to make perfect decisions. Instead, it used a simple probabilistic approach using the current and historic demand on the network to predict which fog nodes required a status change to improve the state of the fog computing network. These operations weren't able to replicate the theoretical optimum, but were objectively correct based on the factors available to the network at that time window.

The heuristic model parameters were tested and ideal values were found for the type

of networks tested. However, a service provider utilizing this technique is recommended to evaluate its historic data and determine the values that are ideal for a given practical application. The peak performance of the heuristic model over all tests was over 95%. However, in most tests, the heuristic model was able to achieve results between 78 and 88% of the theoretical upper bound for various network sizes. There was also a direct correlation between predictability of the input data set and the relative proximity to the upper bound. This relationship is logical since the status change operations of fog nodes rely on the relevance of historic to future input data sets. Interestingly, the heuristic model performed better in larger networks where the impact of a few unpredictable fog node state changes that were optimally best was smaller overall than in networks with fewer assignment operations.

5.2 Overview of Thesis Contributions

There are three major contributions from this thesis:

- The development of a mathematical model to provide the optimal profit of an input set of demanded tasks to a fog computing network. The data sets in this problem have multiple parameters used to constrain the mathematical model to realistic scenarios.
- The development of a heuristic model to provide a real-time solution to optimize profit. The change between exposing elements of the task set in real-time greatly increases the complexity of the decision making required.
- A thorough evaluation of the heuristic model against the mathematical model to provide a performance analysis and determine areas of divergence between both implementations.

5.3 Future Work

The contributions of this thesis have some limitations. The following sub-sections outline items for future work that could possibly alleviate some of these restrictions and extend the functionalities of the models presented in this research.

5.3.1 Task Service Level Agreements

The current vision of the implementation of the dynamic task allocation of fog computing nodes assigns revenue to a service provider at each time window. This simulates a volume-based billing model that would be implemented for charging customers. However, a task-based billing process provides an additional dimension of decisions to the problem.

The implementation of service levels agreements where tasks only generate a profit to the service provider when they are assigned to fog resources for a minimum quantity of time (e.g. 90% of its total duration). This would increase the importance of the predictive de-allocation of tasks implemented in this thesis. In addition, it favours a network that can precisely predict the duration of tasks and meet, but not exceed the quality of service threshold. The assignment decisions would also evolve in over subscribed scenarios to consider predicted task length and profit levels in addition with the green computing elements explored in this thesis.

For these reasons, the implementation of a task service level agreement is an interesting future element of the dynamic task allocation to fog computing networks.

5.3.2 Include Internal Fog Computing Network Routing of Tasks

A network is composed of several network elements that are interconnected. The simulated networks designed in this thesis present fog nodes as purely network components at the edge of the network with no interactions between themselves. A task being presented to the network might be in range of n fog nodes based on the latency requirements. Once a fog node is at capacity, it can't service a new task. However, an evolution of a fog computing network would include fog nodes that both provide computing resources to directly service tasks, but also low latency links to neighboring fog nodes using a high speed point to point technology such as a fibre optic connection.

The addition of routing creates an additional dimension of complexity in both the creation of a tailored routing protocol to accommodate the input data sets. It also creates an additional consideration when positioning fog nodes in a network. Some

fog nodes should be placed in high density areas, but others should be placed in areas with easy access to a backbone network connection to facilitate task routing.

The addition of internal routing to the fog computing network resources would add a significant complexity to task allocation, but allow service providers to increase flexibility in their network planning to better accommodate an evolving network demand.

5.3.3 Improvements to the Input Data Set Resources

A number of assumptions were made in this thesis as an initial step to real-time task assignments in fog network applications. These assumptions were necessary to simplify the problem in order to evaluate the performance of the heuristic in a simulated environment. The model can be adapted to better represent real world applications using a number of changes to the input data set structure.

Firstly, the latency in this thesis was simplified to be the distance between a task and each fog node. This can be improved to include measured values using benchmarks from practical network appliances and communication links. These latency values should include fluctuations based on various network states such as network congestion to better simulate a public network connection. This information should be communicated to the network controller by the fog computing network as part of its input data.

Secondly, additional demanded and supplied resources would improve the model to better approximate the real world. This work could introduce the concept of different types of demand where some tasks do not require every type of resource to be serviced by the fog network.

Finally, the concept of split processing could be introduced where some tasks could be serviced by multiple fog nodes during the same time window to improve the task allocation rate in the network. This would entail some resources (such as memory) to be supplied in duplicate to a task over multiple fog nodes, increasing the demand for those resources on the overall fog computing network. Other resources (such as processing) could be split between fog nodes, reducing the demand for those resources on the individual fog nodes. These considerations would require an adaptation of the assignment algorithms.

5.3.4 Improvement to the Fog Node Status Change Prediction Modules

The fog node status change algorithms had some areas that performed less optimally when compared to the theoretic model. Additional work can be made to improve those predictive status change algorithms to better reproduce the theoretical model allocation and fog node states. For example, the use of machine learning techniques may provide an improved analysis of long term historical network behaviours as part of its decision making.

The inclusion of these concepts could integrate historical data validation benchmarks to the heuristic model during its processing so it can adapt its configuration to better represent the current state of the network.

5.3.5 Decentralization of the Fog Computing Network Controller

The communication protocol between a network controller and its managed elements (i.e. tasks and fog nodes) was outside the scope of this thesis. However, an analysis of the communication requirements (e.g. cost, overhead, position within the network, etc) are important topics of research for the continuing development of the fog computing network architecture.

The network controller is an essential service to the models proposed in this thesis. Its architecture was introduced in Chapter 3 and some design considerations were proposed for service carriers to adopt in order to maintain a robust controller to support a fog computing network. This thesis proposed a network controller as a centralized service. Any operation depends on the controller having access to complete network state information.

The decentralization of the network controller, either partially or completely, is an interesting consideration since it further delegates operations to the network edge. However, this decentralization further complicates the network protocols required for the fog computing network to effectively communicate the overall state of the network between fog nodes. Additional efforts being dedicated to the decentralization of the network controller are recommended.

5.3.6 Network Element Mobility Considerations

Some fog computing network applications, notably Vehicular Ad-Hoc Networks (VANET), inherently require mobility support in a resource allocation model. The models proposed in this thesis could be adapted by changing the network element locations at each time window to support mobility since it can handle task handover in its design. However, their effectiveness under those parameters were not tested as part of the scope of this thesis. It is expected that the performance of the heuristic model will diminish in heavily dynamic environments. Undoubtedly, these types of applications of fog computing networks would benefit from the development of additional real-time allocation models with mobility performance considerations built into their design.

List of References

[1] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on Fog Computing: Architecture, Key Technologies, Applications and Open Issues," *Journal of Network and Computer Applications*, vol. 98, pp. 27–42, 2017.

[2] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, "Cloud RAN for Mobile Networks—a Technology Overview," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 405–426, 2014.

[3] "Canada - 2021 Forecast Highlights." https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Canada_2021_Forecast_Highlights.pdf. Accessed: 2020-05-02.

[4] M. S. Elbamby, M. Bennis, and W. Saad, "Proactive Eedge Ccomputing in Latency-Constrained Fog Networks," in *2017 European Conference on Networks and Communications (EuCNC)*, pp. 1–6, IEEE, 2017.

[5] L. F. Bittencourt, M. M. Lopes, I. Petri, and O. F. Rana, "Towards Virtual Machine Migration in Fog Computing," in *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pp. 1–8, IEEE, 2015.

[6] O. Consortium and Others, "Openfog Rreference Architecture for Fog Computing," *Architecture Working Group*, pp. 1–162, 2017.

[7] L. Jiao, R. Friedman, X. Fu, S. Secci, Z. Smoreda, and H. Tschofenig, "Cloud-Based Computation Offloading for Mobile Devices: State of the Art, Challenges and Opportunities," in *2013 Future Network & Mobile Summit*, pp. 1–11, IEEE, 2013.

[8] W. Lee, K. Nam, H.-G. Roh, and S.-H. Kim, "A Gateway Based Fog Computing Architecture for Wireless Sensors and Actuator Networks," in *2016 18th International Conference on Advanced Communication Technology (ICACT)*, pp. 210–213, IEEE, 2016.

[9] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 416–464, 2017.

[10] M. Satyanarayanan *et al.*, "Pervasive Computing: Vision and Challenges," *IEEE Personal Communications*, vol. 8, no. 4, pp. 10–17, 2001.

[11] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A Survey of Computation Offloading for Mobile Systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, 2013.

[12] R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H.-I. Yang, "The Case for Cyber Foraging," in *Proceedings of the 10th Workshop on ACM SIGOPS European Workshop*, pp. 87–92, ACM, 2002.

[13] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic Resource Allocation and Parallel Execution in the Cloud for Mobile Code offloading," in *2012 Proceedings IEEE Infocom*, pp. 945–953, IEEE, 2012.

[14] S. Yi, C. Li, and Q. Li, "A Survey of Fog Computing: Concepts, Applications and issues," in *Proceedings of the 2015 Workshop on Mobile Big Data*, pp. 37–42, ACM, 2015.

[15] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A Survey of Mobile Cloud computing: Architecture, Applications, and Approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[16] T. X. Tran and D. Pompili, "Joint Task Offloading and Resource Allocation for Multi-Server Mobile-Edge Computing Networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2018.

[17] Z. Lu, J. Zhao, Y. Wu, and G. Cao, "Task Allocation for Mobile Cloud Computing in Heterogeneous Wireless Networks," in *2015 24th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–9, IEEE, 2015.

[18] S. Jošilo and G. Dán, "Decentralized Algorithm for Randomized Task Allocation in Fog Computing Systems," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 85–97, 2018.

[19] D. Puthal, M. S. Obaidat, P. Nanda, M. Prasad, S. P. Mohanty, and A. Y. Zomaya, "Secure and Sustainable Load Balancing of Edge Data Centers in Fog Computing," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 60–65, 2018.

[20] C. Zhu, G. Pastor, Y. Xiao, Y. Li, and A. Ylae-Jaeaeski, "Fog Following Me: Latency and Quality Balanced Task Allocation in Vehicular Fog Computing," in *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–9, IEEE, 2018.

[21] M. Peng, C. Wang, V. Lau, and H. V. Poor, "Fronthaul-Constrained Cloud Radio Access Networks: Insights and Challenges," *arXiv Preprint arXiv:1503.01187*, 2015.

[22] M. Peng, Y. Li, J. Jiang, J. Li, and C. Wang, "Heterogeneous Cloud Radio Access Networks: A New Perspective for Enhancing Spectral and Energy Efficiencies," *arXiv Preprint arXiv:1410.3028*, 2014.

[23] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog Computing Based Radio Access Networks: Issues and Challenges," *arXiv Preprint arXiv:1506.04233*, 2015.

[24] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, "Fog computing: Focusing on Mobile Users at the Edge," *arXiv Preprint arXiv:1502.01815*, 2015.

[25] J. Ni, K. Zhang, X. Lin, and X. S. Shen, "Securing Fog Computing for Internet of Things Applications: Challenges and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 601–628, 2017.

[26] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and its Role in the Internet of Things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, pp. 13–16, ACM, 2012.

[27] K. Mershad and H. Artail, "Finding a Star in a Vehicular Cloud," *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 2, pp. 55–68, 2013.

[28] D. Baby, R. Sabareesh, R. Saravanaguru, and A. Thangavelu, "Vcr: Vehicular Cloud for Road Side Scenarios," in *Advances in Computing and Information Technology*, pp. 541–552, Springer, 2013.

[29] S. Olariu, T. Hristov, and G. Yan, "The Next Paradigm Shift: From Vehicular Networks to Vehicular Clouds," *Mobile Ad Hoc Networking: Cutting Edge Directions*, vol. 56, no. 6, pp. 645–700, 2013.

[30] S. Bitam, A. Mellouk, and S. Zeadally, "VANET-Cloud: a Generic Cloud Computing Model for Vehicular Ad Hoc Networks," *IEEE Wireless Communications*, vol. 22, no. 1, pp. 96–102, 2015.

[31] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected Vehicles: Solutions and Challenges," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 289–299, 2014.

[32] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An Analysis of Internet Content Delivery Systems," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 315–327, 2002.

[33] T. N. Gia, M. Jiang, A.-M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "Fog Computing in Healthcare Internet of Things: A Case Study on ECG Feature Extraction," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pp. 356–363, IEEE, 2015.

[34] F. Haider, D. Zhang, M. St-Hilaire, and C. Makaya, "On the Planning and Design Problem of Fog Computing Networks," *IEEE Transactions on Cloud Computing*, 2018.

[35] D. Zhang, F. Haider, M. St-Hilaire, and C. Makaya, "Model and Algorithms for the Planning of Fog Computing Networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3873–3884, 2019.

[36] A. Tchernykh, U. Schwiegelsohn, V. Alexandrov, and E.-g. Talbi, "Towards Understanding Uncertainty in Cloud Computing Resource Provisioning," *Procedia Computer Science*, vol. 51, pp. 1772–1781, 2015.

[37] D. Kliazovich, J. E. Pecero, A. Tchernykh, P. Bouvry, S. U. Khan, and A. Y. Zomaya, "Ca-dag: Modeling Communication-Aware Applications for Scheduling in Cloud Computing," *Journal of Grid Computing*, vol. 14, no. 1, pp. 23–39, 2016.

[38] U. Schwiegelshohn and A. Tchernykh, "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, 2012.

[39] N. R. Herbst, S. Kounev, and R. Reussner, "Elasticity in Cloud Computing: What it is, and What it is Not," in *Proceedings of the 10th International Conference on Autonomic Computing ({ICAC} 13)*, pp. 23–27, 2013.

[40] S. Agarwal, S. Yadav, and A. K. Yadav, "An Efficient Architecture and Algorithm for Resource Provisioning in Fog Computing," *International Journal of Information Engineering and Electronic Business*, vol. 8, no. 1, p. 48, 2016.

[41] O. Osanaiye, S. Chen, Z. Yan, R. Lu, K.-K. R. Choo, and M. Dlodlo, "From Cloud to Fog Computing: A Review and a Conceptual Live VM Migration Framework," *IEEE Access*, vol. 5, pp. 8284–8300, 2017.

[42] A. Tchernykh, D. Trystram, C. Brizuela, and I. Scherson, "Idle Regulation in Non-Clairvoyant Scheduling of Parallel Jobs," *Discrete Applied Mathematics*, vol. 157, no. 2, pp. 364–376, 2009.

[43] A. Quezada-Pina, A. Tchernykh, J. L. González-García, A. Hirales-Carbajal, J. M. Ramírez-Alcaraz, U. Schwiegelshohn, R. Yahyapour, and V. Miranda-López, "Adaptive Parallel Job Scheduling with Resource Admissible Allocation on Two-Level Hierarchical Grids," *Future Generation Computer Systems*, vol. 28, no. 7, pp. 965–976, 2012.

[44] A. Tchernykh, J. E. Pecero, A. Barrondo, and E. Schaeffer, "Adaptive Energy Efficient Scheduling in Peer-to-Peer Desktop Grids," *Future Generation Computer Systems*, vol. 36, pp. 209–220, 2014.

[45] N. Megow, M. Uetz, and T. Vredeveld, "Models and Algorithms for Stochastic Online Scheduling," *Mathematics of Operations Research*, vol. 31, no. 3, pp. 513–525, 2006.

[46] N. Megow and T. Vredeveld, "Approximation in Preemptive Stochastic Online Scheduling," in *European Symposium on Algorithms*, pp. 516–527, Springer, 2006.

[47] T. Vredeveld, "Stochastic Online Scheduling," *Computer Science-Research and Development*, vol. 27, no. 3, pp. 181–187, 2012.

[48] X. Cai, X. Wu, L. Zhang, and X. Zhou, "Scheduling with Stochastic Approaches," in *Sequencing and Scheduling with Inaccurate Data*, pp. 3–45, Nova Science Publishers, 2014.

[49] S. Kianpisheh, S. Jalili, and N. Charkari, "Predicting Job Wait Time in Grid Environment by Applying Machine Learning Methods on Historical Information," *International Journal of Grid and Distributed Computing*, vol. 5, no. 3, pp. 11–22, 2012.

[50] W. Smith, I. Foster, and V. Taylor, "Predicting Application Run Times Using Historical Information," in *Workshop on Job Scheduling Strategies for Parallel Processing*, pp. 122–142, Springer, 1998.

[51] J. M. Ramírez-Alcaraz, A. Tchernykh, R. Yahyapour, U. Schwiegelshohn, A. Quezada-Pina, J. L. González-García, and A. Hirales-Carbajal, "Job Allocation Strategies with User Run Time Estimates for Online Scheduling in Hierarchical Grids," *Journal of Grid Computing*, vol. 9, no. 1, pp. 95–116, 2011.

[52] M. Aazam and E.-N. Huh, "Fog Computing Micro Datacenter Based Dynamic Resource Estimation and Pricing Model for Internet of Things," in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pp. 687–694, IEEE, 2015.

[53] C. T. Do, N. H. Tran, C. Pham, M. G. R. Alam, J. H. Son, and C. S. Hong, "A Proximal Algorithm for Joint Resource Allocation and Minimizing Carbon Footprint in Geo-Distributed Fog Computing," in *2015 International Conference on Information Networking (ICOIN)*, pp. 324–329, IEEE, 2015.

[54] A. M. Bloch, "Steepest Descent, Linear Programming and Hamiltonian Flows," *Contemp. Math. AMS*, vol. 114, pp. 77–88, 1990.

[55] Z. Xiao, W. Song, and Q. Chen, "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, 2012.

[56] W. Wei, X. Fan, H. Song, X. Fan, and J. Yang, "Imperfect Information Dynamic Stackelberg Game Based Resource Allocation Using Hidden Markov for Cloud Computing," *IEEE Transactions on Services Computing*, vol. 11, no. 1, pp. 78–89, 2016.

[57] P. Lai, Q. He, G. Cui, X. Xia, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Edge User Allocation with Dynamic Quality of Service," in *International Conference on Service-Oriented Computing*, pp. 86–101, Springer, 2019.

[58] F. Haider, "On the Planning and Design Problem of Fog Networks," Master's thesis, Carleton University, 2018.

[59] D. Zhang, "Exact and Approximation Algorithms for the Planning and Design of Fog Networks," Master's thesis, Carleton University, 2018.

[60] J. Ejarque, A. Micsik, R. Sirvent, P. Pallinger, L. Kovacs, and R. M. Badia, "Semantic Resource Allocation with Historical Data Based Predictions," 2010.

[61] S. Schneider, F. Seifert, and A. Sunyaev, "Market Potential Analysis and Branch Network Planning: Application in a German Retail Bank," in *2014 47th Hawaii International Conference on System Sciences*, pp. 1122–1131, IEEE, 2014.

[62] F. Glover and M. Laguna, "Tabu Search," in *Handbook of Combinatorial Optimization*, pp. 2093–2229, Springer, 1998.

[63] J. Kennedy, "Swarm Intelligence," in *Handbook of Nature-Inspired and Innovative Computing*, pp. 187–219, Springer, 2006.

[64] P. J. Van Laarhoven and E. H. Aarts, "Simulated Annealing," in *Simulated Annealing: Theory and Applications*, pp. 7–15, Springer, 1987.

[65] D. Whitley, "A Genetic Algorithm Tutorial," *Statistics and Computing*, vol. 4, no. 2, pp. 65–85, 1994.

[66] N. Funabiki and Y. Takefuji, "A Neural Network Parallel Algorithm for Channel Assignment Problems in Cellular Radio Networks," *IEEE Transactions on Vehicular Technology*, vol. 41, no. 4, pp. 430–437, 1992.

[67] J. A. Suykens and J. Vandewalle, "Least Squares Support Vector Machine Classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[68] M. J. Beckmann, "City Hierarchies and the Distribution of City Size," *Economic Development and Cultural Change*, vol. 6, no. 3, pp. 243–248, 1958.

Appendix A

Theoretical Model Results Additional Data

This appendix presents additional results used in the testing of the theoretical model.

A.1 Theoretical Model Constraint Validation Detailed Results

The following 8 results (Results 5, 6, 7, 8, 9, 10, 11 and 12) present the detailed outcome of the theoretical model for Section 4.2 validating each constraint set.

Result 5: Theoretic model results for Constraint 1

1 **Time Window 0** Running fog node: None
2 Start of task 1
3 $\mu_{10} \leftarrow 1$
4 Task 1 assigned to the cloud
5
6 **Time Window 1** Running fog node: 1
7 End of task 1
8 Start of task 2
9 $x_{2-1-1} \leftarrow 1$; $\theta_{1-1} \leftarrow 1$
10 Task 2 assigned to fog node 1
11
12 **Time Window 2** Running fog node: 1
13 End of task 2
14 Start of task 3
15 $x_{3-1-2} \leftarrow 1$; $\theta_{1-2} \leftarrow 1$
16 Task 3 assigned to fog node 1
17
18 **Time Window 3** Running fog node: 1
19 End of task 3
20 Start of task 4
21 $x_{4-1-3} \leftarrow 1$ $\theta_{1-3} \leftarrow 1$
22 Task 4 assigned to fog node 1
23
24 **Time Window 4** Running fog node: 1
25 End of task 4
26 Start of task 5
27 $x_{5-1-4} \leftarrow 1$ $\theta_{1-4} \leftarrow 1$
28 Task 5 assigned to fog node 1
29
30 **Time Window 5** Running fog node: 1
31 End of task 5
32 $\theta_{1-5} \leftarrow 1$
33
34 $\boldsymbol{Profit = r(x_{2-1-1} + x_{3-1-2} + x_{4-1-3} + x_{5-1-4}) - cu(\mu_{1-0}) - mn(\theta_{1-1} + \theta_{1-2} + \theta_{1-3} + \theta_{1-4} + \theta_{1-5}) = 3.0}$

Result 6: Theoretic model results for Constraint 2

1 **Time Window 0** Running fog node: None

2 Start of tasks 1, 2, 3, 4 and 5

3 $\mu_{1-0} \leftarrow 1$; $\mu_{2-0} \leftarrow 1$; $\mu_{3-0} \leftarrow 1$; $\mu_{4-0} \leftarrow 1$; $\mu_{5-0} \leftarrow 1$

4 Tasks 1, 2, 3, 4 and 5 assigned to the cloud

5

6 **Time Window 1** Running fog nodes: 1, 2, 3, 4, 5

7 $x_{111} \leftarrow 1$; $x_{221} \leftarrow 1$; $x_{331} \leftarrow 1$; $x_{441} \leftarrow 1$; $x_{551} \leftarrow 1$

8 $\theta_{11} \leftarrow 1$; $\theta_{21} \leftarrow 1$; $\theta_{31} \leftarrow 1$; $\theta_{41} \leftarrow 1$; $\theta_{51} \leftarrow 1$

9 Tasks 1, 2, 3, 4 and 5 assigned to fog node 1, 2, 3, 4 and 5

10

11 **Time Window 2** Running fog nodes: 1, 2, 3, 4, 5

12 End of tasks 1, 2, 3, 4, 5

13 $\theta_{12} \leftarrow 1$; $\theta_{22} \leftarrow 1$; $\theta_{32} \leftarrow 1$; $\theta_{42} \leftarrow 1$; $\theta_{52} \leftarrow 1$

14

15 $\boldsymbol{Profit = r(x_{111}+x_{221}+x_{331}+x_{441}+x_{551})-cu(\mu_{10}+\mu_{20}+\mu_{30}+\mu_{40}+\mu_{50})-mn(\theta_{11}+\theta_{12}+\theta_{21}+\theta_{22}+\theta_{31}+\theta_{32}+\theta_{41}+\theta_{42}+\theta_{51}+\theta_{52}) = 1.5}$

Result 7: Theoretic model results for Constraint 3

1 **Time Window 0** Running fog node: None
2 Start of tasks 1, 2, 3, 4 and 5
3 $\mu_{10} \leftarrow 1$
4 Tasks 1, 2, 3, 4 and 5 assigned to the cloud
5
6 **Time Window 1** Running fog node: 1
7 $x_{111} \leftarrow 1$; $x_{211} \leftarrow 1$
8 $\theta_{11} \leftarrow 1$
9 Tasks 1 and 2 assigned to fog node 1 and 2
10 Tasks 3, 4 and 5 assigned to the cloud
11
12 **Time Window 2** Running fog node: 1
13 End of tasks 1, 2, 3, 4, 5
14 $\theta_{12} \leftarrow 1$;
15
16 $\boldsymbol{Profit = r(x_{111} + x_{211}) - cu(\mu_{10}) - mn(\theta_{11} + \theta_{12}) = 1.3}$

Result 8: Theoretic model results for Constraint 6

1 **Time Window 0** Running fog node: None
2 Start of task 1
3 $\mu_{10} \leftarrow 1$
4 Task 1 assigned to the cloud
5
6 **Time Window 1** Running fog node: 1
7 $x_{111} \leftarrow 1$
8 $\theta_{11} \leftarrow 1$
9 Task 1 assigned to fog node 1
10
11 **Time Window 2** Running fog node: 1
12 End of task 1
13 $\theta_{12} \leftarrow 1$;
14
15 $\boldsymbol{Profit = r(x_{111}) - cu(\mu_{10}) - mn(\theta_{11} + \theta_{12}) = 0.3}$

Result 9: Theoretic model results for Constraint 7

1 **Time Window 0** Running fog node: None
2 Start of task 1
3 Task 1 assigned to the cloud
4
5 **Time Window 1** Running fog node: None
6 Task 1 assigned to the cloud
7
8 **Time Window 2** Running fog node: None
9 End of task 1
10
11 $\boldsymbol{Profit = 0}$

Result 10: Theoretic model results for Constraint 8

1 **Time Window 0** Running fog node: None

2

3 **Time Window 1** Running fog node: None

4

5 **Time Window 2** Running fog node: None

6 $\mu_{12} \leftarrow 1$

7

8 **Time Window 3** Running fog node: 1

9 Start of Task 1

10 $x_{113} \leftarrow 1$

11 $\theta_{13} \leftarrow 1$

12 Task 1 assigned to fog node 1

13

14 **Time Window 4** Running fog node: 1

15 $x_{114} \leftarrow 1$

16 $\theta_{14} \leftarrow 1$

17 Task 1 assigned to fog node 1

18

19 **Time Window 5** Running fog node: 1

20 End of task 1

21 $\theta_{15} \leftarrow 1$

22

23 $\boldsymbol{Profit = r(x_{113} + x_{114}) - cu(\mu_{12}) - mn(\theta_{13} + \theta_{14} + \theta_{15}) = 1.2}$

Result 11: Theoretic model results for Constraint 10

Time Window 0 Running fog node: None
    Start of task 1
    Task 1 assigned to the cloud

Time Window 1 Running fog node: None
    Task 1 assigned to the cloud

Time Window 2 Running fog node: None
    End of task 1

$\boldsymbol{Profit = 0}$

Result 12: Theoretic model results for Constraint 11

Time Window 0 Running fog node: None
    Start of task 1
    $\mu_{10} \leftarrow 1$
    Task 1 assigned to the cloud

Time Window 1 Running fog node: 1
    $x_{111} \leftarrow 1$
    $\theta_{11} \leftarrow 1$
    Task 1 assigned to fog node 1

Time Window 2 Running fog node: None
    End of task 1
    $\phi_{12} \leftarrow 1$

$\boldsymbol{Profit = r(x_{111}) - cu(\mu_{11}) - cs(\phi_{12}) - mn(\theta_{11}) = 0.4}$

A.2 Theoretical Model Computational Complexity Detailed Results

Table A.1 presents the complete computational complexity results abbreviated in Section 4.2.9.

Table A.1: Theoretic Model Average Processing Time Results (in seconds)

$\|I\|$	$\|J\|$	Theoretical Model						TL Theoretical Model (180s)					
		Profit			Time			Profit			Time		
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
1	1	0.3	0.7	1.2	0.0	0.1	0.2	1.0	1.0	1.0	0.0	0.0	0.1
	3	0.3	0.8	1.2	0.0	0.1	0.1	1.0	1.0	1.0	0.0	0.1	0.1
	5	0.3	0.7	1.2	0.1	0.1	0.1	1.0	1.0	1.0	0.1	0.1	0.1
	7	0.3	1.0	1.2	0.1	0.1	0.1	1.0	1.0	1.0	0.1	0.1	0.1
	9	0.3	0.8	1.2	0.1	0.1	0.2	1.0	1.0	1.0	0.1	0.1	0.2
	11	0.3	0.7	1.2	0.1	0.1	0.2	1.0	1.0	1.0	0.1	0.1	0.2
	13	1.2	1.2	1.2	0.2	0.2	0.2	1.0	1.0	1.0	0.2	0.2	0.2
	15	0.3	0.8	1.2	0.1	0.2	0.2	1.0	1.0	1.0	0.1	0.2	0.2
	17	0.3	0.8	1.2	0.1	0.2	0.3	1.0	1.0	1.0	0.1	0.2	0.3
	19	0.3	1.0	1.2	0.2	0.3	0.3	1.0	1.0	1.0	0.2	0.3	0.3
	21	0.3	1.0	1.2	0.2	0.3	0.4	1.0	1.0	1.0	0.2	0.3	0.4
6	1	3.0	7.2	9.0	0.1	0.1	0.1	1.0	1.0	1.0	0.1	0.1	0.1
	3	7.0	9.9	13.1	0.1	0.2	0.2	1.0	1.0	1.0	0.1	0.2	0.3
	5	10.2	12.1	14.0	0.4	1.2	3.8	1.0	1.0	1.0	0.4	1.2	3.8
	7	12.1	13.1	14.0	0.8	1.1	1.5	1.0	1.0	1.0	0.8	1.1	1.6
	9	10.5	12.2	14.1	0.7	1.5	2.7	1.0	1.0	1.0	0.6	1.5	2.7
	11	13.0	13.3	14.0	1.5	2.7	3.7	1.0	1.0	1.0	1.5	2.7	3.5
	13	12.1	13.4	14.0	2.4	2.7	3.4	1.0	1.0	1.0	1.9	2.5	3.2
	15	13.1	13.8	14.0	2.5	3.6	6.1	1.0	1.0	1.0	2.4	3.6	6.1
	17	12.1	13.3	14.0	2.7	5.1	7.9	1.0	1.0	1.0	2.7	5.0	7.8
	19	11.1	13.3	14.9	1.9	4.2	6.5	1.0	1.0	1.0	1.9	4.1	6.2
	21	12.1	13.4	15.9	4.4	6.6	11.1	1.0	1.0	1.0	4.5	6.5	10.4
11	1	4.8	10.5	17.7	0.1	0.1	0.1	1.0	1.0	1.0	0.1	0.1	0.1
	3	15.1	18.9	25.6	0.4	0.6	0.8	1.0	1.0	1.0	0.4	0.6	0.8
	5	19.2	24.5	28.5	1.0	2.1	3.7	1.0	1.0	1.0	1.0	2.1	3.7
	7	23.6	25.7	27.8	1.3	2.5	3.6	1.0	1.0	1.0	1.2	2.5	3.6

	9	24.6	25.2	26.6	1.9	4.0	6.5	1.0	1.0	1.0	1.9	4.0	6.6
	11	23.9	25.8	28.5	3.4	8.0	13.3	1.0	1.0	1.0	3.4	8.0	13.4
	13	25.8	27.4	29.5	7.1	20.2	55.5	1.0	1.0	1.0	7.1	20.3	55.1
	15	25.7	26.6	28.7	7.1	26.7	83.2	1.0	1.0	1.0	7.1	25.5	83.2
	17	25.5	26.1	26.8	12.6	81.9	241.7	1.0	1.0	1.0	12.7	69.6	180.0
	19	25.8	28.3	31.3	15.1	82.6	329.2	1.0	1.0	1.0	15.0	52.8	180.0
	21	24.8	27.3	28.6	19.4	86.1	269.7	1.0	1.0	1.0	19.3	69.8	180.0
16	1	7.5	12.7	22.6	0.1	0.1	0.2	1.0	1.0	1.0	0.1	0.1	0.2
	3	35.7	39.2	43.1	0.8	1.7	2.8	1.0	1.0	1.0	0.9	1.7	2.9
	5	34.4	38.0	42.0	2.0	4.0	7.1	1.0	1.0	1.0	2.0	4.0	7.1
	7	39.1	41.7	43.1	2.5	5.6	9.6	1.0	1.0	1.0	2.5	5.7	9.7
	9	39.1	40.9	42.2	7.9	12.5	20.6	1.0	1.0	1.0	8.0	12.5	20.4
	11	38.8	40.6	43.9	14.2	25.1	43.0	1.0	1.0	1.0	14.3	22.6	43.1
	13	39.8	41.5	44.1	14.6	39.0	79.1	1.0	1.0	1.0	16.3	40.2	83.3
	15	37.0	40.4	43.2	21.3	33.5	68.4	1.0	1.0	1.0	20.5	33.8	71.7
	17	39.0	41.2	43.1	21.1	59.0	185.6	1.0	1.0	1.0	21.2	58.5	180.0
	19	1.9	45.0	120.4	24.0	59.0	120.4	1.0	1.0	1.0	24.0	59.0	119.7
	21	38.0	41.5	44.1	27.8	1486.4	5000.1	1.0	1.0	1.0	27.8	107.5	180.0
21	1	9.3	20.6	32.2	0.2	0.2	0.3	1.0	1.0	1.0	0.2	0.2	0.3
	3	25.9	35.4	56.9	0.7	1.1	1.6	1.0	1.0	1.0	0.7	1.1	1.6
	5	50.4	55.0	57.3	3.0	8.8	19.1	1.0	1.0	1.0	3.0	8.8	19.2
	7	51.0	53.0	56.5	3.4	10.1	19.9	1.0	1.0	1.0	3.4	10.1	20.1
	9	48.9	53.1	55.6	5.7	15.9	23.0	1.0	1.0	1.0	5.7	16.1	23.2
	11	52.8	56.9	59.5	24.7	29.5	38.8	1.0	1.0	1.0	24.5	29.7	39.4
	13	55.4	57.1	59.6	24.6	422.3	1399.7	1.0	1.0	1.0	24.5	90.6	180.0
	15	51.6	55.2	60.1	42.7	2163.6	5000.1	1.0	1.0	1.0	43.0	138.2	180.1
	17	53.1	55.5	57.6	46.9	2048.6	5009.8	1.0	1.0	1.0	46.9	118.6	180.1
	19	52.5	56.1	61.4	31.4	172.3	555.2	1.0	1.0	1.0	31.3	97.4	180.0
	21	52.8	55.2	57.6	57.2	2747.6	5005.8	1.0	1.0	1.0	57.1	136.5	180.1
26	1	12.0	20.7	39.0	0.2	0.2	0.3	1.0	1.0	1.0	0.2	0.2	0.3
	3	35.0	52.4	70.1	1.5	2.3	3.3	1.0	1.0	1.0	1.5	2.3	3.3
	5	59.6	65.2	71.0	5.6	8.2	10.1	1.0	1.0	1.0	5.7	8.3	10.3
	7	62.8	67.2	70.3	9.2	16.0	23.3	1.0	1.0	1.0	9.1	16.0	23.2
	9	66.4	69.1	70.5	10.3	19.8	26.3	1.0	1.0	1.0	10.4	19.8	26.0
	11	64.7	68.9	72.7	16.1	28.7	46.8	1.0	1.0	1.0	17.6	37.5	59.3
	13	66.2	70.1	74.7	42.7	1360.8	5000.1	1.0	1.0	1.0	45.5	116.9	180.0
	15	64.4	67.2	70.9	33.4	698.8	3134.9	1.0	1.0	1.0	33.4	103.6	180.0

	17	67.9	69.7	71.9	101.7	3071.5	5005.1	1.0	1.0	1.0	102.3	164.5	180.1
	19	67.8	70.0	71.7	48.8	2114.8	5000.1	1.0	1.0	1.0	48.7	134.9	180.1
	21	66.9	70.3	73.9	415.0	3286.5	5000.1	0.0	0.8	1.0	180.0	180.1	180.2
31	1	13.8	31.9	49.6	0.2	0.3	0.3	1.0	1.0	1.0	0.3	0.3	0.3
	3	42.1	65.9	74.2	1.8	2.4	3.0	1.0	1.0	1.0	1.8	2.4	3.0
	5	70.6	77.6	86.8	5.2	7.6	10.1	1.0	1.0	1.0	5.1	7.6	10.1
	7	77.7	80.8	83.2	9.6	13.4	22.7	1.0	1.0	1.0	9.8	13.2	22.1
	9	76.8	82.6	85.8	20.6	39.9	76.8	1.0	1.0	1.0	20.3	39.2	76.1
	11	80.2	83.1	87.1	22.1	627.0	2862.5	1.0	1.0	1.0	21.8	98.0	180.0
	13	78.5	83.8	86.7	59.7	2088.3	5007.0	1.0	1.0	1.0	60.8	132.6	180.0
	15	82.6	84.8	88.5	83.6	2348.1	5000.1	1.0	1.0	1.0	81.0	160.3	180.1
	17	80.9	84.4	89.9	83.7	2495.0	5000.1	0.0	0.8	1.0	83.5	160.7	180.1
	19	78.6	81.1	84.7	119.2	1572.7	5000.1	1.0	1.0	1.0	118.2	160.3	180.1
	21	85.1	85.5	86.1	242.1	3194.5	5000.2	0.0	0.6	1.0	180.0	180.0	180.0
36	1	15.6	28.4	52.5	0.3	0.4	0.4	1.0	1.0	1.0	0.3	0.3	0.4
	3	68.3	83.0	94.0	3.0	4.0	5.3	1.0	1.0	1.0	3.0	4.0	5.3
	5	95.4	96.6	98.1	5.0	7.8	11.1	1.0	1.0	1.0	5.0	7.8	11.1
	7	91.7	97.5	101.8	9.2	35.9	61.0	1.0	1.0	1.0	8.9	35.8	62.0
	9	96.1	97.5	99.8	69.6	1104.8	5000.1	1.0	1.0	1.0	63.4	121.7	180.1
	11	94.1	97.8	99.5	69.3	1085.9	5000.1	1.0	1.0	1.0	69.9	118.7	180.1
	13	92.4	98.2	102.7	99.3	886.5	2183.5	1.0	1.0	1.0	99.0	145.6	180.0
	15	95.3	99.3	103.1	303.1	3125.3	5007.0	1.0	1.0	1.0	180.0	180.0	180.1
	17	95.0	96.7	98.3	163.8	2161.4	5009.3	0.0	0.8	1.0	165.9	177.2	180.1
	19	96.1	100.3	102.5	294.5	4060.8	5009.2	0.0	0.8	1.0	180.0	180.1	180.1
	21	94.1	99.1	104.1	334.6	4067.0	5000.2	0.0	0.0	0.0	180.0	180.0	180.0
41	1	19.2	32.6	60.3	0.3	0.4	0.5	1.0	1.0	1.0	0.3	0.4	0.5
	3	58.3	72.9	95.8	2.1	3.0	4.4	1.0	1.0	1.0	2.1	3.0	4.4
	5	84.7	104.2	113.6	4.1	18.4	53.7	1.0	1.0	1.0	4.0	18.4	54.0
	7	107.8	110.0	113.4	15.0	39.2	67.2	1.0	1.0	1.0	15.3	38.4	63.4
	9	109.4	113.2	119.0	34.0	739.8	2709.2	1.0	1.0	1.0	33.7	95.7	180.0
	11	109.6	112.9	118.1	42.3	2076.8	5000.1	1.0	1.0	1.0	42.8	144.1	180.1
	13	107.9	112.5	117.7	81.9	2118.6	5005.1	1.0	1.0	1.0	83.5	142.8	180.1
	15	108.6	112.5	120.9	257.3	2480.6	5000.2	0.0	0.6	0.9	180.0	180.0	180.1
	17	108.5	113.1	117.2	0.0	1473.1	5000.2	0.0	0.6	1.0	152.5	174.5	180.1
	19	110.3	112.4	114.6	165.8	4035.4	5010.5	0.0	0.2	1.0	164.5	176.9	180.0
	21	110.3	110.7	111.3	436.1	3182.6	5000.2	0.0	0.2	1.0	180.0	180.0	180.0
46	1	21.0	32.1	73.8	0.3	0.3	0.5	1.0	1.0	1.0	0.3	0.4	0.5

	3	61.9	87.0	122.1	2.3	4.6	10.4	1.0	1.0	1.0	2.3	4.6	10.4
	5	109.7	121.3	124.7	7.0	11.7	18.7	1.0	1.0	1.0	7.1	11.3	16.8
	7	123.2	125.0	126.7	14.0	229.0	958.8	1.0	1.0	1.0	13.9	73.9	180.0
	9	117.9	125.6	129.7	18.4	1444.4	5000.0	1.0	1.0	1.0	20.1	129.4	180.1
	11	125.1	127.3	128.7	129.2	2139.2	5007.1	1.0	1.0	1.0	129.1	169.9	180.1
	13	121.2	126.7	132.1	101.8	2098.4	5002.3	1.0	1.0	1.0	101.5	149.3	180.1
	15	125.8	127.6	129.8	292.9	2342.7	5000.2	0.0	0.8	1.0	180.0	180.0	180.1
	17	122.9	128.3	131.0	198.8	2495.4	5000.1	0.0	0.4	1.0	180.0	180.0	180.0
	19	116.3	123.9	127.7	327.3	4065.6	5000.2	0.0	0.0	0.0	180.0	180.0	180.1
	21	126.9	128.7	131.4	3519.3	4704.7	5003.7	0.0	0.0	0.0	180.0	180.0	180.1
51	1	23.7	50.3	76.7	0.3	0.5	0.6	1.0	1.0	1.0	0.3	0.5	0.6
	3	72.7	100.9	144.4	2.7	6.5	13.3	1.0	1.0	1.0	2.6	6.3	13.3
	5	113.7	133.8	141.1	10.5	24.6	59.5	1.0	1.0	1.0	10.8	24.9	59.5
	7	128.9	133.7	137.8	9.4	26.7	51.4	1.0	1.0	1.0	9.4	27.2	51.0
	9	137.7	140.3	144.5	46.7	69.1	99.5	1.0	1.0	1.0	46.2	68.4	99.3
	11	137.3	139.8	142.9	87.8	1163.3	5010.3	1.0	1.0	1.0	87.9	157.0	180.0
	13	134.8	140.8	145.8	135.3	3079.9	5000.1	1.0	1.0	1.0	136.1	171.3	180.1
	15	140.2	143.9	150.8	5000.1	5001.8	5008.6	0.0	0.2	0.9	180.0	180.0	180.0
	17	137.3	140.9	144.5	589.1	4121.1	5011.1	0.0	0.0	0.0	180.0	180.0	180.1
	19	137.8	143.3	147.1	1318.9	4263.9	5000.2	0.0	0.0	0.0	180.0	180.0	180.1
	21	138.1	143.1	147.0	3869.8	4719.4	5007.3	0.0	0.0	0.0	180.0	180.0	180.1

Appendix B

Heuristic Model Results Additional Data

This appendix presents additional results used in the configuration testing of the heuristic model.

B.1 Task Order Manipulation Additional Data

Tables B.1 and B.2 present the detailed results generated for Section 4.3.1 (task order manipulation parameters).

Table B.1: Task Manipulation Weight Testing Detailed Example Results

Test Number	Task Density	Heuristic Density	Upper Bound	Comment
1	4	25	25	
2	4	28	28	
3	4	26	26	
4	4	**28***	**29**	Fog node options provided the wrong order, prioritizing age would have been better.
5	4	27	27	
6	4	24	24	
7	4	28	28	
8	4	26	26	
9	4	26	26	
10	4	23	23	
1	6	22	22	
2	6	20	20	
3	6	22	22	
4	6	23	23	
5	6	21	21	
6	6	23	23	
7	6	19	19	
8	6	22	22	
9	6	20	20	
10	6	22	22	
1	8	**18**	**19**	Required a seemingly random order for one time window.
2	8	17	17	
3	8	**16**	**17**	Required a seemingly random order for one time window.
4	8	17	17	
5	8	18	18	
6	8	18	18	
7	8	17	17	
8	8	17	17	
9	8	20	20	
10	8	16	16	
1	10	15	15	
2	10	18	18	
3	10	15	15	
4	10	15	15	
5	10	13	13	
6	10	17	17	
7	10	17	17	
8	10	15	15	
9	10	16	16	
10	10	16	16	
1	12	16	16	
2	12	14	14	
3	12	15	15	
4	12	35	35	
5	12	14	14	
6	12	**15**	**17**	Required a seemingly random order for one time window.
7	12	14	14	
8	12	15	15	
9	12	15	15	
10	12	16	16	

Table B.2: Task Manipulation Weight Testing Detailed Example Results

Test Number	Task Density	Heuristic Density	Upper Bound	Comment
1	4	31	31	
2	4	28	28	
3	4	**33***	**34**	Fog node options provided the wrong order, prioritizing age would have been better.
4	4	35	35	
5	4	36	36	
6	4	**32**	**33**	Required a seemingly random order for one time window.
7	4	36	36	
8	4	35	35	
9	4	32	32	
10	4	29	29	
1	6	27	27	
2	6	26	26	
3	6	22	22	
4	6	**24***	**26**	Fog node options provided the wrong order, prioritizing age would have been better.
5	6	28	28	
6	6	**25**	**26**	Required a seemingly random order for one time window.
7	6	28	28	
8	6	26	26	
9	6	27	27	
10	6	24	24	
1	8	23	23	
2	8	22	22	
3	8	21	21	
4	8	23	23	
5	8	22	22	
6	8	23	23	
7	8	22	22	
8	8	23	24	
9	8	20	20	
10	8	24	24	
1	10	19	19	
2	10	20	20	
3	10	19	19	
4	10	20	20	
5	10	19	19	
6	10	20	20	
7	10	19	19	
8	10	20	20	
9	10	20	20	
10	10	21	21	
1	12	19	19	
2	12	19	19	
3	12	17	17	
4	12	18	18	
5	12	15	15	
6	12	19	19	
7	12	19	19	
8	12	18	18	
9	12	18	18	
10	12	18	18	

B.2 Fog Node Memory Threshold Additional Data

Figures B.3, B.4, B.5, B.6, B.7, B.8, B.9, B.10, B.11 and B.12 present the fog node memory results for each of the 10 tests averaged for Section 4.3.2.1.

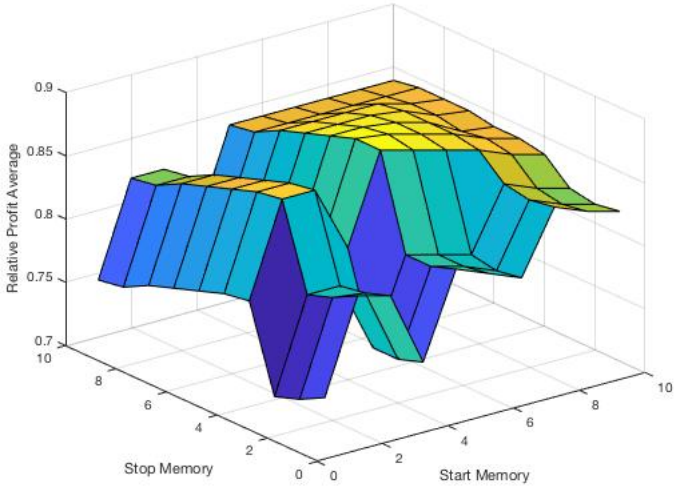

Figure B.1: Heuristic Model Start Stop Memory Results Test 1).

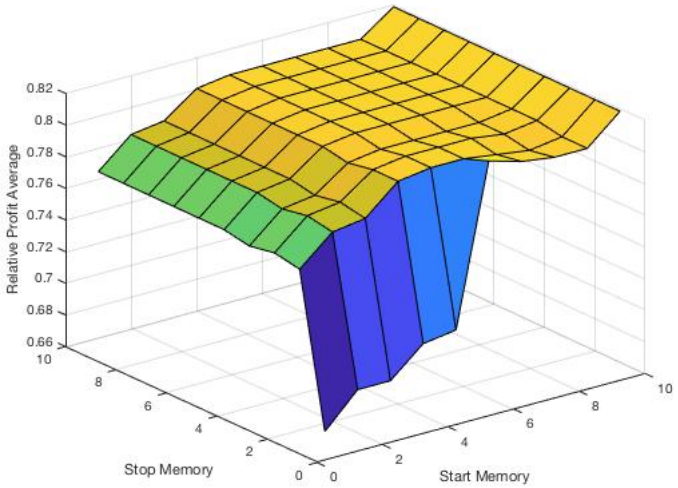

Figure B.2: Heuristic Model Start Stop Memory Results Test 2).

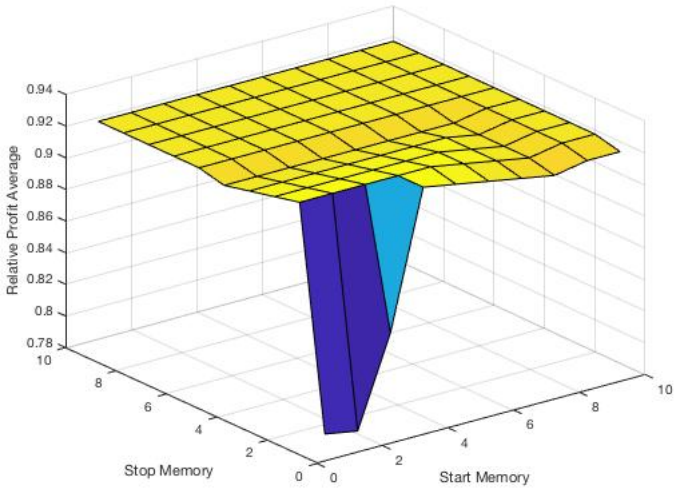

Figure B.3: Heuristic Model Start Stop Memory Results Test 3).

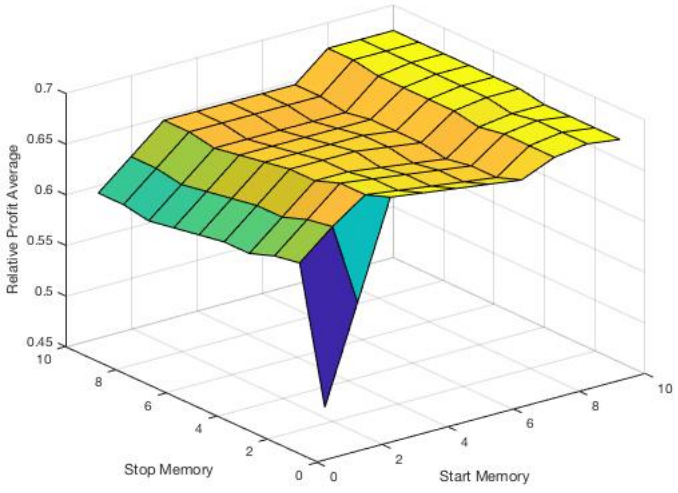

Figure B.4: Heuristic Model Start Stop Memory Results Test 4).

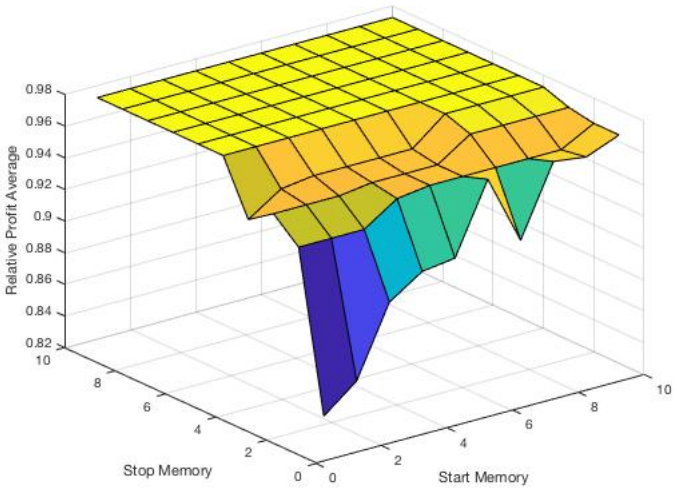

Figure B.5: Heuristic Model Start Stop Memory Results Test 5).

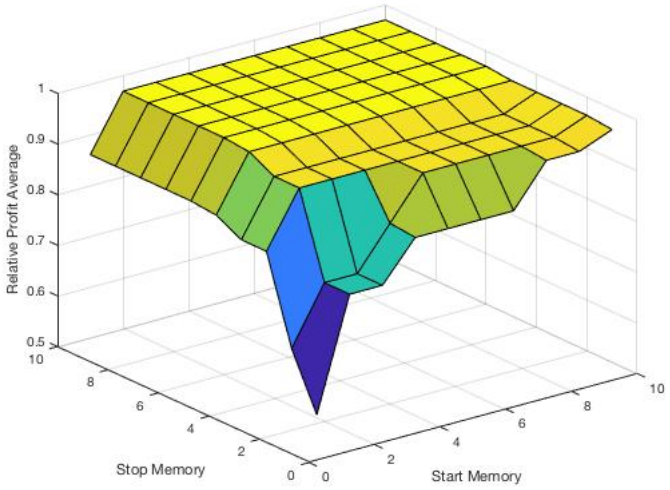

Figure B.6: Heuristic Model Start Stop Memory Results Test 6).

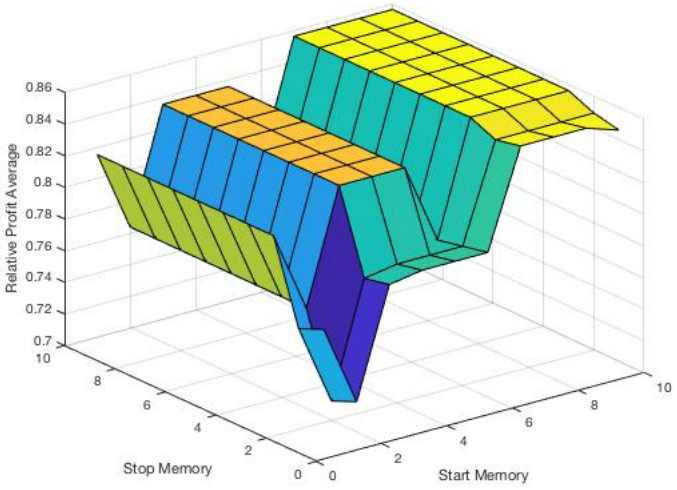

Figure B.7: Heuristic Model Start Stop Memory Results Test 7).

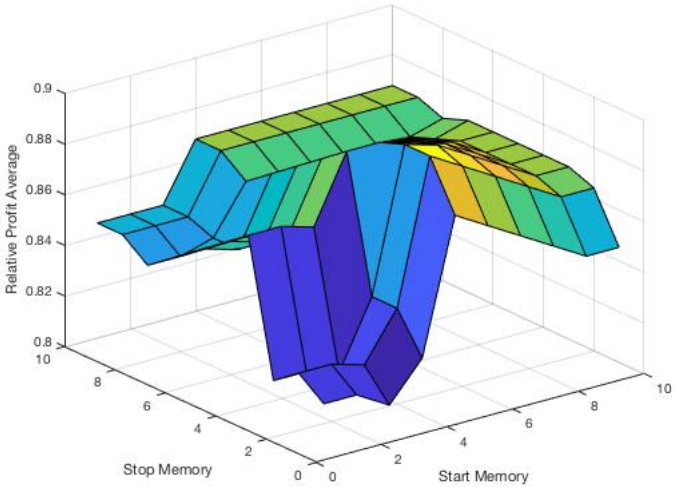

Figure B.8: Heuristic Model Start Stop Memory Results Test 8).

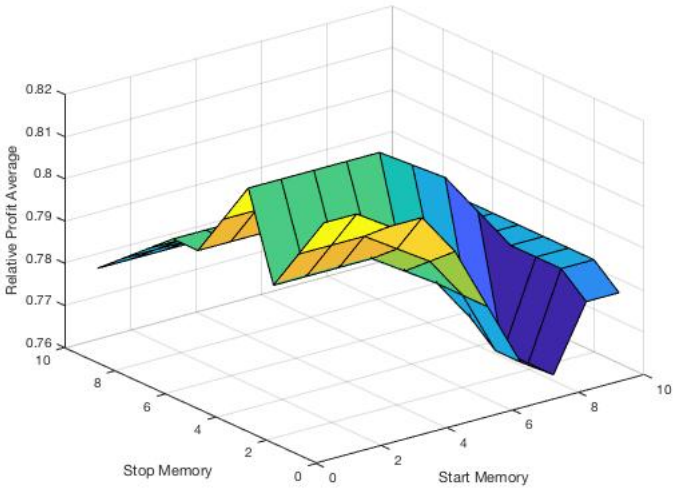

Figure B.9: Heuristic Model Start Stop Memory Results Test 9).

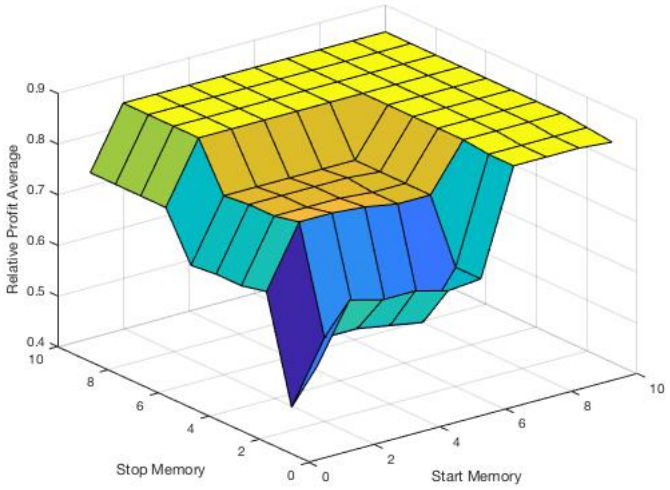

Figure B.10: Heuristic Model Start Stop Memory Results Test 10).

B.3 Fog node Simulated Tasks Additional Data

Tables B.3, B.3, B.4, B.5, B.6, B.7, B.8, B.9, B.10, B.11 and B.12 present the complete results for simulated tasks used for Section 4.3.2.2.

Table B.3: Fog Node State Manipulation Simulated Tasks Test1 Results

Nb Tasks	Nb Fog Nodes	Number of Simulated Tasks	Heuristic Profit (normalized)	Processing Time
20	5	1	0.83	0.20
20	5	3	0.90	0.41
20	5	5	0.90	0.45
20	5	7	0.86	0.52
20	5	9	0.86	0.62
20	5	11	0.89	0.63
20	5	13	0.92	0.92
20	5	15	0.92	1.02
20	5	17	0.92	1.12
20	5	19	0.92	1.22
20	5	21	0.92	1.31

Table B.4: Fog Node State Manipulation Simulated Tasks Test2 Results

Nb Tasks	Nb Fog Nodes	Number of Simulated Tasks	Heuristic Profit (normalized)	Processing Time
20	5	1	0.91	0.25
20	5	3	0.95	0.46
20	5	5	0.95	0.49
20	5	7	0.92	0.51
20	5	9	0.91	0.55
20	5	11	0.88	0.67
20	5	13	0.91	0.76
20	5	15	0.87	0.80
20	5	17	0.94	1.15
20	5	19	0.94	1.24
20	5	21	0.94	1.33

Table B.5: Fog Node State Manipulation Simulated Tasks Test3 Results

Nb Tasks	Nb Fog Nodes	Number of Simulated Tasks	Heuristic Profit (normalized)	Processing Time
20	5	1	0.85	0.20
20	5	3	0.90	0.39
20	5	5	0.89	0.47
20	5	7	0.89	0.51
20	5	9	0.84	0.57
20	5	11	0.82	0.58
20	5	13	0.91	0.89
20	5	15	0.91	1.04
20	5	17	0.93	1.20
20	5	19	0.89	1.32
20	5	21	0.93	1.57

Table B.6: Fog Node State Manipulation Simulated Tasks Test4 Results

Nb Tasks	Nb Fog Nodes	Number of Simulated Tasks	Heuristic Profit (normalized)	Processing Time
20	5	1	0.89	0.31
20	5	3	0.89	0.45
20	5	5	0.89	0.50
20	5	7	0.89	0.60
20	5	9	0.92	0.66
20	5	11	0.95	0.83
20	5	13	0.96	1.06
20	5	15	0.95	1.11
20	5	17	0.95	1.15
20	5	19	0.95	1.28
20	5	21	0.96	1.40

Table B.7: Fog Node State Manipulation Simulated Tasks Test5 Results

Nb Tasks	Nb Fog Nodes	Number of Simulated Tasks	Heuristic Profit (normalized)	Processing Time
20	5	1	0.86	0.25
20	5	3	0.92	0.38
20	5	5	0.89	0.42
20	5	7	0.89	0.51
20	5	9	0.92	0.68
20	5	11	0.86	0.74
20	5	13	0.93	0.95
20	5	15	0.92	1.05
20	5	17	0.92	1.15
20	5	19	0.91	1.21
20	5	21	0.93	1.41

Table B.8: Fog Node State Manipulation Simulated Tasks Test6 Results

Nb Tasks	Nb Fog Nodes	Number of Simulated Tasks	Heuristic Profit (normalized)	Processing Time
20	5	1	0.90	0.20
20	5	3	0.95	0.42
20	5	5	0.95	0.50
20	5	7	0.90	0.47
20	5	9	0.87	0.62
20	5	11	0.93	0.88
20	5	13	0.96	1.06
20	5	15	0.89	0.94
20	5	17	0.96	1.30
20	5	19	0.96	1.43
20	5	21	0.96	1.56

Table B.9: Fog Node State Manipulation Simulated Tasks TEst7 Results

Nb Tasks	Nb Fog Nodes	Number of Simulated Tasks	Heuristic Profit (normalized)	Processing Time
20	5	1	0.91	0.29
20	5	3	0.97	0.49
20	5	5	0.96	0.58
20	5	7	0.93	0.63
20	5	9	0.91	0.61
20	5	11	0.87	0.72
20	5	13	0.98	1.11
20	5	15	0.98	1.24
20	5	17	0.98	1.36
20	5	19	0.98	1.49
20	5	21	0.98	1.60

Table B.10: Fog Node State Manipulation Simulated Tasks Test8 Results

Nb Tasks	Nb Fog Nodes	Number of Simulated Tasks	Heuristic Profit (normalized)	Processing Time
20	5	1	0.85	0.20
20	5	3	0.94	0.44
20	5	5	0.92	0.56
20	5	7	0.86	0.56
20	5	9	0.92	0.75
20	5	11	0.86	0.81
20	5	13	0.96	1.06
20	5	15	0.96	1.19
20	5	17	0.96	1.33
20	5	19	0.96	1.41
20	5	21	0.96	1.54

Table B.11: Fog Node State Manipulation Simulated Tasks Test9 Results

Nb Tasks	Nb Fog Nodes	Number of Simulated Tasks	Heuristic Profit (normalized)	Processing Time
20	5	1	0.88	0.26
20	5	3	0.91	0.52
20	5	5	0.91	0.59
20	5	7	0.88	0.58
20	5	9	0.85	0.67
20	5	11	0.86	0.75
20	5	13	0.90	1.00
20	5	15	0.87	1.06
20	5	17	0.91	1.29
20	5	19	0.91	1.40
20	5	21	0.91	1.55

Table B.12: Fog Node State Manipulation Simulated Tasks Test10 Results

Nb Tasks	Nb Fog Nodes	Number of Simulated Tasks	Heuristic Profit (normalized)	Processing Time
20	5	1	0.91	0.26
20	5	3	0.94	0.48
20	5	5	0.94	0.55
20	5	7	0.94	0.67
20	5	9	0.93	0.72
20	5	11	0.88	0.80
20	5	13	0.94	1.02
20	5	15	0.94	1.13
20	5	17	0.94	1.26
20	5	19	0.94	1.38
20	5	21	0.94	1.49

B.4 Fog node Status Change Sensitivity Threshold

Tables B.13 and B.14 present the complete results for the status change sensitivity threshold used for Section 4.3.2.3.

Table B.13: Fog Node Simulated Status Change Probability Threshold Bulk Results (15 Tasks, 5 Fog Nodes)

Start Prob	3;0	3;0.05	3;0.10	3;0.15	3;0.2	4;0	4;0.05	4;0.01	4;0.15	4;0.2	5;0	5;0.05	5;0.01	5;0.15	5;0.2	6;0	6;0.05	6;0.01	6;0.15	6;0.2
0.0	0.7959	0.7901	0.7892	0.8143	0.859	0.834	0.8561	0.8397	0.8286	0.8396	0.8374	0.8453	0.8632	0.851	0.8445	0.8937	0.8682	0.8857	0.8598	0.853
0.2	0.7998	0.7934	0.7955	0.8074	0.846	0.8312	0.8545	0.8524	0.8366	0.8282	0.8353	0.845	0.8638	0.8493	0.8366	0.8961	0.8626	0.8882	0.8614	0.8579
0.4	0.8017	0.7896	0.7927	0.8202	0.8455	0.8412	0.8483	0.8437	0.8362	0.8386	0.8385	0.852	0.8574	0.855	0.8337	0.8879	0.8624	0.8861	0.8667	0.8526
0.6	0.7873	0.7812	0.7894	0.8217	0.8494	0.8481	0.8471	0.8462	0.8407	0.8306	0.8398	0.8437	0.8574	0.8551	0.8446	0.8962	0.8545	0.889	0.8615	0.8592
0.8	0.787	0.7981	0.7951	0.8104	0.8506	0.8443	0.8493	0.8356	0.838	0.8369	0.836	0.8454	0.8669	0.8535	0.838	0.8897	0.8576	0.8854	0.8584	0.8545
1.0	0.7897	0.7919	0.7919	0.8076	0.8528	0.8453	0.8616	0.8445	0.8317	0.8348	0.8324	0.8483	0.8746	0.8615	0.8425	0.892	0.8564	0.8843	0.8629	0.8666
Stop Prob	3;0	3;0.05	3;0.10	3;0.15	3;0.2	4;0	4;0.05	4;0.01	4;0.15	4;0.2	5;0	5;0.05	5;0.01	5;0.15	5;0.2	6;0	6;0.05	6;0.01	6;0.15	6;0.2
0.0	0.949	0.958	0.9465	0.9635	0.9611	0.9392	0.9706	0.9244	0.9436	0.9185	0.9192	0.9366	0.9758	0.9423	0.9385	0.9685	0.9437	0.984	0.9837	0.9591
0.2	0.925	0.8862	0.9317	0.9325	0.8977	0.9709	0.9683	0.9767	0.9589	0.9693	0.9473	0.9409	0.9772	0.9558	0.9566	0.9779	0.9588	0.93	0.9546	0.9566
0.4	0.8646	0.8436	0.8798	0.886	0.9124	0.9066	0.9346	0.8967	0.9062	0.8748	0.8086	0.8398	0.8898	0.8654	0.847	0.9348	0.9099	0.8976	0.9224	0.9053
0.6	0.7006	0.7139	0.6883	0.7217	0.7914	0.8105	0.8276	0.8192	0.8294	0.8288	0.8574	0.8488	0.873	0.8653	0.8509	0.8297	0.7778	0.8392	0.7865	0.7981
0.8	0.671	0.6846	0.6495	0.6991	0.7725	0.6914	0.713	0.7305	0.6861	0.716	0.7223	0.7681	0.7246	0.7615	0.7246	0.8257	0.7788	0.8342	0.7573	0.7541
1.0	0.6516	0.6582	0.6573	0.6783	0.7676	0.7252	0.7032	0.7146	0.6875	0.701	0.7641	0.7459	0.7427	0.7352	0.7216	0.819	0.7923	0.8342	0.7666	0.7709

Table B.14: Fog Node Simulated Status Change Probability Threshold Bulk Results (15 Tasks, 5 Fog Nodes)

Start Prob	3;0	3;0.05	3;0.10	3;0.15	3;0.2	4;0	4;0.05	4;0.01	4;0.15	4;0.2	5;0	5;0.05	5;0.01	5;0.15	5;0.2	6;0	6;0.05	6;0.01	6;0.15	6;0.2
0.0	0.7621	0.7233	0.7596	0.7269	0.7809	0.7795	0.8	0.8024	0.7908	0.82	0.8048	0.7634	0.8083	0.7837	0.7967	0.8187	0.8303	0.8538	0.8307	0.788
0.2	0.7511	0.7397	0.7591	0.7243	0.7681	0.7782	0.8038	0.8041	0.7968	0.8292	0.8032	0.7692	0.8056	0.7869	0.7985	0.8132	0.8207	0.8519	0.8269	0.7867
0.4	0.7612	0.7246	0.7548	0.722	0.7836	0.782	0.7967	0.7994	0.7974	0.8245	0.799	0.7697	0.8046	0.7933	0.8021	0.8237	0.8238	0.8534	0.8254	0.7789
0.6	0.7606	0.7313	0.7536	0.721	0.7846	0.7784	0.8071	0.8076	0.7956	0.8194	0.8027	0.7628	0.8044	0.7801	0.8051	0.8155	0.823	0.8463	0.8279	0.784
0.8	0.742	0.7267	0.7548	0.7225	0.7876	0.7754	0.7919	0.8044	0.7923	0.8253	0.8041	0.7683	0.8078	0.7934	0.8058	0.8163	0.8235	0.8552	0.8247	0.7812
1.0	0.7476	0.7325	0.7606	0.7298	0.7856	0.778	0.798	0.8069	0.8035	0.825	0.8137	0.7646	0.7927	0.7928	0.7997	0.8187	0.8293	0.8506	0.8296	0.7796
Stop Prob	3;0	3;0.05	3;0.10	3;0.15	3;0.2	4;0	4;0.05	4;0.01	4;0.15	4;0.2	5;0	5;0.05	5;0.01	5;0.15	5;0.2	6;0	6;0.05	6;0.01	6;0.15	6;0.2
0.0	0.9473	0.9546	0.9428	0.95	0.969	0.9683	0.9657	0.9775	0.9571	0.9611	0.946	0.9266	0.9701	0.8729	0.9583	0.9134	0.929	0.9409	0.9365	0.9419
0.2	0.8824	0.8725	0.8414	0.8161	0.8931	0.8535	0.9274	0.9443	0.9018	0.9557	0.9338	0.9417	0.9744	0.9536	0.9704	0.9352	0.9436	0.979	0.9622	0.9478
0.4	0.8163	0.8278	0.8009	0.7668	0.8474	0.8418	0.8959	0.9126	0.8661	0.9097	0.7674	0.6868	0.8031	0.7566	0.8265	0.8383	0.8473	0.8493	0.8671	0.8389
0.6	0.658	0.5951	0.6721	0.6171	0.6894	0.699	0.7193	0.7007	0.7615	0.7491	0.7747	0.7012	0.7376	0.7879	0.7409	0.7837	0.7737	0.806	0.7482	0.7101
0.8	0.6255	0.5753	0.652	0.6052	0.6458	0.6638	0.6449	0.644	0.6442	0.6844	0.6919	0.6642	0.6678	0.6595	0.6669	0.7213	0.7454	0.7484	0.7251	0.6367
1.0	0.5953	0.5528	0.6341	0.5909	0.6457	0.6453	0.6437	0.6457	0.6464	0.6837	0.7139	0.6773	0.6709	0.6999	0.6449	0.7139	0.712	0.7878	0.726	0.6222

Appendix C

Large Scale Testing Additional Data

This appendix presents additional results used in the large scale testing of the heuristic model.

C.1 Performance Analysis Additional Data

Tables C.1 and C.2 present the complete bulk results used in Section 4.4.2.

Table C.1: 40 task 20 fog node bulk results for performance analysis

Rand H	Rand T	Rand R	3 H	3 T	3 R	5 H	5 T	5 R	10 H	10 T	10 R
120.6	165.8	0.7273823884	145.1	182.1	0.7968149368	128	172.6	0.741599073	145.1	192.1	0.7553357626
125.5	171.7	0.7309260338	149.6	185.8	0.8051668461	129.3	166	0.7789156627	137	167.4	0.8183990442
98.5	131.3	0.7501904037	140.7	178.1	0.7900056148	113.2	144.6	0.7828492393	119.9	155.1	0.7730496454
121.9	161.2	0.7562034739	152.4	188.1	0.8102073365	113.1	143.2	0.7898044693	139.7	174.8	0.7991990847
114.5	151.2	0.7572751323	153	183.8	0.8324265506	117.1	147.8	0.7922868742	141.1	173.9	0.8113858539
116.1	153.1	0.7583278903	143.4	181.2	0.7913907285	156.6	196.8	0.7957317073	114.7	146.6	0.7824010914
122.7	160.1	0.7663960025	155.3	182.5	0.8509589041	129.7	162.7	0.7971727105	150.4	186.3	0.8073000537
118.6	154.6	0.7671410091	152.2	181.9	0.8367234744	140.3	175.2	0.8007990868	119.4	147.9	0.8073022312
143.8	185.2	0.7764578834	132.9	165.2	0.8044794189	122.6	153	0.8013071895	115.3	149	0.7738255034
119.7	153.9	0.7777777778	135.1	162.8	0.8298525799	149.1	186	0.8016129032	121.5	151.9	0.7998683344
123.2	158.4	0.7777777778	149.2	180.5	0.8265927978	140.2	173.6	0.8076036866	141.7	171.9	0.8243164631
112.7	144.1	0.7820957668	159.9	185.9	0.8601398601	137.4	169.5	0.810619469	112.9	136.2	0.828928047
125.6	160.4	0.783042394	135.1	171	0.7900584795	126.2	155.2	0.8131443299	146.2	180.5	0.8099722992
123	156.6	0.785440613	137.7	167.2	0.8235645933	133.6	164.2	0.8136419001	124.6	157.8	0.7896070976
114.2	145.2	0.7865013774	134.3	153.1	0.8772044415	152.4	187	0.814973262	132.9	161.4	0.8234200743
109.6	139	0.7884892086	143.5	172.8	0.8304398148	136.5	167.3	0.8158995816	120	149.3	0.8037508372
117.2	147.8	0.7929634641	160.5	189.2	0.8483086681	136.5	167.1	0.8168761221	125	151.8	0.8234519104
101.8	128.1	0.7946916472	159.9	187.8	0.8514376997	143.9	174.9	0.8227558605	120.2	150.7	0.797611148
107.3	134.7	0.7965850037	129.9	164	0.7920731707	148.4	180.2	0.8235294118	126.5	156.8	0.8067602041
127.9	157.4	0.8125794155	169.4	195.3	0.8673835125	162.5	196.6	0.8265513733	116.9	146.4	0.7984972678
104.7	128.6	0.8141524106	117.8	146.1	0.8062970568	148.8	179.4	0.8294314381	145.1	172.9	0.8392134182
124.6	152.8	0.8154450262	138.8	173	0.8023121387	141.8	169.3	0.8375664501	135.8	169.2	0.8026004728
135.7	166	0.8174698795	167.9	195.4	0.8592630502	139.2	165.7	0.84007242	122.4	155.5	0.7871382637
128.5	157.1	0.8179503501	128.9	160.7	0.8021157436	163.6	194.4	0.841563786	134.8	163.9	0.8224527151
135.5	164.1	0.8257160268	140	171.6	0.8158508159	163.3	193.5	0.8439276486	113.1	145.3	0.7783895389
120.2	145.4	0.8266850069	144.5	180.7	0.7996679579	154.5	183	0.8442622951	144	175.6	0.8200455581
131.9	158.8	0.830604534	137.4	168.1	0.8173706127	152.4	179.2	0.8504464286	143.6	173.5	0.8276657061
136.5	163.6	0.8343520782	146.6	173.4	0.84544406	139.3	163.7	0.850946854	133.8	160.2	0.8352059925
125.8	147.9	0.8505747126	175.2	200.2	0.8751248751	149	169.8	0.8775029446	138.1	170.3	0.8109219025

Table C.2: 200 task 20 fog node bulk results for performance analysis

Rand	3	5	10
592.4	833.1	569.2	689.8
637.2	770.4	641.9	694
648.8	738	702.8	558.8
650.3	756.9	645.1	621.8
655.1	819.7	798	660.6
607	732.1	704.1	666.1
610.7	770	728.2	677.5
599.8	809.3	694.9	726.9
680.8	798.8	697.6	666.8
642	729.4	746.6	688.1
584.7	794.4	725.5	757.2
608.1	758.7	813.2	664.8
595.1	668	731.7	617.8
559.1	896.6	770.3	664.5
590.1	793.9	754.4	731.7
635.3	668.5	745.7	653
642.5	739.9	769.6	738.3
644.6	682.8	595.4	636.4
606	761.7	700.8	724.9
615.8	737.9	772.4	640.7
591	719.2	693.3	681.6
574.7	758.1	737.7	695.1
634.1	729.1	681.6	707.5
629.5	764.6	690.5	652
652.8	745	734.5	700.4
566.8	763.1	681	640.8
577.2	786.4	707.3	712.9
584.2	737	738.3	676.9
637.5	676.3	749.3	638.1
575.7	756.6	700.3	698

C.2 Analysis of Differences Between the Heuristic and Theoretic Models Additional Data

Table C.3 and Results 13, 14, 15, 16 and 17 present the complete set of results used in Section 4.4.3.

Table C.3: 20 Fog Node Set used in the analysis for Section 4.4.3

FnId	FnGridX	FnGridY	FnMaxTasks	FnVcpu	FnMem	FnState
1	54.549	48.254	4	5	5	0
2	8.5505	93.061	2	2	2	0
3	89.705	88.184	2	2	2	0
4	94.472	47.151	3	6	6	0
5	67.835	87.434	2	3	4	0
6	3.511	10.086	4	5	5	0
7	52.957	13.465	2	2	2	0
8	21.157	70.689	2	2	2	0
9	46.917	84.553	4	5	5	0
10	89.835	66.863	3	6	6	0
11	67.498	26.375	3	6	6	0
12	17.746	25.247	3	6	6	0
13	24.603	53.664	4	5	5	0
14	41.031	85.878	2	3	4	0
15	83.888	16.824	1	2	2	0
16	69.502	10.28	4	5	5	0
17	78.403	59.04	1	2	2	0
18	34.758	38.875	2	2	2	0
19	10.799	45.22	2	2	2	0
20	25.389	9.833	3	6	6	0

Result 13: Detailed Results Used in the Analysis for Section 4.4.3 Part 1

Time Window 0

$x_{3-0-0} \leftarrow 1$ $x_{2-0-0} \leftarrow 1$ $x_{1-0-0} \leftarrow 1$

$\mu_{4-0} \leftarrow 1$ $\mu_{5-0} \leftarrow 1$ $\mu_{11-0} \leftarrow 1$ $\mu_{12-0} \leftarrow 1$ $\mu_{11-0} \leftarrow 1$

Time Window 1

$x_{4-0-1} \leftarrow 1$ $x_{1-0-1} \leftarrow 1$ $x_{3-11-1} \leftarrow 1$ $x_{5-12-1} \leftarrow 1$ $x_{2-4-1} \leftarrow 1$

$\theta_{4-1} \leftarrow 1$ $\theta_{5-1} \leftarrow 1$ $\theta_{11-1} \leftarrow 1$ $\theta_{12-1} \leftarrow 1$

$\mu_{7-1} \leftarrow 1$

Time Window 2

$x_{6-5-2} \leftarrow 1$ $x_{4-7-2} \leftarrow 1$ $x_{1-0-2} \leftarrow 1$ $x_{3-11-2} \leftarrow 1$ $x_{5-12-2} \leftarrow 1$ $x_{2-4-2} \leftarrow 1$

$\theta_{4-2} \leftarrow 1$ $\theta_{5-2} \leftarrow 1$ $\theta_{7-2} \leftarrow 1$ $\theta_{11-2} \leftarrow 1$ $\theta_{12-2} \leftarrow 1$

Time Window 3

$x_{8-0-3} \leftarrow 1$ $x_{6-5-3} \leftarrow 1$ $x_{9-7-3} \leftarrow 1$ $x_{4-7-3} \leftarrow 1$ $x_{3-11-3} \leftarrow 1$ $x_{5-12-3} \leftarrow 1$ $x_{7-11-3} \leftarrow 1$

$\theta_{4-3} \leftarrow 1$ $\theta_{5-3} \leftarrow 1$ $\theta_{7-3} \leftarrow 1$ $\theta_{11-3} \leftarrow 1$ $\theta_{12-3} \leftarrow 1$

Time Window 4

$x_{8-0-4} \leftarrow 1$ $x_{6-5-4} \leftarrow 1$ $x_{11-4-4} \leftarrow 1$ $x_{9-7-4} \leftarrow 1$ $x_{10-12-4} \leftarrow 1$ $x_{4-7-4} \leftarrow 1$ $x_{3-11-4} \leftarrow 1$ $x_{5-0-4} \leftarrow 1$ $x_{7-11-4} \leftarrow 1$

$\theta_{4-4} \leftarrow 1$ $\theta_{5-4} \leftarrow 1$ $\theta_{7-4} \leftarrow 1$ $\theta_{11-4} \leftarrow 1$ $\theta_{12-4} \leftarrow 1$

$\mu_{6-4} \leftarrow 1$

Time Window 5

$x_{13-0-5} \leftarrow 1$ $x_{8-0-5} \leftarrow 1$ $x_{6-5-5} \leftarrow 1$ $x_{12-0-5} \leftarrow 1$ $x_{11-4-5} \leftarrow 1$ $x_{9-7-5} \leftarrow 1$ $x_{4-7-5} \leftarrow 1$ $x_{10-6-5} \leftarrow 1$ $x_{3-11-5} \leftarrow 1$ $x_{5-12-5} \leftarrow 1$

$\theta_{4-5} \leftarrow 1$ $\theta_{5-5} \leftarrow 1$ $\theta_{6-5} \leftarrow 1$ $\theta_{7-5} \leftarrow 1$ $\theta_{11-5} \leftarrow 1$ $\theta_{12-5} \leftarrow 1$

$\mu_{10-5} \leftarrow 1$ $\mu_{16-5} \leftarrow 1$ $\mu_{18-5} \leftarrow 1$

Time Window 6

$x_{13-18-6} \leftarrow 1$ $x_{15-11-6} \leftarrow 1$ $x_{8-0-6} \leftarrow 1$ $x_{14-4-6} \leftarrow 1$ $x_{6-5-6} \leftarrow 1$ $x_{12-10-6} \leftarrow 1$ $x_{11-5-6} \leftarrow 1$ $x_{9-7-6} \leftarrow 1$ $x_{10-6-6} \leftarrow 1$ $x_{4-7-6} \leftarrow 1$ $x_{3-11-6} \leftarrow 1$ $x_{5-12-6} \leftarrow 1$

$\theta_{4-6} \leftarrow 1$ $\theta_{5-6} \leftarrow 1$ $\theta_{6-6} \leftarrow 1$ $\theta_{7-6} \leftarrow 1$ $\theta_{10-6} \leftarrow 1$ $\theta_{11-6} \leftarrow 1$ $\theta_{12-6} \leftarrow 1$ $\theta_{16-6} \leftarrow 1$ $\theta_{18-6} \leftarrow 1$

Result 14: Detailed Results Used in the Analysis for Section 4.4.3 Part 2

Time Window 7

$x_{16-4-7} \leftarrow 1$ $x_{13-18-7} \leftarrow 1$ $x_{15-11-7} \leftarrow 1$ $x_{8-0-7} \leftarrow 1$
$x_{14-5-7} \leftarrow 1$ $x_{6-5-7} \leftarrow 1$ $x_{12-10-7} \leftarrow 1$ $x_{11-5-7} \leftarrow 1$
$x_{9-7-7} \leftarrow 1$ $x_{10-6-7} \leftarrow 1$

$\theta_{4-7} \leftarrow 1$ $\theta_{5-7} \leftarrow 1$ $\theta_{6-7} \leftarrow 1$ $\theta_{7-7} \leftarrow 1$ $\theta_{10-7} \leftarrow 1$
$\theta_{11-7} \leftarrow 1$ $\theta_{12-7} \leftarrow 1$ $\theta_{16-7} \leftarrow 1$ $\theta_{18-7} \leftarrow 1$

Time Window 8

$x_{18-0-8} \leftarrow 1$ $x_{16-4-8} \leftarrow 1$ $x_{13-18-8} \leftarrow 1$ $x_{15-11-8} \leftarrow 1$
$x_{17-11-8} \leftarrow 1$ $x_{14-5-8} \leftarrow 1$ $x_{12-10-8} \leftarrow 1$ $x_{11-5-8} \leftarrow 1$
$x_{9-7-8} \leftarrow 1$ $x_{10-6-8} \leftarrow 1$

$\theta_{4-8} \leftarrow 1$ $\theta_{5-8} \leftarrow 1$ $\theta_{6-8} \leftarrow 1$ $\theta_{7-8} \leftarrow 1$ $\theta_{10-8} \leftarrow 1$
$\theta_{11-8} \leftarrow 1$ $\theta_{12-8} \leftarrow 1$ $\theta_{16-8} \leftarrow 1$ $\theta_{18-8} \leftarrow 1$

Time Window 9

$x_{18-0-9} \leftarrow 1$ $x_{20-0-9} \leftarrow 1$ $x_{16-4-9} \leftarrow 1$ $x_{13-18-9} \leftarrow 1$
$x_{15-11-9} \leftarrow 1$ $x_{19-5-9} \leftarrow 1$ $x_{17-11-9} \leftarrow 1$ $x_{14-5-9} \leftarrow 1$
$x_{12-10-9} \leftarrow 1$ $x_{11-5-9} \leftarrow 1$ $x_{10-6-9} \leftarrow 1$

$\theta_{4-9} \leftarrow 1$ $\theta_{5-9} \leftarrow 1$ $\theta_{6-9} \leftarrow 1$ $\theta_{7-9} \leftarrow 1$ $\theta_{10-9} \leftarrow 1$
$\theta_{11-9} \leftarrow 1$ $\theta_{12-9} \leftarrow 1$ $\theta_{16-9} \leftarrow 1$ $\theta_{18-9} \leftarrow 1$

$\mu_{13-9} \leftarrow 1$

Time Window 10

$x_{18-0-10} \leftarrow 1$ $x_{20-13-10} \leftarrow 1$ $x_{21-11-10} \leftarrow 1$ $x_{16-4-10} \leftarrow 1$
$x_{13-18-10} \leftarrow 1$ $x_{15-11-10} \leftarrow 1$ $x_{22-5-10} \leftarrow 1$ $x_{19-5-10} \leftarrow 1$
$x_{17-7-10} \leftarrow 1$ $x_{14-5-10} \leftarrow 1$ $x_{11-0-10} \leftarrow 1$

$\phi_{4-10} \leftarrow 1$ sim

$\theta_{4-10} \leftarrow 1$ $\theta_{5-10} \leftarrow 1$ $\theta_{6-10} \leftarrow 1$ $\theta_{7-10} \leftarrow 1$ $\theta_{10-10} \leftarrow 1$
$\theta_{11-10} \leftarrow 1$ $\theta_{12-10} \leftarrow 1$ $\theta_{13-10} \leftarrow 1$ $\theta_{16-10} \leftarrow 1$ $\theta_{18-10} \leftarrow 1$

$\mu_{9-10} \leftarrow 1$ $\mu_{4-10} \leftarrow 1$

Time Window 11

$x_{24-0-11} \leftarrow 1$ $x_{18-0-11} \leftarrow 1$ $x_{20-13-11} \leftarrow 1$ $x_{21-11-11} \leftarrow 1$
$x_{15-11-11} \leftarrow 1$ $x_{22-4-11} \leftarrow 1$ $x_{17-7-11} \leftarrow 1$ $x_{14-5-11} \leftarrow 1$
$x_{23-7-11} \leftarrow 1$ $x_{11-9-11} \leftarrow 1$

$\phi_{5-11} \leftarrow 1$ sim

$\theta_{4-11} \leftarrow 1$ $\theta_{5-11} \leftarrow 1$ $\theta_{6-11} \leftarrow 1$ $\theta_{7-11} \leftarrow 1$ $\theta_{9-11} \leftarrow 1$
$\theta_{10-11} \leftarrow 1$ $\theta_{11-11} \leftarrow 1$ $\theta_{12-11} \leftarrow 1$ $\theta_{13-11} \leftarrow 1$ $\theta_{16-11} \leftarrow 1$
$\theta_{18-11} \leftarrow 1$

Result 15: Detailed Results Used in the Analysis for Section 4.4.3 Part 3

1 **Time Window 12**

2 $x_{24-0-12} \leftarrow 1$ $x_{26-0-12} \leftarrow 1$ $x_{18-0-12} \leftarrow 1$ $x_{20-13-12} \leftarrow 1$
$x_{25-10-12} \leftarrow 1$ $x_{21-11-12} \leftarrow 1$ $x_{15-11-12} \leftarrow 1$ $x_{22-4-12} \leftarrow 1$
$x_{17-7-12} \leftarrow 1$ $x_{14-0-12} \leftarrow 1$ $x_{23-7-12} \leftarrow 1$

3 $\phi_{11-12} \leftarrow 1$ sim $\phi_{12-12} \leftarrow 1$ sim

4 $\theta_{4-12} \leftarrow 1$ $\theta_{6-12} \leftarrow 1$ $\theta_{7-12} \leftarrow 1$ $\theta_{9-12} \leftarrow 1$ $\theta_{10-12} \leftarrow 1$
$\theta_{11-12} \leftarrow 1$ $\theta_{12-12} \leftarrow 1$ $\theta_{13-12} \leftarrow 1$ $\theta_{16-12} \leftarrow 1$ $\theta_{18-12} \leftarrow 1$

5 $\mu_{3-12} \leftarrow 1$ $\mu_{5-12} \leftarrow 1$ $\mu_{11-12} \leftarrow 1$ $\mu_{14-12} \leftarrow 1$ $\mu_{11-12} \leftarrow 1$

6 **Time Window 13**

7 $x_{28-0-13} \leftarrow 1$ $x_{24-0-13} \leftarrow 1$ $x_{26-14-13} \leftarrow 1$ $x_{18-0-13} \leftarrow 1$
$x_{20-13-13} \leftarrow 1$ $x_{25-10-13} \leftarrow 1$ $x_{27-4-13} \leftarrow 1$ $x_{21-11-13} \leftarrow 1$
$x_{17-7-13} \leftarrow 1$ $x_{14-3-13} \leftarrow 1$ $x_{23-7-13} \leftarrow 1$

8 $\phi_{7-13} \leftarrow 1$ sim

9 $\theta_{3-13} \leftarrow 1$ $\theta_{4-13} \leftarrow 1$ $\theta_{5-13} \leftarrow 1$ $\theta_{6-13} \leftarrow 1$ $\theta_{7-13} \leftarrow 1$
$\theta_{9-13} \leftarrow 1$ $\theta_{10-13} \leftarrow 1$ $\theta_{11-13} \leftarrow 1$ $\theta_{13-13} \leftarrow 1$ $\theta_{14-13} \leftarrow 1$
$\theta_{16-13} \leftarrow 1$ $\theta_{18-13} \leftarrow 1$

10 $\mu_{15-13} \leftarrow 1$

11 **Time Window 14**

12 $x_{30-14-14} \leftarrow 1$ $x_{28-15-14} \leftarrow 1$ $x_{24-0-14} \leftarrow 1$ $x_{26-0-14} \leftarrow 1$
$x_{29-6-14} \leftarrow 1$ $x_{20-13-14} \leftarrow 1$ $x_{25-10-14} \leftarrow 1$ $x_{27-4-14} \leftarrow 1$
$x_{21-11-14} \leftarrow 1$ $x_{17-16-14} \leftarrow 1$

13 $\theta_{3-14} \leftarrow 1$ $\theta_{4-14} \leftarrow 1$ $\theta_{5-14} \leftarrow 1$ $\theta_{6-14} \leftarrow 1$ $\theta_{9-14} \leftarrow 1$
$\theta_{10-14} \leftarrow 1$ $\theta_{11-14} \leftarrow 1$ $\theta_{13-14} \leftarrow 1$ $\theta_{14-14} \leftarrow 1$ $\theta_{15-14} \leftarrow 1$
$\theta_{16-14} \leftarrow 1$ $\theta_{18-14} \leftarrow 1$

14 **Time Window 15**

15 $x_{32-10-15} \leftarrow 1$ $x_{30-14-15} \leftarrow 1$ $x_{28-15-15} \leftarrow 1$ $x_{24-0-15} \leftarrow 1$
$x_{26-0-15} \leftarrow 1$ $x_{31-11-15} \leftarrow 1$ $x_{29-6-15} \leftarrow 1$ $x_{25-18-15} \leftarrow 1$
$x_{21-15-15} \leftarrow 1$

16 $\phi_{6-15} \leftarrow 1$ sim $\phi_{16-15} \leftarrow 1$ sim $\phi_{18-15} \leftarrow 1$ sim

17 $\theta_{3-15} \leftarrow 1$ $\theta_{4-15} \leftarrow 1$ $\theta_{5-15} \leftarrow 1$ $\theta_{6-15} \leftarrow 1$ $\theta_{9-15} \leftarrow 1$
$\theta_{10-15} \leftarrow 1$ $\theta_{11-15} \leftarrow 1$ $\theta_{13-15} \leftarrow 1$ $\theta_{14-15} \leftarrow 1$ $\theta_{15-15} \leftarrow 1$
$\theta_{16-15} \leftarrow 1$ $\theta_{18-15} \leftarrow 1$

Result 16: Detailed Results Used in the Analysis for Section 4.4.3 Part 4

Time Window 16

$x_{34-14-16} \leftarrow 1$ $x_{32-10-16} \leftarrow 1$ $x_{30-0-16} \leftarrow 1$ $x_{28-15-16} \leftarrow 1$ $x_{31-11-16} \leftarrow 1$ $x_{29-0-16} \leftarrow 1$ $x_{25-13-16} \leftarrow 1$ $x_{33-11-16} \leftarrow 1$ $x_{21-15-16} \leftarrow 1$

$\theta_{3-16} \leftarrow 1$ $\theta_{4-16} \leftarrow 1$ $\theta_{5-16} \leftarrow 1$ $\theta_{9-16} \leftarrow 1$ $\theta_{10-16} \leftarrow 1$ $\theta_{11-16} \leftarrow 1$ $\theta_{13-16} \leftarrow 1$ $\theta_{14-16} \leftarrow 1$ $\theta_{15-16} \leftarrow 1$

$\mu_{1-16} \leftarrow 1$ $\mu_{2-16} \leftarrow 1$

Time Window 17

$x_{34-14-17} \leftarrow 1$ $x_{35-2-17} \leftarrow 1$ $x_{32-10-17} \leftarrow 1$ $x_{30-1-17} \leftarrow 1$ $x_{31-11-17} \leftarrow 1$ $x_{33-11-17} \leftarrow 1$ $x_{21-15-17} \leftarrow 1$

$\theta_{1-17} \leftarrow 1$ $\theta_{2-17} \leftarrow 1$ $\theta_{3-17} \leftarrow 1$ $\theta_{4-17} \leftarrow 1$ $\theta_{5-17} \leftarrow 1$ $\theta_{9-17} \leftarrow 1$ $\theta_{10-17} \leftarrow 1$ $\theta_{11-17} \leftarrow 1$ $\theta_{13-17} \leftarrow 1$ $\theta_{14-17} \leftarrow 1$ $\theta_{15-17} \leftarrow 1$

Time Window 18

$x_{36-9-18} \leftarrow 1$ $x_{37-0-18} \leftarrow 1$ $x_{34-14-18} \leftarrow 1$ $x_{35-2-18} \leftarrow 1$ $x_{32-10-18} \leftarrow 1$ $x_{30-1-18} \leftarrow 1$ $x_{31-11-18} \leftarrow 1$ $x_{33-11-18} \leftarrow 1$

$\theta_{1-18} \leftarrow 1$ $\theta_{2-18} \leftarrow 1$ $\theta_{3-18} \leftarrow 1$ $\theta_{4-18} \leftarrow 1$ $\theta_{5-18} \leftarrow 1$ $\theta_{9-18} \leftarrow 1$ $\theta_{10-18} \leftarrow 1$ $\theta_{11-18} \leftarrow 1$ $\theta_{13-18} \leftarrow 1$ $\theta_{14-18} \leftarrow 1$ $\theta_{15-18} \leftarrow 1$

$\mu_{6-18} \leftarrow 1$

Time Window 19

$x_{38-4-19} \leftarrow 1$ $x_{36-9-19} \leftarrow 1$ $x_{37-6-19} \leftarrow 1$ $x_{34-14-19} \leftarrow 1$ $x_{35-2-19} \leftarrow 1$ $x_{32-10-19} \leftarrow 1$ $x_{30-1-19} \leftarrow 1$ $x_{39-3-19} \leftarrow 1$ $x_{31-11-19} \leftarrow 1$

$\phi_{13-19} \leftarrow 1$ sim

$\theta_{1-19} \leftarrow 1$ $\theta_{2-19} \leftarrow 1$ $\theta_{3-19} \leftarrow 1$ $\theta_{4-19} \leftarrow 1$ $\theta_{5-19} \leftarrow 1$ $\theta_{6-19} \leftarrow 1$ $\theta_{9-19} \leftarrow 1$ $\theta_{10-19} \leftarrow 1$ $\theta_{11-19} \leftarrow 1$ $\theta_{13-19} \leftarrow 1$ $\theta_{14-19} \leftarrow 1$ $\theta_{15-19} \leftarrow 1$

Time Window 20

$x_{40-0-20} \leftarrow 1$ $x_{38-4-20} \leftarrow 1$ $x_{36-9-20} \leftarrow 1$ $x_{37-6-20} \leftarrow 1$ $x_{34-14-20} \leftarrow 1$ $x_{35-2-20} \leftarrow 1$ $x_{32-10-20} \leftarrow 1$ $x_{39-3-20} \leftarrow 1$ $x_{31-11-20} \leftarrow 1$

$\theta_{1-20} \leftarrow 1$ $\theta_{2-20} \leftarrow 1$ $\theta_{3-20} \leftarrow 1$ $\theta_{4-20} \leftarrow 1$ $\theta_{5-20} \leftarrow 1$ $\theta_{6-20} \leftarrow 1$ $\theta_{9-20} \leftarrow 1$ $\theta_{10-20} \leftarrow 1$ $\theta_{11-20} \leftarrow 1$ $\theta_{14-20} \leftarrow 1$ $\theta_{15-20} \leftarrow 1$

Result 17: Detailed Results Used in the Analysis for Section 4.4.3 Part 5

1 **Time Window 21**

2 $x_{40-0-21} \leftarrow 1$ $x_{38-4-21} \leftarrow 1$ $x_{36-9-21} \leftarrow 1$ $x_{37-6-21} \leftarrow 1$ $x_{34-14-21} \leftarrow 1$ $x_{35-2-21} \leftarrow 1$ $x_{39-3-21} \leftarrow 1$ $x_{31-11-21} \leftarrow 1$

3 $\theta_{1-21} \leftarrow 1$ $\theta_{2-21} \leftarrow 1$ $\theta_{3-21} \leftarrow 1$ $\theta_{4-21} \leftarrow 1$ $\theta_{5-21} \leftarrow 1$ $\theta_{6-21} \leftarrow 1$ $\theta_{9-21} \leftarrow 1$ $\theta_{10-21} \leftarrow 1$ $\theta_{11-21} \leftarrow 1$ $\theta_{14-21} \leftarrow 1$ $\theta_{15-21} \leftarrow 1$

4 **Time Window 22**

5 $x_{40-0-22} \leftarrow 1$ $x_{38-4-22} \leftarrow 1$ $x_{37-6-22} \leftarrow 1$ $x_{34-14-22} \leftarrow 1$ $x_{39-9-22} \leftarrow 1$

6 $\phi_{4-22} \leftarrow 1$ sim $\phi_{5-22} \leftarrow 1$ $\phi_{9-22} \leftarrow 1$ sim $\phi_{10-22} \leftarrow 1$ sim

7 **Time Window 23**

8 $x_{38-0-23} \leftarrow 1$

9 $\phi_{14-23} \leftarrow 1$ sim

10 $\mu_{4-23} \leftarrow 1$ $\mu_{5-23} \leftarrow 1$ $\mu_{4-23} \leftarrow 1$

11 $Profit \leftarrow 1.248000e+02$