

**A Thesis Submitted for the Degree of PhD at the University of Warwick**

**Permanent WRAP URL:**

<http://wrap.warwick.ac.uk/78802>

**Copyright and reuse:**

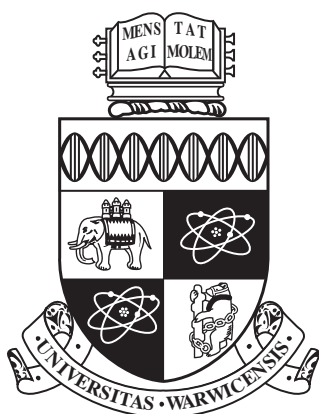
This thesis is made available online and is protected by original copyright.

Please scroll down to view the document itself.

Please refer to the repository record for this item for information to help you to cite it.

Our policy information is available from the repository home page.

For more information, please contact the WRAP Team at: [wrap@warwick.ac.uk](mailto:wrap@warwick.ac.uk)



# Developing Energy-Aware Workload Offloading Frameworks in Mobile Cloud Computing

by

**Bo Gao**

A thesis submitted to The University of Warwick

in partial fulfilment of the requirements

for admission to the degree of

**Doctor of Philosophy**

**Department of Computer Science**

The University of Warwick

June 2015

# Abstract

Mobile cloud computing is an emerging field of research that aims to provide a platform on which intelligent and feature-rich applications are delivered to the user at any time and at anywhere. Computation offload between mobile and cloud plays a key role in this vision and ensures that the integration between mobile and cloud is both seamless and energy-efficient. In this thesis, we develop a suite of energy-aware workload offloading frameworks to accommodate the efficient execution of mobile workflows on a mobile cloud platform. We start by looking at two energy objectives of a mobile cloud platform. While the first objective aims at minimising the overall energy cost of the platform, the second objective aims at the longevity of the platform taking into account the residual battery power of each device. We construct optimisation models for both objectives and develop two efficient algorithms to approximate the optimal solution. According to simulation results, our greedy autonomous offload (GAO) algorithm is able to efficiently produce allocation schemes that are close to optimal. Next, we look at the task allocation problem from a workflow's perspective and develop energy-aware offloading strategies for time-constrained mobile workflows. We demonstrate the effect of software and hardware characteristics have over the offload-efficiency of mobile workflows with a workflow-oriented greedy autonomous offload (WGAO) algorithm, an extension to the GAO algorithm. Thirdly, we propose a novel network I-O model to describe the bandwidth dependencies and allocation problem in mobile networks. This model lays the foundation for further objective developments such as the cost-based and adaptive bandwidth allocation schemes which we also present in this thesis. Lastly, we apply a game theoretical approach to model the non-cooperative behaviour of mobile cloud applications that reside on the same device. Mixed-strategy Nash equilibrium is derived for the offload game which further quantifies the price of anarchy of the system.

To my wife, daughter and parents

# Acknowledgements

First, I would like to express my sincere gratitude to my supervisor Dr. Ligang He, whose guidance, encouragement and support have been invaluable to me during my time at the Department of Computer Science at the University of Warwick. I benefited greatly from his insightful advices and comments in finding and solving research problems. I look forward to maintaining our collaboration in the future.

I would like to thank my wife Wenting whose patience, encouragement and unwavering love has been the magical special ingredient in my life for the past four years. I also need to thank her for bringing the little bundle of joy and many other things that is our new born daughter Alex to the world. May I wish here that she is as lucky as her daddy is in finding that special person of her life.

I thank my parents, auntie and uncle for their continuous and unreserved support in my pursuit of an academic career.

I thank Dr. Roger Packwood for his support in solving technical issues and his positive attitude which influenced me greatly.

Last but not the least, I want to thank my fellow lab-mates, particularly Xin Lu, Chao Chen, Huanzhou Zhu, Zhuoer Gu, Peng Jiang, Shenyuan Ren, Xufeng Lin, Phil Taylor and Wilson Tan, for their stimulating discussions in current trends in technology and for creating all the happy memories that we share.

# Declarations

This thesis is submitted to the University of Warwick in support of the author's application for the degree of Doctor of Philosophy. It has been composed by the author and has not been submitted in any previous application for any degree. The work presented was carried out by the author except where acknowledged.

Parts of this thesis have been previously published by the author in the following publications:

- [1] B. Gao, L. He, and S. A. Jarvis, "Offload Decision Models and the Price of Anarchy in Mobile Cloud Application Ecosystems," *IEEE Access, Special Section on Emerging Cloud-Based Wireless Communications and Networks*, vol. 3, pp. 3125–3137, 2016
- [2] B. Gao, L. He, and C. Chen, "Modelling the Bandwidth Allocation Problem in Mobile Service-Oriented Networks," in *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'15)*, pp. 307–311, 2015
- [3] B. Gao, L. He, X. Lu, C. Chang, K. Li, and K. Li, "Developing Energy-Aware Task Allocation Schemes in Cloud-Assisted Mobile Workflows," in *Proceedings of IEEE International Conference on Ubiquitous Computing and Communications (IUCC'15)*, pp. 1266–1273, 2015
- [4] B. Gao and L. He, "Modelling Energy-Aware Task Allocation in Mobile Workflows," in *Proceedings of the 10th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous'13)*, vol. 131, pp. 89–101, 2013
- [5] B. Gao, L. He, L. Liu, K. Li, and S. Jarvis, "From Mobiles to Clouds: Developing Energy-Aware Offloading Strategies for Workflows," in *Proceedings of the 13th ACM/IEEE Interna-*

- 
- tional Conference on Grid Computing (GRID'12)*, pp. 139–146, 2012
- [6] H. Zhu, L. He, B. Gao, K. Li, and K. Li, “Modelling and Developing Co-Scheduling Strategies on Multicore Processors,” in *Proceedings of the 44th International Conference on Parallel Processing (ICPP'15)*, 2015
- [7] C. Chen, L. He, and B. Gao, “Modelling and Optimizing Bandwidth Provision for Interacting Cloud Services,” in *Proceedings of the 13th International Conference on Service Oriented Computing (ICSOC'15)*, 2015
- [8] S. Fu, L. He, X. Liao, C. Huang, K. Li, C. Chang, and B. Gao, “Cadros: The Cloud-Assisted Data Replication in Decentralized Online Social Networks,” in *Proceedings of the 11th IEEE International Conference on Services Computing (SCC'14)*, pp. 43–50, 2014
- [9] S. Fu, L. He, X. Liao, C. Huang, K. Li, C. Chang, and B. Gao, “Modelling and Predicting the Data Availability in Decentralized Online Social Networks,” in *Proceedings of the 21st IEEE International Conference on Web Services (ICWS'14)*, pp. 161–168, 2014
- [10] C. Chen, L. He, H. Chen, J. Sun, B. Gao, and S. A. Jarvis, “Developing Communication-aware Service Placement Frameworks in the Cloud Economy,” in *Proceedings of IEEE International Conference on Cluster Computing (CLUSTER'13)*, pp. 1–8, 2013
- [11] K. Li, Z. Zhang, Y. Xu, B. Gao, and L. He, “Chemical Reaction Optimization for Heterogeneous Computing Environments,” in *Proceedings of the 10th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA'12)*, pp. 17–23, IEEE, 2012
- [12] L. He, C. Huang, K. Li, H. Chen, J. Sun, B. Gao, K. Duan, and S. A. Jarvis, “Modelling and Analyzing the Authorization and Execution of Video Workflows,” in *Proceedings of the 18th International Conference on High Performance Computing (HiPC'11)*, pp. 1–10, 2011

# Sponsorship and Grants

The research presented in this thesis was made possible by the support of the following benefactors and sources:

- Leverhulme Trust:  
Research Project Grant (RPG-101)
- Department of Computer Science, The University of Warwick, United Kingdom:  
Postgraduate Research Scholarship



# Abbreviations

<b>2G</b>	Second Generation (Mobile Telecommunications Technology)
<b>3G</b>	Third Generation (Mobile Telecommunications Technology)
<b>4G</b>	Fourth Generation (Mobile Telecommunications Technology)
<b>App</b>	Applications (Mobile)
<b>CPU</b>	Central Processing Unit
<b>GAO</b>	Greedy Autonomous Offload (Algorithm)
<b>WGAO</b>	Workflow-oriented Greedy Autonomous Offload (Algorithm)
<b>IP</b>	Integer Program
<b>I/O</b>	Input/Output (Module)
<b>I-O</b>	Input-Output (Analysis Model)
<b>LAN</b>	Local Area Network
<b>LP</b>	Linear Program
<b>MCC</b>	Mobile Cloud Computing
<b>MCP</b>	Mobile Cloud Platform
<b>MGECP</b>	Minimum Group Energy Cost Problem
<b>MIQP</b>	Mixed Integer Quadratic Program
<b>MMUP</b>	Minimum Maximum Utilisation Problem
<b>MSON</b>	Mobile Service-Oriented Network
<b>OS</b>	Operating System
<b>P2P</b>	Peer to Peer
<b>QP</b>	Quadratic Program
<b>QAP</b>	Quadratic Assignment Problem
<b>QCP</b>	Quadratically Constrained Program

---

<b>QoS</b>	Quality of Service
<b>SA</b>	Simulated Annealing (Algorithm)
<b>SOA</b>	Service-Oriented Architecture
<b>VANET</b>	Vehicular Ad Hoc Network
<b>VM</b>	Virtual Machine
<b>WAN</b>	Wide Area Network

# Contents

<b>Abstract</b>	<b>ii</b>
<b>Dedication</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>iv</b>
<b>Declarations</b>	<b>v</b>
<b>Sponsorship and Grants</b>	<b>vii</b>
<b>Abbreviations</b>	<b>viii</b>
<b>List of Figures</b>	<b>xvi</b>
<b>List of Tables</b>	<b>xvii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Mobile, Cloud, Cloud via Mobile and Mobile Cloud Computing . . . . .	2
1.2 Mobile Cloud Workflow . . . . .	4
1.3 Outline of Research Contributions . . . . .	5
1.4 Thesis Organisation . . . . .	6
<b>2 Mobile Cloud Computing</b>	<b>8</b>
2.1 Architectures and Components of Mobile Cloud Computing . . . . .	8
2.1.1 Enabling Mobile Computation Offload . . . . .	9
2.1.2 Mobile Workflow Engine . . . . .	10

---

2.2	Impact of Mobile Cloud Computing . . . . .	11
2.2.1	Motivational Applications of Mobile Cloud Computing . . . . .	11
2.2.2	Illustrative Use Cases of Mobile Workflows . . . . .	12
2.2.3	Advantages and Issues of Mobile Cloud Computing . . . . .	15
<b>3</b>	<b>Energy-Aware Task Allocation in Mobile Cloud Platforms</b>	<b>17</b>
3.1	Mobile Cloud Platform Model . . . . .	18
3.1.1	Mobile, Cloud and Network Metrics . . . . .	18
3.1.2	Application Workflow Metrics . . . . .	19
3.1.3	Fixed and Constrained Tasks . . . . .	20
3.1.4	Allocation Scheme and Energy Costs . . . . .	20
3.2	Minimum Group Energy Cost Problem . . . . .	21
3.2.1	Assignment Matrix . . . . .	21
3.2.2	Quadratic Program Formulation . . . . .	22
3.2.3	Convexification . . . . .	23
3.3	Minimum Max-Utilisation Problem . . . . .	24
3.3.1	Device Cost Matrix . . . . .	25
3.3.2	Device Utilisation . . . . .	25
3.3.3	Quadratically Constrained Program Formulation . . . . .	26
3.4	Heuristics . . . . .	27
3.4.1	Simulated Annealing . . . . .	27
3.4.2	Greedy Autonomous Offload . . . . .	30
3.4.3	Joint Search . . . . .	33
3.5	Simulations, Comparisons and Discussion . . . . .	33
3.5.1	Simulation Structure . . . . .	33
3.5.2	MCP Construction . . . . .	36
3.5.3	Results from Solving MGECP . . . . .	37
3.5.4	Results from Solving MMUP . . . . .	41
3.5.5	Comparing MGECP and MMUP . . . . .	42
3.6	Summary . . . . .	48

---

<b>4</b>	<b>Offloading Strategies for Time-Constrained Mobile Workflows</b>	<b>50</b>
4.1	Computation Offload with Cloudlets . . . . .	51
4.2	Offload Strategies for Mobile Workflows . . . . .	53
4.2.1	Preliminaries and Problem Definition . . . . .	53
4.2.2	Workflow-Oriented Greedy Autonomous Offload Algorithm . . . . .	58
4.2.3	Discussion of Variations and Optimisation of WGAO . . . . .	61
4.3	Simulations . . . . .	63
4.3.1	Communication Size and Network Connectivity . . . . .	64
4.3.2	Computation Size and Cloudlet Speed . . . . .	69
4.3.3	Energy Profile . . . . .	70
4.4	Summary . . . . .	71
<b>5</b>	<b>Bandwidth Dependency and Allocation in Mobile Service-Oriented Networks</b>	<b>73</b>
5.1	Mobile Service-Oriented Networks . . . . .	74
5.1.1	Example and Definition . . . . .	74
5.1.2	Service-Oriented Architecture . . . . .	76
5.1.3	Applications of MSON . . . . .	77
5.1.4	Mobile Device as Service Hosts . . . . .	77
5.2	Input-Output Analysis in Economics . . . . .	78
5.3	The Economy of Mobile Service-Oriented Networks . . . . .	79
5.3.1	Network I-O Model . . . . .	84
5.3.2	Network I-O Model with Latency . . . . .	86
5.4	Parametric Evaluation . . . . .	87
5.4.1	Effect of Service Arrival Rate . . . . .	88
5.4.2	Effect of Per Service Data Size . . . . .	89
5.4.3	Effect of Latency . . . . .	89
5.4.4	Alternative Allocation Scheme . . . . .	89
5.5	Cost-Based Bandwidth Allocation . . . . .	91
5.5.1	Problem Formulation . . . . .	91
5.5.2	Simulation . . . . .	92

---

5.6	Adaptive Bandwidth Allocation . . . . .	94
5.6.1	Problem Formulation . . . . .	94
5.6.2	Simulation . . . . .	96
5.7	Summary . . . . .	99
<b>6</b>	<b>Rethinking the Offload Decision Models in Mobile Cloud Application Ecosystems</b>	<b>100</b>
6.1	Mobile Cloud Application Ecosystems . . . . .	101
6.1.1	Problem Statement . . . . .	102
6.1.2	Objective and Contribution . . . . .	105
6.1.3	System Notations . . . . .	105
6.2	Offload with Symmetrically Incomplete Information . . . . .	107
6.3	Offload with Complete Information . . . . .	108
6.3.1	The Offload Game . . . . .	109
6.3.2	Mixed Strategies and Expected Costs . . . . .	109
6.3.3	Nash Equilibrium . . . . .	110
6.3.4	Social Cost and Price of Anarchy . . . . .	111
6.4	Cooperative Decision Model . . . . .	112
6.5	Simulations, Comparisons and Discussion . . . . .	114
6.5.1	Simulation Setup . . . . .	114
6.5.2	Strategy Behaviour of Non-Cooperative Applications . . . . .	116
6.5.3	Social Costs . . . . .	120
6.5.4	Price of Anarchy . . . . .	120
6.6	Summary . . . . .	129
<b>7</b>	<b>Conclusions and Further Work</b>	<b>130</b>
7.1	Energy-Aware Task Allocation . . . . .	131
7.2	Offloading Strategies for Time-Constrained Workflows . . . . .	132
7.3	Efficient Resource Allocation in Mobile Networks . . . . .	132
7.4	Application Ecosystem and Offload Competition . . . . .	133
7.5	Directions for Further Work . . . . .	134

---

<b>Bibliography</b>	<b>135</b>
<b>A MGECP Simulation Results</b>	<b>148</b>
<b>B MMUP Simulation Results</b>	<b>154</b>
<b>C Derivation of <math>p_i^{\mathbb{R}}</math></b>	<b>160</b>

# List of Figures

1.1	Research structure. . . . .	7
2.1	Example use case of an MCC workflow: Business task interaction. . . . .	14
2.2	Example use case of an MCC workflow: 3D scan workflow. . . . .	15
2.3	Example use case of an MCC workflow: Darts game. . . . .	15
3.1	Illustration of a Mobile Cloud Platform. . . . .	19
3.2	Results from solving MGECP: Solution optimality from S0 and M0. . . . .	38
3.3	Results from solving MGECP: Solution optimality from L0 and X0. . . . .	39
3.4	Results from solving MGECP: Solution time from S0, M0, L0 and X0. . . . .	40
3.5	Results from solving MMUP: Solution optimality from S0 and M0. . . . .	44
3.6	Results from solving MMUP: Solution optimality from L0 and X0. . . . .	45
3.7	Results from solving MMUP: Solution time from S0, M0, L0 and X0. . . . .	46
3.8	Prime and by objective values from S0, M0, L0 and X0. . . . .	47
4.1	Example showing cloudlet and faster network connections improve battery life on mobile devices. . . . .	52
4.2	Offload expands the mapping into the Cloudlet space. . . . .	55
4.3	Effect of change in executable size and WiFi availability. . . . .	65
4.4	Offload savings when no WiFi is available. . . . .	67
4.5	Effect of change in communication data size and WiFi availability. . . . .	68
4.6	Effect of density of workload. . . . .	69
4.7	Effect of increase in cloudlet speedup. . . . .	70



---

4.8	Energy distribution before and after offload. . . . .	71
5.1	A simple example of a personal MSON. . . . .	75
5.2	The economy of an MSON. . . . .	80
5.3	Four types of service communication patterns. . . . .	82
5.4	Parametric evaluation of the Network I-O model. . . . .	90
5.5	Effect of reductions in utilisation threshold ( $u_i$ ). . . . .	93
5.6	Comparison of adaptive strategies. . . . .	98
6.1	A mobile cloud application ecosystem . . . . .	102
6.2	Application composition of a mobile cloud application ecosystem. . . . .	106
6.3	Results from S1, S1F and S2: Offload strategy behaviours of application with increasing weight, and the impact on social costs. . . . .	117
6.4	Results from S3 and S4: Offload strategy behaviours of application within increasing weight, and the impact on social costs. . . . .	119
6.5	Results from Y1, Y2 and Y3: $\Theta_P - \Theta_{Opt}$ . . . . .	122
6.6	Results from Y1, Y2 and Y3: $PoA_P = \Theta_P : \Theta_{Opt}$ . . . . .	123
6.7	Results from Y1, Y2 and Y3: $\Theta_B - \Theta_{Opt}$ . . . . .	124
6.8	Results from Y1, Y2 and Y3: $PoA_B = \Theta_B : \Theta_{Opt}$ . . . . .	125
6.9	Results from Y1, Y2 and Y3: Social costs. . . . .	126
6.10	Price of anarchy following changes in platform parameters. . . . .	128

# List of Tables

1.1	Development of mainstream mobile devices . . . . .	2
3.1	Simulation structure . . . . .	34
3.2	Simulation series specific parameters . . . . .	35
6.1	An example showing the effect of different offload decisions . . . . .	104
6.2	Simulation parameters . . . . .	115
A.1	Comparison of algorithms for MGECP - S series - Solution optimality . . . . .	149
A.2	Comparison of algorithms for MGECP - M series - Solution optimality . . . . .	150
A.3	Comparison of algorithms for MGECP - L series - Solution optimality . . . . .	151
A.4	Comparison of algorithms for MGECP - X series - Solution optimality . . . . .	152
A.5	Comparison of algorithms for MGECP - Solution time . . . . .	153
B.1	Comparison of algorithms for MMUP - S series - Solution optimality . . . . .	155
B.2	Comparison of algorithms for MMUP - M series - Solution optimality . . . . .	156
B.3	Comparison of algorithms for MMUP - L series - Solution optimality . . . . .	157
B.4	Comparison of algorithms for MMUP - L series - Solution optimality . . . . .	158
B.5	Comparison of algorithms for MMUP - Solution time . . . . .	159

# Chapter 1

## Introduction

Technologies that were once thought to be futuristic like self-driving vehicles and wearable digital assistants are fast becoming a reality with the proliferation of mobile smart devices. Indeed, mobile computing is radically changing the way applications are delivered. From traditional office productivity applications like word processors and spreadsheet, to emerging smart mobile applications like personal AI assistants embedded in smart vehicles [13] and large-scale urban sensing projects [14], mobile smart devices and services are at the centre of interest for application developments.

Despite the rapid development of mobile computing technologies, mobile applications are constrained by many limiting factors including battery power, storage space and CPU speed. To overcome these issues, researches turn to cloud computing which has an abundance of computing and storage resources accessible in forms of services that are scalable and easy to integrate. Facilitating the intelligent, seamless and adaptive integration of the two platforms in this joint venture is the role of the emerging research topic of Mobile Cloud Computing.

## 1.1 Mobile, Cloud, Cloud via Mobile and Mobile Cloud Computing

**Mobile** smart devices are becoming the platform of choice for both enterprise and personal computing needs. It was predicted that mobile application development projects would outnumber desktop projects by a ratio of 4:1 by the end of 2015 [15]. Indeed, in recent years the mobile platform’s ability to enable ubiquitous access to services on the move has broadened the usability of many social and entertainment media and created great successes.

With the convenience provided by the compactness of mobile devices come many limitations to its hardware, especially its battery size. In comparison to other components, the pace of development in improving the energy density of smartphone batteries has been slow. In Table 1.1, we list the component specifications of several mainstream smartphones from their first generation to the latest. We see that battery size has not improved in the same order of magnitude as the other components. Furthermore, improvements made at a hardware level have often been taken up by extended software functionalities [16, 17]. Therefore, energy-awareness is crucial for all mobile-related developments.

This energy-awareness also limits the processor’s performance on mobile devices. To prevent over-heating and reduce energy consumption, mobile CPUs are considerably less powerful than their desktop counter parts of the same frequency.

In contrast, **Cloud** computing has a resource rich infrastructure. Implemented over networks of servers and data centres, computing power and storage space on a cloud seem “limitless” when compared with their counter parts on mobile devices. What lacks in cloud computing is its accessibility to the end user. These characteristics make cloud computing a perfect complementary

Table 1.1: Development of mainstream mobile devices

OS	Name	Year	CPU	Memory	Display	Network	App Store	Battery
iOS	iPhone 1	2007	412MHz	128MB	320×480	2G	500	1400mAh
iOS	iPhone 6	2014	Dual 1.4GHz	1GB	750×1334	4G	1.2million	1810mAh
Android	nexus 1	2010	1GHz	512MB	480×800	3G	30k	1400mAh
Android	nexus 6	2014	Quad 2.7GHz	3GB	1440×2560	4G	1.4million	3220mAh

technology to overcome the disadvantages of mobile computing.

Mobile application developer has long adopted cloud services as the back end of mobile applications. In this development paradigm, mobile devices act as an interface to the cloud services which implements the actual logic of the mobile application and holds its database. This **Cloud via Mobile** approach simply defines a client-server structure between mobile and cloud. We further distinguish a mobile application with a cloud back end and a mobile cloud application as we discuss in this thesis in Section 6.1.

As opposed to native mobile applications, applications with a cloud back end are able to take advantage of the richness of cloud services, and are often more feature-rich. For example, user data can be synchronised and shared across all devices registered to a user; more sophisticated algorithms can be applied without concern over energy consumption; sensor data recorded on mobile devices can be used to customise user experience of cloud services.

However, with this client-server structure and its dependency of a wireless connection come new challenges:

1. In a client-server structure, mobile devices rely on wireless data links to communicate with cloud services. Due to the high mobility of mobile devices, wireless data links have fluctuating capacities which greatly affects service QoS.
2. Compared to local computation, wireless communication is associated with higher energy costs. Constant communication with the server quickly drains the battery of mobile devices.
3. Applications of a client-server structure are only available when data link to the server is established. The application does not function locally on device. The user can not use the application unless a wireless data connection is established to the server.
4. The local computation capability of mobile devices has dramatically improved in recent years (Table 1.1). The energy efficiency of these components has also improved. These local computation resources are not fully utilised in a client-server structure.
5. When a group of mobile devices are to cooperate on a workflow, their communication has to be routed via a cloud server. This not only affects the availability of the workflow, when

these devices are of close geographical proximity to each other, this centralised network topology also dictates a higher energy cost than that of a local ad hoc network.

Researches in **Mobile Cloud Computing** aim to address these issues in addition to the limited energy, storage and processing capacities of mobile computing. As we discuss in this thesis, mobile cloud computing extends the integration of cloud and mobile computing further from the client-server architecture and covers a broad spectrum of networking architectures. Mobile cloud computing is more than making cloud services accessible through mobile devices. With techniques like computation and storage offloading (Section 2.1.1), an intelligent and seamless integration of mobile and cloud computing is envisioned in the research of mobile cloud computing.

We give a more detailed overview of mobile cloud computing, its key techniques like computation offloading and its research challenges in Chapter 2.

## 1.2 Mobile Cloud Workflow

Mobile computing and cloud computing are two of the most influential technologies that look set to change the face of computing in the coming years. Combination of the two provides us with an unprecedented opportunity to provide highly portable and yet content-rich and computation-intensive applications to the end user.

With the rapid development of the smartphone and tablet market comes a new generation of handheld devices equipped with faster processors, bigger memories and higher quality displays that have not been seen before in the mobile world. With this improved hardware capability, sophisticated, intelligent and mission-critical processes are being adapted from desktop computers to mobile devices. We also expect to see novel applications utilising the unique features of the devices developed for the mobile world. A workflow is the abstraction of the processes involved in the execution of an application.

A mobile workflow, as discussed in this thesis, consists of a sequence of interactive components<sup>1</sup> that are deployed over a network of distributed mobile devices and cloud servers. In [18],

---

<sup>1</sup>In this thesis, the workload of one such component is referred to as a “task” in the context of a workflow. The actual implementation of each task is referred to as a “service” that is deployed on either a cloud server or on a mobile device.

two scenarios are used to demonstrate how a mass of mobile devices, each used as a rich sensor, organised in a cooperative mobile workflow, can be used to solve real-life problems that could not have been solved by traditional methods. In the first scenario, photos taken on smartphones within a two miles perimeter at a crowded location are gathered to locate a missing child. In the second scenario, photos taken on smartphones in a disaster-stricken area are used to construct detailed maps in aid of rescuing efforts.

Indeed, with the ability to collect rich sensor data and process data at anywhere and at anytime, applications deployed over a network of mobile devices provide the user with much more flexibility than the traditional desktop-based work environments.

To oversee and schedule the execution of mobile cloud workflows, a mobile workflow engine plays an essential part in the development of mobile cloud workflows. The development of a mobile workflow engines as shown in [19, 20, 21] suggests that an organisation is able to rely on the computing and connectivity capabilities within the mobile devices as a substitute to a technology back end server infrastructure.

The main logics and algorithms we discuss in this thesis reside on a workflow engine. We give further description and use cases of mobile cloud workflows and workflow engines in Section 2.2.1.

## 1.3 Outline of Research Contributions

Workload offload [22, 23, 24, 25] is a key technique applied in mobile cloud computing. Aimed at reducing the energy costs of mobile devices, its principle approach is to reduce the computation workload on mobile devices by offloading it to the cloud. Most existing researches are constrained within a one-to-one model in which offloading is carried out between a single device and a single cloud.

In this thesis, we extend the existing research framework of mobile cloud computing and investigate the issues embedded in the many-to-many model of mobile cloud computing which underlies the future developments of the platform.

First, from the platform's perspective, a quadratic program is constructed first to model the energy-aware task allocation problem of a mobile cloud platform. Two energy objectives are investigated looking at minimising the total energy cost and maximising the longevity of the

platform respectively. Two heuristic algorithms are proposed to approximate the solution to both objectives.

Second, from a workflow's perspective, time constraints are considered as well as energy constraints in developing offload strategies. A heuristic algorithm are proposed to produce offload strategies that satisfies both constraints. Effect of hardware and software characteristics over the offload-ability of the workflow are demonstrated.

Third, in order to address the need for efficient resource allocation in mobile networks, a novel network I-O model is proposed to model the bandwidth dependencies between interactive services of a mobile network. The network I-O model lays the foundation for further objective developments. Based on the network I-O model, a cost-based bandwidth allocation scheme is proposed, and models for adaptive bandwidth allocation strategies are also constructed.

Lastly, to address the competition between applications that share the same device, an extension to the classic load balancing game is developed. Three offload decision models of cooperative and non-cooperative nature were constructed. Mixed-strategy Nash equilibrium is derived for the non-cooperative offload game with complete information which further quantifies the price of anarchy in such ecosystems.

More detailed summaries of our contributions are given at the end of Chapter 3, Chapter 4, Chapter 5 and Chapter 6.

## 1.4 Thesis Organisation

This chapter provided a brief overview of the motivations of the research of mobile cloud computing and mobile workflows. It is important to distinguish and understand the relations between mobile cloud computing and the cloud-based mobile application development paradigm as we discussed at the beginning of this chapter.

In the next chapter, we give description of the mobile cloud computing architecture, and motivating applications of the mobile cloud platform. Computation workload offloading techniques which is the key feature of mobile cloud computing and the basis for the development of our efficient frameworks are discussed further in Section 2.1.1.

Each of the four technical chapters (Chapters 3 to 6) of this thesis looks at the mobile cloud



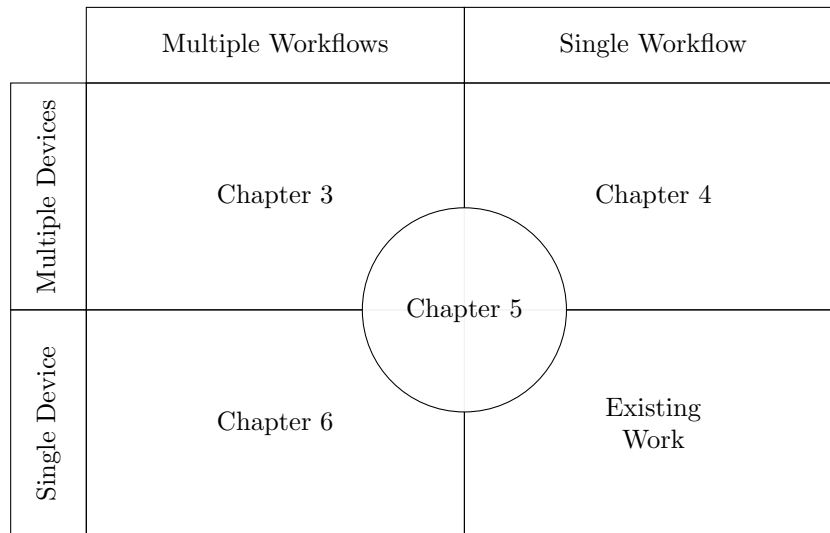


Figure 1.1: Research structure.

computing platform from a different perspective. One way to understand the relations between these chapters is by the number of workflows and the number of devices considered in the scenario which is concerned with in that chapter as illustrated in Fig. 1.1.

From Fig. 1.1, we see that we start this thesis in Chapter 3 which extends existing work from energy-aware offloading between single workflow on a single device to that of multiple workflows on multiple devices. Then in Chapter 4, we take into account the time constraint that is associated with individual workflows and develop offload strategies accordingly. In Chapter 5, we develop a network I-O model to describe the bandwidth dependencies of general mobile networks as a foundation for further efficient bandwidth allocation schemes. Lastly in Chapter 6, we focus on the competition between mobile cloud applications on a single device which give further insight into the importance of a coordinated offloading framework.

## Chapter 2

# Mobile Cloud Computing

This chapter gives description of the research topic of mobile cloud computing. We define the scope of the platform as we discuss in this thesis and its architecture. We focus on existing implementations which enable the computation workload offloading ability of mobile cloud computing platforms. Motivating scenarios and applications of mobile cloud computing are illustrated followed by a discussion of the advantages and disadvantages of the platform.

### 2.1 Architectures and Components of Mobile Cloud Computing

Mobile cloud computing is an emerging field of research that aims to provide a platform on which intelligent and feature-rich applications are delivered to the user's fingertips efficiently. This efficiency comes from the adaptive offload ability of mobile cloud applications which is key to the seamless integration of mobile devices and cloud servers. Pioneered by the likes of MAUI [22], CloneCloud [23] and ThinkAir [26], adaptive computation offload as a core technology in mobile cloud computing has gathered momentum in recent years and has grown from a futuristic concept to a practical means to improve and augment the user's experience of mobile applications.

In this thesis we give two illustrations of the architecture of mobile cloud computing, in Fig. 3.1 and Fig. 4.1. While Fig. 3.1 gives a simple illustration of the underlying network structure of

a mobile cloud platform and a change in task allocation, Fig. 4.1 adds an extra layer of cloud computing resources referred to as **cloudlets** (Section 4.1) into the mobile cloud computing picture and illustrate the benefit of computation offloading with cloudlets in the field.

Commonly applied to enable access to cloud resources a service-oriented architecture (SOA) is also observed in a mobile cloud platform. We give further description of SOA in Section 5.1.2 and related researches that use mobile devices as service hosts in Section 5.1.4. In the rest of this section, we focus on the techniques that enables computation offloading and researches related to the development of mobile workflow engines which are most relevant to all technical chapters of this thesis.

### 2.1.1 Enabling Mobile Computation Offload

The integration of mobile and cloud computing promises the user with convenient access to powerful applications at any time and at any where. The research of computation offload plays an essential part in this vision and ensures that this integration process is both seamless and energy-efficient.

The idea of transferring computation to a nearby processing unit in order to improve mobile application's performance and reduce local energy cost has been researched along with the maturity of mobile technologies. Many ideas and techniques we use in this paper are inspired by this work.

Early research focuses on the partition schemes of an application. Aimed at energy management, a compile-time framework supporting remote task execution was first introduced in [27]. Based on the same approach, a more detailed cost graph was used in [28] with a parametric analysis on its effect at runtime presented in [29]. Another compiler-assisted approach was introduced in [30], which turns the focus to reducing the application's overall execution time. Spectra [31] adds application fidelity (a run-time QoS measurement) into the decision making process and uses it to leverage execution time and energy usage in its utility function. Spectra monitors the hardware environment at run-time and choose between programmer pre-defined execution plans. Chroma [32] builds on Spectra but constructs the utility function externally in a more automated fashion. MAUI [22] also reduces the programmer's workload by automating some

of the partitioning process models. The offload decision engine applies an integer programming techniques to produce allocation schemes.

Before Cloud, opportunistic use of surrogates (untrusted machines) was adopted in [33] and [34]. Slingshot [34] also identifies wireless hotspots as a platform to accommodate the virtual machine capsule. As Cloud Computing and Virtual Machine technologies become mainstream, more researches turned to the Cloud in search of a more secure, accessible and powerful offload platform. OS supported VM migration was introduced in CloneCloud [35]. Calling-the-cloud [36] add a middleware platform that manages an application's execution between the phone and the cloud. A consumption graph is used to model the application. Wishbone [37] looks at the partitioning of sensor network applications in particular and models the decision making process as a integer program. Aimed at reducing the communication costs [38] proposes the concept of cloudlets, which brings the distant Cloud to the more commonly accessible WiFi hotspots. A dynamic VM synthesis approach is also suggested in [38].

Our research is distinguishable to existing work since the applications that we investigate have computation tasks scattered over a group of distributed mobile devices (i.e. a mobile workflow), whereas existing researches look at applications that are implemented on one device only.

Besides existing research level implementations of mobile cloud applications (we recommend three excellent surveys, [39], [40] and [41], to the interested readers for a comprehensive list of existing researches in mobile cloud computing), we argue that the increasing popularity of HTML5 as a mobile application development framework also greatly shortens the time required to develop applications that are deployable both natively on the mobile device and remotely as cloud services. The use of HTML5 and platforms like Apache Cordova help significantly lower the level of technical challenges involved in the development of mobile cloud applications. We expect to see an increasing number of mobile applications to adopt the adaptive execution approach proposed by the research of mobile cloud computing in the near future.

### 2.1.2 Mobile Workflow Engine

A workflow engine is often required to oversee the execution of mobile workflows. In [19] a detailed mobile workflow engine is implemented and tested on Nokia devices. A decentralised

workflow coordination architecture designed for mobile devices is presented in [42] for use in biological studies and the supply-chain industry. Authors of [20] propose a rapid application development framework based on a dynamic workflow engine for creating mobile web services. A Field Service application is presented in [20] as an example use case.

Several researches has been carried out in workflow management issues in mobile social content sharing applications [43, 44, 45, 46]. A mobile P2P social content sharing framework was proposed in [44]. In [45], a Java API based mobile workflow system was proposed. A content distribution protocol was proposed in [47] for vehicular ad hoc networks (VANET). Clusters of mobile devices has been proposed in [48] to support the execution of parallel applications. A workflow engine is present in all these researches to oversee the scheduling of tasks.

The common approach towards an allocation problem often model the problem as a linear programs (LP) [22, 49, 50] in the workflow engine. LPs are suitable for modelling situations where communication time is not considered or when there are only two devices involved in the process. However, in the cases of mobile workflows, communication tasks are an essential part of the workload and occur significant amount of energy cost[51]. Thus we construct our model's objective function as a quadratic program in Chapter 3 in order to accurately capture the communication costs.

## 2.2 Impact of Mobile Cloud Computing

### 2.2.1 Motivational Applications of Mobile Cloud Computing

Mobile cloud applications combine the accessibility, agility and the rich sensing ability of the mobile computing platform with the abundance of services provided by the cloud computing platform and is applicable in many fields of researches. [41]

#### **Healthcare**

Mobile medical treatment and health monitoring devices are constrained by its physical size just like smartphones. mHealthMon [52] is a mobile health monitoring platform which adopts the computation offloading technique in mobile cloud computing to improve the energy efficiency of

the process of continuously gather, process and update sensor readings for patients. In mHealth-Mon, users publish and access sensor data via a cloud-based P2P overlay network. @HealthCloud [53] proposes the implementation of a mobile system that enables electronic healthcare data storage, update and retrieval utilising cloud services provided by Amazon S3 and an Android app. Although computation offload is not applied in @HealthCloud, the system benefit greatly from the cooperation of mobile and cloud computing.

### Image and Video Processing

Cuckoo [24] implements a face detection over its offloading platform and is able to reduce the energy cost as well as augment the features provided by the application. MAUI [22] is also tested with a face detection application and significant energy savings has been make by applying computation offload. The two scenarios given in [18] as discussed in Section 1.2 are both based on crowd-sourced imaging informations. GigaSight [54] is a scalable video crowd-sourcing application which gathers video information from devices such as Google Glass. Privacy sensitive information is automatically removed from the video based on time, location and content informations. Cloudlets are used to support the computation and storage offloading of GigaSight and also achieve scalability.

### Gaming

A video game and a chess game is tested by MAUI [22]. Although significant energy savings were made under good network conditions, significant savings in execution time has been made in the video game application. In [55] an adaptive rendering technique is applied to maximise user experience of cloud-based mobile gaming given constraints in communication and computation in the wireless network.

### 2.2.2 Illustrative Use Cases of Mobile Workflows

A cloud-assisted mobile workflow as we discuss in this thesis is an application workflow that is implemented over a group of mobile devices with access to cloud resources. The cloud resources may either be a compulsory component in the execution of the workflow, or be in an assistant

role to handle computation offload requests sent from the mobile devices. Mobile application workflows can be found when a group of mobile users are to share or communicate with each other in order to accomplish a certain task. We give two example use cases of such workflows in Fig. 2.1, Fig. 2.2 and Fig. 2.3 to further clarify our objective.

**Enterprise use case:** With increasing adaptation of mobile devices into enterprise business models[15], modern enterprise applications often include or are entirely based on mobile devices. Fig. 2.1 illustrates an application workflow involving three mobile devices and two cloud services of a supply-chain business. Two employees are concerned with the receipt and sale of goods respectively. Both activities require an up to date pricing information at runtime. A manager is concerned with the trends that are developing in the company's inventory in real-time which is produced via a series of tasks like forecast and analysis.

There are two types of tasks in this workflow: tasks like login, record sale and database queries that are fixed either on a mobile device or a cloud node; and tasks like data analysis and forecast that can be offloaded between devices and clouds. Depending on the computation size and communication size of these offload-able tasks, an energy-aware allocation scheme can help optimise the execution of the workflow in term of its overall energy cost imposed onto the mobile cloud platform.

**Consumer use case 1:** Because of its portability, mobile devices encourages the development of collaborative application. In Fig. 2.2, we illustrate the collaboration of three mobile devices in scanning the 3D structure of an object. Pictures of the object is taken on all three devices at the same time and once pre-processed, these pictures are gathered to construct the 3D model of the object. Two cloud services are available to the users, one for storage and one for application hosting and computation offloading.

Similar to the enterprise use case, the tasks involved in this mobile application workflow can either be fixed or offload-able over the mobile cloud platform. Allocation of tasks is critical in deciding the energy-footprint of the workflow. For instance, once the picture has been pre-processed, the subsequent communication size may be reduced. Comparing this reduction to the computation cost of the pre-processing task on the mobile device, it may or may not be beneficial for the device to offload this task to the cloud.

Our research investigates ways to model and optimise the energy efficiency of such workflows

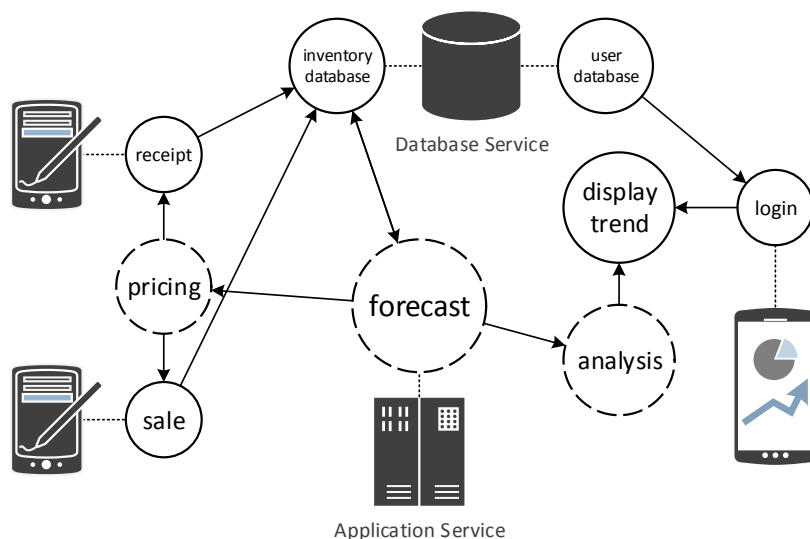


Figure 2.1: Example use case of an MCC workflow: Business task interaction. Solid circles represent tasks that are fixed on a device or a cloud service, whereas dashed circles represent tasks that are free to offload / migrate between devices and clouds. Allocation of tasks significantly affects the energy-efficiency of the mobile cloud platforms.

running atop a platform of mobile and cloud devices. Our goal is to develop ways to produce energy-aware task allocation schemes and provide an energy efficient execution platform for cloud-assisted mobile workflows.

**Consumer use case 2:** Fig. 2.3 illustrates three smartphones and a tablet, and expands on the idea of a popular consumer game application [56] that allows the user to use a smartphone and a tablet to emulate the equipments used in a darts game. The tablet is used to display a dart board, the smartphones are used by each participating player as their darts. The workflow starts when a player throws a dart (by waving the phone towards the tablet). Sensor readings (accelerometer and gyroscope) are then taken from the phone and fed into a calculation module to work out where the dart should land on the board. Once the calculation is finished, the result is then passed on to the display module of the tablet.

Like all multi-player competitions, the game can only function until its weakest player withdraws, which in this case, is the device that runs out of battery first. The module for calculating the landing point might not cause too much energy to run each time, however, repeat execution is required during the game. The device which runs this module consumes considerable amount



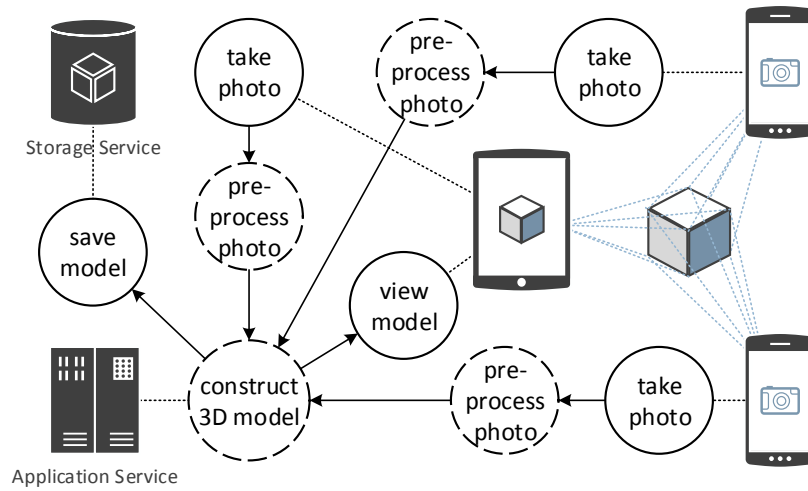


Figure 2.2: Example use case of an MCC workflow: 3D scan workflow.

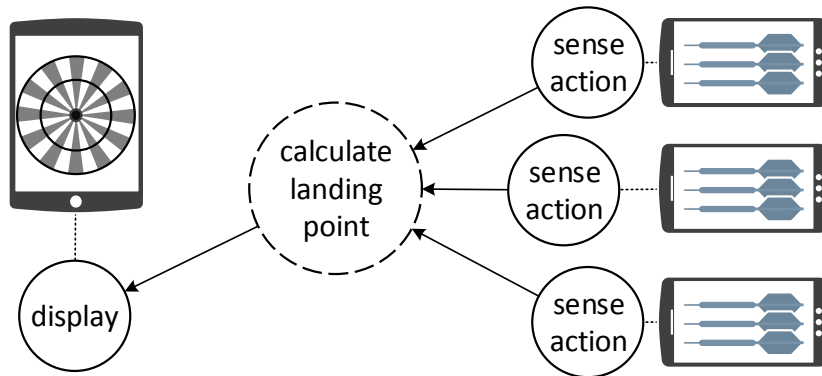


Figure 2.3: Example use case of an MCC workflow: Darts game.

of energy more than the others over time. A fair task allocation, which we study in Section 3.3, is needed in such scenarios to balance the contribution made by participating members of the workflow.

### 2.2.3 Advantages and Issues of Mobile Cloud Computing

As a combination of mobile and cloud computing, mobile cloud computing combines the advantages of both platforms including convenient access to services at anywhere and at any time enabled by the portability of the mobile device, and the on demand access to limitless computation and storage capacities provided by cloud services.

- Access to applications at anywhere and any time
- Greater computation power to implement sophisticated applications
- Bigger, more reliable and synchronised storage space
- On demand scalability of cloud services
- Reduced battery consumption

However, because of use of a wireless connection that is both energy-consuming, unstable and limited in capacity, there are open issues that need to be addressed in order to maximise the benefit of the mobile cloud platform.

**Energy-Awareness** Wireless communication is expensive in energy costs. The wireless radio unit is the most energy-consuming component in smart devices[51, 57]. Therefore task allocation and offloading actions which changes the communication costs of the workflow need to be managed in an energy-efficient manner. We address these issues by developing energy-aware task allocation and offload strategies in Chapter 3 and Chapter 4.

**Network Capacity, Reliability and Cost** Bandwidth is a valuable asset on all types of mobile networks and limits the offload-ability and efficiency of mobile workflows. Therefore knowing the bandwidth dependency between interacting mobile services is key to the resource management aspect of mobile cloud computing. Furthermore, the reliability issue related to the wireless network requires adaptive strategies to accommodate changes in network conditions. Wireless connections can be expensive and therefore the provisioning of bandwidth need to be cost effective. We address these issues by constructing a network I-O model in Chapter 5.

**Competition On Device** Mobile devices are crowded with applications and services of different purposes. Many require the access to cloud resources. The unawareness of information of other applications and the selfish behaviour exhibited by application put the system as a whole in a sub-optimal position, We address this issue by constructing a game theoretical framework to model the offload decision models in a mobile cloud platform and also propose cooperative solutions to optimise system performance in Chapter 6.

## Chapter 3

# Energy-Aware Task Allocation in Mobile Cloud Platforms

Improvement in mobile applications' energy efficiency is one of the principle motivations behind the development of mobile cloud computing technologies. This improvement is facilitated by the computation offloading capability provided by mobile cloud platforms. Relocating offload-able tasks in an application's workflow from one host to another modifies the energy profile of the platform. Therefore, solving the task allocation problem becomes a critical first step in ensuring the energy efficiency of the mobile cloud platform. The optimality of task allocation schemes is a fundamental issue which greatly affects the energy efficiency of application workflows running atop a mobile cloud computing platform.

In this chapter, we investigate the energy-aware task allocation problem in a mobile cloud platform with two distinctive objectives. For our first objective, which we refer to as the Minimum Group Energy Cost Problem (MGECP), we aim to minimise the overall energy cost of the platform. We construct a quadratic program to model the MGECP. Using the quadratic objective function as constraints, we further developed a quadratically constrained program to model our second objective which we refer to as the Minimum Maximum Utilisation Problem (MMUP). In the MMUP, we aim to distribute the tasks fairly while taking into account the size of the residual energy of each mobile devices of the platform.

In a cloud-assisted mobile application workflow, energy is spent on both executing the computation of tasks and the communications between tasks. Therefore, quadratic objectives and constraints are inevitable in modelling the allocation problem in such scenarios. This increases the complexity of finding the exact solution to our problems. To approximate the optimal solution, we developed two heuristics including an implementation of the simulated annealing (SA) algorithm and a greedy autonomous offload (GAO) algorithm.

The solution optimality and execution time cost of our heuristics are compared to that of the QP and QCP solver libraries of CPLEX [58] version 12.6.1, an industry-leading optimisation software package, herein through the analysis of a series of simulation results. We also investigate and report on the relations between MGECP and MMUP based on simulation results.

## 3.1 Mobile Cloud Platform Model

### 3.1.1 Mobile, Cloud and Network Metrics

We consider a mobile cloud platform  $MCP$  consisting of a set of  $p$  processing nodes, denoted  $P = \{P_1, \dots, P_p\}$ . Each processing node may either be a *mobile device* node from set  $P^M \subseteq P$  or a *cloud resource* node from set  $P^C \subseteq P$ , with  $P^M \cap P^C = \emptyset$  and  $P^M \cup P^C = P$ . We denote a mobile device node profile as  $P_i^M(s_i, e_i^{cmp}, e_i^{snd}, e_i^{rcv}, e_i^{mnt}), i \in \{1, \dots, |P^M|\}$  and a cloud resource node profile as  $P_i^C(s_i)^1, i \in \{1, \dots, |P^C|\}$  with parameters defined as follows:

$s_i$ , denotes the peak processing speed of  $P_i \in P$ , measured in the number of clock cycles available in a millisecond;

$e_i^{cmp}$ , denotes the current draw from the battery of  $P_i^M \in P^M$  when the device is executing computation tasks at peak speed;

$e_i^{snd/rcv}$ , denotes the current draw from the battery of  $P_i^M \in P^M$  when the device is sending/receiving data to/from the data network.

---

<sup>1</sup>Although we don't associate any particular energy metrics for a cloud processor, the notation of  $e_i^{snd/rcv/mnt}$  is applied in the rest of this chapter even when  $i \in P^C$  in order to keep the notations easy to follow. The reader may assume that these variable are of value 0.

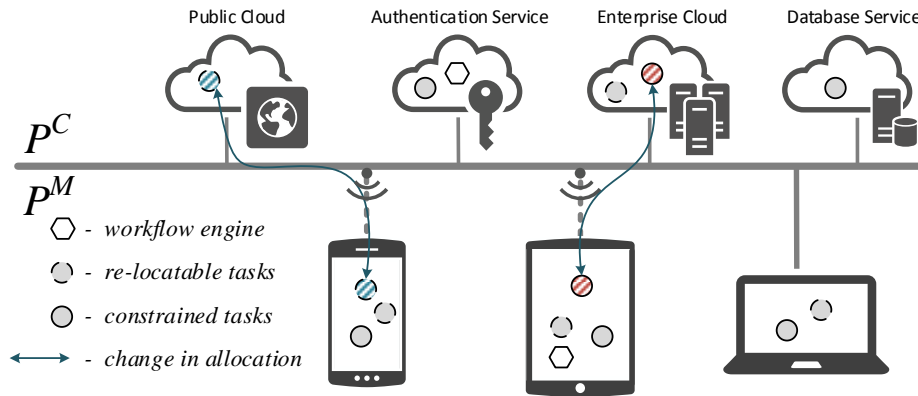


Figure 3.1: Illustration of a Mobile Cloud Platform.

$e_i^{mnt}$ , denotes the current draw from the battery of  $P_i^M \in P^M$  when the device is maintaining the wireless connection alive to anticipate transmission of data.

All nodes ( $P_i \in P$ ) are interconnected via a network (as illustrated in Fig. 3.1 with solid lines being wired connections and dotted being wireless connections), and we use  $b_{ij}$  to denote the bandwidth between devices  $P_i$  and  $P_j$ ,  $i, j \in \{1, 2, \dots, p\}$ . Thus, we have p-matrices  $B = (b_{ij})_{p \times p}$  and  $L = (l_{ij})_{p \times p}$  which hold all of the bandwidth and latency information of the underlying network of the MCP. When two adjacent tasks are assigned to the same device, we assume that they share the same memory address space on the device. Therefore, we assign positive infinite values to the principal diagonal elements of  $B$ , that is  $b_{ii} = +\infty, i \in \{1, 2, \dots, p\}$ , and zeros to the principal diagonal elements of  $L$ , that is  $l_{ii} = 0, i \in \{1, 2, \dots, p\}$ .

Our choice of energy profile parameters is in line with researches that investigate the energy characteristics of mobile devices and wireless networks [51, 59, 60, 61, 62, 63, 64] especially for the wireless module. We especially emphasis the difference in energy consumption between the sender and the receiver of the data, and that it cost energy to maintain a live connection in anticipation of data transmission. We refer the interested reader to [57] for a comprehensive survey in energy consumption models and techniques of modern mobile devices.

### 3.1.2 Application Workflow Metrics

The tasks of application workflows hosted on MCP and their interactions are represented by a directed graph  $W = (T, R)$  whose vertex set  $T = \{t_1, \dots, t_n\}$  denotes the set of *tasks* of the

workflows. The ordering of the tasks are defined as the topological order given by  $W$ . We assume that all tasks are defined via a service-oriented architecture (SOA) and that services may be available on more than one device. An  $n$ -matrix  $D = (d_{ab})_{n \times n}$  denotes the weighted adjacency matrix of  $W$ , where  $d_{ab}$  is the size of the data package that is to be sent from  $t_a$  to  $t_b$  for  $(t_a, t_b) \in R$ . All principle diagonal elements of  $D$  are zeros.

Each task has profile  $t_a(d_{(\cdot,a)}, d_{(a,\cdot)}, c_a)$ ,  $a \in \{1, \dots, n\}$  where  $d_{(\cdot,a)}$  and  $d_{(a,\cdot)}$  are the  $a$ -th column and the  $a$ -th row of  $D$  which represent the incoming and outgoing data respectively.  $c_a$  denotes the workload size of the task.

### 3.1.3 Fixed and Constrained Tasks

Not all tasks of a mobile workflow are suitable for offload from its host. For instance, a task that authenticates the user's identity using the fingerprint reader on the smartphone has to be executed on the smartphone; a task that manages database files saved on a cloud-storage service has to run locally on that cloud; a task which senses the user's heartbeat resides on the smart-watch, etc. These tasks are *fixed* on their hosts.

Furthermore, there may be tasks that are only allowed to be executed on a subset of the devices from  $P$ . For instance, when a task is associated with sensitive data shared between a group of users, or when the OS of each device varies in versions such that the execution of certain tasks are not supported by all of  $P$ . The allocations of these tasks are *constrained* within a group of devices.

We denote the set of devices that a task  $t_a$  may execute on with  $P^{t_a}$ . For a fixed task,  $P^{t_a}$  has a cardinality of one and contains only its host. For a constrained task,  $P^{t_a}$  includes only devices on which  $t_a$  is installed. For a task which is not at all constrained,  $P^{t_a} = P$ .

### 3.1.4 Allocation Scheme and Energy Costs

Given an allocation scheme  $\psi : T \rightarrow P$ , we first derive the energy cost of computing  $t_a$ ,  $a \in \{1, \dots, n\}$  to be

$$\mathcal{E}_{a\psi(a)}^{cmp} = e_{\psi(a)}^{cmp} \times \frac{c_a}{s_{\psi(a)}} \quad (3.1)$$

where  $\psi(a)$  is the device to which  $t_a$  is assigned. Secondly, we have the energy cost of transferring  $d_{ab}$ ,  $(t_a, t_b) \in R$  as given by

$$\mathcal{E}_{ab\psi(a)\psi(b)}^{tran} = \underbrace{e_{\psi(a)}^{snd} \times \frac{d_{ab}}{b_{\psi(a)\psi(b)}} + e_{\psi(a)}^{mnt} l_{ab}}_{\text{sender's cost}} + \underbrace{e_{\psi(b)}^{rcv} \times \frac{d_{ab}}{b_{\psi(a)\psi(b)}} + e_{\psi(b)}^{mnt} l_{ab}}_{\text{receiver's cost}} \quad (3.2)$$

(3.2) accounts the data transmission cost from both the sender device and the receiver device of the data. Modern wireless transmission components on a mobile device such as WiFi and cellular 3G and 4G modules apply a third power state to intermediate the off and transmit power states. This power state is often referred to as the “idle” power state in related researches such as [63, 64, 65, 66]. We refer to this power state as the “maintenance” cost of the data link, denoted by  $e_i^{mnt}$  with  $i \in \{1, \dots, |P^M|\}$ . In this power state, a device is ready to transmit or receive but because of the latency between the devices, the actual transmission or receiving action has yet to happen, therefore the higher power states ( $e_i^{snd}$  and  $e_i^{rcv}$ ) do not yet apply.

## 3.2 Minimum Group Energy Cost Problem

In this section, our objective is to minimise the energy cost of the MCP as a whole. We first show that the Minimum Group Energy Cost Problem (MGECP) can be modelled as a generalised Quadratic Assignment Problem (QAP) [67] and then we convexify the objective function in order to obtain the optimal solution using a QP solver.

### 3.2.1 Assignment Matrix

To represent an allocation scheme  $\psi$ , we first construct an  $n \times p$  binary matrix  $X^\psi = (x_{ai}^\psi)_{n \times p}$ , such that

$$x_{ai}^\psi = \begin{cases} 1 & \text{if } \psi(a) = i, \quad t_a \in T, \quad P_i \in P \\ 0 & \text{otherwise.} \end{cases} \quad (3.3)$$

We call matrix  $X^\psi$  an *assignment matrix* and a valid assignment must satisfy the following constraints

$$\sum_{P_i \in P^t_a} x_{ai}^\psi = 1, \quad a = 1, 2, \dots, n, \quad (3.4)$$

$$x_{ai}^\psi \in \{0, 1\}, \quad a = 1, 2, \dots, n, \quad i = 1, 2, \dots, p. \quad (3.5)$$

(3.4) ensures that every task must be assigned to one and only one device within the group of devices which it is able to execute. (3.5) states that all tasks are indivisible.

### 3.2.2 Quadratic Program Formulation

With (3.1) (3.2) and (3.3), we can derive the group energy cost function as

$$\begin{aligned} \mathcal{E}^\psi &= \sum_{b=1}^n \sum_{j=1}^p \sum_{a=1}^n \sum_{i=1}^p \left( (e_i^{snd} + e_j^{rcv}) \frac{d_{ab}}{b_{ij}} + (e_i^{mnt} + e_j^{mnt}) l_{ab} \right) x_{ai}^\psi x_{bj}^\psi \\ &+ \sum_{a=1}^n \sum_{i=1}^p e_i^{cmp} \frac{c_a}{s_i} x_{ai}^\psi \end{aligned} \quad (3.6)$$

The quadratic terms in (3.6) gives the total energy cost for data transmission, whereas the linear term gives the total energy cost for executing computing tasks. Next we introduce  $(pn)^2$  coefficients  $q_{aibj}$  as given by

$$q_{aibj} := \begin{cases} e_i^{cmp} \frac{c_a}{s_i} + \underbrace{(e_i^{snd} + e_j^{rcv}) \frac{d_{ab}}{b_{ij}} + (e_i^{mnt} + e_j^{mnt}) l_{ab}}_{=0} & \text{if } (a, i) = (b, j), \\ e_i^{snd} \frac{d_{ab}}{b_{ij}} + e_i^{mnt} l_{ab} & a < b, \\ e_j^{rcv} \frac{d_{ba}}{b_{ij}} + e_j^{mnt} l_{ab} & a > b. \end{cases} \quad (3.7)$$

With (3.7) we can transform (3.6) to

$$\text{minimise: } \mathcal{E}^\psi = \sum_{b=1}^n \sum_{j=1}^p \sum_{a=1}^n \sum_{i=1}^p q_{aibj} x_{ai}^\psi x_{bj}^\psi \quad (3.8)$$

as the objective function of our quadratic program.



**Theorem 3.2.1.** *Let coefficients  $q_{aibj}$  be the entries of an  $pn \times pn$  matrix  $Q$ , such that  $q_{aibj}$  is on row  $(i-1)n+a$  and column  $(j-1)n+b$ , and  $\text{vec}(X^\psi) = (x_{11}^\psi, x_{12}^\psi, \dots, x_{1n}^\psi, x_{21}^\psi, \dots, x_{pn}^\psi)^T$  be the vector formed from the columns of  $X^\psi$ .*

*Equivalent formulations for the MGECP objective function are given by (3.8) and*

$$\text{minimise: } \mathcal{E}^\psi = \text{vec}(X^\psi)^T \cdot Q \cdot \text{vec}(X^\psi) \quad (3.9)$$

*Proof.* From the construction of  $\text{vec}(X)$ , we observe that its  $u$ -th element  $\text{vec}(X^\psi)_u = x_{ai}^\psi \Leftrightarrow u = (i-1)n+a$ . Furthermore, given  $u = (i-1)n+a$  and  $v = (j-1)n+b$ ,  $u, v \in \{1, 2, \dots, pn\}$ , we also get  $Q_{uv} = q_{aibj}$ . Hence,

$$(3.9) = \sum_{v=1}^{pn} \sum_{u=1}^{pn} \text{vec}(X^\psi)_u^T Q_{uv} \text{vec}(X^\psi)_v = \sum_{b=1}^n \sum_{j=1}^p \sum_{a=1}^n \sum_{i=1}^p x_{ai}^\psi q_{aibj} x_{bj}^\psi = (3.8)$$

□

Therefore (3.9) constrained by (3.4) and (3.5) completes our quadratic program formulation for the MGECP. The assignment matrix for the MGECP is given by

$$\arg \min_{X^\psi} (\mathcal{E}^\psi) \quad (3.10)$$

### 3.2.3 Convexification

With the quadratic program formulated, we can find its exact optimal solution using QP solvers. In order to exploit the power of modern QP solvers, we first need to pre-process the problem and convexify the objective function [68]. There are a number of ways of convexification. Our process is similar to that use in [69].

**Theorem 3.2.2.** *Let  $Q^* := 1/2(Q + Q^T) + \alpha I$ , where  $I$  is the  $pn \times pn$  identity matrix, then  $Q^*$  is positive definite if scalar  $\alpha = 1 + \|Q\|_\infty$*

*Proof.* We observe that  $Q^*$  preserve the properties of being a square, non-negative matrix from  $Q$ , and that it is also symmetric  $\Leftrightarrow q_{uv}^* = q_{vu}^*$ . Recall that  $\|Q\|_\infty = \max_{1 \leq u \leq mn} \{\sum_{v=1}^{mn} |q_{uv}|\}$ .

Therefore we have

$$\begin{aligned}
 x^T Q^* x &= \sum_{u=1}^{mn} q_{uu}^* x_u^2 + 2 \sum_{u=1}^{mn-1} \sum_{v=u+1}^{mn} q_{uv}^* x_u x_v & (3.11) \\
 &= \sum_{u=1}^{mn} \left( q_{uu}^* - \sum_{\substack{v=1 \\ v \neq u}}^{mn} q_{uv}^* \right) x_u^2 + \sum_{u=1}^{mn-1} \sum_{v=u+1}^{mn} q_{uv}^* (x_u + x_v)^2 \\
 &= \sum_{u=1}^{mn} \left( q_{uu} + \alpha - \sum_{\substack{v=1 \\ v \neq u}}^{mn} q_{uv}^* \right) x_u^2 + \sum_{u=1}^{mn-1} \sum_{v=u+1}^{mn} q_{uv}^* (x_u + x_v)^2 \\
 &= \sum_{u=1}^{mn} \left( 2q_{uu} + \alpha - \sum_{v=1}^{mn} q_{uv} \right) x_u^2 + \sum_{u=1}^{mn-1} \sum_{v=u+1}^{mn} q_{uv} (x_u + x_v)^2 \\
 &\geq \sum_{u=1}^{mn} \left( 2q_{uu} + \underbrace{\alpha - \sum_{v=1}^{mn} q_{uv}}_{\geq 1} \right) x_u^2
 \end{aligned}$$

Hence  $x^T Q^* x > 0$  for any  $x \neq 0$  and the proof is complete. Similar formulation was first introduced in [69] to construct a negative definite matrix.  $\square$

Addition of a constant on the main diagonal of  $Q$  only adds a constant to the value of (3.9) and does not change its optimal solution to (3.10). Hence we can rewrite our objective function as

$$\text{minimise: } \text{vec}(X)^T Q^* \text{vec}(X) \quad (3.12)$$

This together with (3.4) and (3.5) completes the formulation of the QP solvable optimisation problem of the MGECP. The positive definite property of  $Q^*$  ensures that (3.12) is strictly convex and a global minimum can be found by existing QP solvers.

### 3.3 Minimum Max-Utilisation Problem

While solutions generated by the MGECP ensures that the application workflows running atop the MCP consumes minimum amount of energy from the mobile devices as a collection, it does not consider the stress it has on individual devices. This could result in unfair energy cost distribution within the MCP, and create *over-utilised* devices. Having such workflow executed repeatedly over

time without adjustment to its task allocation scheme could lead to early retirements of over-utilised devices from the MCP. In a business environment, it is common to have authorisation constrained services or tasks taking critical roles within workflows. In such cases, the MCP's inclusion of these authorised devices is critical to the fulfilment of the application workflow's functionalities. Retirement or withdrawal of devices which support this set of critical tasks could lead to the retirement of the entire workflow. This leads to the shift in the workflow engine's priority from reducing the total energy cost of the group to ensuring the availability of individual devices.

Hence in this section, we look at ways to adjust the task allocation provided by the MGECP so that the availability period (or the number of run counts) of a workflow can be lengthened. We refer to this class of problem as the Minimum Max-Utilisation Problem (MMUP).

### 3.3.1 Device Cost Matrix

Unlike the MGECP, we need to add additional constraint to individual devices' energy cost in order to control the outcome of the adjustment action. In order to formulate these constraints, we first introduce the device specific cost matrix  $Q^i$ :

**Theorem 3.3.1.** *Let*

$$Q_{uv}^i = \begin{cases} Q_{uv} & \text{if } n \times (i-1) < u \leq n \times i, \\ 0 & \text{otherwise.} \end{cases} \quad (3.13)$$

for  $i \in \{1, \dots, |P^M|\}$ . Then given an allocation scheme  $\psi$  and its allocation matrix  $X^\psi$ , we have the energy cost of  $M_i$  to be

$$\mathcal{E}_i^\psi = \text{vec}(X^\psi)^T Q^i \text{vec}(X^\psi) \quad (3.14)$$

*Proof.* Proof is similar to the proof of Theorem 3.2.1 and can be worked out easily.  $\square$

### 3.3.2 Device Utilisation

Next, we introduce the measure of device utilisation:

**Definition 1.** Given an allocation scheme  $\psi$ , the *utilisation* of  $M_i$ , denoted  $\mathcal{U}_i^\psi$ , equals  $\mathcal{E}_i^\psi / \mathcal{E}_i^R$ , for  $P_i \in P^M$ , where  $\mathcal{E}_i^\psi$  is the energy cost of  $P_i$  under  $\psi$  and  $\mathcal{E}_i^R$  is the size of the residual energy of  $P_i$ .

It is easy to see that the reciprocal of a device's utilisation,  $(1/\mathcal{U}_i^\psi)$ , is the number of times  $P_i^M$  can support the workflow before it runs out of battery. The wholeness of the workflow is limited to the device which can afford the least number of runs. This implies that the availability period of a workflow is hence constrained by the member with the highest value of utilisation. Redistribution of workload from over-utilised devices to under-utilised devices can help reduce  $\mathcal{E}_i$  and thus lengthen the workflow's availability prospect.

By definition, another aspect that determines a device's utilisation value is its residual battery value  $\mathcal{E}_i^R$ . Because the resources on most mobile devices are often not exclusively managed in coordination with the services that serve the cause of the workflows, the workflow engine cannot accurately predict the residual battery value over time. For instance, a user's decision to watch an online video over 3G network can quickly drain the battery, and that in turn will affect the prospect of the workflows in which it plays a critical role. Furthermore, unlike devices with fixed locations, mobile devices are constantly exposed to the changing conditions of the open world. This is especially true with the changing signal strengths of the data links which can put extra stress on the device's battery.

### 3.3.3 Quadratically Constrained Program Formulation

The MMUP is a min-max problem over each device's utilisation ratio:

$$\text{minimise: } \max\{ \mathcal{U}_i^\psi \}, \quad i \in 1, \dots, |P^M| \quad (3.15)$$

subject to: (3.4) and (3.5)

With the introduction of an auxiliary variable

$$y \geq \mathcal{U}_i^\psi, \quad i \in 1, \dots, |P^M| \quad (3.16)$$

we transform the formulation of the MMUP to a Quadratically Constrained Problem (QCP):

$$\begin{aligned} & \text{minimise:} && y && (3.17) \\ & \text{subject to:} && (3.16), (3.4) \text{ and } (3.5) \end{aligned}$$

Our formulation of the MMUP is now complete. The problem is now solvable by QCP solvers. We apply the same convexification process (Section 3.2.3) to the quadratic constraints (3.16).

## 3.4 Heuristics

Amongst all combinatorial optimisation problems, QAP is one of the hardest to solve [70]. Despite the development of modern QP solvers, finding the global optimal solution of a QAP remains a computational consuming task. When the problem size gets bigger ( $p \times n > 200$  in our experience on a laptop with a first generation Core i7 processor and 8GB of memory), finding the exact solution becomes impractical. This is especially true with the QCP formulation of the MMUP because of the number of quadratic constrains follows the number of mobile devices in an MCP. Therefore we designed two heuristics to approximate the solutions.

Note that in both heuristics, we assume that an allocation scheme  $\psi$  is currently in place to schedule all tasks on the MCP.

### 3.4.1 Simulated Annealing

Simulated annealing [71, 72] is a meta-heuristic algorithm often applied to NP-hard combinatorial optimisation problems including QAP [70]. One of the main features of simulated annealing is that by occasionally allowing inferior solutions on its search path, the algorithm is able to perform uphill<sup>2</sup> search steps so that its solution needs not get stuck at local optimal points.

The algorithm has a simple structure. As illustrated in Algorithm 1, the main procedure has two nested loops. The outer loop (line 4 - 17) iterates for a number of *cycles*. Each cycle

---

<sup>2</sup>Note that because of the random uphill actions in a simulated annealing search, the solution from its last trial may not be the best it has ever come across. Therefore, as well as letting the algorithm to develop a best possible solution by itself, we also keep a record of the best solution throughout the searching process. We return this best solution as the final output of the algorithm. We omit this from Algorithm 1 so that the structure of the algorithm remains clear to the reader.

corresponds to a temperature  $T$  which is trialed in the inner loop (line 5 - 15) and *cooled* by a cooling ratio ( $0 < T_c < 1$ ) at the end of each cycle. At the start of the inner loop, a candidate ( $\psi'$ ) is randomly chosen from the local neighbourhood of the current solution ( $\psi^*$ ). This candidate is then accepted as the best solution either through the fact that it improves the value of our objective function or with probability  $\min\{1, \exp(-|Obj^{\psi'} - Obj^{\psi^*}|/T)\}$  regardless of whether it improves the objective function or not.

The structure of the simulated annealing algorithm is independent from the problem it applies to. Therefore our implementation for both MGECP and MMUP share the same structure as presented in Algorithm 1. What distinguishes the two solutions are their implementation of the objective function ( $Obj^{\psi}$ ), for the MGECP this is (3.9), and for the MMUP this is (3.15).

Next, we discuss details of our implementations of the simulated annealing algorithm.

### Cooling Schedule

The use of the exponential function (line 10) in the simulated annealing algorithm means that the probability ( $p$ ) of a candidate / incumbent solution ( $\psi'$ ) being accepted is directly related to the temperature ( $T$ ) of each cycle. The higher the temperature, the higher chance that an inferior candidate solution may be accepted. Likewise, following the cooling (reduction) of  $T$  at the end of each cycle, the probability of an inferior solution being accepted by the algorithm is gradually reduced towards the end of the algorithm.

Because of this property, the performance of an implementation of the simulated annealing algorithm greatly depends on its choice of a *cooling schedule*. On one hand, the *initial temperature* must be high enough such that the final solution is independent from the initial solution ( $\psi_0$ ). A low initial temperature constraints the development of the solution by assigning low or zero probability to inferior solutions from the start of the algorithm. On the other hand, the *exit temperature* needs to be small enough so that the development of the final solution is adequately constrained by the algorithm. A solution produced by a high exit temperature is less refined and may be randomly further away from the optimal solution.

In our implementation of the simulated annealing algorithm we set the initial and exit temperature as  $T_0 = (\log(p_0))^{-1}$  and  $T_e = (\log(p_e))^{-1}$  where  $p_0$  and  $p_e$  are the initial and exit acceptance probabilities that we set out. Then we have  $T_c = (T_e - T_0)^{\frac{1}{\text{NumberOfCycles} - 1}}$  to complete the cooling

schedule. We also normalise  $|\mathcal{E}^{\psi'} - \mathcal{E}^{\psi^*}|$  by its averages in each cycle at line 10. To preserve the essential structures of a simulated annealing algorithm, fine tunings of the algorithm are not presented in detail in Algorithm 1.

### Calculating $|\mathcal{E}^{\psi'} - \mathcal{E}^{\psi^*}|$ for MGECP and $|\mathcal{E}_i^{\psi'} - \mathcal{E}_i^{\psi^*}|$ for MMUP

In each trial of the algorithm, the objective of the candidate solution ( $\mathcal{E}^{\psi'}$  for MGECP,  $\mathcal{E}_i^{\psi'}$  for MMUP) is to be re-calculated (line 10). Therefore its calculation is crucial to the time complexity of the algorithm as a whole. Because  $Q$  is of size  $(pn)^2$ , the calculation of  $\mathcal{E}^{\psi'}$  using (3.9) becomes a time consuming task with increases in either  $p$  or  $n$  or both. In our experience, as the complexity of the problem increases, it becomes less feasible to apply the heuristic than that of an QP solver if (3.9) is used to calculate  $\mathcal{E}^{\psi'}$ .

To overcome this issue, we observe the following:

**Theorem 3.4.1.** *If  $\psi'$  is the allocation scheme which alters only the assignment of  $a \in T$  from  $i$  to  $j$  (with  $i, j \in P$ ) when compared with  $\psi^*$ , then*

$$\mathcal{E}^{\psi'} - \mathcal{E}^{\psi^*} = Q_{(j-1)n.} \text{vec}(X^{\psi'}) + Q_{.(j-1)n} \text{vec}(X^{\psi'})^T - Q_{(i-1)n.} \text{vec}(X^{\psi^*}) - Q_{.(i-1)n} \text{vec}(X^{\psi^*})^T \quad (3.18)$$

where  $Q_{(j-1)n.}$  and  $Q_{(i-1)n.}$  denote the  $(j-1)n$ -th and the  $(i-1)n$ -th row of  $Q$ ,  $Q_{.(j-1)n}$  and  $Q_{.(i-1)n}$  denote the  $(j-1)n$ -th and the  $(i-1)n$ -th column of  $Q$  respectively.

*Proof.* Observe that an allocation matrix  $X$  has only binary values and that when one application changes allocation from  $\psi$  to  $\psi'$ , only a pair of values of that allocation matrix is exchanged. Apply these observations to (3.9), the reader should not find it difficult to come to (3.18).  $\square$

With (3.18), we can quickly assign the probability of accepting a candidate solution. We reduced the time complexity of calculating  $|\mathcal{E}^{\psi'} - \mathcal{E}^{\psi^*}|$  from a complexity that is greater than  $O((pn)^2)$  for multiple vector-matrix multiplications to  $O(pn)$  for the sum of vector dot products.

Note that the same technique is also applied in our second heuristic introduced in the next section.

**Algorithm 1** Simulated Annealing - MGECP and MMUP

---

```

1: procedure SANNEALING( $\psi_0, Q$ )
2:    $T \leftarrow T_0$ 
3:    $\psi^* \leftarrow \psi_0$ 
4:   for  $c \leftarrow 1, \text{NumberOfCycles}$  do
5:     for  $t \leftarrow 1, \text{NumberOfTrials}$  do
6:        $\psi' \leftarrow \text{local}(\psi^*)$ 
7:       if  $\text{Obj}^{\psi'} < \text{Obj}^{\psi^*}$  then
8:          $\psi^* \leftarrow \psi'$ 
9:       else
10:         $p = \exp(-|\text{Obj}^{\psi'} - \text{Obj}^{\psi^*}|/T)$ 
11:        if  $p > \text{rand}(0, 1)$  then
12:           $\psi^* \leftarrow \psi'$ 
13:        end if
14:      end if
15:    end for
16:     $T = T \times T_c$ 
17:  end for
18:  return  $\psi^*$ 
19: end procedure

```

---

**3.4.2 Greedy Autonomous Offload**

Heuristics that are used in the literature to approximate QAPs (e.g. simulated annealing) often share a common evolution-like structure which iteratively improves on a best-known result. The optimality of the final solution is often dependent on the number of iterations the algorithm is allowed to run. In an MCP environment, workflows need to be nimble and adaptable to the constantly changing network conditions of mobile devices. It is often not practical to let the algorithm running for a large number of iterations. Therefore in the design of our second heuristic, we take a step back from the established algorithms and aim to build an algorithm that is most practical to the MCP.

In the design of this heuristic, as shown in Algorithm 2, we emphasis on the core feature of an MCP which is computation offload (or migration) from mobile to cloud. On a workflow engine level, the adjustment to the initial allocation scheme is carried out in rounds (line 3 to 9 for MGECP and line 17 to 21 for MMUP) triggered either by changes in MCP or periodically. On a device level, we first associate each mobile device with the cloud which it has the best connection with (line 25). Then all tasks currently located on this device and are not fixed to this device is measured against each other in terms of the energy savings (or losses) that may



**Algorithm 2** Greedy Autonomous Offload

---

```

1: procedure GAO-MGECP( $\psi_0, Q$ )
   This procedure is executed by the workflow engine, triggered by the changes in network
   conditions or periodically.
2:    $\psi' \leftarrow \psi_0$ 
3:   repeat
4:      $\psi^* \leftarrow \psi'$ 
5:     for  $d \leftarrow 1, |P^M|$  do
6:        $\psi'_d \leftarrow \text{GAO-DEVICE}(d, \psi')$ 
7:     end for
8:      $\psi' \leftarrow \text{Reduce}(\psi'_d)$ 
9:   until MaxIterations or  $\psi' == \psi^*$ 
10:  return  $\psi'$ 
11: end procedure

12: procedure GAO-MMUP( $\psi_0, Q$ )
13:   $\psi' \leftarrow \psi_0$ 
14:  for  $d \leftarrow 1, |P^M|$  do
15:     $\text{maxheap.insert}(\mathcal{U}_d^{\psi'}, d)$ 
16:  end for
17:  repeat
18:     $[\mathcal{U}_{max}, d] \leftarrow \text{maxheap.extract}()$ 
19:     $\psi' \leftarrow \text{GAO-DEVICE}(d, \psi')$ 
20:     $\text{maxheap.insert}(\mathcal{U}_d^{\psi'}, d)$ 
21:  until MaxIterations or  $\mathcal{U}_d^{\psi'} == \mathcal{U}_{max}$ 
22:  return  $\psi'$ 
23: end procedure

24: procedure GAO-DEVICE( $d, \psi'$ )
   This procedure may either execute on the mobile devices or on the workflow engine.
25:   $c \leftarrow \text{BestConnectedCloud}$ 
26:   $\Delta\mathcal{E}^* \leftarrow 0$ 
27:   $\psi'_{Me.ID} \leftarrow \psi'$ 
28:  for all  $a \in \text{Me.Offloadables}$  do
29:     $\psi'' \leftarrow \psi'_{Me.ID}$ 
30:     $\Delta\mathcal{E} \leftarrow \mathcal{E}_{Me.ID}^{\psi''(a)=c} - \mathcal{E}_{Me.ID}^{\psi''}$ 
31:    if  $\Delta\mathcal{E} > \Delta\mathcal{E}^*$  then
32:       $\psi'_{Me.ID}(a) \leftarrow c$ 
33:    end if
34:  end for
35:   $\psi'(a) \leftarrow c$ 
36:  return  $\psi'_{Me.ID}$ 
37: end procedure

```

---

occur if it is offloaded to this designated cloud (line 28 to 34). This constitutes the device-level decision making process of our algorithm.

The first two procedures of the algorithm are entry points for solving the MGECP and MMUP respectively. For the MGECP, all devices from  $|P^M|$  are allowed to offload computation in each iteration until no device is able to reduce its energy cost any further. For the MMUP, a heap data structure is in place to hold the utilisation value of all devices. The device with the highest value of utilisation is picked from the top of this max-heap in each iteration to offload its computation. The procedure finishes when the device picked is unable to reduce its utilisation further.

This algorithm is *greedy* in that each device offloads the one most “profitable” task to the most “promising” location known to it. This means that the algorithm is quick and cheap (in terms of energy cost) to execute on the device. Although it does not apply exhaustive search methods for the optimal offload scheme, it produces good results which we demonstrate in the next chapter. As a possible extension to this algorithm we could model the per-device offload decision as an integer program as in [22] to obtain the optimal solution.

This algorithm is *autonomous* because it allows each device to make its own offload decisions independently. This is due to the fact that the device-level procedure (line 24) of the algorithm may execute locally on mobile devices. Note that although  $\psi'$  is requested by the procedure as input, this does not create any extra communication workload for the devices. The communication of  $\psi'$  between the devices and the workflow engine is requested to guide the execution of the workflow regardless of any offload requests. Once an offload decision has been made on the device, only the difference between  $\psi'$  and  $\psi'_{Me.ID}$  is to be returned at line 36. This autonomous behaviour also mimics the cooperation of mobile devices when each is equipped with one-to-one mobile cloud computing offload schemes as suggested by [22, 23, 24].

Another benefit of the greedy autonomous structure is that the workflow engine is able to react to the changing network conditions more efficiently. For instance, when a device is temporarily cut-off from the MCP network, the workflow engine may pause the procedure and wait for the device to come back online. Depending on the new connection speed that device has to the MCP when it recovers, the workflow engine can decide whether to restart the whole procedure or continue the existing procedure. Likewise, when a new cloud resource becomes available on the MCP, the device can adjust its favourite offload destination and revise its decisions.

### 3.4.3 Joint Search

The solution produced by the greedy autonomous offload algorithm is sub-optimal because each device only offload tasks to a destination which is considered the best by itself, whereas a joint offload action with one of its neighbouring devices to the cloud of their choice may produce a better result. This issue does not exist in the simulated annealing algorithm.

One the other hand, a disadvantage of the simulated annealing algorithm is that it does not apply exhaustive local search around its solution, therefore its solution is not guaranteed to be locally optimal. This local optimality gap can be reduced by applying the greedy autonomous offload algorithm on the solution provided by the simulated annealing algorithm.

Therefore, in our experiments, we use the combination of the two heuristics, namely GAO+SA and SA+GAO, to attempt to reduce the impact from each algorithm's disadvantages.

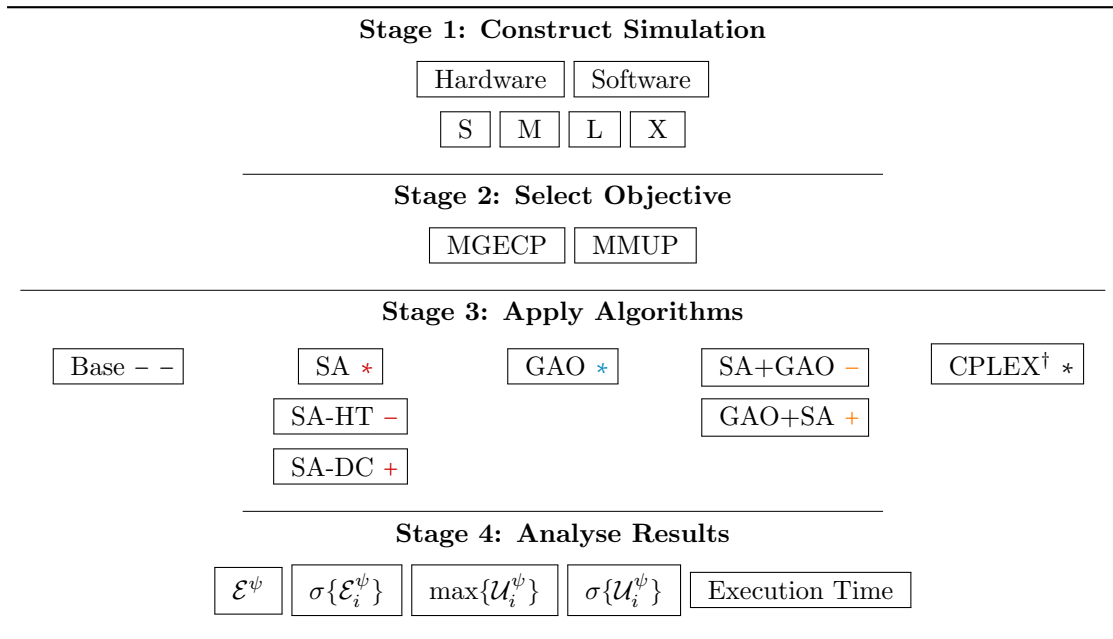
## 3.5 Simulations, Comparisons and Discussion

In this section, we carry out simulation studies to verify and compare the results produced by the proposed algorithms. We first layout the structure of our simulation in Section 3.5.1, and then give details of the hardware and software parameters used in the simulations in Section 3.5.2. Results produced from the simulations are analysed in the rest of this section.

### 3.5.1 Simulation Structure

There are four stages in our simulations as shown in Table 3.1. Each simulation test is constructed at stage 1 with stochastic hardware and software parameters. We give details of the ranges of these parameters in the next section (3.5.2). Since scalability is one of the main concerns which led to the development of our heuristics, we construct four series of simulations of different complexities to compare the performance of our heuristics under different circumstances. Each series includes five test groups details of which are given in Table 3.2. We refer to each test group by their ID (e.g. S0, M3) in the rest of this section. For brevity, we only discuss the results from the first test group of each series in this section, i.e. S0, M0, L0 and X0. Data from all other groups are attached in Appendix A and B for further reference.

Table 3.1: Simulation structure



† - QP and QCP solvers are applied for MGECP and MMUP respectively.

Once a simulation environment has been constructed, we then solve each of the two objectives at stage 2 with each of the algorithms listed at stage 3. Within stage 3, we first apply a baseline algorithm (Base) which attempts to reduce the total energy cost by distributing the number of tasks evenly across the MCP including both mobile and cloud nodes. This algorithm provides a good baseline value because although it does not seek the benefit of using an energy efficient device, its chance of being able to take that advantage is consistent. This baseline algorithm also utilises cloud resources to execute a fair portion of the workloads. The allocation scheme produced by Base is used as the initial allocation scheme for our heuristics.

Next, we apply both of our heuristics (SA and GAO) to solve the MGECP and MMUP of the simulation. To demonstrate the effect of the two key parameters, *NumberOfCycles* and *NumberOfTrials*, used in SA (Algorithm 1), we apply two variations of our standard SA algorithm: *SA-HT*, which apply only half the number of trials as compared to our standard SA; *SA-DC*, which apply twice the number of cycles as compared to our standard SA. In terms of solution quality, as compared to the standard SA algorithm, we expect SA-HT to produce a less optimal

Table 3.2: Simulation series specific parameters

S Series - Test Groups			M Series - Test Groups		
ID	$ P ( P^C )$	$ T / R $	ID	$ P ( P^C )$	$ T / R $
S0	10(2)	60/90	M0	20(2)	120/180
S1	10(2)	60/60	M1	20(2)	120/120
S2	10(2)	60/120	M2	20(2)	120/240
S3	10(4)	60/90	M3	20(4)	120/180
S4	10(2)	30/45	M4	20(2)	60/90
L Series - Test Groups			X Series - Test Groups		
ID	$ P ( P^C )$	$ T / R $	ID	$ P ( P^C )$	$ T / R $
L0	30(4)	180/270	X0	40(4)	240/360
L1	30(4)	180/180	X1	40(4)	240/240
L2	30(4)	180/360	X2	40(4)	240/480
L3	30(8)	180/270	X3	40(8)	240/360
L4	30(4)	90/135	X4	40(4)	120/180

solution with half the execution time and SA-DC to produce a better solution but twice the execution time.

Joint search methods (SA+GAO and GAO+SA) are then applied to the simulated MCP. Lastly, we apply QP and QCP solvers (CPLEX Component Libraries v12.6.1) for MGECP and MMUP respectively. Because of the complexity of the QCP constructed for MMUP, we limit the solver's execution time at 5 times the execution time taken by our standard SA algorithm so the solution produced by QCP is time constrained and is not necessarily optimal. No time limit is applied to the QP while solving the MGECP and so the solutions produced by QP for MGECP are all optimal. We also illustrate the symbols and colours which represent the results of each algorithm in the rest of this section in Table 3.1.

Finally at stage 4, from every solution produced at stage 3, we collect key energy performance indicators of the MCP including the mobile device group's total energy cost ( $\mathcal{E}^\psi$ ) and the maximum energy utilisation ( $\max\{\mathcal{U}_i^\psi\}$ ,  $i \in P^M$ ) since these directly reflects the qualities of the solutions in terms of MGECP and MMUP respectively. Additionally, we also record the standard deviation of the per-device energy cost ( $\sigma\{\mathcal{E}_i^\psi\}$ ,  $i \in P^M$ ) and the standard deviation of the per-device energy utilisation ( $\sigma\{\mathcal{U}_i^\psi\}$ ,  $i \in P^M$ ) to reflect the fairness of energy costs within

the MCP under different objectives. We also record the execution time of each algorithm to compare their efficiency.

### 3.5.2 MCP Construction

While it is intractable to cover all possible use cases of mobile cloud platforms, we aim to base our simulation closely to the characteristics of an average modern mobile device with wireless connectivities typically ranged within the capacities of existing wireless technologies (e.g. WiFi, 3G and LTE). On a hardware level, we construct our simulation with two building blocks: a *typical mobile device* and a *typical wireless connection*:

**Definition 2.** A *typical mobile device* has a battery capacity of 2000mAh, draws a current of 200-300mA during data transmission (with uplink drawing 20% more current than downlink) and 100-200mA when executing local computation tasks.

**Definition 3.** A *typical wireless connection* has an uplink bandwidth of 2-10Mbps, and a latency of 10-50ms.

The values in Definition 2 are based on the data presented in recent researches [51, 57, 59, 60, 66]. Characterisation of energy consumptions in smart devices and wireless networks is a challenging research topic. Because of the rapid development of new devices and emerging network standards, it is unrealistic to associate exact quantities to activities on mobile smart devices. Therefore, we use value ranges to characterise the energy consumptions of devices and networks in Definition 2 to simulate the variety of devices and power characteristics which may exist in a mobile cloud computing environment. We also used the tools presented in [63] to verify the values used in the definition. The network data in Definition 3 is based on the combination of 3G and LTE data presented in a recent report produced by Ofcom [73].

Likewise, on a software task interaction level, we construct our simulation with two basic unit workloads:

**Definition 4.** A task,  $t_a \in T$ , has a *unit computation workload* if its execution,  $c_a$ , takes 1 second to complete on a typical device.

**Definition 5.** Two tasks,  $\{t_a, t_b\} \subseteq T$ ,  $(t_a, t_b) \in R$ , have a *unit communication workload* if the size of the data sent from  $t_a$  to  $t_b$ ,  $d_{ab}$ , takes 1 second to complete on a typical wireless connection.

In our simulation, we specify each task's workload size using multiples (real numbers between 1 and 20) of a unit computation workload and a unit communication workload. The size of  $W$  of each simulation group is as specified in Table 3.2.

### 3.5.3 Results from Solving MGECP

We now compare the quality of the solutions produced by each algorithm in solving MGECP. We examine the solution quality in terms of both optimality and time. We plot the results from S0, M0, L0 and X0 in Fig. 3.2, Fig. 3.3 and Fig. 3.4. The optimality of each algorithm's solutions ( $\mathcal{E}^\psi$ ) are plotted in Fig. 3.2 and Fig. 3.3. The corresponding execution times of each algorithm are summarised in Fig. 3.4.

Each plot in Fig. 3.2 and Fig. 3.3 gives the solutions from 100 simulations produced by all algorithms listed in stage 3 of Table 3.1 for S0, M0, L0 and X0 respectively. Since no time limit is set for the CPLEX QP solver in solving MGECP, the allocation schemes produced by the solver are optimal. For illustration purposes, in (a1) and (b1) of Fig. 3.2 and Fig. 3.3, we normalise  $\mathcal{E}^\psi$  across the 100 simulations by their base value and sort them in ascending order of the optimal value given by the QP solver.

In MGECP, our objective is to minimise the energy cost  $\mathcal{E}^\psi$  of the MCP as a whole. From Fig. 3.2 and Fig. 3.3, we see that in all four (S0, M0, L0 and X0) groups, the proposed algorithms are all able to reduce  $\mathcal{E}^\psi$  to some extent. When the scale of the problem is small (S0 and M0), the differences between our heuristics and the solver is relatively small. This gap increases when the scale of simulation gets larger as seen in the results from L0 and X0. The quality of the solutions produced by the GAO algorithm are ranked second best consistently to that of the optimal solution produced by the QP solver in all four simulation groups. Although not as consistent, the joint search algorithm SA+GAO also produces good solutions that are better than GAO at times. The GAO algorithm is also superior than all other algorithms in terms of execution time as shown in Fig. 3.4.

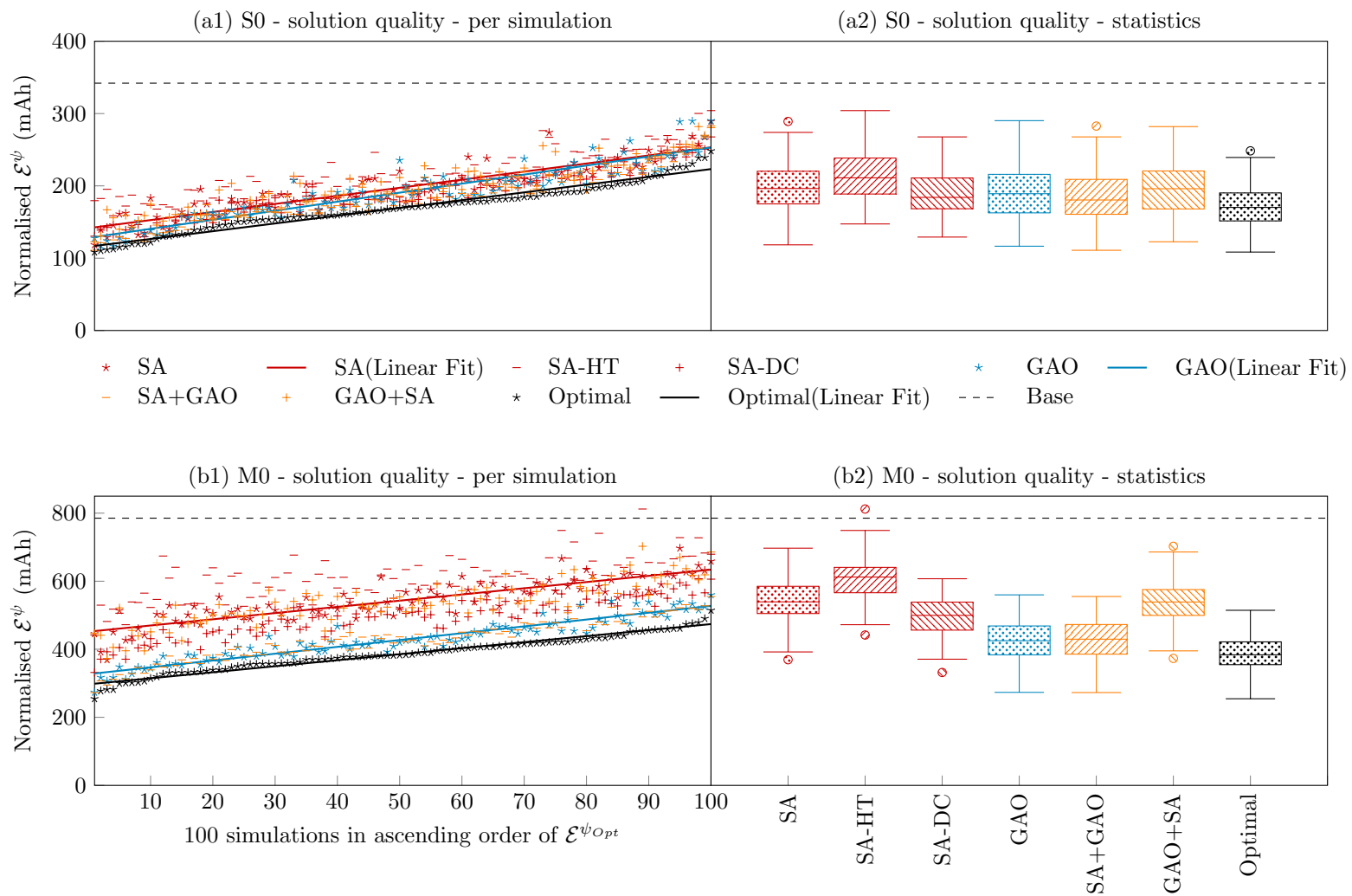


Figure 3.2: Results from solving MGECP: Solution optimality from S0 and M0.



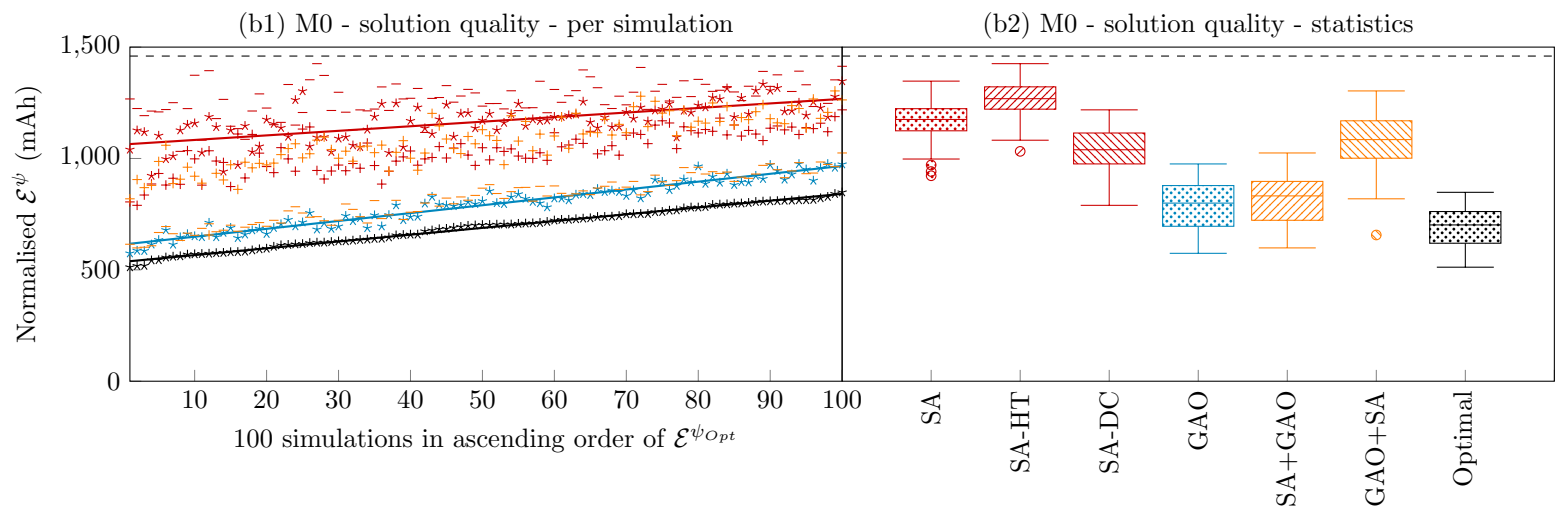
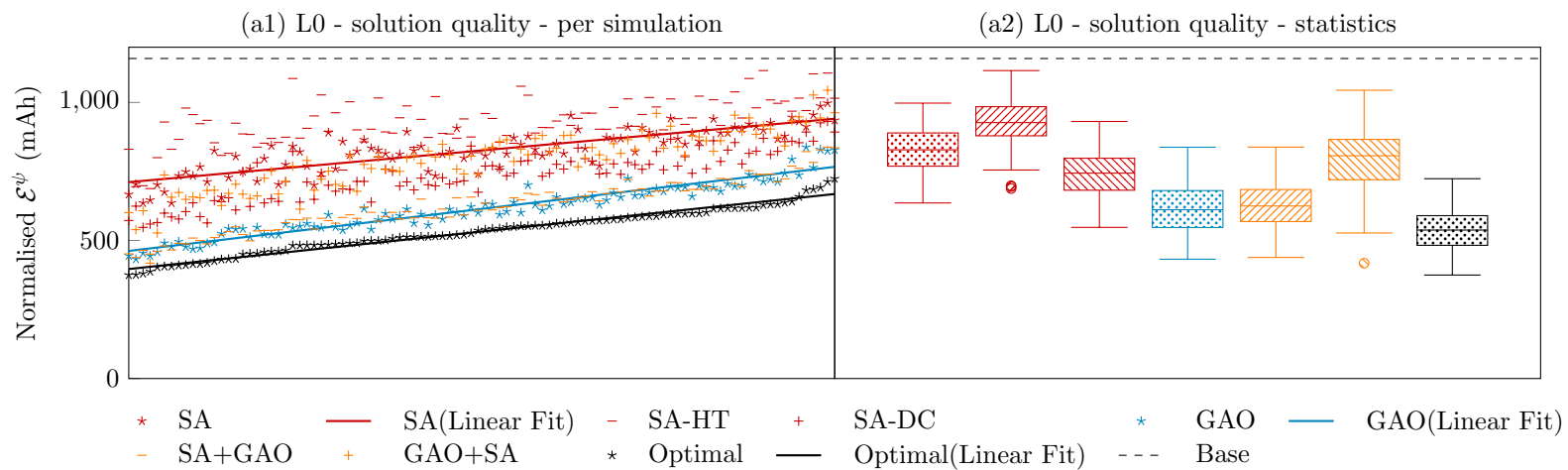


Figure 3.3: Results from solving MGECP: Solution optimality from L0 and X0.

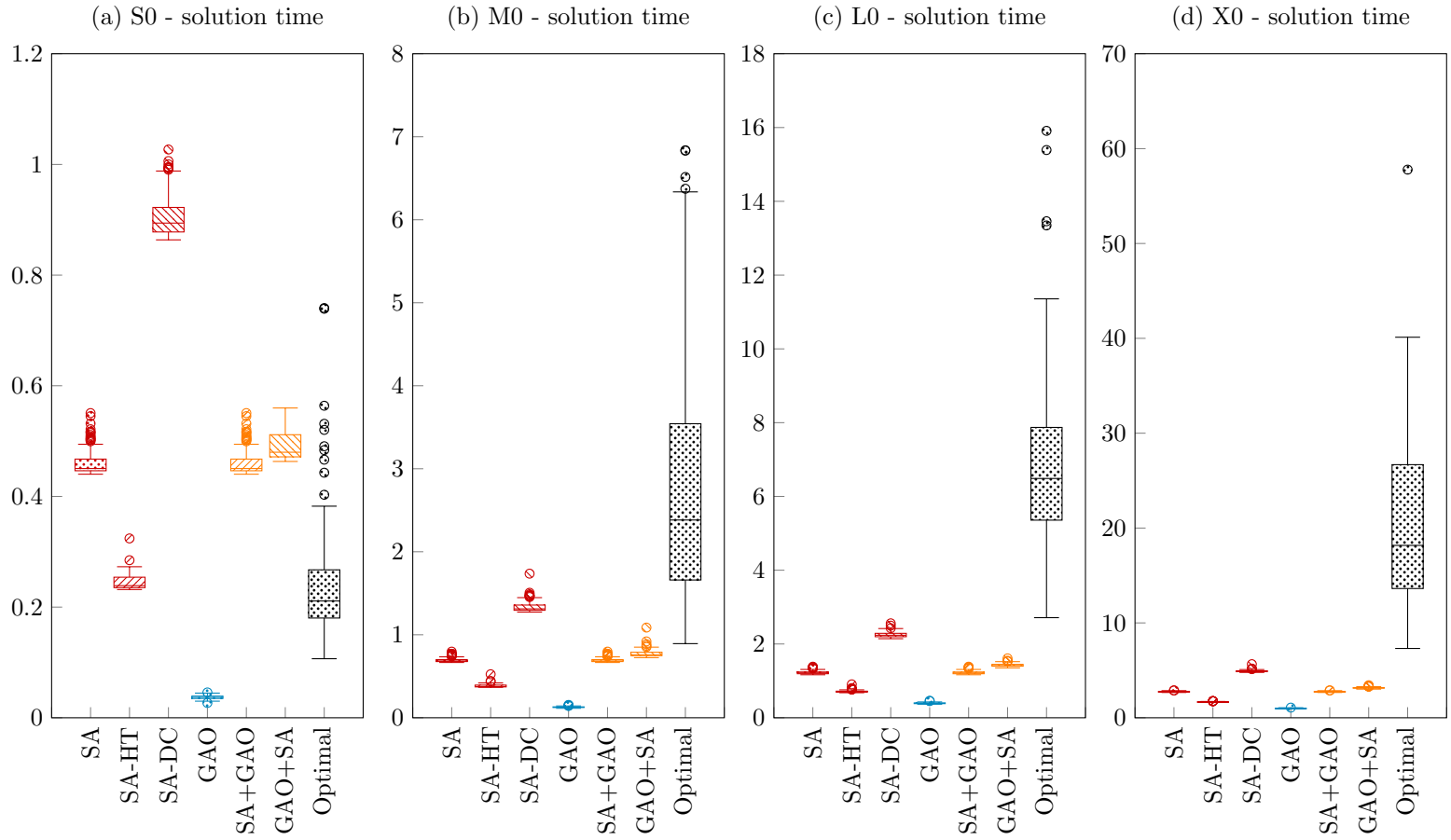


Figure 3.4: Results from solving MGECP: Solution time from S0, M0, L0 and X0.

As expected, of all three SA algorithms, SA-DC produces the best results with double the amount of execution time taken by the standard SA algorithm, and SA-HT produces the worst results within roughly half the amount of time. The difference in solution quality is less tangible in S0 and M0 than in L0 and X0. Compared to GAO, solutions produced by SA is further away from the optimal solution, and because of the stochastic nature of the algorithm, its solution quality is also not consistent.

Between the two joint search algorithms, SA+GAO produces much better results than GAO+SA. The fact that GAO+SA produces worse solutions than GAO is due to the random search methods applied by the SA at its initial stage which simply put the results already produced by GAO to waste. On the contrary, because GAO only applies allocation changes when it is beneficial, therefore in SA+GAO, the optimality of the result already produced by SA is preserved and improved upon.

Optimal solutions are guaranteed by the CPLEX QP solver, and the solver's time cost is also acceptable in small scale problems like those simulated in S0 as shown in (a) of Fig. 3.4. However, the scalability of the QP solver is very poor as shown by results from M0, L0 and X0 in (b-d) of Fig. 3.4. Additionally, it can be seen from (a-d) of Fig. 3.4 that the amount of time it takes the solver to produce the optimal result occupies a large range of values (e.g. from less than 3 seconds to more than 16 seconds in L0) and therefore is difficult to predict. In mobile computing scenarios, the allocation of tasks need to be decided quickly to react to the network conditions that are constantly changing and the poor scalability of the QP solver makes it less favourable for solving scheduling problems in MCPs.

To summarise our results for the objective of MGECP, we see that the GAO algorithm is the best when both scalability and solution optimality is considered. The QP solver is useful in solving small scale problems.

### 3.5.4 Results from Solving MMUP

With a similar structure to the previous section, we now compare the quality of the solutions produced by each algorithm for solving the MMUP in MCP. Results from S0, M0, L0 and X0 are plotted in Fig. 3.5, Fig. 3.6 and Fig. 3.7. Results of the adjusted allocation schemes are shown

in Fig. 3.5 and Fig. 3.6. The execution times of each algorithm is shown in Fig. 3.7.

In MMUP, our objective is to minimise the maximum utilisation of the mobile device group ( $\max\{\mathcal{U}_i^\psi, i \in P^M\}$ ). Our results show that not all of the algorithms applied are able to adjust the allocation schemes provided by the baseline algorithm to reduce the maximum utilisation value for all simulations. This is especially true for the time-limited QCP solver and our greedy algorithm GAO. However, GAO compensates this poor performance with very short execution times as shown in Fig. 3.7.

Allocation schemes given by the SA algorithms provide the best results overall. In S0 both variants of the standard SA share similar results to that of the standard SA. When the simulation scale increases in M0, L0 and X0, the differences in solution optimality become more noticeable. However, the relatively small reduction or improvement in solution optimality makes it difficult to definitively choose one of the SA algorithms over another. It is also worth noting that because of the random nature of SA, there is no guarantee that an increase in number of search trials and cycles would produce better allocation schemes. This can be observed from the scatter plots of Fig. 3.5 and Fig. 3.6. Given that the two variants cost significantly different amount of time to execute, a choice can be made according to the requirements of specific use cases.

Neither of the two joint search methods (SA+GAO and GAO+SA) provides significant improvement to the solution's optimality whilst costing more time than the standard SA and GAO algorithms.

### 3.5.5 Comparing MGECP and MMUP

We now discuss the similarities and contradictions of solving MGECP and MMUP. Recall that with each allocation scheme produced by our algorithms, as specified in stage 4 of Table 3.1, we collect not only the objective value for the chosen problem, but also for the other problem. That is to say, for instance, that when an allocation scheme ( $\psi$ ) is given by an algorithm with the objective to solve MGECP, we record not only the overall energy cost of the MCP under such an allocation scheme ( $\mathcal{E}^\psi$ ), but also the maximum utilisation value of the MCP ( $\max\{\mathcal{U}_i^\psi\}, i \in P^M$ ) which is the objective value for MMUP. We refer to the objective value which is given to the algorithm as the *prime objective value* and the objective value of the other problem not given

to the algorithm as the *by objective value*. By doing this we are able to gain further insight into the relations between solving MGECP and MMUP.

We plot and compare the prime and by objective values from S0, M0, L0 and X0 in Fig. 3.8. In the first row of Fig. 3.8, we compare the overall energy cost of the allocation schemes produced by solving MGECP and MMUP. In this row, results from the algorithms which have MGECP as the given objective (e.g. SA-MGECP, GAO-MGECP) are their prime objective values, whereas results from the algorithms which have MMUP as the given objective (e.g. SA-MMUP, GAO-MMUP) are their by objective values.

It is clear from all four simulation groups that the total energy cost of MCP is significantly lower when it is the prime objective of the algorithms. It is also worth noting that apart from S0, by solving MMUP in M0, L0 and X0, the total energy cost of the MCP as a by objective increases when compared with the baseline allocation scheme. This is because in MMUP, our objective is to reduce the energy cost of individual devices. In doing so, a task may be offloaded to a less energy-efficient device because it has greater amount of residual energy. Therefore, the energy cost of executing this task becomes greater rather than being reduced.

In the second row of Fig. 3.8, we give the maximum utilisation values produced by solving MGECP and MMUP. In this case, results from the algorithms which have MGECP as the given objective are their by objective values, where as results from the algorithms which have MMUP as the given objective are their prime objective values.

Here, results produced by SA are in line with what we observed from the first row of Fig. 3.8 meaning that it is preferable to set MMUP as the prime objective when we wish to minimise the maximum utilisation value of the MCP. In contrast, we observe the opposite with results from GAO and CPLEX (QCP solver). Although this gap is not obvious with GAO, we do not see a huge improvement from the by objective results to the prime objective results. This reflects the poor performance of GAO and CPLEX in solving MMUP as we have shown in the previous section (3.5.4).

Contrary to what we observed from the first row of Fig. 3.8, we do not observe an increase in the by objective values when compared with the baseline value. This is because in minimising the overall energy cost of the MCP, the energy cost of each device is also reduced and the reduction in utilisation follows.

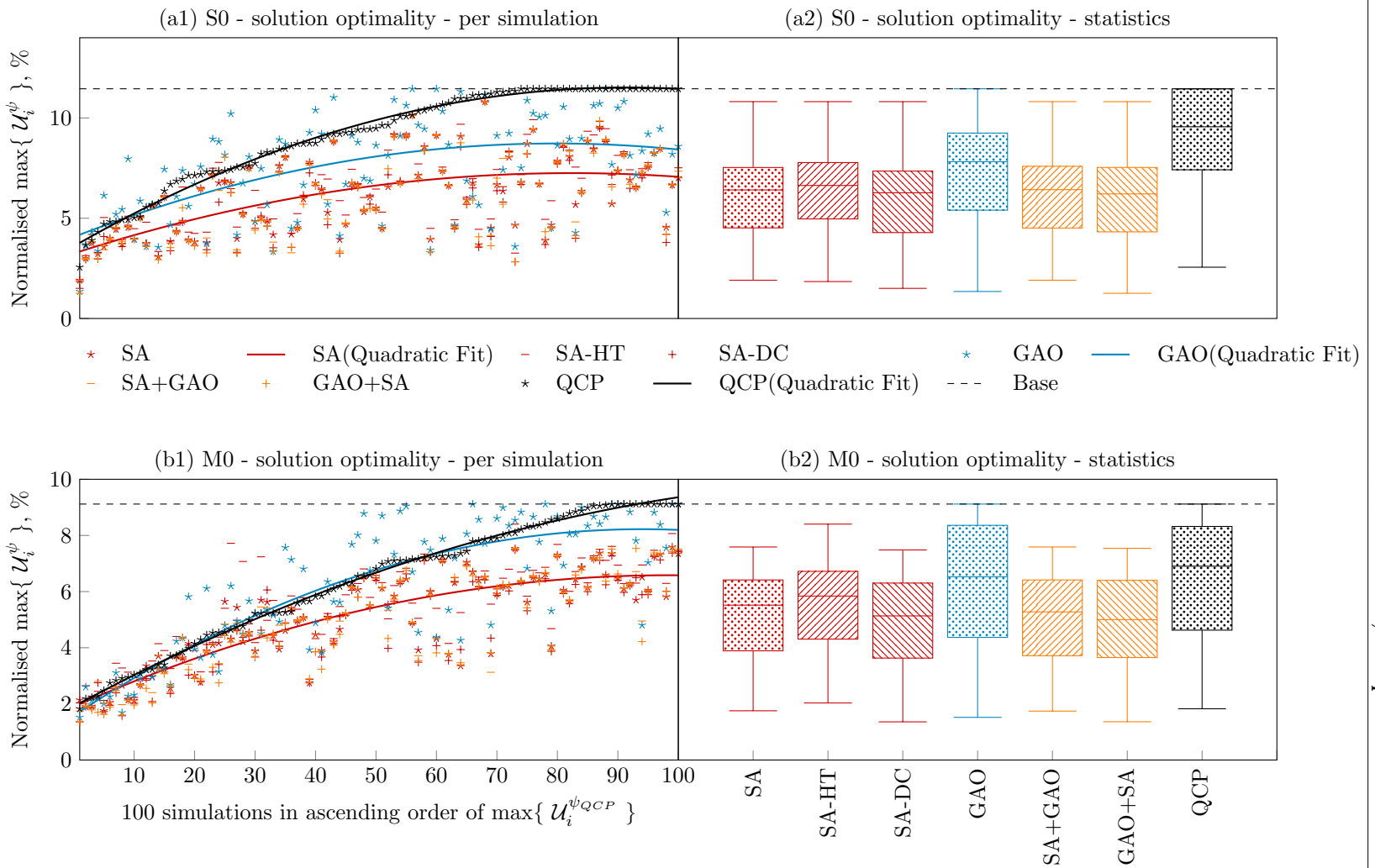


Figure 3.5: Results from solving MMUP: Solution optimality from S0 and M0.

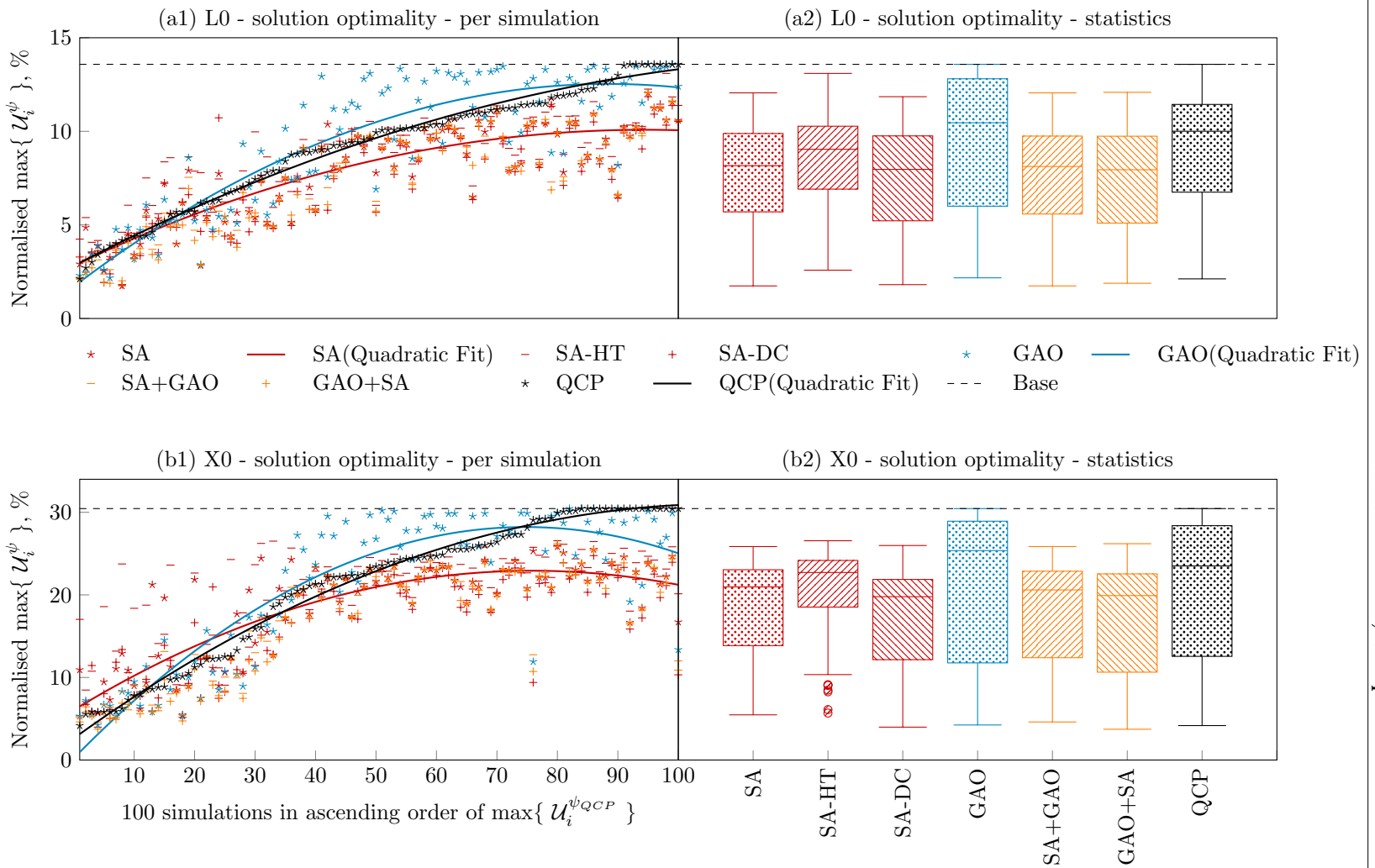


Figure 3.6: Results from solving MMUP: Solution optimality from L0 and X0.

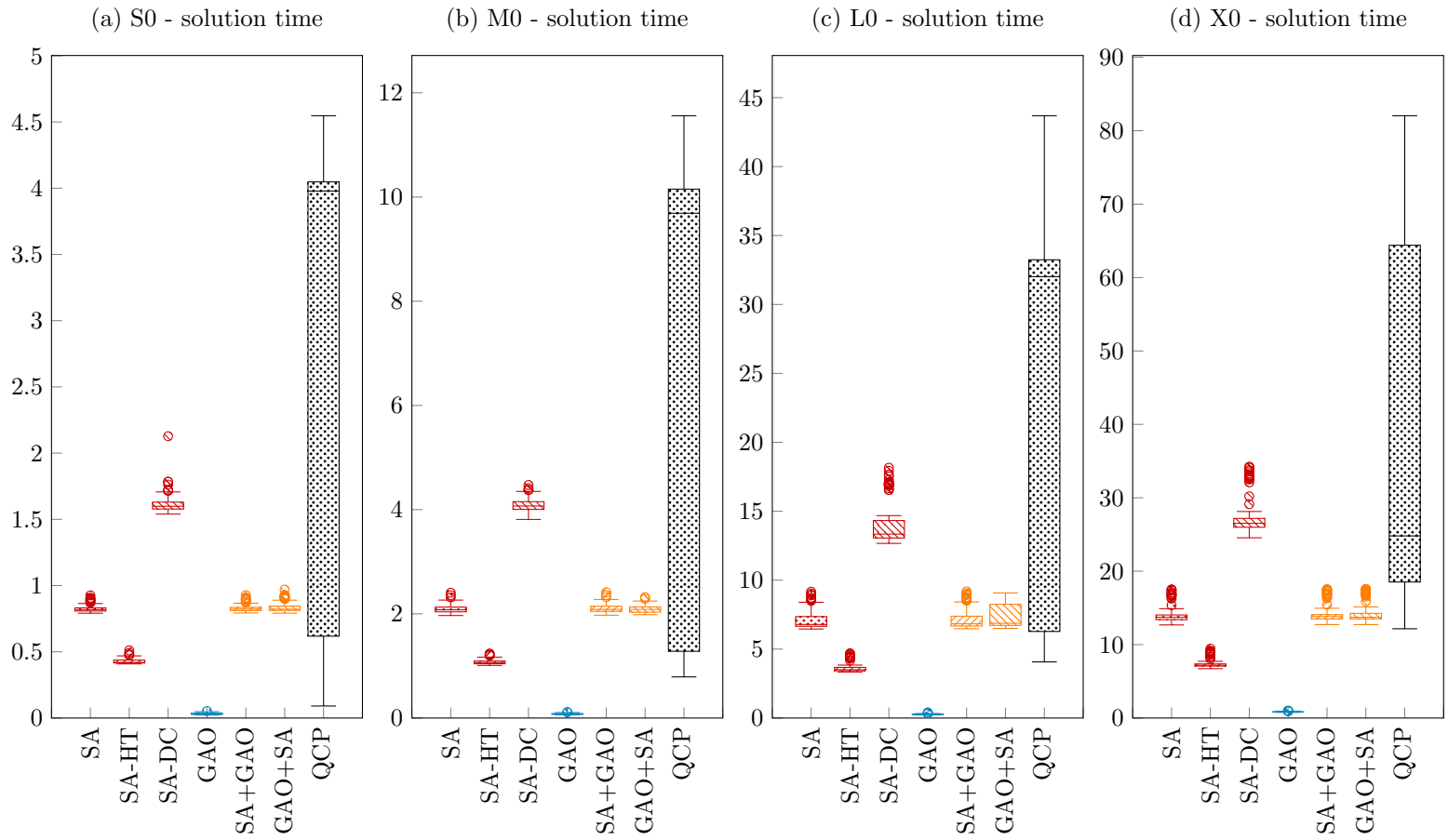


Figure 3.7: Results from solving MMUP: Solution time from S0, M0, L0 and X0.



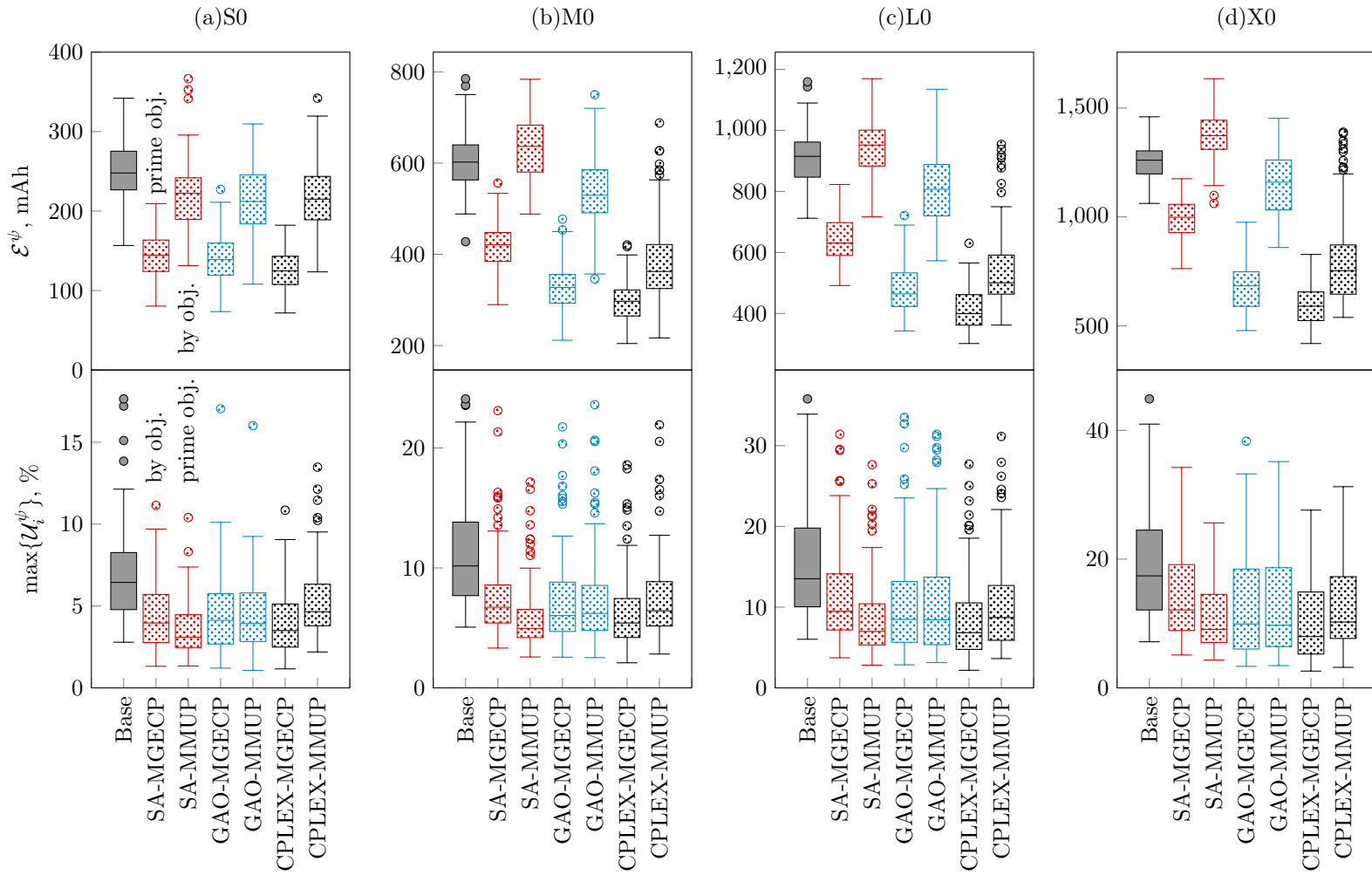


Figure 3.8: Prime and by objective values from S0, M0, L0 and X0.

From these results we conclude that allocation schemes generated when MGECP is the prime objective with MMUP as a by objective have good level of optimality in achieving both objectives in MGECP and MMUP. Solving MGECP not only reduces the energy cost of the MCP as a whole, but also in some degree reduces the maximum utilisation of the MCP and boost the lifetime of the MCP as a platform. When MMUP is a priority, the SA algorithm can be used to further reduce the maximum utilisation of the MCP.

### 3.6 Summary

In this chapter, we first introduced the common structure of a multi-device and multi-workflow mobile cloud platform and the task scheduling problems in such platforms. This distinguishes our work from existing researches which model only the task scheduling problem in single-device and single-workflow scenarios of mobile cloud computing.

Two aspects of the energy-aware requirements of a mobile cloud platform are investigated in this chapter, namely the Minimum Group Energy Cost Problem (MGECP) and the Minimum Maximum Utilisation Problem (MMUP). In MGECP, our objective is to minimise the overall energy cost of the platform, whereas in MMUP, our objective is to maximise the energy-life of the platform. Both problems are realistic and critical to improving the energy-efficiency of mobile cloud platforms.

In order to model both objectives of the energy-aware task scheduling problem of a mobile cloud platform, we first characterised the computation and communication costs of the platform by abstracting the key parameters of the platform. We then summarised the total energy cost of a mobile cloud platform with a quadratic binary program the solution of which provides an optimal allocation scheme of the MGECP. Next, a quadratically constrained variant of the quadratic program was developed to model the the MMUP.

Due to the computational complexity of solving the quadratic programs to optimal, we developed two heuristics to approximate the optimal solution for both scheduling problems in MGECP and MMUP. In the first heuristic, we implemented a simulated annealing algorithm (SA) which is often applied to combinatorial optimisation problems, especially for quadratic programs, in literature. In the second heuristic, we took a greedy (GAO) approach to the problems, which

allows each device to make task offload decisions autonomously. We tailored both algorithms to fit the requirements of our problems so that allocation schemes can be produced efficiently.

A comprehensive simulation study is carried out to verify and compare the performance and quality of our algorithms to that of the solvers provided by CPLEX, an industry-leading optimisation package. In these simulations, we also applied two variants of our standard simulated annealing algorithm to demonstrate and verify our choice of parameters in the standard simulated annealing algorithm. Furthermore, we implemented two joint search algorithms which combines SA and GAO in an attempt to overcome the disadvantages of each of the two heuristics.

From the simulation results, we evaluate each algorithm from two aspects: the optimality of the allocation schemes produce by the algorithm, and the amount of time it took the algorithm to execute. For the MGECP, we find that GAO is the best algorithm amongst all in terms of both solution optimality and execution time. The joint search method SA+GAO is able to improve the optimality of the solution while costing more time to execute. While guaranteeing the optimal scheduling schemes, the QP solver from CPLEX is only applicable for small scale problems for its high and unpredictable execution time. For the MMUP, we find that SA produces the best solution, while GAO's time cost remains small. The QCP solver from CPLEX is not able to produce good solutions within five times the execution time of the standard version of our SA algorithm.

Finally, we compare our results from MGECP and MMUP in terms of their contribution to each other's objective. Although both problems are critical in an energy-aware mobile cloud platform, we wish to see the effect of solving one of the problems on the other, so that a decision can be made when only one problem (objective) is to be selected. We find that solving MGECP also has a positive effect towards our goal in the MMUP. On the contrary, solving MMUP alone increases the overall energy cost of the MCP which is opposite to the goal of MGECP.

Based on our findings in this chapter, we see that a tailored greedy algorithm is able to efficiently produce good solutions for energy-aware task scheduling problems in mobile cloud computing scenarios.

The modelling technique we presented in this chapter is also applicable to other energy critical scenarios.

## Chapter 4

# Offloading Strategies for Time-Constrained Mobile Workflows

Whereas energy cost is the priority for mobile cloud platforms, time cost is of equal importance when an individual workflow is concerned. As well as energy, time is another important metric associated with mobile application workflows. In this chapter, we investigate further into the task allocation problem within a mobile cloud environment and look at developing offload strategies for mobile workflows while taking into account both time and energy requirements of the workflow. This distinguishes our work to existing researches which consider only one of these two aspects (in time efficiency [30, 34, 38, 74] and in energy saving [27, 28]). A similar analysis on the offload-abilities of tasks is included in [75], but not in any great detail, and also is only based on single smartphone nodes.

To broaden our vision of a ubiquitous mobile cloud environment, we further introduce the concept of cloudlets into our platform model which acts as an additional layer of execution platform in our mobile cloud platform. Cloudlets as proposed in [38] are not as powerful as standard cloud services. The advantage of cloudlets is their accessibility to mobile devices.

Located at the edges of the network, they have very close physical proximity to the mobile devices. This greatly reduces the communication energy cost to the mobile cloud platform and time cost to the mobile workflows.

Also different from the scenarios we looked at in Chapter 3 is that, in this chapter, we assume all tasks are originally located on mobile devices. When an offload action is scheduled, the executables associated with that task must be transmitted to the cloud or cloudlet before that task can be executed remotely. This creates extra communications between mobile and cloud. Consequently, this extra cost need to be considered when offloading decisions are made.

We develop an algorithm WGAO to produce the offload strategies in this chapter which is based on the GAO algorithm. A comprehensive analysis of the simulation results give further insight of the relation between different characteristics of a workflow and its offload-ability.

## 4.1 Computation Offload with Cloudlets

A new layer of network infrastructure referred to as *cloudlets* is the term used to capture the offload destination in this chapter. The concept of a cloudlet was first introduced in [38] at the end of the last decade, and was subsequently discussed in [18], [76] and [22]. In [38], a cloudlet is described as a “data centre in a box” and is “self-managing, requiring little than power, Internet connectivity, and access control for setup.”

In Fig. 4.1, we present an example in which an enterprise workflow is deployed over four mobile devices including a laptop and three smartphones. A cloudlet is deployed next to a WiFi hotspot in a coffee shop that is accessible to the user of the second smartphone. Another cloudlet is deployed at a more remote location which is accessible to both the third and the fourth smartphone. If we assume that all tasks of the workflow require the same amount of energy per second to run on their host devices and that the communications between each task are of the same size. We also assume that no other application draws energy from these devices whilst the workflow is being executed, then when the workflow is run repeatedly, we can predict that the first phone’s battery is to go flat first because its user is sitting in traffic and can only communicate with the other phones over a 3G connection. Its offload activity (if any) also has to go through a 3G connection (3G is generally more expensive than WiFi [77, 78, 79]).

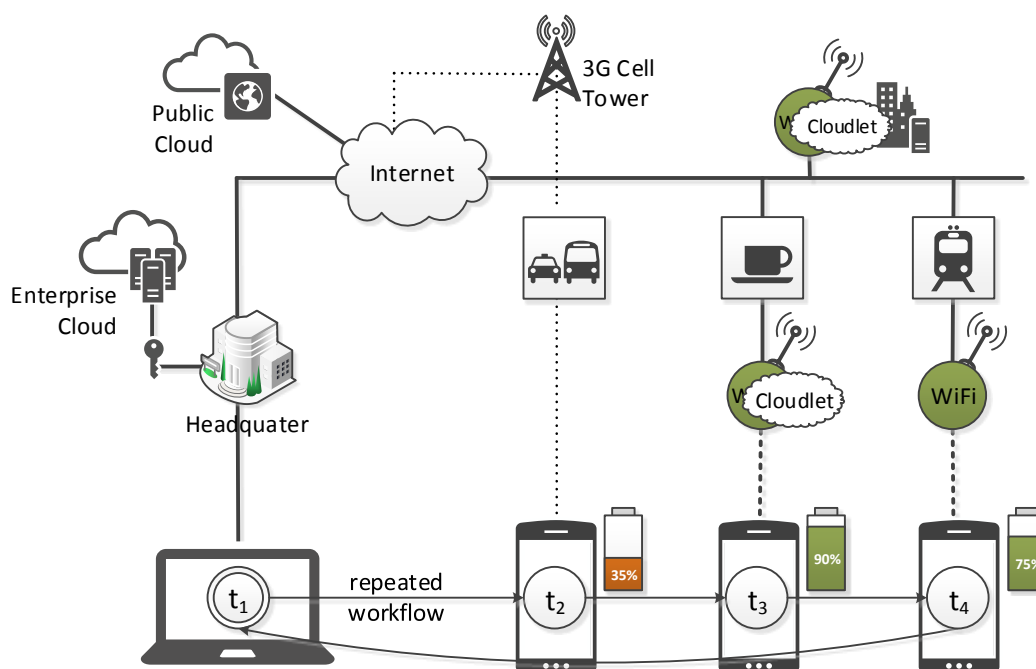


Figure 4.1: Example showing cloudlet and faster network connections improve battery life on mobile devices.

The second smartphone communicates with others over the coffee shop's WiFi and is able to use its cloudlet to offload  $t_3$ 's computation cost. Therefore its battery gets consumed the slowest. The user of the last handset has access to a WiFi hotspot whilst travelling on the train. However the train does not have a cloudlet deployed, so to offload  $t_4$ 's computation it has to send the executables to the more distant cloudlet which takes longer to reach and thus consumes more energy.

In Fig. 4.1, an enterprise cloud at the firm's headquarters and a distant cloud service on the Internet are also available to support offload. These nodes may have faster processing speeds than the cloudlets. Offload to these clouds could prove more beneficial if the network connection is of high speed.

## 4.2 Offload Strategies for Mobile Workflows

In this section, we first set the scene by abstracting the mobile workflow and its execution platform into two graphs. We demonstrate the impact of an offload action to various interest groups of a workflow with a simple example. Trade-offs in time and energy of an offload action vary depending on the characteristics of the workflow and the hardware network that carries it. We thus build these variables into our model and construct our objective functions. We then present the algorithm and discuss the design philosophies behind these.

### 4.2.1 Preliminaries and Problem Definition

Two graphs are used in our definition, each annotates the workflow and the hardware network respectively. Firstly, we annotate our mobile workflow as a directed acyclic graph  $W = (T, E)$  whose vertices are the set of tasks of the workflow and whose edges are the communications between these tasks. Each task requires a number of instructions to be processed in order to complete its computation, which is given by function  $I$ . For example  $I(t_i)$  gives the number of instructions  $t_i$  requires. Since to run the offloaded task on the cloudlet, the executable of the task needs to be transmitted to the cloudlet, we have function  $U(t_i)$  to represent the size of  $t_i$ 's executable. The size of the data carried within each communication call is given by function  $D$ . Hence we have  $D(t_i, t_j)$  to represent the size of the message sent from  $t_i$  to  $t_j$ .

Our second graph  $H = (N, R)$  represents the hardware platform on which our workflow is to be executed. Graph  $H$ 's vertices are the processing nodes, and its edges represents the data links between these nodes. A processing node  $n \in N$  must be either a local smartphone ( $n_s \in N_s$ ) or a cloudlet server ( $n_c \in N_c$ ) but not both, and hence we have  $N = N_s \cup N_c$  and  $N_s \cap N_c = \emptyset$ . Effectively, this divides the hardware graph  $H$  into two processing spaces: the smartphone space  $H_s$  and the cloudlet space  $H_c$ . Edges within the  $H_s$  space interconnect the smartphones together, which in practice is most likely to be carried over via a 3G or 4G data network unless both phones have established WiFi links. Cloudlet nodes within the  $H_c$  space are interconnected via Wide Area Networks (WANs). A data link between the two spaces (i.e. a data link from a smartphone to a cloudlet) is dependent on the smartphone's location and can be either a 3G/4G or WiFi connection in practice. The bandwidth of each data link varies depending on its carrier, and in

our model we annotate function  $B$  to obtain the bandwidth property of an edge. For instance we have  $B(n_a, n_b)$  which gives the bandwidth between node  $n_a$  and  $n_b$ . We also annotate function  $S$  to give the processing speed of each node, for instance,  $S(n_a)$  represents the processing speed of node  $n_a$ .

The mobile workflow graph  $W$  is mapped onto the hardware graph  $H$  by two mapping functions:  $\alpha: T \mapsto N$  and  $\beta: E \mapsto R$  to represent the execution plan of the workflow:

$$\alpha(t_i) = n_a \Leftrightarrow \text{task } t_i \text{ is executed on node } n_a$$

$$\text{edge } e \text{ joins } t_i \text{ to } t_j \Leftrightarrow \beta(e) \text{ joins } \alpha(t_i) \text{ to } \alpha(t_j)$$

Before any offload action takes place, our workflow is executed on the smartphone space only:

$$(\forall t)(t \in T \rightarrow \alpha(t) \in N_s)$$

Fig. 4.2 shows an example of a workflow consisting of three tasks, and the workflow is originally mapped to the smartphone nodes only:

$$\alpha: T \rightarrow N, \alpha(t_1) = n_{s1}, \alpha(t_2) = n_{s2}, \alpha(t_3) = n_{s3}$$

$$\beta: E \rightarrow R, \beta(e_1) = r(n_{s1}, n_{s2}), \beta(e_2) = r(n_{s2}, n_{s3})$$

In order to reduce the energy cost of the smartphone space and also to take advantage of the fast processing speed provided by the cloudlet space, our general agenda is to shift the workflow's tasks over to the cloudlet space as much as possible. In our example in Fig. 4.2, task  $t_2$  is offloaded from its local smartphone node  $n_{s2}$  to cloudlet node  $n_{c1}$ , and this changes the mapping functions from  $W$  to  $H$  as:

$$\alpha': T \rightarrow N, \alpha'(t_1) = n_{s1}, \alpha'(t_2) = n_{c1}, \alpha'(t_3) = n_{s3}$$

$$\beta': E \rightarrow R, \beta'(e_1) = r(n_{s1}, n_{c1}), \beta'(e_2) = r(n_{c1}, n_{s3})$$



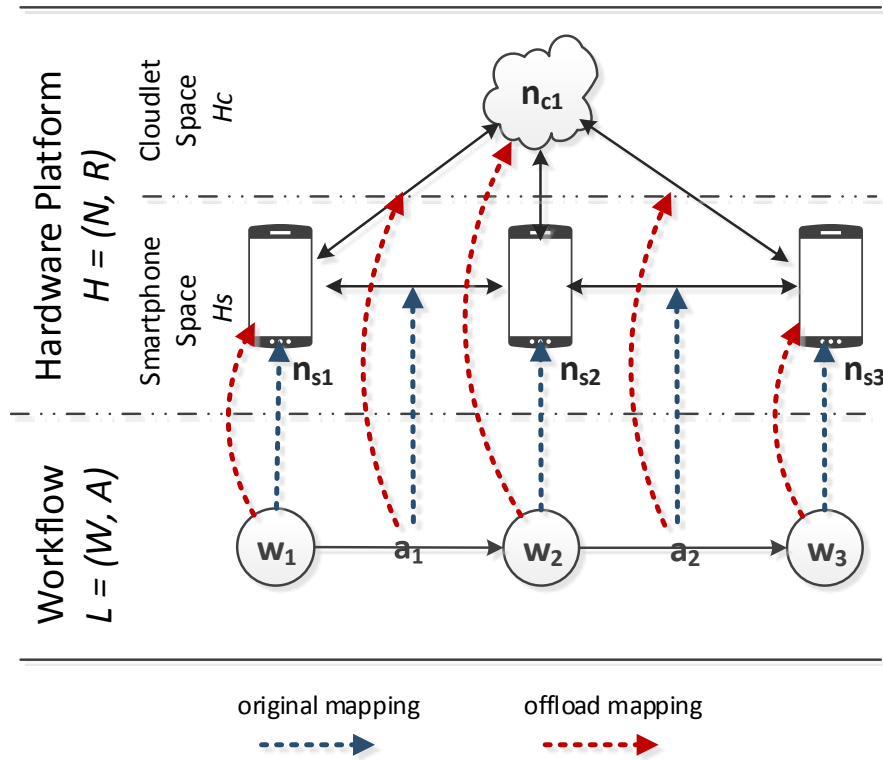


Figure 4.2: Offload expands the mapping into the Cloudlet space.

This change effectively expands graph  $W$ 's destination graph from  $H$ 's sub-graph  $H_s$  to the rest of  $H$  and with this expansion comes a series of trade-offs to various interest groups of the workflow:

- (a) To the user of smartphone node  $n_{s2}$ , because task  $t_2$ 's computation is no longer executed locally, this reduces the energy cost of his handset. Moreover, because the workflow is redirected away from his handset, he also avoids sending and receiving messages to the other handsets which also reduces the energy cost to his handset. The only extra cost incurred from the offload action is that the executables of task  $t_2$  needs to be transmitted to the cloudlet node  $n_{c1}$ , which costs energy in this example.

Notice that in a real mobile application, as identified in several papers [22, 24, 37, 80], and as we discussed earlier in Section 3.1.3, not all components are suitable for offload. In the most common cases, components which require I/O access must be executed locally on the

handset, the same also applies to user interface modules. Thus it is unlikely that a handset can offload all of its duties from the workflow. In such cases those components which are pinned on the handset require active connections to be kept between the handset and its neighbours or the cloudlet depending on its relation with other tasks in the workflow. Consequently offload becomes a less attractive option to the user.

- (b) To the users of  $n_{s1}$  and  $n_{s3}$ , this offload has a negative impact if the distance from it to cloudlet  $n_{c1}$  is greater than that to  $n_{s2}$ . For instance, consider an enterprise workflow and a time in which both  $n_{s1}$  and  $n_{s2}$  resides in the same building and are connected through the building's local area network (LAN). Cloudlet  $n_{c1}$  however sits externally to this LAN. In such a situation, at least one more network hop is required to complete the communication between  $t_1$  and  $t_2$ , which means that  $n_{s1}$  must remain active for a longer period of time (with a higher energy cost) in order to confirm a safe exit from the workflow. On the other hand, in a case where  $n_{s1}$  is connected to  $n_{s2}$  over a long distance network, it is possible that communication from  $n_{s1}$  to  $n_{c1}$  is shorter than that to  $n_{s2}$ , thus the offload is beneficial to the user of  $n_{s1}$ .
- (c) Execution of a typical IT workflow is often constrained by time. While users of individual handset might prioritise energy saving on their phone, the overall time-efficiency of the workflow also needs to be ensured.

From this simple example, we see that managing the trade-offs between time and energy in various aspects of the workflow is the key element to our algorithm's decision making process. Hence we first capture the time and energy cost both before and after the offload action, and then with these functions we set our objectives to ensure the offload option has at least a positive effect.

### Time Constraint

Consider a task  $t_i$  which is local to smartphone node  $n_{t_i}^l$ , we want to see if offloading it to cloudlet node  $n_c$  is a beneficial option. We have the time cost before ( $M^l$ ) and after ( $M^r$ ) the offload as:

$$\begin{aligned}
 M^l(t_i) &= \frac{I(t_i)}{S(n_{t_i}^l)} + \sum_{(t_j, t_i) \in E} \frac{D(t_j, t_i)}{B(n_{t_j}, n_{t_i}^l)} + \sum_{(t_i, t_j) \in E} \frac{D(t_i, t_j)}{B(n_{t_i}^l, n_{t_j})} \\
 M^r(t_i, n_c) &= \frac{I(t_i)}{S(n_c)} + \sum_{(t_j, t_i) \in E} \frac{D(t_j, t_i)}{B(n_{t_j}, n_c)} + \sum_{(t_i, t_j) \in E} \frac{D(t_i, t_j)}{B(n_c, n_{t_j})} + \frac{U(t_i)}{B(n_{t_i}^l, n_c)}
 \end{aligned}$$

The first term in both functions gives the amount of time task  $t_i$  takes to execute on the smartphone and the target cloudlet respectively. The second and third terms are the inbound and outbound communication time costs. Note that  $n_{t_j}$  is the node which task  $t_j$  is currently assigned to. It can be either task  $t_j$ 's local smartphone node or a cloudlet node which task  $t_j$  is already offloaded to. The last term in the second function is the amount of time it takes to transmit task  $t_j$ 's executables to  $n_c$ .

Our objective is to ensure that the offload action does not delay the workflow's progress. We denote the slack time of task  $t_i$  with  $M_{t_i}^{slack}$  (the slack time is calculated according to the workflow's critical path) and have our time constraint as:

$$M^r(t_i, n_c) < M^l(t_i) + M_{t_i}^{slack} \quad (4.1)$$

### Energy Constraint

Suppose the current draw on a smartphone node  $n_s$ , in mAh, is  $P_c(n_s)$  for computing,  $P_i(n_s)$  when it is idle,  $P_{ts}(n_s)$  for sending data and  $P_{tc}(n_s)$  for receiving data. We have the energy cost on the smartphone before ( $G^l$ ) and after ( $G^r$ ) offloading as:

$$\begin{aligned}
 G^l(t_i) &= \frac{I(t_i)}{S(n_{t_i}^l)} \times P_c(n_{t_i}^l) \\
 &+ \sum_{(t_j, t_i) \in E} \frac{D(t_j, t_i)}{B(n_{t_j}, n_{t_i}^l)} \times P_{tc}(n_{t_i}^l) + \sum_{(t_i, t_j) \in E} \frac{D(t_i, t_j)}{B(n_{t_i}^l, n_{t_j})} \times P_{ts}(n_{t_i}^l) \\
 G^r(t_i, n_c) &= \frac{I(t_i)}{S(n_c)} \times P_i(n_{t_i}^l) \\
 &+ \sum_{(t_j, t_i) \in E \wedge n_{t_j} = n_{t_i}^l} \frac{D(t_j, t_i)}{B(n_{t_j}, n_c)} \times P_{tc}(n_{t_i}^l) + \sum_{(t_i, t_j) \in E \wedge n_{t_j} = n_{t_i}^l} \frac{D(t_i, t_j)}{B(n_c, n_{t_j})} \times P_{ts}(n_{t_i}^l) \\
 &+ \frac{U(t_i)}{B(n_{t_i}^l, n_c)} \times P_{ts}(n_{t_i}^l)
 \end{aligned}$$

The first term in both functions give the amount of energy the smartphone spends whilst the task is being executed. The next two terms are the amount of energy spent receiving and sending data to the neighbouring nodes respectively. Note that if the other end of the communication is on a different node ( $n_{t_j} \neq n_{t_i}^l$ ), no energy is spent at  $t_i$ 's local node for sending or receive messages.

In order to guarantee that the offload action does not cause the smartphone to consume more energy than its original setting, we set our energy constraint to:

$$G^r(t_i, n_c) < G^l(t_i) \quad (4.2)$$

For use in our algorithm, we also denote:

$EP_M(t_i, n_c)$  = it satisfies the time constraint (4.2) to offload  $t_i$  to  $n_c$

$EP_G(t_i, n_c)$  = it satisfies the energy constraint (4.1) to offload  $t_i$  to  $n_c$

### 4.2.2 Workflow-Oriented Greedy Autonomous Offload Algorithm

We now present a Workflow-Oriented Greedy Autonomous Offload (WGAO) Algorithm to produce offload strategies for mobile workflows, as shown in Algorithm 3. In this algorithm, we apply a similar structure to that of the Greedy Autonomous Offload (GAO) algorithm which we developed in Section 3.4.2 of Chapter 3.

We partition WGAO into two stages so that it can be implemented on the mobile nodes and the workflow's monitoring server respectively. The first two procedures (line 1 to 20) of our algorithm are executed by the workflow engine. WGAO-MAIN (line 1) traverses the list of tasks and communicates with each task's host to see if any offload action is possible. If the host's feedback is positive, then the workflow engine tries to construct an offload tree cluster with that task being the root using WGAO-TREE (line 11).

The third procedure (line 21) is implemented on the smartphones and helps its host to find the best possible offload point for its tasks according to the environmental parameters it gathers in real-time. For each of its tasks, out of all cloudlets that satisfy both time and energy constraints (if any), it selects the one which gives the largest amount of energy savings as its

offload destination. A user has the ability to set a task’s property to “isfixed” in order to protect the relevant content from being offloaded. At line 25  $N_c(n_{t_i}^l)$  represents the set of cloudlets that are visible to task  $t_i$ ’s local mobile node at that time.

The following document some of the algorithm’s desired properties that we identified in designing the algorithm:

**Autonomous Decision Making Ability** Each participating smartphone node should have the ability to make simple offload decisions based on the environment it is currently situated in without prior knowledge or instruction from the server. A mobile wireless data connection, especially when implemented over a cellular network, is prone to connectivity disruption. In such cases, the isolated node should be able to carry on executing its own tasks in an energy-efficient manner. WGAO-DEVICE is designed to take on such duty.

**Offload Authorisation** Not all resources on a mobile device are dedicated to a specific workflow. Although an offload action might be beneficial to the overall performance of the workflow, the owner of the device should still be able to have the authority to stop a task and its relevant data to be offloaded. Examples of which include sensitive or private information that the user is not prepared to share; extra financial expenditure for using a faster wireless connection in range, etc. Hence the isfixed property as used in WGAO-TREE and WGAO-DEVICE.

This is especially true in choosing the type of wireless connections for the smartphone nodes. In practice, although 3G and WiFi modules can be enabled at the same time on a smartphone, it is normally up to the local OS to decide which connection is to be used for data transfer tasks. A remote workflow decision engine’s role is to give advice to the user rather than altering the existing settings on the device.

Furthermore, as discussed in Section 3.1.3, some tasks are not suitable to be offloaded. This includes user interface processes, I/O components and processes that are observed by external processes that require the output to be produced on the local node only [22, 24, 37, 80].

**Task Clustering** Offloading two tasks to the same cloudlet greatly reduces the energy consumption in completing communication tasks between the two. Especially when those tasks

**Algorithm 3** Workflow-Oriented Greedy Autonomous Offload

---

```

1: procedure WGAO-MAIN( $W$ )
   This procedure is executed by the workflow engine, triggered by the changes in network
   conditions or periodically.
2:   sort workloads in set  $W$  in topological order
3:   for  $t_i \in T$  do
4:     if  $t_i.n_{off} == null$  then
5:       if WGAO-DEVICE( $t_i$ )  $\neq null$  then
6:         WGAO-TREE( $t_i$ )
7:       end if
8:     end if
9:   end for
10: end procedure

11: procedure WGAO-TREE( $t_i$ )
12:   for  $t_j : (t_j \in T \wedge (t_i, t_j) \in E)$  do
13:     if  $t_j.n_{off} == null \wedge \neg t_j.isfixed$  then
14:       if  $EP_M(t_i, n_c) \wedge EP_M(t_i, n_c)$  then
15:          $t_j.n_{off} \leftarrow t_i.n_{off}$ 
16:         WGAO-TREE( $t_j$ )
17:       end if
18:     end if
19:   end for
20: end procedure

21: procedure WGAO-DEVICE( $t_i$ )
   This procedure may either execute on the mobile devices or on the workflow engine.
22:    $n_c \leftarrow null$ 
23:    $g_{max} \leftarrow 0$  // maximum energy saving
24:   if  $t_i.isfixed$  then return  $n_c$  end if
25:   for all  $n_j \in N_c(n_{t_i}^l)$  do
26:     if  $EP_M(t_i, n_c) \wedge EP_M(t_i, n_c) \wedge G^l(t_i) - G^r(t_i, n_c) > g_{max}$  then
27:        $n_c \leftarrow n_j$ 
28:        $g_{max} \leftarrow G^l(t_i) - G^r(t_i, n_c)$ 
29:     end if
30:   end for
31:   if  $n_c \neq null$  then  $t_i.n_{off} \leftarrow n_j$  end if
32:   return  $n_c$ 
33: end procedure

```

---

belong to different smartphone nodes, clustering essentially eliminates the need to transfer data over a wireless connection between the mobile nodes. In WGAO-TREE, once a task has been approved to offload to a cloudlet, we then attempt to exploit the same offload route and offload the same task’s leaf tasks to the same cloudlet. Recursive calls to WGAO-TREE expand the offload cluster.

**Update on Event Mechanism** The outcome of the decision making process depends heavily on the mobile node’s real-time environmental parameters. Thus accuracy of this information directly affects the offload’s efficiency. However, it is expensive in both time and energy to constantly update the information onto the server [81], especially when no changes have occurred between updates. One solution to this problem is to use the wake-on-event mechanism provided by the mobile’s operating systems [51], especially on events like entering a WiFi zone or moving into the range of a Cloudlet as demonstrated in [82].

Our algorithm is designed so that WGAO-DEVICE is triggered on the handset when significant change has occurred in its network connectivity. Updated information including a new local offload plan is then feedback to the workflow engine.

### 4.2.3 Discussion of Variations and Optimisation of WGAO

In WGAO both constraints for time and energy, as specified by (4.1) and (4.2), are to be satisfied in order for an offload decision to be approved. However, in some cases the workflow would have preference in gaining saving in one metric over the other. For instance, in a business environment, users of the workflow are highly mobile and the handheld device’s up time is critical for the users to be able to answer voice calls at all time. A non-time-critical workflow within such an environment has strong preference in saving battery life over execution time. Thus sacrifices in task execution time can be made in order to help reduce the energy consumption on handsets.

Derived from this philosophy to trade-off gains and loses between time and energy, we describe two variations of WGAO:

**Minimum Battery Cost** Our first variation prioritises energy saving over time costs. An acceptable time delay  $M^{allowed\ delay}$  is added into the time constraint statement. We have

the new time constraint as:

$$M^r(t_i, n_c) < M^l(t_i) + M_{t_i}^{slack} + M_{t_i}^{allowed\ delay} \quad (4.3)$$

This acceptable delay can be either a static value or a dynamic value that is dependent the device's current status (e.g. the current battery level, additional energy saving generated and etc.).

**Shortest Schedule Length** In some cases, when the ability to re-charge the battery of the smartphone is assured, it is often preferable to take advantage of this opportunity to accelerate the execution of the workflow. In contrast to the first variation, we commit extra energy consumption in exchange for faster execution speed in the second variation. We introduce  $G^{extra}$  to the energy constraint and have the modified energy constraint:

$$G^r(t_i, n_c) < G^l(t_i) + G^{extra} \quad (4.4)$$

In the extreme case where the mobile device is docked to a charging station, we can remove energy constraint  $EP_G$  from WGAO-DEVICE and WGAO-TREE completely, so that the offload decisions are free from energy constraints.

**Optimal Condition Expression** Improvements in hardware resources can increase the workflow's offload-ability. However, there is a limit to the hardware's performance. For instance, an individual user's available bandwidth to a WiFi hotspot is often capped. So to send a message of a certain size over this connection takes at least  $\frac{D(t_i, t_j)}{BandwidthCap}$  seconds.

In order to reduce the complexity of our algorithm in real-time, we can use an optimal conditional expression to pre-test a task to see if the time and energy constraints can be satisfied provided that the device is in the best available hardware environments. For instance we can



take a bandwidth cap value of 1Mbps into the time constraint and have:

$$M^{r-opt}(t_i, n_c) = \frac{I(t_i)}{S(n_c)} + \sum_{(t_j, t_i) \in E} \frac{D(t_j, t_i)}{1Mbps} + \sum_{(t_i, t_j) \in E} \frac{D(t_i, t_j)}{1Mbps} + \frac{U(t_i)}{1Mbps} \quad (4.5)$$

If the value given by this expression is greater than the local running time  $G^l(t_i)$ , this clearly implies that task  $t_i$  is not suitable to be offloaded to cloudlet  $n_c$ . Increases in cloudlet processing speed also have limited effect on improving the workflow’s offload-ability as we discuss further in our simulation study. Pre-testing the workflow with this optimal conditional expression can significantly reduce the algorithm’s workload at run time.

### 4.3 Simulations

We now present the results of the simulations conducted using our algorithm. Our aim is to find out the impact of our offloading algorithm over workflows of various distinct characteristics on top of different hardware environments. The key parameters of this study are the savings made on the workflow’s total energy consumption and its schedule length. We vary the hardware (e.g. processor speed, 3G/WiFi availability) and software (e.g. computation, executable size) specifications and study their effects on the two metrics. For each environmental setup, we conduct 100 runs of the simulation and use the averages as the experimental result. At the start of each run, our model generates a random workflow which includes 40 independent workloads. Then various parameters are fed into the model to construct a simulation of desired characteristics before we let the offloading algorithm take action. The measured metrics are recorded within each run before and after the offload for analysis.

In the simulation, we expect to see two pairs of metrics affect the offloading decision the most: *communication size and network connectivity*, and *computation size and cloudlet processing speed*. We also profile the energy consumption in our simulation as to what activity it is spent on, and analyse the energy profile of the workflow before and after offload.

### 4.3.1 Communication Size and Network Connectivity

In this group of simulations we aim to find out the impact of an increase in communication size over a workflow's offload-ability, and also see if improvements in the wireless connectivity between the smartphone space and the cloudlet domain can help expand the benefits of the offload activity. In order to eliminate the impact from the other critical attributes of a workflow, the computation size, we fix the mean local (smartphone) processing time to  $1/1000$  of the mean communication time, so that the offloading decisions in this group of simulations are all only dependent on the workflow's communication size.

An offloaded workflow's communication expense comes from two sources: the process to send the executable to the cloudlet and the re-routed inter-workload communication calls. We look at their impact separately:

**Executable Size** As shown in Fig. 4.3 increases in a workflow's mean workload executable size (100KB, 200KB, 400KB, 800KB to 1600KB) derives a decrease in the saving generated by the offload. More WiFi connection reduces the extra cost of transferring the executables and thus generates better offload result. Sending a copy of the executable to the cloudlet server is a procedure solely created to enable the offload action and only makes the offload a more expensive in time and energy.

Like the app stores provided on iOS, Android and Windows Mobile, the concept of an enterprise application store has been widely accepted by the industry and is becoming a common practice in business environments. This eliminates the cost to transfer executables to the server. Similar framework can be found in MAUI [22], which keep a code repository on the server which contain a copy of all executables to overcome this issue.

Although both plots in Fig. 4.3 look very similar to each other, we notice that at the 0% WiFi connectivity mark, Fig. 4.3 (b) shows that the savings made in schedule length are mostly zero, whereas Fig. 4.3 (a) indicates that of the same tests energy savings are positive. One intuitive assumption would expect the saving in time and energy to be synchronised with each other, and this contradiction seems impossible on first inspection. Furthermore, as in Fig. 4.4 (a), out of the 100 runs which the WiFi connectivity was set to zero, the number of runs which occurred saving in energy consumption is more than twice the number of runs

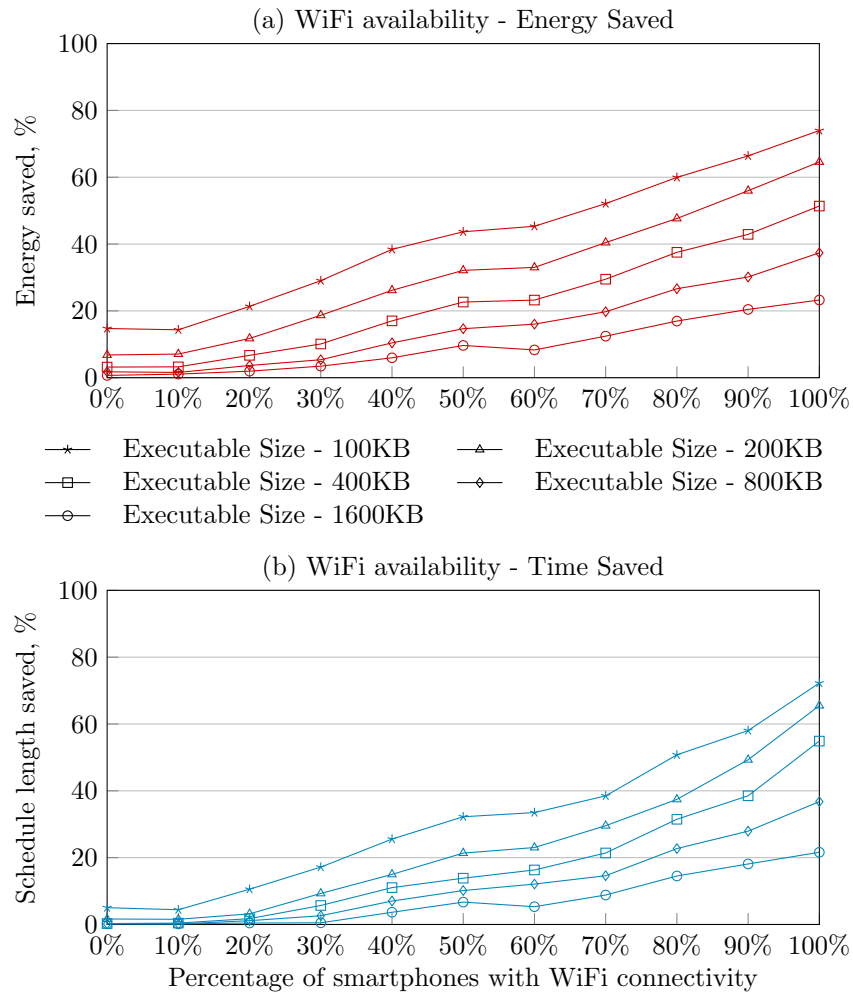


Figure 4.3: Effect of change in executable size and WiFi availability.

As the size of executables increase, fewer savings can be made in the workflow's total energy cost and schedule length (critical path). More WiFi connection makes offload appear more beneficial in both metrics.

with shortened schedule lengths.

In order to understand this result we decomposed this data (at 0% WiFi availability) and found that the extra energy savings come from the tasks that do not reside on the critical path as shown in Fig. 4.4 (b). This analysis indicates that to ensure the workflow gets completed no longer than its original schedule length, tasks on its critical path cannot be offloaded with poor network connectivity. However, away from the critical path where the extra communication time created by an offload can be compensated by its slack time, offload is still a feasible choice and helps preserve energy on the mobile nodes.

**Inter-Task Communication Size** It seems like a intuitive presumption in MCC that an increase in communication size makes offload less favourable. Our simulation shown in Fig. 4.3, which has an increasing executable size brings us to the same conclusion. However, our next set of simulations with increasing inter-task communication size (100KB, 200KB, 400KB, 800KB to 1600KB) gives us an entirely different picture.

In this group of tests, we exchange the value used for executable size and inter-task communication size in the previous simulation. The remainder of the workflow's attributes stay unchanged. As shown in both plots in Fig. 4.5, the lines are intertwined with each other, which indicate that the increase in inter-task communication size did not have a significant effect on how a workflow is offloaded.

To understand this we need to look at one of the fundamental differences our research has over other work, which is that our experiment is based on a workflow whose tasks are scattered across many different smartphone nodes, rather than all concentrated on one device. In such cases, because the tasks are not all local to the same processor node, every inter-task communication of the workflow would have already had a sizeable cost in both time and energy in the original state. Therefore re-routing these tasks does not necessarily occur any additional costs.

Fig. 4.6 shows a comparison between two groups of simulations with contrasting smartphone to workload ratios. It is very clear in the graph that the workflow that has a higher concentration of workload reacts negatively when its inter-task communication size increases, whereas the other workflow which has half the workload concentration rate shows an opposite trend.

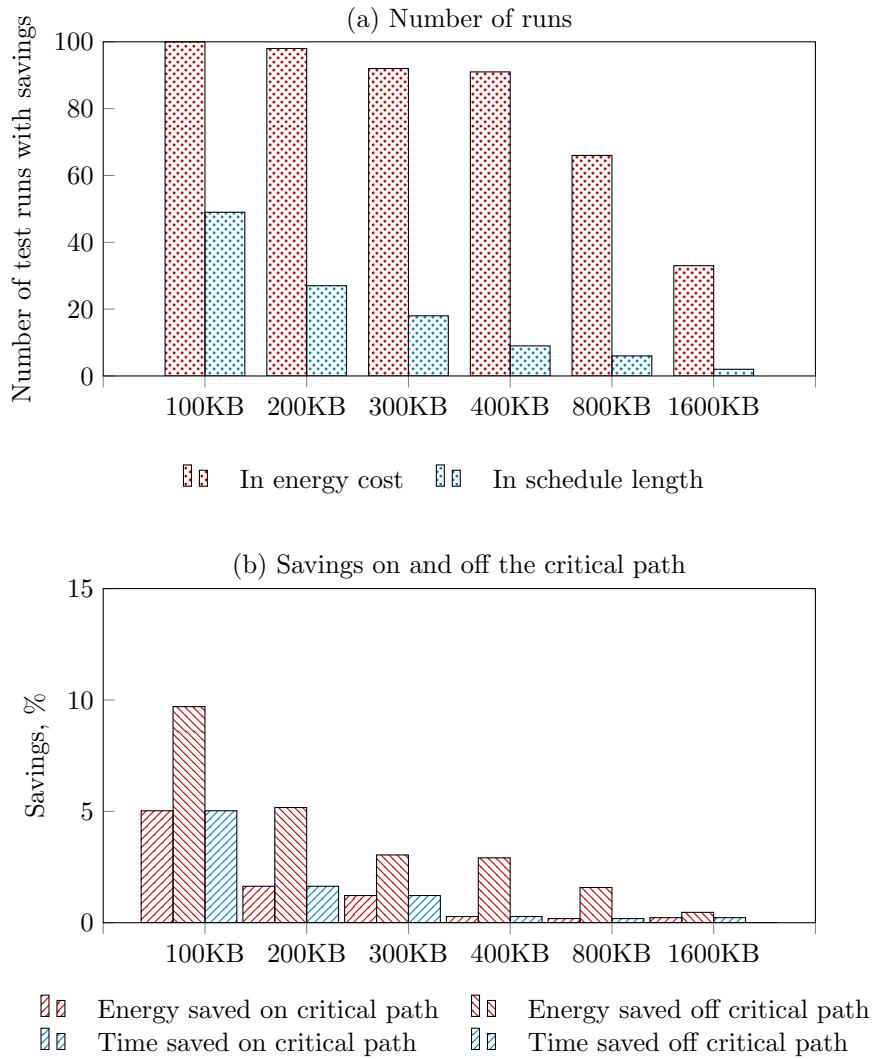


Figure 4.4: Offload savings when no WiFi is available.

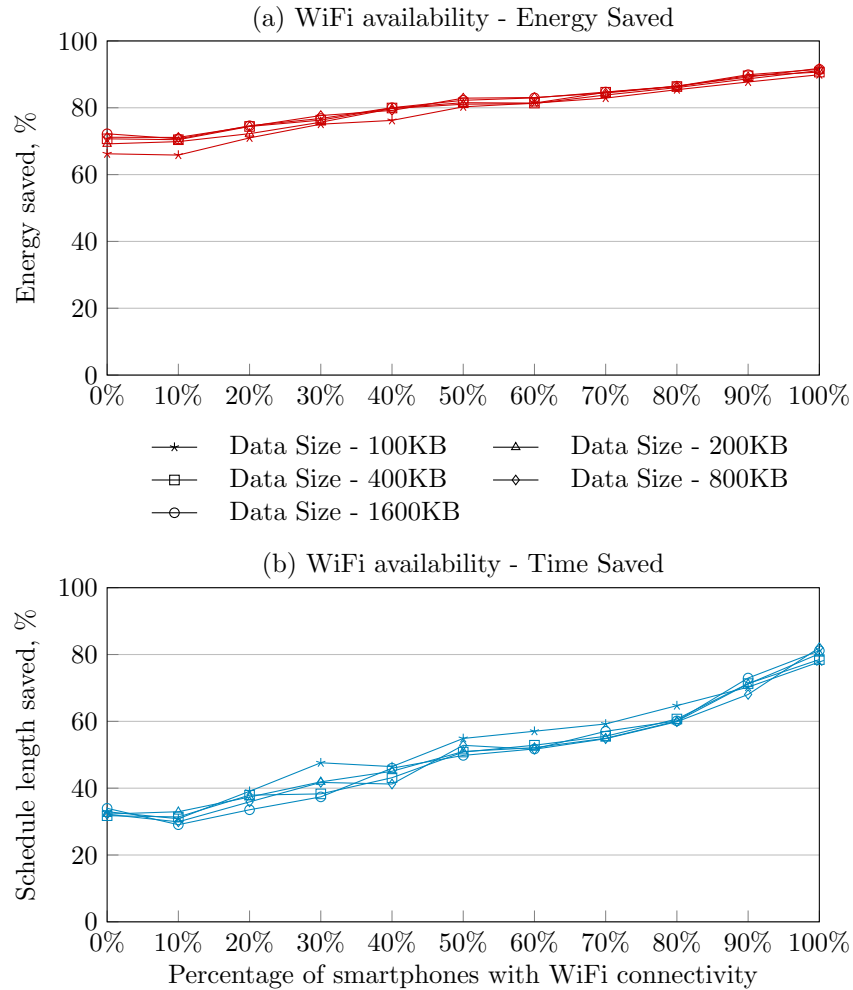


Figure 4.5: Effect of change in communication data size and WiFi availability. When scattered, increase in workflow's local inter-task message data size does not affect its offload-ability. More WiFi connection makes offload appear more beneficial in both metrics.

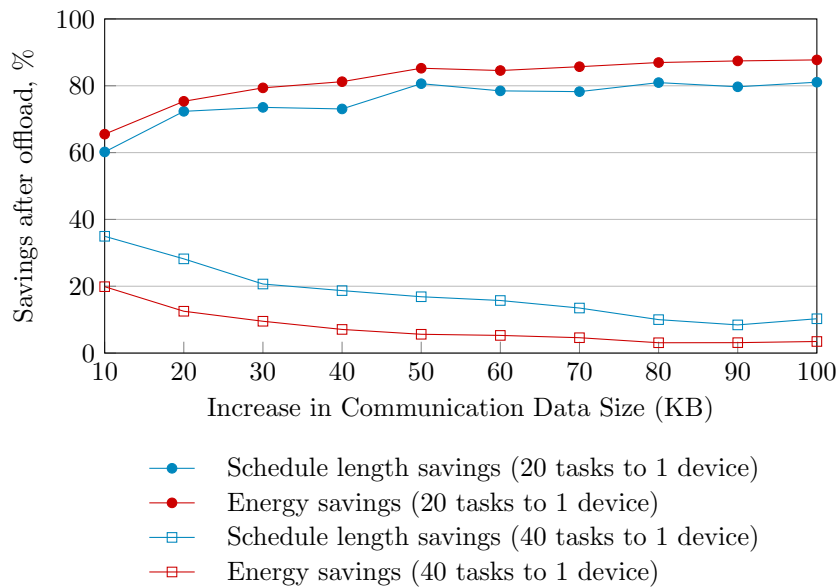


Figure 4.6: Effect of density of workload.

When tasks (workloads) are clustered onto a small number of smartphones, workflows with a bigger local communication size produce less energy gain after offload. The contrary applies when tasks are scattered. All devices have WiFi connection.

### 4.3.2 Computation Size and Cloudlet Speed

The fast processing speed provided by the cloudlet space helps reduce the execution time of tasks and helps to reduce the overall schedule length of the workflow. Energy wise, although the device might need to be in idle mode whilst waiting for the task to be executed on the cloudlet, the energy cost in idle is much lower than that of computation [51]. Hence we expect a group of cloudlets with higher processing speed to produce better offload gains.

In this group of simulations, we set the bandwidth of smartphone-to-cloudlet and smartphone-to-smartphone connections to be the same in order to prevent the result from being influenced by network parameters. Fig. 4.7 shows the variation of savings made in both schedule length and energy consumption with respect to the increase in cloudlet processing speed (from 1 to 1024 times greater to the smartphone speed). We observe that the benefit of offload increases as the cloudlets gets faster. However, both lines fall flat after the smartphone-to-cloudlet gets beyond 1:16.

Recall our discussion on an optimal condition expression at the end of the algorithm section.

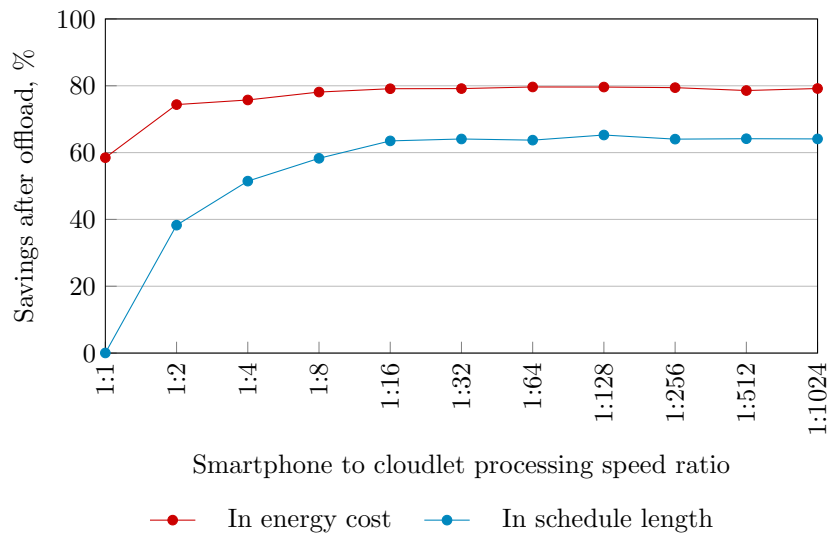


Figure 4.7: Effect of increase in cloudlet speedup.  
Improvements in both savings fall become flat as cloudlet processing speed increases.

Although we do not apply a cap to the cloudlet’s speed in our simulation, the effect from a faster cloudlet is capped to a certain level. In our functions which work out constraint functions (4.1) and (4.2), we can see that this is because the functions all follow a reciprocal relation to the processor speeds on the cloudlets i.e.  $S(n_c)$ . The same also applies to network bandwidth  $B(n_i, n_j)$ . This indicates that improvements in hardware environments help increase the offload-ability of workflows but excessive investment is not necessary.

### 4.3.3 Energy Profile

In our simulations, as well as seeing the savings made by offload, we are also interested in what activities (computation or communication) the energy was spent on before and after the offload. Fig. 4.8 includes two stacked bar plots. The one in the background shows the energy distributions of the original workflow. The second plot, in the foreground, shows the energy distributions of the offloaded workflow. The top section of both plots indicate the share of energy that is spent on communication. The data is grouped so that on the very left is the data gathered from workflows that are offloaded only to take advantage of the fast processing speeds of the cloudlet (i.e. with poor network bandwidth). On the very right are data from workflows that are offloaded only



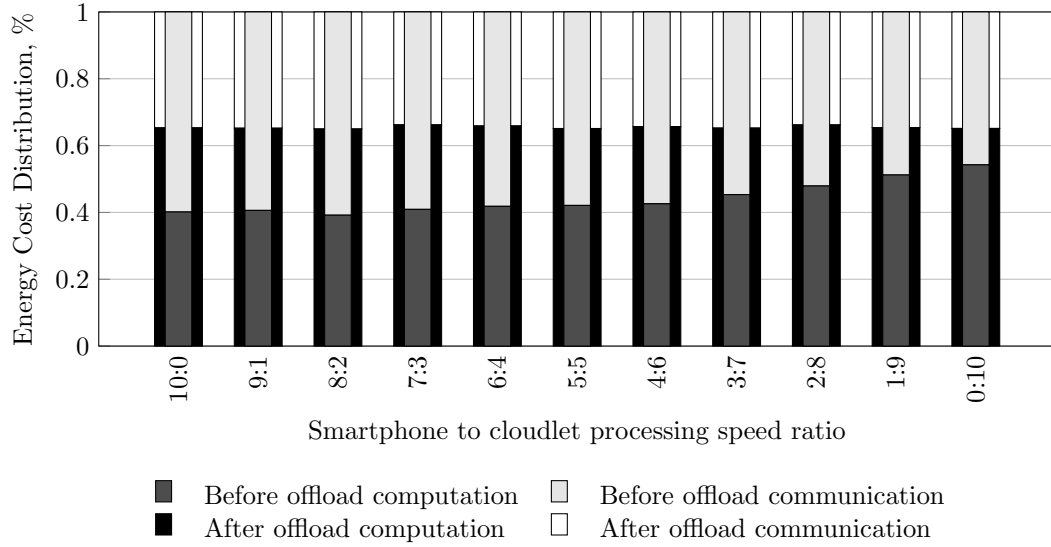


Figure 4.8: Energy distribution before and after offload.  
Offloaded workflows are proportionally more reliant on the network.

to eliminate communication costs (slow cloudlet speed). We can clearly see that in all groups, the share of energy spent on communication has been increased after offload. This is a clear indication that an offloaded workflow is proportionally more reliant on network connectivity than its original form.

## 4.4 Summary

In this chapter, we presented our approach to managing a mobile workflow over its supporting platform in an energy- and time-aware manner. With a model which reflects the software and hardware characteristics of the scenario, we developed a heuristic algorithm to build and update the offload plan dynamically based on the time and energy constraints of the workflow. Variations of the objective functions are also discussed together with optimisation of the algorithm.

A series of simulation studies concludes that:

1. When no code repository is available at the server side, a large executable size invariably generates a negative effect on a workflow's offload-ability.
2. Large inter-task communication size within a workflow only makes offload less feasible when

tasks are concentrated on a small number of smartphones.

3. Energy savings can be found easier on workloads that are not on the workflow's critical path, so even when offload is proven not to be preferable by the time constraint, savings can still be made in the workflow's overall energy consumption.
4. The significance of the savings brought about by offload follow a reciprocal relation to the hardware metrics.
5. Offloaded workflows are proportionally more reliant on the network.

## Chapter 5

# Bandwidth Dependency and Allocation in Mobile Service-Oriented Networks

Bandwidth is another influential factor to the QoS of mobile applications and services alongside energy. As we have demonstrated in the previous chapters, the bandwidth variable plays a pivotal role in the management of mobile cloud computing platforms. When the services requested by mobile application workflows are distributed over a network of cooperative mobile smart devices and clouds, the question arises as to which service should be allocated with how much bandwidth and when in order to satisfy service demands? Moreover, how to adjust the bandwidth allocation to accommodate changes in service demands whilst maintaining service QoS? Furthermore, the mobility of smart mobile devices brings forward the challenge to determine how changes in mobile network conditions affect the bandwidth requirements of interacting services.

To answer these questions, in this chapter, we investigate the resource management aspect of mobile cloud computing platforms, more specifically on modelling the bandwidth dependencies between interactive components of mobile application workflows. We generalise the bandwidth allocation problem in mobile cloud computing platforms to that of generic mobile wireless net-

works which we refer to as Mobile Service-Oriented Networks (MSON)s, so that the model is applicable to a wider scope of problems. We assume all computation nodes are mobile devices and don't specifically include cloud nodes in our model. This generalisation does not alter the structure of the model since we wish to quantify the bandwidth requirements of each computation nodes and whether the node is mobile or cloud does not affect this value. We give further definition of an MSON in Section 5.1.

In Section 5.2 we give introduction to the Leontief I-O model which is th foundation of economic studies. Then in Section 5.3 we adopt and extend on the analytical framework of the Leontief Input-Output model and develop a network I-O model to describe the bandwidth dependencies within an MSON. Various factors such as bandwidth, latency, service demand and costs are accounted for in the model. A set of equilibrium equations are derived to produce the bandwidth requirements of mobile devices.

Based on the Network I-O model, first a cost-based bandwidth allocation scheme is developed to maximise service benefit in Section 5.5. Next, a set of adaptive bandwidth allocation strategies is also proposed to accommodate changes in service demands and network conditions while minimising the overall impact on application workflows in Section 5.5. Results from simulation studies are presented at the end of each section to demonstrate the effectiveness of the proposed methods.

## 5.1 Mobile Service-Oriented Networks

In this chapter, we assume that mobile (cloud) application workflows are manifested as dynamic compositions of services located on mobile devices. We refer to the underlying network structure as a Mobile Service-Oriented Network (MSON).

### 5.1.1 Example and Definition

Consider as a example a wireless network of personal mobile smart devices as shown in Fig. 5.1. The network consists of a smartphone, a tablet and a smart TV. A remote file storage service is also accessible from the network. Each device provides (within the dotted boxes) services (illustrated in solid boxes) to the user. Each service module (e.g. in the form of mobile apps)

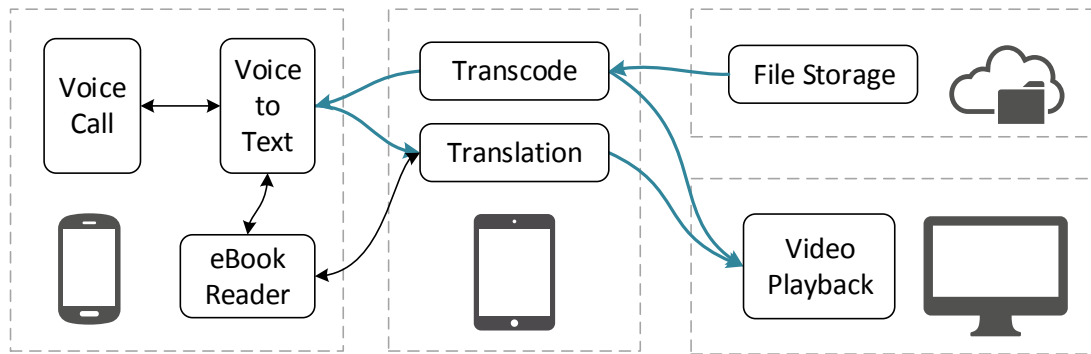


Figure 5.1: A simple example of a personal MSON.

serves a different purpose to the user and is able to run independently. However, when connected via a network, these services can also be dynamically combined to serve more complex mobile application workflows.

For instance, as illustrated in Fig. 5.1 by the coloured lines, the user can use the tablet to stream a remote video file from the storage service by transcoding it to a format that is readable by the smart TV. At the same time, if the smartphone is also available (has adequate bandwidth) to the network, the tablet can transcode the video file to a voice stream that can be transcribed to a text stream on the smartphone. This text information can then be translated on the tablet to a language chosen by the user and streamed to the smart TV as subtitles.

Observe that the process of service composition is dynamic and non-deterministic. The composition decision may be influenced by many factors such as 1) network conditions: the smartphone might not have enough bandwidth therefore the “Voice-to-Text” service on the smartphone cannot receive the voice stream from the “Transcode” service on the tablet; 2) dynamic application information: the film may be in a language which the user can understand, and therefore the “Translation” service is not included in the workflow.

In this chapter, we refer to this type of mobile networks as *Mobile Service-Oriented Networks* (MSON)s on which a universe of services is distributed, and mobile applications are manifested as dynamic compositions of these services. It is easy to see that one of the key challenges that comes with the research in MSON is the constrained and unpredictable wireless network connection capacity (e.g. bandwidth). In contrast to desktop based SOA networks [83, 84, 85], focus of the bandwidth allocation problem has shifted from the centre of the network (considered fast in an

MSON) to the access points at the edge of the network. If the required bandwidth for a service were not met, it is not just a matter of delaying the execution of other interacting services, but more importantly the matter of the collective waste of bandwidth that is created down the application workflow pipeline [86].

Bandwidth allocation schemes are commonly found residing in the one-station-to-many-device cell structures, and are in charge of distributing available bandwidth of the mobile station to devices within its cell proximity. The bandwidth allocation problem of a mobile station with three radio interfaces (IEEE 802.11, WLAN and CDMA) is modelled as a bankruptcy game in [87]. In [88], a bandwidth allocation scheme is proposed for the WCDMA system. In [89], the multiple fractional channel reservation strategy is used to maximise wireless channel capacity in wireless networks with multiple radio interfaces. We refer the interested reader to [90] for a comprehensive survey of bandwidth allocation strategies in such settings.

In contrast, our Network I-O model applies to the service-to-service cooperative network structures, such as MSONs, supporting mobile application workflows. The allocation schemes derived from the model are technology independent in that the backbone of this network may be a LAN, a WAN or a WANET, the access network for each device may also vary from WiFi, 3G to WiMAX, LTE. We focus on the interactions between services hosted on mobile devices that are connected through a backbone network infrastructure. These devices may be distributed geographically and do not necessarily need to be connected to the same mobile station. Allocations are done at a device level for services supporting mobile application workflows. We also don't assume the knowledge of call graphs between services as in Chapter 3 and Chapter 4.

### 5.1.2 Service-Oriented Architecture

The characteristics and challenges faced by mobile application workflows motivates the adoption of the service-oriented architecture (SOA) [91] which advocates:

**Encapsulation** of core functionalities, therefore an application workflow can be developed as a composition of service modules. For instance, a train timetable application may go one step further and invoke a traffic and navigation service to guide the user to the train station, a video streaming service may further compose a voice to text service to add subtitles to the

video as we have shown in Fig. 5.1.

**Loose coupling** principle of SOA ensures that the dependency between two services is minimised, and that the services only maintain a knowledge of the existence of each other, with composition of services done on demand. The P2P file sharing architecture presented in [46] and the mobile based social interaction system proposed in [92] exploits this principle. Therefore our timetable application may choose to overpass the traffic service if the current network bandwidth is limited or when the residual battery life is low.

**Service discovery** makes the detection of services over the network possible, and facilitates the usage between services. For instance, in order to construct a virtual traffic light system as presented in [93], vehicles approaching the same junction must be able to discover the service of each other to cooperate, a tour guide service may be located at a tourist attraction to enrich the user's experience.

### 5.1.3 Applications of MSON

An MSON infrastructure can be observed from many research areas. In vehicular wireless systems (VANETs), many applications [13] are built on top of cooperative networks of mobile smart devices installed on smart vehicles. Exemplar applications includes BitTorrent-styled location significant content downloading [94] and vehicle-to-vehicle environment and safety sensing [93]. A mobile-based social interaction application is presented in [92]. In biomedical applications, a mobile application workflow is used in [95] to describe a sensor-based biomedical application which includes mobile devices used as both sensors and data processing units. The motivating scenario addressed in [84] describes the benefit of using service composition in a hospital resource scheduling application. In other mobile application areas, a mobile P2P file sharing framework is presented in [46]. A framework for mobile P2P social content sharing is presented in [44]. These studies all share the same underlying MSON infrastructure.

### 5.1.4 Mobile Device as Service Hosts

Extensive research has been carried out to use mobile devices as service hosts. The idea of mobile devices as service hosts is also an active research topic. In [96], an SOA-based approach

is presented to support interactions between business applications running on J2ME. In [97], opportunistic composition of sequentially-connected service over a decentralised mobile ad hoc network is proposed. Experiments conducted in [98] demonstrates that this opportunistic communication is viable at scale. A power-aware mobile service composition algorithm is presented in [86]. In the same work, the problem of wasted network resource down a service workflow pipeline is also discussed. A middleware is created in [99] to reduce user perceived latency while accessing remote services on mobiles by pre-fetching and caching data according to a sequence prediction algorithm. The same technique has been shown to reduce battery cost. All of these studies are about developing the service-oriented architecture on mobile devices. None of the work discusses the bandwidth requirement of mobile services.

## 5.2 Input-Output Analysis in Economics

Suppose a nation's economy is divided into  $n$  sectors that produce goods or services. Let  $x_i$  be the value of goods or services produced by sector  $i$ , we then have a *production vector*  $\mathbf{x} \in \mathbb{R}^n$  to list the output from all sectors of the economy. In order to avoid waste and deficiency, production is planned in accordance to the demand of goods and services which originates from two channels: **External demand** represents consumer demands, exports, planned surplus, etc. from the economy. Let  $d_i$  be the external demand of sector  $i$ , then  $\mathbf{d} \in \mathbb{R}^n$ , namely the *external demand vector*, lists the external demand (output) of all sectors of the economy. **Intermediate demand**, represents intra-sector demand of good and services. For instance, assume a small town with two primary industries: a steel plant and a railway. Then in order to produce goods, the steel plant requires services from the railway. To represent the intermediate demand, a square matrix  $A \in \mathbb{R}^{n \times n}$ , namely the *consumption matrix*, is assumed, in which  $a_{ij}$  denotes the production (input) needed from sector  $i$  per unit of production (output) by sector  $j$ . With this definition, we have that in order to produce  $x_j$  units of good or services, sector  $j$  will demand  $x_j a_{ij}$  units from sector  $i$ , that is the intermediate demand by sector  $j$  from sector  $i$ . When the



economy's production balances the total demand for that production exactly, we have:

$$\underbrace{\mathbf{x}}_{\text{production}} = \underbrace{A\mathbf{x}}_{\text{intermediate demand}} + \underbrace{\mathbf{d}}_{\text{external demand}} \quad (5.1)$$

which is the cornerstone of the Leontief Input-Output model of economics. This model helps economists understand how changes in one sector affect others, and predict the production level required to balance the demand exactly. Such is the significance of his research in the I-O analysis of economics, Leontief was awarded the Nobel Prize in Economics in 1973.

### 5.3 The Economy of Mobile Service-Oriented Networks

We consider each service as a sector of the network economy. Entailed by the SOA paradigm, services are combined and possibly recombined to create complex applications (workflows) that serve the demand of the end users. This composition of services is a dynamic run-time decision process which adapts to: the fluctuating network conditions (which is especially true for a mobile network), various application-dependent QoS level requirements [100, 101, 102] and dynamic application information (e.g., whether the user requires translation for a video). This means that the exact execution sequence of services is not predefined and therefore the communication demands between services are non-deterministic. This behaviour is similar to that of the common economies analysed by the Leontief I-O model. For instance, consider manufacturing and raw material as two sectors of an economy. Each product of the manufacturing sector has its own bill of materials and may require different amount of input from the raw material sector. Furthermore, a repair service may avoid input from the raw material sector completely if it does not require any replacement parts.

Besides the non-deterministic behaviour, when exchanges between economic entities are made, interactions within an MSON manifest many common characteristics to that of interacting sectors of a common economy. Notwithstanding these similarities, key economic entities need to be identified and their behaviours modelled before the bandwidth I-O model of an MSON can be constructed. We describe the construction of such an economy in the rest of this section which is supplemented by Fig. 5.2 throughout.

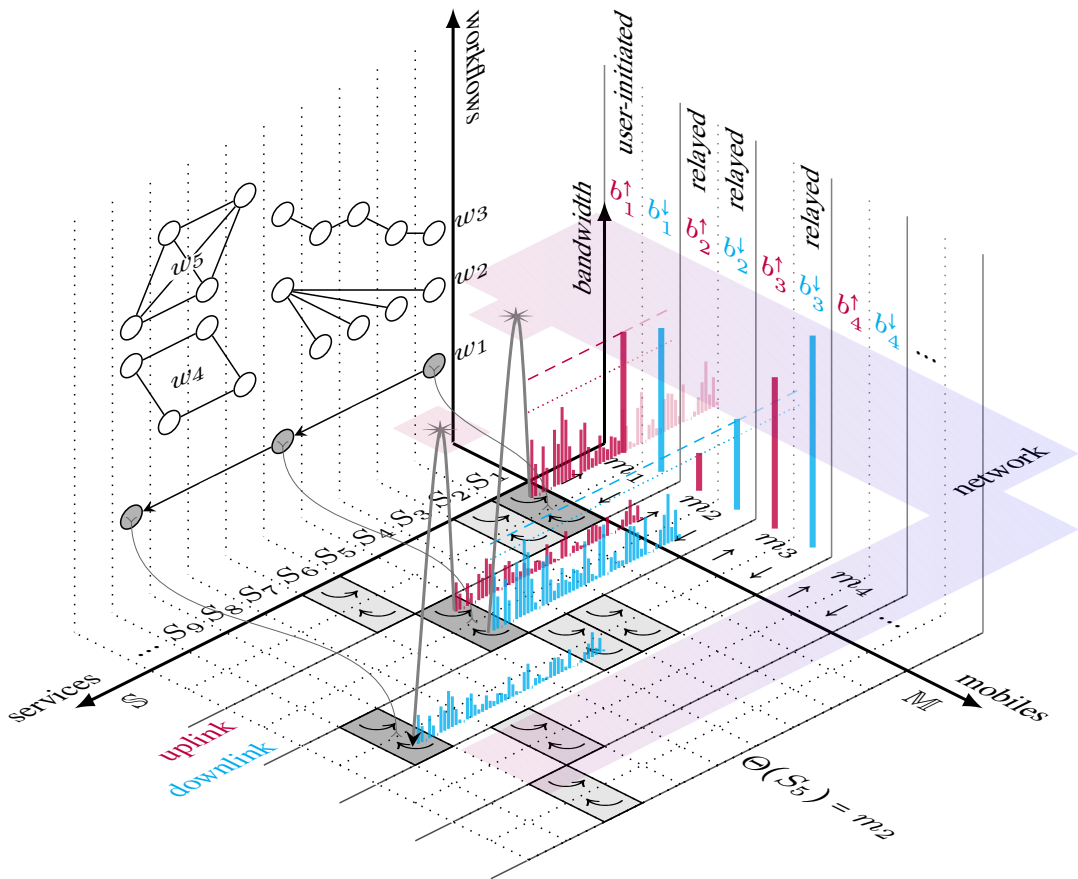


Figure 5.2: The economy of an MSON.

There are three planes in this illustration. The service-mobile plane laying flat in the centre gives the allocation scheme ( $\Theta$ ). The one standing vertically on the left shows that the communication behaviour between services (represented by circles) is a statistical process of all possible service compositions (workflows). The plane standing on the right shows that each mobile device has its uplink and downlink bandwidth to facilitate its communication to the network.

### Data as Commodity

Services (sectors) of an MSON economy produce and exchange data to serve the demands of its end users. This data as a commodity may carry information requested by the user (e.g., query services), which may be the product in accordance to user input (e.g., image processing service), or simply be the confirmation from the service that the user's request has been recorded (e.g., flight check-in service).

### Bandwidth as Currency

Exchange of data is facilitated by the network. One unit of network bandwidth facilitates the exchange of one unit of data in one unit of time. Similar to the common currency (e.g., one US dollar) used in an economy to measure goods and services of different sectors, one unit of bandwidth is the common currency of an MSON economy.

### Exchange of Data

Let  $\mathbb{S}$  denote the universe of services distributed over the network containing a set  $\mathbb{M}$  of mobile devices, according to a mapping scheme  $\Theta : \mathbb{S} \rightarrow \mathbb{M}$ . (In Fig. 5.2, we have  $\Theta(s_1) = m_1, \Theta(s_2) = m_1, \Theta(s_3) = m_3$ , and so on.) For each service  $s_i \in \mathbb{S}$ , assuming that historical data (e.g., collected by filtering logging data) are available [103, 104], and let  $\beta_i$ , measured in units of bandwidth, denote the (average) size of data produced by each run of  $s_i$  as an intermediate step of a service composition. The effect of  $\beta_i$  is three-fold:

First, as an intermediate product,  $\beta_i$  needs to be communicated to the next service(s)  $s_j \in \mathbb{S}$  as instructed by the service composition (application workflow). If  $s_j$  is not located on the same device as  $s_i$ , then  $\beta_i$  needs to be sent from its host  $\Theta(s_i) \in \mathbb{M}$  over the MSON. We define a co-location indicator

$$\omega_{ij} = \begin{cases} 0 & \text{if } \Theta(s_i) = \Theta(s_j), \\ 1 & \text{otherwise.} \end{cases} \quad (5.2)$$

so that  $\omega_{ij} = 1$  indicates that, if destined to  $s_j$ , the task of sending  $\beta_i$  would consume the uplink bandwidth of  $\Theta(s_i)$ . As illustrated in Fig. 5.2, when a service workflow  $w_1$  is initiated on  $m_1$ , because the next service ( $s_5$ ) is located on a different device ( $m_2$ ),  $m_1$  has to first upload the

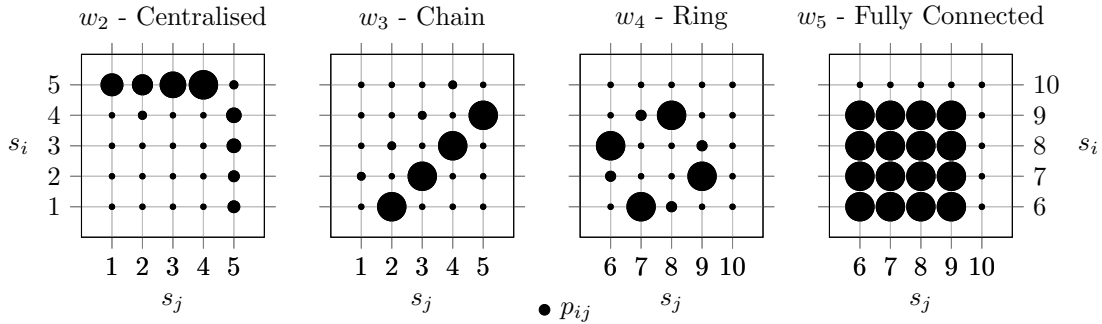


Figure 5.3: Four types of service communication patterns.

data to the network. We refer to this type of bandwidth cost as *self-initiated cost* later on in this subsection. We assume that a square matrix  $P = [p_{ij}]_{|\mathbb{S}| \times |\mathbb{S}|}$ , in which  $p_{ij}$  denotes the probability that a run of  $s_i$  is to be succeeded by a call to  $s_j$  ( $\beta_i$  is to be sent to  $s_j$ ), is known through profiling [103, 104]. Together with our co-location indicator  $\omega_{ij}$ , we define

$$\rho_i = \sum_j p_{ij} \omega_{ij} \quad (5.3)$$

which gives the probability that each unit product (data) of  $s_i$  is to be uploaded to the MSON by  $\Theta(s_i)$ . Fig. 5.3 illustrates the communication patterns of four typical service composition patterns,  $w_2$ ,  $w_3$ ,  $w_4$  and  $w_5$ . These are also illustrated in the left vertical plane of Fig. 5.2. Note that  $\sum_j p_{ij}$  is not necessarily one, because each run of  $s_i$  is not necessarily succeeded by a call to another service. We also assume that data is passed on to the user (e.g., at the end of an application workflow) by the service located on the user's mobile device. Therefore all data presented to the user is considered a local resource on the user's device and does not consume any bandwidth.

Second, for a service  $s_j$  to receive  $\beta_i$ , the downlink bandwidth of  $\Theta(s_j)$  is consumed. In the example illustrated in Fig. 5.2, this receive action is taken by  $m_2$  which hosts  $s_5$ . This creates a dependency between the consumption of the uplink bandwidth of the sender device and the downlink bandwidth of the receiver device in the MSON economy. We define

$$\eta_{ij} = \frac{p_{ij} \omega_{ij}}{\sum_k p_{ik} \omega_{ik}}, \quad s_k \in \mathbb{S} \quad (5.4)$$

which gives the probability that a unit of data sent by  $s_i$  to the MSON is to be received by  $s_j$ . We refer to the cost occurred in this type of process as *relayed cost* later on in this section.

Third, depending on the specification of the application workflow, the service which received  $\beta_i$  may be requested to further communicate with other services. Take for instance the example workflow illustrated in Fig. 5.2,  $s_5$  is to continue the workflow and communicate with  $s_9$ . This action consumes the uplink bandwidth of  $m_2$  and the downlink bandwidth of  $m_3$ . As such, following a service workflow, a sequence of services in the service composition would be requested to perform communication tasks. This chain effect exists in every application and is triggered by the execution of the head services of the workflow. In the following section, we refer to the data produced by the head services as *self-initiated cost* and all subsequent production of data as *relayed cost*, and discuss in more details.

### Self-initiated Cost vs. Relayed Cost

Recall that one of the key characteristics of the Leontief I-O model is that it classifies production of goods and services by their corresponding demand into two groups, namely external demand and intermediate demand. Following previous discussion, we discover that the production of data, and thus the cost of bandwidth, in a MSON can also be classified into two classes: ***Self-initiated*** production of data refers to data generated by the head services executed at the start of every application workflow and exhibits the same characteristics as the external demands in Leontief's model. Devices that hosts these head services bear the cost of sending data to subsequent services. These costs are in the form of uplink bandwidth of the sender device, and are initiated solely by the service itself (e.g., triggered by user action). Let  $\lambda_i$  denote the arrival rate of  $s_i$ , then the self-initiated cost to the uplink of  $\Theta(s_i)$  which we denote  $c_i^\dagger$  is given by

$$c_i^\dagger = \lambda_i \beta_i \rho_i \quad (5.5)$$

The other class of data production is in contrast caused by services that executed prior in the application workflow and thus no consequent cost is self-inflicted. We refer to this as ***relayed*** production of data. Bandwidth cost from this class of data production can be in forms of both uplink and downlink bandwidth. We derive the cost function of this class in the proof of Theorem

## 5.3.1.

Note that the same data which is considered to be self-initiated by its sender, is classified as relayed by the receiving service(s). This is because that the receiving service is selected by a dynamic service composition process (reflected in  $P$ ), and this selection process is not deterministic.

**Markets: Uplink vs. Downlink**

We consider uplink and downlink as two related markets on which data are traded. The difference between the two is that the data exchanged through the uplink market includes both self-initiated and relayed data, whereas the demands for downlink bandwidth are all relayed from the uplink market. Thus in the next section we derive one set of I-O equations for each of the two markets.

**5.3.1 Network I-O Model**

Given a service  $s_i \in \mathbb{S}$ , let  $x_i = x_i^\uparrow + x_i^\downarrow$  denote its total, uplink and downlink bandwidth costs respectively. We now construct a model that derives these values with the limited information we have about the MSON, i.e.,  $\Theta$ ,  $P$ ,  $\beta$  and  $\lambda$ .

**Definition 6.** For each pair of services  $\{s_i, s_j\} \in \mathbb{S}^2$ , the elements of the *uplink consumption coefficient matrix* of  $\mathbb{S}$ , denoted  $A^\uparrow = [a_{ij}^\uparrow]_{|\mathbb{S}| \times |\mathbb{S}|}$  is given by

$$a_{ij}^\uparrow = \frac{1}{\beta_j \rho_j} p_{ji} \beta_i \rho_i \quad (5.6)$$

**Theorem 5.3.1.** Let  $\mathbf{x}^\uparrow = [x_i^\uparrow]_{|\mathbb{S}| \times 1}$  denote the uplink bandwidth demand vector of  $\mathbb{S}$ , and  $\mathbf{c}^\uparrow = [c_i^\uparrow]_{|\mathbb{S}| \times 1}$  denote the self-initiated demand vector of  $\mathbb{S}$ , then when the network is in equilibrium (meaning that each service is given the amount of bandwidth it requires to run without delay) the following equation holds

$$\underbrace{\mathbf{x}^\uparrow}_{\text{uplink cost}} = \underbrace{A^\uparrow \mathbf{x}^\uparrow}_{\text{relayed uplink demand}} + \underbrace{\mathbf{c}^\uparrow}_{\text{self-initiated demand}} \quad (5.7)$$

*Proof.* From our earlier discussion in 5.3, we know that the send (uplink) action of a service  $s_i$

is triggered by two sources, namely self-initiated and relayed. With  $c_i^\uparrow$  defined in (5.5), let  $h_{ji}^\uparrow$  denote the uplink demand that is relayed from  $s_j$  to  $s_i$ , i.e., when  $s_j$  immediately precedes  $s_i$  in an application workflow. Therefore

$$x_i^\uparrow = \sum_j h_{ji}^\uparrow + c_i^\uparrow \quad (5.8)$$

With (5.3) we derive that each run of  $s_j$  and  $s_i$  is to generate data of size  $\beta_j \rho_j$  and  $\beta_i \rho_i$  respectively. If service  $s_j$  were to be allocated an uplink bandwidth of  $x_j^\uparrow$ , as an equilibrium entails,  $s_j$  would execute  $x_j^\uparrow / \beta_j \rho_j$  times. From the communication probability matrix  $P$ , we know that for every one run of  $s_j$  there is a probability  $p_{ji}$  a subsequent run of  $s_i$  is triggered. Therefore we have

$$h_{ji}^\uparrow = \frac{x_j^\uparrow}{\beta_j \rho_j} p_{ji} \beta_i \rho_i \stackrel{(5.8)}{\Rightarrow} x_i^\uparrow = \sum_j \frac{x_j^\uparrow}{\beta_j \rho_j} p_{ji} \beta_i \rho_i + c_i^\uparrow \quad (5.9)$$

Consider  $i \in \{1, 2, \dots, |\mathbb{S}|\}$ , (5.9) derives the same set of equations as given by taking (5.6) into (5.7).  $\square$

**Definition 7.** For each pair of services  $\{s_i, s_j\} \in \mathbb{S}^2$ , the elements of the *downlink consumption coefficient matrix* of  $\mathbb{S}$ , denoted  $A^\downarrow = [a_{ij}^\downarrow]_{|\mathbb{S}| \times |\mathbb{S}|}$  is given by

$$a_{ij}^\downarrow = \eta_{ji} = \frac{p_{ji} \omega_{ji}}{\sum_k p_{jk} \omega_{jk}}, \quad s_k \in \mathbb{S} \quad (5.10)$$

**Theorem 5.3.2.** Let  $\mathbf{x}^\downarrow = [x_i^\downarrow]_{|\mathbb{S}| \times 1}$  denote the downlink bandwidth demand vector of  $\mathbb{S}$ , then when the network is in equilibrium (meaning that each service is given the amount of bandwidth it requires to run without delay) the following equation holds

$$\underbrace{\mathbf{x}^\downarrow}_{\text{downlink cost}} = \underbrace{A^\downarrow \mathbf{x}^\uparrow}_{\text{relayed downlink demand}} \quad (5.11)$$

*Proof.* It is easy to understand that within the MSON, the downlink cost is totally dependent on the uplink cost in the sense that no receive action is required if no data was sent, and that all data sent by a service in context of the MSON must be received by another service of the MSON. On this basis, let  $h_{ji}^\downarrow$  denote the downlink cost relayed from data sent from  $s_j$  to  $s_i$ , i.e.,

the amount of data sent from  $s_j$  to  $s_i$ , and we have

$$x_i^\downarrow = \sum_j h_{ji}^\downarrow \quad (5.12)$$

Recall from (5.4) that the probability that a unit of data sent by  $s_i$  to  $s_j$  is given by  $\eta_{ij}$ , we derive

$$h_{ji}^\downarrow = x_j^\uparrow \eta_{ji} \stackrel{(5.12)}{\Rightarrow} x_i^\downarrow = \sum_j x_j^\uparrow \frac{p_{ji} \omega_{ji}}{\sum_k p_{jk} \omega_{jk}}, \quad s_k \in \mathbb{S} \quad (5.13)$$

Similarly to the proof of theorem 5.3.1, by enumerating (5.13) with  $i \in \{1, 2, \dots, |\mathbb{S}|\}$ , we get the same set of equations as given by taking (5.10) into (5.11).  $\square$

Observe that from the definition given by (5.2), we know that  $\omega_{ji} = 0$  when  $i = j$ , therefore the entries on the main diagonal of  $A^\downarrow$  are all zero. In contrast, the main diagonal of  $A^\uparrow$  are not necessarily all zero. These properties of the two coefficient matrices match the behaviour of a service in a practical sense. When a service (recursively) calls on itself (refer to the communication pattern given by [103] and [104]), the pair of send and receive action itself is local and thus does not cost the hosting device's bandwidth to the access network. However, the consequence of this communication does not exclude the possibility of data being produced by the newly invoked service call. This new data has a non-zero possibility (if  $\rho_i > 0$ ) to be destined to services that are not locally available, and thus would incur a cost to the hosing device's uplink bandwidth.

To conclude the network I-O model, we gather the per-service cost from both markets and derive the total bandwidth cost for a host device  $m \in \mathbb{M}$  as

$$b_m = b_m^\uparrow + b_m^\downarrow = \sum_i x_i^\uparrow + \sum_i x_i^\downarrow = \sum_i x_i, \quad \Theta(s_i) = m \quad (5.14)$$

with  $b_m$ ,  $b_m^\uparrow$  and  $b_m^\downarrow$  denote the total, uplink and downlink bandwidth cost of  $m$ .

### 5.3.2 Network I-O Model with Latency

Network latency is another crucial factor in network performance modelling. Whereas the bandwidth based network I-O model we presented so far describes the dependencies of the bandwidth demands between mobile services when the effect of network latency is negligible (in cases where



latency may be modelled as a constant reduction of available bandwidth), it does not provide the necessary means to reflect the dynamics of network latency. Therefore in this section, we extend the network I-O model to incorporate the network latency factor and demonstrate its effect in an MSON. From the definition of network latency and bandwidth we know

$$\text{time} = \text{latency} + \frac{\text{data size}}{\text{bandwidth}} \quad (5.15)$$

which indicates that when latency increases, the amount of bandwidth that is required in order to transfer the same amount of data in the same amount of time also increases. We denote  $l_i$  and  $x_i^\uparrow$  to be the latency and uplink bandwidth of service  $s_i$  (when latency is considered) respectively. Similar to bandwidth (uplink at source and downlink at destination), the network latency between two services also depends on the latencies at both ends of the link between the service pair, therefore the latency between  $s_i$  and  $s_j$  is given by  $l_{ij} = (l_i + l_j)\omega_{ij}$ , with  $\omega_{ij}$  as defined by (5.2). The amount of data needs to be transferred (in a unit of time  $t = 1$  second) with latency considered remains the same as is given by  $x_i^\uparrow$ . Therefore from (5.15) we derive

$$t = \sum_j \frac{x_i^\uparrow}{\beta_i \rho_i} p_{ij} l_{ij} + \frac{x_i^\uparrow}{x_i^\uparrow} \Rightarrow x_i^\downarrow = x_i^\uparrow / (t - \sum_j \frac{x_i^\uparrow}{\beta_i \rho_i} p_{ij} l_{ij}) \quad (5.16)$$

which establishes the adjustment from  $x_i^\uparrow$  to  $x_i^\downarrow$ .

For the second part of our extension to the network I-O model with latency, we establish the relay relation from the uplink bandwidth of the sender service to the downlink bandwidth of the receiver service. We denote  $x_i^\downarrow$  as the downlink bandwidth of  $s_i$  (when latency is considered). Deriving from (5.15) again, we have

$$t = \sum_j \frac{x_j^\uparrow}{\beta_j \rho_j} p_{ji} l_{ji} + \frac{x_i^\downarrow}{x_i^\downarrow} \Rightarrow x_i^\downarrow = x_j^\uparrow / (t - \sum_j \frac{x_j^\uparrow}{\beta_j \rho_j} p_{ji} l_{ji}) \quad (5.17)$$

## 5.4 Parametric Evaluation

To summarise, our network I-O model (given by (5.7) and (5.11)) describes the dependencies of an MSON's properties: Given a mapping scheme  $\Theta$ , a communication pattern  $P$ , a latency

vector  $\mathbf{l}$  and two service QoS metrics vectors  $\beta$  and  $\lambda$ , we can derive a per service bandwidth allocation vector  $\mathbf{x}$  which further derives a per device bandwidth allocation vector  $\mathbf{b}$ .

In this section, we conduct a series of simulation studies based on two types of service topologies: centralised and chain (illustrated by  $w_2$  and  $w_3$  in Fig. 5.2 and Fig. 5.3) to demonstrate the basic dynamics of the network I-O model. We assume a service-to-mobile allocation scheme given as  $\Theta(s_1) = \Theta(s_2) = M_1$ ,  $\Theta(s_3) = \Theta(s_4) = M_2$  and  $\Theta(s_5) = M_3$  in both sets of experiments.

#### 5.4.1 Effect of Service Arrival Rate

In comparison to the other properties of an MSON, the service arrival rate vector  $\lambda$  in reality is likely to have short term fluctuations. Recall that all bandwidth costs are either directly inflicted by the service's self-initiated actions, or are the relay consequence of the self-initiated actions of other services. Therefore each service request initiated in the MSON may inflict bandwidth costs to both its host device and those it communicates with.

In this set of simulations, we demonstrate the dynamics of the network I-O model by examining the effect of increase in  $\lambda$  on the bandwidth costs of all services in  $\mathbb{S}$ . Furthermore, we map each service to a mobile device and examine the effect of the same action on each device's total bandwidth requirement. Changes in  $\beta$ , e.g., change in per frame resolution of video streamed from one user to the other due to network connection changes, can be examined in a similar fashion.

In a centralised topology ( $w_2$ ), we identify  $s_5$  to be the core service and gradually increase  $\lambda_5$  from 20 to 40. Results as illustrated in the first row of Fig. 5.4 show that the increased traffic is evenly relayed to the downlink bandwidth cost of the other services (due to the service topology), and because the traffic relayed back from the leaf services are less significant (due to the communication pattern),  $M_3$  which hosts  $s_5$  does not require great increase in downlink capacity.

In a chain topology ( $w_3$ ), we identify the head service  $s_1$  to be the core service and increase  $\lambda_1$  to double its initial value. As shown in the second row of Fig. 5.4,  $s_1$  itself does not demand much extra bandwidth since its succeeding service is located on the same device ( $M_1$ ). This co-location factor also explains why only  $x_2^\uparrow$  and  $x_4^\uparrow$  is showing an increase in the first plot and  $x_3^\downarrow$

and  $x_5^\downarrow$  in the second plot. When these values are summarised per device,  $b_2$  shows the greatest increase because it has to accommodate both the increase in  $x_3^\downarrow$  and  $x_4^\uparrow$ .

#### 5.4.2 Effect of Per Service Data Size

In this set of simulations, we examine the effect of increase in per request data size (i.e.,  $\beta$ ). In practice, this can be observed when the per frame resolution of a video stream from one user to the other is changed. As shown in the third row of Fig. 5.4, as we increase  $\beta_5$ , both  $x_5^\uparrow$  and  $b_5^\uparrow$  increase as they do in the first row of Fig. 5.4. However,  $x_5^\downarrow$  remains unchanged. Furthermore, the uplink bandwidth demand of all other services and their hosts remain unchanged. This is because the increase in  $\beta_5$  does not affect the relayed uplink bandwidth of the service which is called by  $s_5$ , therefore the effect of increase in  $\beta$  is more confined within the MSON than that in  $\lambda$ . The same can be observed from the fourth row of Fig. 5.4 which illustrates the result from a chain topology (increase in  $\beta_1$ ).

#### 5.4.3 Effect of Latency

We apply a 10ms latency to the key service's host device ( $M_3$  for centralised and  $M_1$  for chain) in all simulations and compare the results with the originals. The increase in bandwidth demand as a consequence is shown in the latter three columns of Fig. 5.4. In the second and fourth row of the figure, it shows that because of the chain service topology, the effect of this added latency is almost entirely passed on to  $M_2$  which then acted as a filter to stop this effect to be passed on to  $M_3$ . When available bandwidth is capped or limited, high latency implies a lower service rate.

#### 5.4.4 Alternative Allocation Scheme

One common bandwidth allocation scheme, as an alternative scheme to our network I-O model, evenly distributes the available bandwidth to the services it hosts. As a result, the service rate of an MSON is prematurely capped by the service which requires the most amount of bandwidth as shown by  $\lambda'$  of Fig. 5.4 (zoom). It can be seen that the scheme as given by the I-O model, capped at  $\lambda''$ , realise greater potential from the MSON.

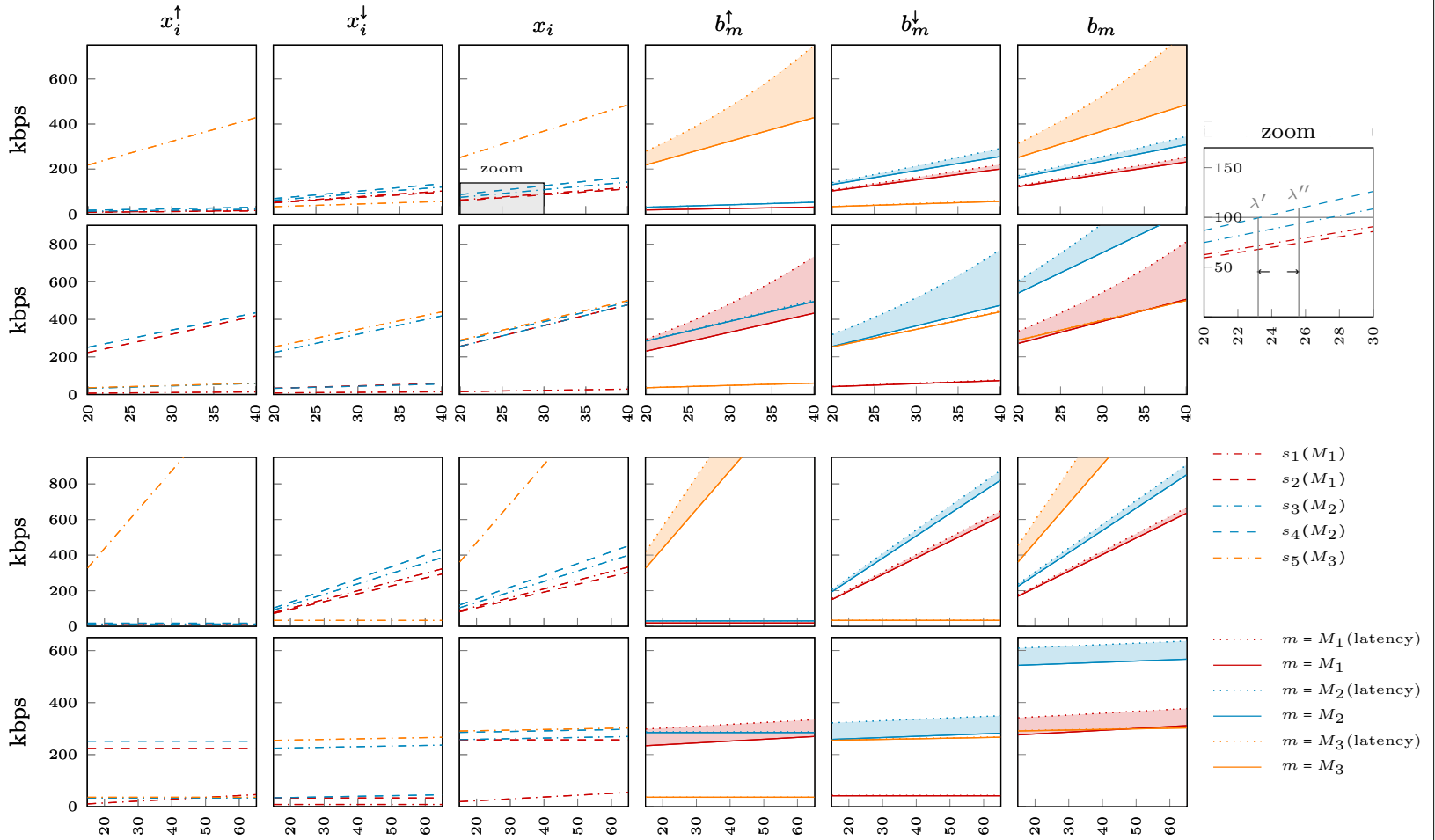


Figure 5.4: Parametric evaluation of the Network I-O model.

The increases in  $\lambda_5$  of  $w_2$ ,  $\lambda_1$  of  $w_3$ ,  $\beta_5$  of  $w_2$  and  $\beta_1$  of  $w_3$  are projected onto the x-axis of each of the four rows of plots respectively. The effects of these increases including the uplink ( $x_i^\uparrow$ ), downlink ( $x_i^\downarrow$ ) and total ( $x_i$ ) bandwidth demands of each service ( $s_1$  to  $s_5$  as in plot legends), and the uplink ( $b_m^\uparrow$ ), downlink ( $b_m^\downarrow$ ) and total ( $b_m$ ) bandwidth demands of each device ( $M_1$  to  $M_3$  as in plot legends) are presented in each of the six columns of plots respectively.

## 5.5 Cost-Based Bandwidth Allocation

In a dynamic mobile environment, a number of QoS metrics must be supported by the network in order to provide performance guarantees for mobile applications. Once we have derived the bandwidth demand vector from the network I-O model, the question arises as *How much bandwidth should the user provision when cost is considered?* In this section, we propose a cost-based method for the bandwidth allocation problem and then demonstrate the solution with a case study.

### 5.5.1 Problem Formulation

The QoS required from an MSON is diverse (e.g., throughput, loss and jitter guarantees). We assume that these QoS metrics are mapped to a utilisation threshold of each service's access network link (a technique that is commonly used by network service providers) [83], and define a utilisation threshold  $u_i$  for each  $s_i \in \mathbb{S}$ , such that QoS is ensured for  $s_i$  as long as its average link utilisation in a unit of time is no greater than  $u_i$ . Given  $s_i$  with an average bandwidth demand of  $\bar{x}_i$ , we allocate  $\bar{x}_i/u_i$  units of bandwidth to  $s_i$  to ensure its QoS. It is easy to see that a low  $u_i$  provides better QoS guarantee. However, this increased *over-provisioning* of bandwidth also brings a greater overall cost (e.g., A 4G network provides more bandwidth than 3G but is also more expensive) to the MSON. Furthermore, the QoS gain from a lower network utilisation is also capped. Therefore, a trade-off relation exists between service QoS and the overall cost.

Let  $r_i$ ,  $\kappa_i$  and  $\varphi_i$  denote the revenue (per service request), provision cost (per unit of time for reserving one unit of bandwidth) and the penalty cost (per unit bandwidth that is requested over the reserved bandwidth limit per unit of time) of  $s_i \in \mathbb{S}$  respectively<sup>1</sup>. Then we derive the total net income of the network in a unit of time ( $t$ ) as

$$V_t = \underbrace{\sum_i \lambda_i^t r_i}_{\text{revenue}} - \underbrace{\sum_i \frac{\bar{x}_i}{u_i} \kappa_i}_{\text{cost}} - \underbrace{\sum_i \max(x_i^t - \frac{\bar{x}_i}{u_i}, 0) \varphi_i}_{\text{penalty}} \quad (5.18)$$

<sup>1</sup>In practice, the revenue, cost and penalty values may be better presented as non-linear increasing functions of service request and bandwidth. The exact algorithm varies in different scenarios and technologies. We approximate the effect of these functions with increasing linear functions to keep our approach generic rather than technology-specific.

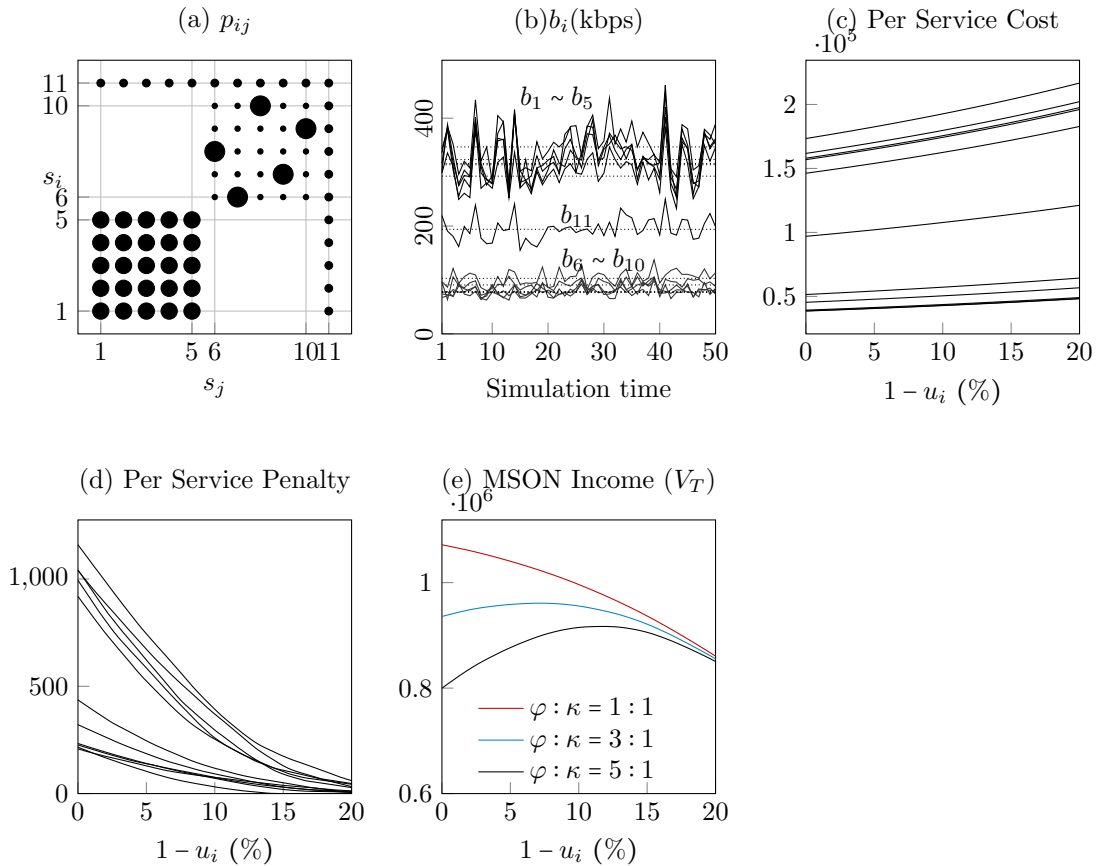
with  $\lambda_i^t$  and  $x_i^t$  denote the arrival rate and bandwidth demand of  $s_i$  in time  $t$  respectively. One underlying dependency exists between  $\lambda_i^t$  and  $x_i^t$  which is given by (5.7) and (5.5) of the network I-O model. Over a period of time  $T = \{1, 2, 3, \dots\}$ , the total income is given by  $V_T = \sum_{t \in T} V_t$ .

Observe that for a given period of time: the revenue term in (5.18) is constant because the total number of request made to the MSON is constant in the same period of time; the cost term is an increasing function when  $u_i$  is reduced because a lower link utility implies a higher provisioning cost; and the penalty term is a decreasing function when  $u_i$  is reduced because a lower link utility implies a smaller probability that  $x_i^t$  is greater than  $\bar{x}_i/u_i$ . Therefore it is easy to see that there exist an optimal  $u_i$  which balances the cost and penalty term and gives a maximum  $V_T$ . However, deriving the optimal  $u_i$  is not a straightforward task. This is largely due to the complexity in predicting the value of the penalty term. From the network I-O model, we know that  $x_i^t$  can be dependent on the arrival rate of all services in MSON which may follow different probability functions. Therefore we apply a numerical approach based on simulation to illustrate the trade-off and approximate the optimal  $u_i$  and leave further analysis to future work.

### 5.5.2 Simulation

Consider an MSON with two groups of five services ( $s_1 - s_5$  and  $s_6 - s_{10}$ ) and an administrative service ( $s_{11}$ ). This grouping setup is borrowed from the datasets presented in [103] and [104]. Assume that the services in the first group is fully connected and the services in the second group has a ring like topology, then the communication pattern of this MSON is as illustrate by Fig. 5.5(a). We set both  $\lambda$  and  $\beta$  to follow normal distributions with means of 20 and 10 Kb respectively.

We model the arrival process of each service's request as a Poisson process (note that a different probability function or a different mixture of probability functions would produce different results) with mean  $\lambda_i$ , and conduct simulations that last 50 units of time,  $T = \{1, 2, \dots, 50\}$  as shown in Fig. 5.5(b). Although each service has an independent arrival process, the fluctuations of bandwidth demand (as illustrated by the solid lines plotted in Fig. 5.5(b)) have great similarities within each group of services. This is because each service's bandwidth demand is not only generated by its own arrival rate, but also by that of the others', especially by the services

Figure 5.5: Effect of reductions in utilisation threshold ( $u_i$ ).

with which it frequently communicates. This result also verifies our classification of bandwidth demands in Section 5.3. Each value in vector  $\lambda$  is plotted as a dotted lines in Fig. 5.5(b).

As we reduce  $u_i$  from 100% to 80%, the bandwidth provision cost increases as shown in Fig. 5.5(c), and the penalty is gradually reduced as shown in Fig. 5.5(d). Note that each service has a line in both plots, this may not be clearly seen from the figures because of overlaps.

Combine the results from Fig. 5.5(c) and Fig. 5.5(d) together with a constant revenue value we get the trade-off curve between  $u_i$  and  $V_T$  as illustrated in Fig. 5.5(e). We conduct this simulation with different penalty to cost ratios because a higher penalty to cost ratio implies a greater incentive to reduce the penalty (i.e., a lower  $u_i$ ). Each curve clearly has its own peak (optimal value for  $u_i$ ). In the case of the 1:1 ratio, there is no incentive for a  $u_i$  lower than 1 because of the relatively low penalty to cost ratio. Note that the data presented in Fig. 5.5(b-d)

have  $\varphi : \kappa = 5 : 1$ . In reality,  $\varphi$  can be seen as a temporary purchase of bandwidth from the network provider, or as an additional bandwidth usage that is above the contracted limit. In such situations, it is common to have a higher bandwidth price than that of long term contracts ( $\kappa$ ).

## 5.6 Adaptive Bandwidth Allocation

Wireless connections are prone to change. Mobile smart devices frequently travel from areas covered by one network to that of another (e.g., switch from WiFi to 3G). The differences between each network's capacity not only affect the QoS of the services that are locally hosted on that device, but also affect that of the remote services which require communication with these local services. In this section, we consider the arrival rates of the service requests that the service can accommodate as the key QoS metric of the MSON and investigate the impact of the reduction in bandwidth with different adaptive strategies.

### 5.6.1 Problem Formulation

Assuming that the physical bandwidth of device  $\mu$  is reduced from  $B_\mu$  to  $\check{B}_\mu$ , with  $\check{B}_\mu < b_\mu < B_\mu$ . Then if  $\Theta$ ,  $P$  and  $\beta$  are to remain the same, reduction has to be made in  $\lambda$  (with  $\lambda_i$  as in (5.5)), i.e., the arrival rate of some or all of the services in  $\mathbb{S}$  is to be reduced. Let  $\lambda$  and  $\check{\lambda}$  denote the arrival rate vector of  $\mathbb{S}$  before and after the reduction respectively. Then one intuitive objective<sup>2</sup> when adapting to this reduction is to find a new bandwidth allocation vector  $\check{\mathbf{x}} = \check{\mathbf{x}}^\uparrow + \check{\mathbf{x}}^\downarrow$  for  $\mathbb{S}$  which minimises the difference between  $\lambda$  and  $\check{\lambda}$ , i.e., share the impact of the reduction with minimal deviation from the original state. Formally, we formulate this problem as a constrained

<sup>2</sup>This objective often derives from the nature of the mobile applications and can vary significantly between different types of MSONs. For instance, the changes in  $\lambda$  can be weighted if each service has a priority assigned to it. Furthermore, a cost-based function similar to (5.18) as given in Section 5.5 can also be used as an objective function to the problem. We use (5.19) in our demonstration for its brevity and intuitiveness.



least squares problem:

$$\min_{\tilde{\lambda}} \|\lambda - \tilde{\lambda}\|_2 \quad (5.19)$$

$$\text{s.t. } (A^\dagger - I)\tilde{\mathbf{x}}^\dagger + \tilde{\lambda} \circ \beta \circ \rho = \mathbf{0} \quad (5.20)$$

$$A^\downarrow \tilde{\mathbf{x}}^\dagger - \tilde{\mathbf{x}}^\downarrow = \mathbf{0} \quad (5.21)$$

$$\check{b}_m = \sum_i \check{x}_i^\dagger + \sum_i \check{x}_i^\downarrow, \quad \Theta(s_i) = m \quad (5.22)$$

$$\check{b}_m \leq b_m, \quad m \in \{\mathbb{M} - \mu\} \quad (5.23)$$

$$\check{b}_\mu \leq \check{B}_\mu \quad (5.24)$$

(5.19) states our objective which is to minimise the deviation from the original state. The first set of equality constraints, ensuring the dependency between  $\tilde{\lambda}$  and  $\tilde{\mathbf{x}}^\dagger$ , are given by (5.20) which is derived from (5.5) and (5.7) with  $\circ$  denotes the Hadamard product (also known as the element wise product) of two vectors. The second set of equality constraints as given by (5.21) ensures the dependency between  $\tilde{\mathbf{x}}^\dagger$  and  $\tilde{\mathbf{x}}^\downarrow$ . Recall from (5.14) that  $b_m$  denotes the total bandwidth cost of  $m \in \mathbb{M}$  given  $\mathbf{x}^\dagger$ ,  $\mathbf{x}^\downarrow$  and  $\Theta$ . Let  $\check{b}_m$  denote the new bandwidth cost of  $m$  by summarising  $\tilde{\mathbf{x}}^\dagger$ ,  $\tilde{\mathbf{x}}^\downarrow$  using (5.14), we have (5.22). (5.23) states that the new bandwidth cost should not be greater than the original values. (5.24) ensures that the new bandwidth cost for  $\mu$  cannot be greater than  $\check{B}_\mu$  which denotes the current (after reduction) physical (available) bandwidth of  $\mu$ .

The adaptive strategy as given by (5.19) to (5.24) describes a scenario in which all devices are cooperative towards the reduction in one device's bandwidth availability and are willing to reduce the service rate of the services it hosts. However, this universal unselfishness might not be true. Therefore, as a comparison, we give two alternative adaptive strategies to present varying degrees of selfishness.

In the first alternative strategy,  $\mu$  keeps the arrival rates of its own services (i.e., workflows initiated by  $\mu$ ) unchanged and rejects only requests received from other devices. We denote this alternative solution with  $\tilde{\lambda}'$ , then the set of constraints for this adaptive problem, in addition to

(5.21)-(5.24), becomes

$$\text{s.t. } (A^\dagger - I)\tilde{\mathbf{x}}^\dagger + \tilde{\lambda}' \circ \beta \circ \rho = \mathbf{0} \quad (5.25)$$

$$\tilde{\lambda}'_i = \lambda_i, \quad \Theta(s_i) = \mu \quad (5.26)$$

In practice, this strategy is equivalent to a device which priorities requests of its owner, and when bandwidth is limited, choose to reject requests sent by remote services first.

In contrast, a second alternative strategy keeps all service arrival rates unchanged except those hosted by  $\mu$ . We denote this alternative solution with  $\tilde{\lambda}''$ , then the set of constraints for this adaptive problem, in addition to (5.21)-(5.24), becomes

$$\text{s.t. } (A^\dagger - I)\tilde{\mathbf{x}}^\dagger + \tilde{\lambda}'' \circ \beta \circ \rho = \mathbf{0} \quad (5.27)$$

$$\tilde{\lambda}''_i = \lambda_i, \quad \Theta(s_i) \neq \mu \quad (5.28)$$

In practice, this strategy is equivalent to a scenario in which the device which suffered a loss of bandwidth would be forced to prioritise requests arrived from other devices and delay the requests generated locally. Note that a positive solution to the adaptive problem is not guaranteed (e.g., when the physical bandwidth of  $\mu$  is very small).

## 5.6.2 Simulation

### Basic Set

Our first set of simulations is based on the four scenarios presented in Fig. 5.3. We first set the initial arrival rates of all five services at 20 and calculate the initial bandwidth demand vector  $\mathbf{x}$  accordingly. We assume each service is assigned to a unique device for illustration purposes and select the host of  $s_3$  ( $s_8$  in the latter two cases) to be  $\mu$  and reduce  $b_3$  by 10%, i.e.,  $\tilde{B}_\mu = b_\mu * 90\% = x_3 * 90\%$ . We then apply all three adaptive strategies and solve each problem as a linear program. Results from this set of experiments are presented in Fig. 5.6 (a-d). In each plot, there are three groups of five vertical bars representing the arrival rate of each of the five services given by the three adaptive strategies.

From Fig. 5.6 (a), we see that being the core service  $s_5$  is affected the most in both  $\check{\lambda}$  and  $\check{\lambda}'$ . Because of their relative independent status from each other in a centralised setting, the selfish behaviour of  $s_3$  in  $\check{\lambda}'$  has relatively small impact towards  $s_1$ ,  $s_2$  and  $s_4$ . In Fig. 5.6 (b) when  $s_3$  is selfish, both  $\lambda_1$  and  $\lambda_2$  have to be reduced. Without the cooperation from other services,  $\check{\lambda}_3''$  is greatly reduced from its original value. From Fig. 5.6 (c) we see that  $s_9$  is affected when  $s_8$  is being selfish (recall that  $p_{98}$  is of relatively high value in  $w_4$  of Fig. 5.3), whereas in Fig. 5.6 (d) this effect is also observed from  $s_6$  and  $s_7$  because the topology in the latter entails a more equal status among the services. In all four cases, when the other services are non-cooperative,  $\lambda_3$  has the greatest reductions. Note that in Fig. 5.6 (c & d), because  $s_{10}$  is very loosely connected to the others,  $\lambda_{10}$  is not significantly affected in all cases.

### Random Set

In the second set of simulation, we construct a case of larger scale containing 51 services which while has a similar two group plus a administrator pattern similar to that of Fig. 5.5 (a), contains 25 services in each of the two main groups. Both groups are of a fully connected topology. We also assume that each service is assigned to a unique device as in the first set of simulations and that  $b_3$  is reduced by 90%. We set the mean arrival rate (normal distribution) of all services in the first group (which includes  $s_3$ ) and the second group at 10 and 15 respectively (in order to distinguish the data from the two groups when plotted), and give 6 as the arrival rate of the administrator service  $s_{51}$ . Fig. 5.6 (e) gives a comparison between  $\lambda$ ,  $\check{\lambda}$  and  $\check{\lambda}'$  (we omit results from the second alternative strategy,  $\check{\lambda}''$ , because in this scenario it is difficult for the host device to accommodate all bandwidth reductions on its own, therefore a positive solution is unlikely).

In Fig. 5.6 (e), the set of circles along the diagonal indicates the values of original service arrival rates. The deviation of the set of + marks from the diagonal indicates a reduction in arrival rates according to the fully cooperative adaptive strategy ( $\check{\lambda}$ ). The set of x further deviates from the diagonal indicating further reductions made in service arrival rates according to the selfish adaptive strategy ( $\check{\lambda}'$ ). The communication pattern can be clearly seen from the data presented in Fig. 5.6 (e). Because the reduction is made in  $s_3$  which belongs to the first group (group A in Fig. 5.6 (e)), the arrival rate of the second communication group is not affected. Reduction is made in the administrator  $s_{51}$  because of its communication with  $s_3$ . The selfish

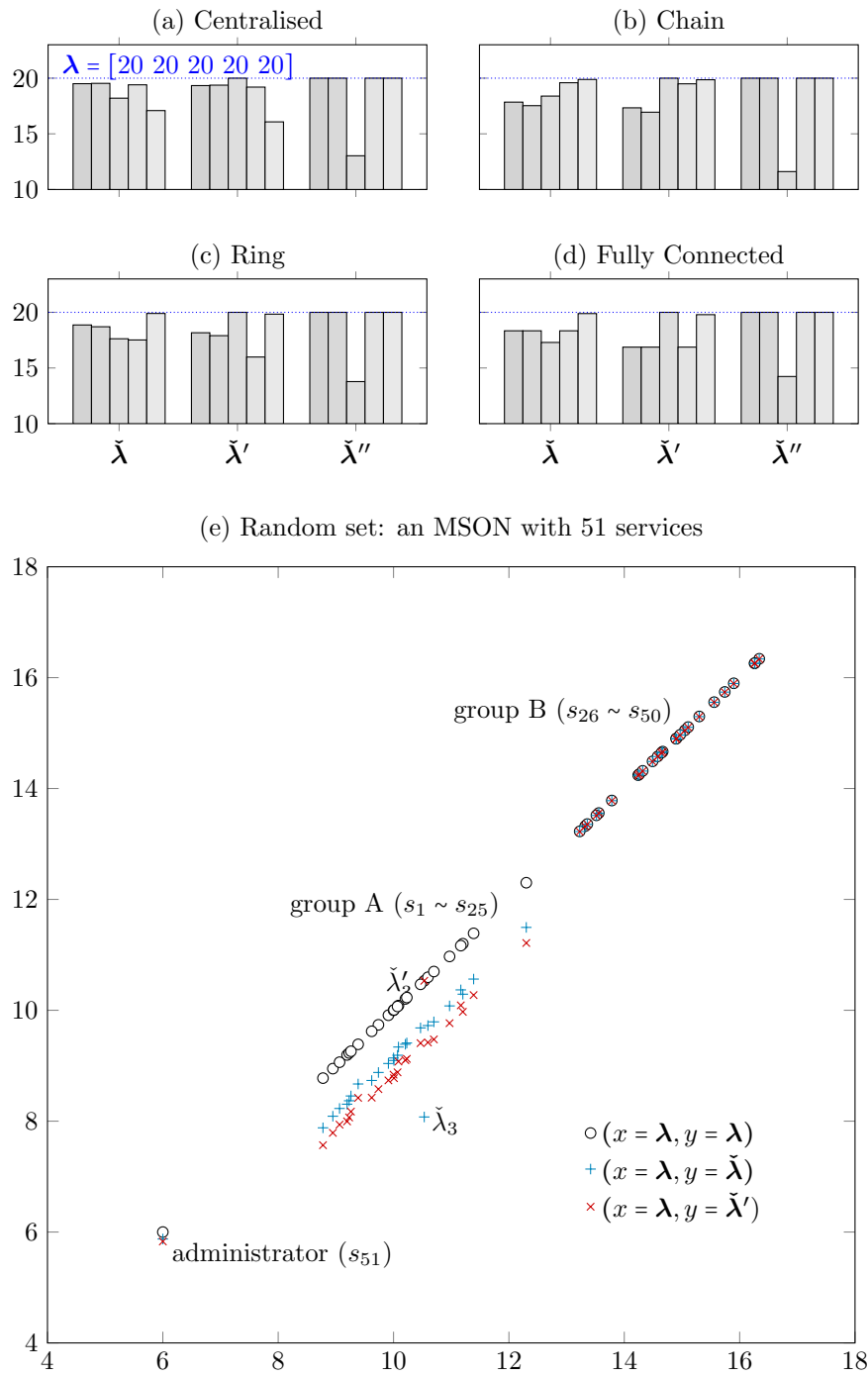


Figure 5.6: Comparison of adaptive strategies.

strategy sees  $\check{\lambda}'_3$  positioned along the diagonal in contrast to the reduction seen in  $\check{\lambda}_3$ .

## 5.7 Summary

Besides energy, bandwidth is another critical factor limiting the QoS of mobile services and applications. This significance is further amplified in mobile cloud computing settings where application workflows are to be executed over a group of mobile devices and clouds the communications between which are purely reliant on the conditions of the underlying wireless network (e.g. MSON).

In this chapter, we assume mobile application workflows are manifested dynamically as compositions of mobile services. In order to describe the bandwidth dependencies between services, we proposed a novel network I-O model which extends on the original framework proposed in the Leontief Input-Output analysis in economics. In the construction of our network I-O model, we interpreted a number of factors of an MSON including bandwidth, latency and service arrival rate to describe the underlying structure of mobile network economies. A set of equilibrium equations are derived to produce the bandwidth requirements of mobile devices. A series of parametric studies is carried out to validate and demonstrate the dynamics of the proposed network I-O model.

Based on the proposed network I-O model, we further developed bandwidth allocation schemes that are applicable to two common optimisation objectives of MSONs. We selected two objective applications of our network I-O model to cover the cases of static as well as dynamic conditions of MSONs. The maximisation of service benefit is considered as the first objective in which the trade-off between application QoS and bandwidth cost is demonstrated. In our second application of the network I-O model, a set of adaptive bandwidth allocation schemes are formulated and compared to demonstrate how service QoS is affected by the changing conditions of the wireless network. The effect of cooperative and uncooperative status of the mobile devices is also demonstrated.

Our network I-O model is not only applicable to mobile cloud computing scenarios, but also lays the foundation for further objective developments mobile networks in general.

## Chapter 6

# Rethinking the Offload Decision

## Models in Mobile Cloud

## Application Ecosystems

With the increasing popularity and maturity of technologies such as HTML5 and JavaScript, mobile cloud computing as a new design paradigm of mobile application developments is becoming increasingly accessible to the developers of mobile applications. With this increase in popularity, multiple mobile cloud applications will reside on the same device in the near future, and they will be competing for the limited resources available on a mobile device. Furthermore, even if there is only one mobile cloud application installed on the device, it still cannot ignore the existence of other standard (native or remote) applications that are also installed on the same mobile device. This competition for resources affects an application's perception of the mobile cloud platform and its offload decision making process. However this competition is not yet considered in existing researches of mobile cloud computing.

In the final contribution of this thesis, we take this potential competition into account and examine how it affects the behaviour of mobile cloud applications. We look at the mobile cloud computing platform from a per-device perspective and develop a theoretical framework to analyse

and compare the outcomes of different offload decision models. One of our prime contribution in this chapter is the game theoretical model we constructed for the offload game in Section 6.3, and the derivation of the mixed-strategy Nash equilibrium of the game that follows.

In Section 6.1, we give more detailed context of a mobile cloud application ecosystem and outline the problem statement, objective and contribution of this chapter.

## 6.1 Mobile Cloud Application Ecosystems

A mobile cloud application as we discuss in this chapter is an application whose main functionality may be executed independently on either a mobile device or a cloud server. This means that the application is able to offload or migrate itself seamlessly between the two platforms. This offload decision is often taken at runtime according to the current network condition and the anticipated workload size [22, 75], as we have demonstrated in Chapter 3 and 4. We refer to this class of applications as *Hybrids* as opposed to *Native* and *Remote* to distinguish applications by their designated execution platform.

It is important to note that our use of these three terms, hybrid, native and remote, refers to the place of execution of the application's main functionality and especially its ability to seamlessly offload or migrate between mobile devices and cloud, rather than the traditional usage of these terms in mobile application developments where they refer to the environment it is developed in. Traditionally for an application developer, when an application is written in a native language like Objective-C for iOS devices or Java for Android devices, it is referred to as a native application; an application that is run on a web server (cloud back end) and delivered to the user via a browser is referred to as a remote mobile web application. A hybrid application in this sense is a crossover between these two approaches. The majority of a hybrid application's code is usually written in HTML5 and JavaScript and rendered by the device's web engine, so the code is portable between platforms. A hybrid application also include native codes to refine user experience and get access to a wider range of device functionalities. This code portability is an attractive option for the development of mobile cloud applications. However, a hybrid application as in mobile cloud computing is more intelligent in utilising different platforms at runtime.

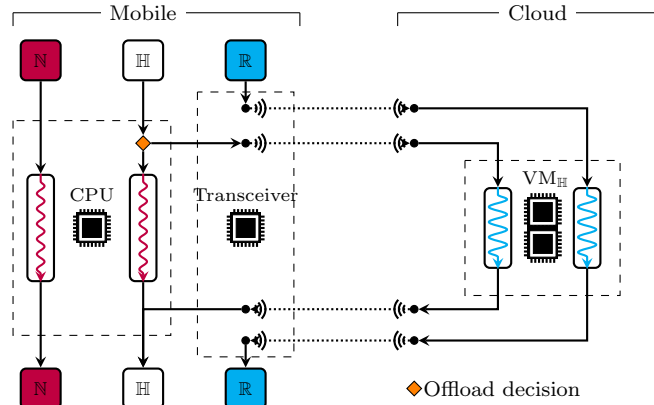


Figure 6.1: A mobile cloud application ecosystem

In order to qualify as a hybrid application as we discuss in this chapter, the application need not only be deployable to different platforms, but also make offload decisions at runtime to improve user experience. The code portability of a hybrid mobile cloud application also need not be limited to the use of HTML5, MAUI [22] is written in C# for Microsoft's .NET Common Language Runtime, CloneCloud [23] modified the Dalvik VM for code migration on Android OS, ThinkAir [26] builds its offload platform with a modified version of Android x86. A more recent work [105] utilises a modified version of WebKit to support the offload of HTML5 workers.

### 6.1.1 Problem Statement

With the increasing popularity of mobile cloud applications come one problem currently missing from the researches of mobile cloud computing which is the recognition of the competition for resources between applications on mobile devices.

Applications are selfish entities each aims to maximise its own performance. Notwithstanding the cooperative interactions that may exist within certain application workflows, given a host device, each application's performance is proportional to the exclusivity it has over its host's resources. Therefore the competition for resources underlies each community of applications that lives on the same computing device. Recognising the existence of this competition is especially important for the applications that are hosted by resource constrained mobile devices.

With the popularity of mobile applications (or apps in short), the real estate of a mobile



device has already been heavily competed on by the many apps that are currently installed on each device. According to the data published by Google's Our Mobile Planet report [106] for 2013, on average 28.5 apps are installed on each smartphone in the UK which is just above the overall average (26) among the 47 countries included in the survey. In South Korea and Switzerland this number is higher at 40 apps per smartphone. A similar figure is reported by Nielsen in their early 2014 report [107] which includes both Android and iOS users. [108] report an average number of 177 apps installed on their participant's android devices.

We illustrate the resource competition in a mobile cloud application ecosystem with Fig. 6.1. Three classes of applications share the same mobile device. A wireless connection is established to a remote cloud service supporting computation offload<sup>1</sup>. The main functionality of a native application is carried out on the local CPU, whereas a remote application carry out the majority of its computation via cloud services. To access a cloud service, data is sent via the transceiver of equipped on the mobile device. A hybrid application has the ability to choose between the two platforms. Its offload decision precedes the execution of its main functionality.

Competition of resources comes with either options for a hybrid application. The path of native execution is shared with other native applications at the CPU, whereas the path of remote execution is firstly bottlenecked at the transceiver, and consequently congested at the supporting cloud server. This competition is apparent between hybrids and other two classes of applications, but more importantly it exists within the hybrid class itself.

Existing researches in mobile cloud computing focus on the application's ability to offload computation between mobile and cloud. Offload decisions in existing work are based on the device's parameters without taking into account that it may not be the only application that's using these resources (i.e. the processing unit and the wireless data connection). This uninformed decision making process means that the offload decision made may not be as beneficial as predicated.

Table 6.1: An example showing the effect of different offload decisions

$i \in [s]^H$	Scenario A		Scenario B		Scenario C	
	N	R	N	R	N	R
$i = 1$	15	(10)	(15)	10	15	(10)
$i = 2$	18	(15)	18	(15)	(18)	15
Cost/Platform	0	(25)	15	15	(18)	15
Social Cost		25		15		18

### Example

We demonstrate the effect of an uninformed offload decision with a simple example as shown in Table 6.1. We assume three scenarios where two hybrid applications share a device. Each number in the table represents the amount of time it takes the application to run on a platform assuming exclusive usage of the device’s resources, in seconds. A circle represents the decision made by the application. Scenario A is a typical example of applications making uninformed decisions. Both applications assume that it is the only application running on the device and the cost comparison between the two platforms means both applications prefer to execute on the cloud. This makes  $\mathbb{R}$  congested while leaving the  $\mathbb{N}$  vacant. The total cost on  $\mathbb{R}$  is 25 seconds compared to the cost of 0 on  $\mathbb{N}$ .

From the user’s point of view, the makespan (i.e. social cost as we discuss in detail in Section 6.3.4 and Section 6.4) of the system as a whole is 25 seconds. This social cost is higher than either of the other scenarios where the applications’ choice of platform are split between  $\mathbb{N}$  and  $\mathbb{R}$ .

From each application’s point of view, in A, if applications’ sub tasks are scheduled in a round-robin way on  $\mathbb{R}$ , the expected time costs for both applications are 25 seconds; if the scheduling order is randomly chosen between the two applications as a whole, the expected cost is 17.5 seconds for  $i = 1$ , and 20 seconds for  $i = 2$ , all higher than the cost if it were run on  $\mathbb{N}$ .

<sup>1</sup>A remote application does not have to run on the same cloud server as the hybrid applications. A proprietary application (e.g. Facebook or Twitter) is usually supported by its own servers. Furthermore, a proprietary server is also unlikely to accept offload requests from a personal device. For these type of remote applications, we set  $w_i^b$  to be zero in our model since they don’t consume the computation resources on our cloud.

### 6.1.2 Objective and Contribution

In this chapter, we model each of the three offload decision models that applies to a mobile cloud computing scenario in Section 6.2, 6.3 and 6.4. We especially focus on the game theoretical modelling of the offload game with complete information. We derive the mixed-strategy Nash equilibrium of the game and its social cost at equilibrium in Section 6.3. The derivation of the Nash equilibrium is significant that it provides a basis for measuring the distance (referred to as the “price of anarchy” of the game which we introduce in Section 6.3.4) between a non-cooperative and a cooperative application ecosystem. With the model we present in Section 6.3, we also extend the classic load balancing game [109] which has been highly cited since its publication. Comprehensive simulation experiments has been conducted and presented in Section 6.5. Results from the comparisons between the three models provide us with a rare insight into the behaviours of applications within a community (ecosystem).

The impact and future direction of this chapter is in two folds. First, from the user’s perspective, we provide a suite of modelling tools to quantify the costs and benefits of different offload decision making processes so that an informed decision can be made on a global level. Our results pave the way for future development of manager services of hybrid applications on the device to provide a cooperative environment. Second, from a hybrid application’s point of view, in absence of a cooperative mechanism, it is able to derive an offload strategy that’s most beneficial to itself.

### 6.1.3 System Notations

To describe a mobile cloud application ecosystem, we assume a set of  $n$  independent applications sharing the same mobile device, denoted  $[n] = \{1, \dots, n\}$ . Each application  $i \in [n]$  is to choose between two parallel execution platforms, which we refer to as the remote cloud  $\mathbb{R}$  and the native processing unit  $\mathbb{N}$  in our model, in order to minimise its execution time cost.

Let  $a_i^j$  be a binary variable indicating  $i$ ’s decision to execute on platform  $j$ . All  $a_i^j$  together constitute an assignment  $A : [n] \rightarrow \{\mathbb{R}, \mathbb{N}\}$  with  $A(i)$  denotes the chosen platform for  $i$ . The weight of each application  $i$  has two components:

$w_i^d$ , denotes the size of the **d**ata that is to be transmitted over the wireless network if appli-

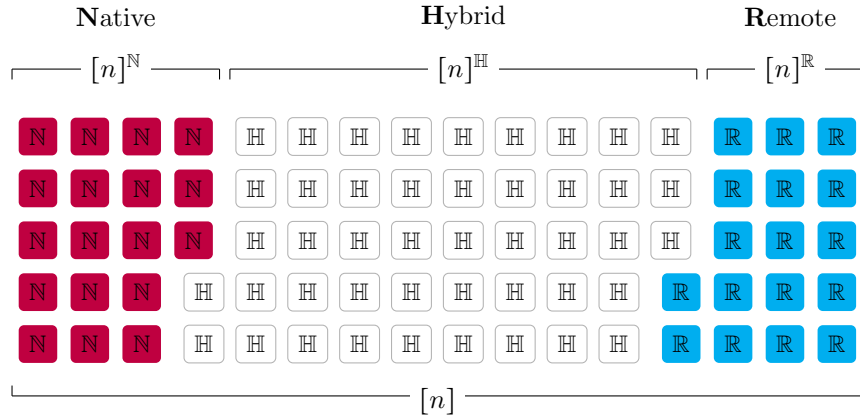


Figure 6.2: Application composition of a mobile cloud application ecosystem.

cation  $i$  is offloaded to  $\mathbb{R}$ ,

$w_i^b$ , denotes the amount of computation binary that is associated with  $i$ .

In correspondence, the speed in which each platform  $j \in \{\mathbb{R}, \mathbb{N}\}$  can process an application also consists of two components:

$s_j^d$ , denotes the data transmission speed<sup>2</sup> to  $j$ , with  $s_{\mathbb{N}}^d = inf$  and  $s_{\mathbb{R}}^d =$  bandwidth between  $\mathbb{N}$  and  $\mathbb{R}$ ,

$s_j^b$ , denotes the computation speed of  $j$ 's processing unit, we assume  $s_{\mathbb{N}}^b < s_{\mathbb{R}}^b$ .

Not all mobile applications in  $[n]$  has the ability to migrate between  $\mathbb{N}$  and  $\mathbb{R}$ . Some are fixed to run natively (locally), whereas some may rely on an active data connection to run remotely. To represent this distinction within  $[n]$ , we divide  $[n]$  into three distinct subsets:

$[n]^N$ , for native applications fixed to run on  $\mathbb{N}$ ,

$[n]^R$ , for remote applications fixed to run on  $\mathbb{R}$ ,

$[n]^H$ , for hybrid (mobile cloud) applications that may run on either  $\mathbb{N}$  or  $\mathbb{R}$ .

This composition of applications is illustrated by Fig. 6.2.

<sup>2</sup>When an application is run on the local device, we assume that the speed at which its binary reaches the processor is infinite. This way we keep the equations generic, and we don't have to add an indicator variable inside the subsequent equations (e.g. (6.2)).

Note that we use subscripts for applications and superscripts for platforms when a variable is associated with both sets. With these notations, we first derive the classic offload decision model.

In a mobile cloud computing scenario, applications have the option to either execute locally on its host device or offload and execute remotely on a supporting cloud platform. The application must estimate the cost and benefit of an offload action prior to making a decision. Depending on how much information this application has of other applications running on the same device, this decision making process may yield different results. In the next three sections (Section 6.2, Section 6.3 and Section 6.4), we formulate the different decision models of a mobile cloud application ecosystem.

## 6.2 Offload with Symmetrically Incomplete Information

In this scenario, each application  $i$  knows the properties of both platforms  $(s_j^d, s_j^b, j \in \mathbb{N}, \mathbb{R})$  and of its own task  $(w_i^d, w_i^b)$ , but is unaware of the other applications who also share the resources provided by the same device. Due to this limitation, exclusive usage of the device's data connection and processor is assumed by all applications. Hence the offload decision of  $i$  is given by

$$a_i^{\mathbb{R}} = \begin{cases} 1, & \text{If } \frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b} < \frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b} \\ 0, & \text{Otherwise.} \end{cases} \quad (6.1)$$

with  $a_i^{\mathbb{N}} = 1 - a_i^{\mathbb{R}}$ .

Depending on the capacity of the device's wireless data connection  $(s_{\mathbb{R}}^d)$ , the benefit of remote execution (i.e. reduced execution time, given by  $w_i^b/s_{\mathbb{N}}^b - w_i^b/s_{\mathbb{R}}^b$ ) may be offset by the additional communication cost (between the device and the cloud, given by  $w_i^d/s_{\mathbb{R}}^d$ ) when applications are run on or offloaded to the cloud. Therefore mobile cloud applications often requires that the data connection speed between  $\mathbb{N}$  and  $\mathbb{R}$  to be greater than a certain threshold before an offload action is considered [5, 75]. The capacity of the device's wireless data connection greatly influence the decision making process of offload-able applications.

There are two potential flaws in this offload decision model. First, the wireless connection may be occupied by other applications, which means that the actual data transmission cost is greater than  $w_i^d/s_{\mathbb{R}}^d$ . Second, the local processing unit is also shared with other applications, hence the cost of local execution  $w_i^b/s_{\mathbb{N}}^b$  and therefore the benefit of remote execution as given by  $w_i^b/s_{\mathbb{N}}^b - w_i^b/s_{\mathbb{R}}^b$  are also under-estimated. Both flaws are direct results of the incomplete information given to each application.

To complete the notation of this section, we denote  $\Theta_B$  to represent the social cost of the system under the symmetrically incomplete information decision model. We further discuss the definition of social costs in Section 6.3.4 and Section 6.4. The ‘‘B’’ in this notation comes from the fact that the wireless data bandwidth plays a crucial role in this decision model. Next, we derive the decision model when applications are given complete information of other applications.

### 6.3 Offload with Complete Information

We now consider the scenario in which all applications are given complete information of the weights<sup>3</sup> of all other applications ( $w_i^b, w_i^d, i \in [n]$ ). Given an assignment  $A : [n] \rightarrow \{\mathbb{R}, \mathbb{N}\}$ , the cost (time delay) for application  $i$  is given by

$$c_i = \sum_{\substack{k \in [n] \\ A(k)=A(i)}} \left( \frac{w_k^d}{s_{A(k)}^d} + \frac{w_k^b}{s_{A(k)}^b} \right) \quad (6.2)$$

which assumes that no priority is assigned to any application. That is to say that both data packets over the data connection and instructions in the processor stack are scheduled in a round-robin way. Following this, the cost of platform  $j$  is given by

$$C_j = \sum_{\substack{i \in [n] \\ A(i)=j}} \left( \frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) \quad (6.3)$$

(6.2) and (6.3) together correct the inaccuracy caused by incomplete information.

---

<sup>3</sup>In practical terms, the weights of an application can be predicted based on its historic profiles as done in [108].

### 6.3.1 The Offload Game

It is easy to see that the decision of each application is directly influenced by the decisions made by others. Since each application's goal is to minimise its own cost, the offload decision model with complete information can be described by a non-cooperative game theoretic framework.

In this game, which we refer to as the *offload game*, each application is an agent (player) whose objective is to minimise  $c_i$ . Each application has a strategy profile of  $\{\mathbb{N}, \mathbb{R}\}$ . A collection of pure strategies of all applications  $i \in [n]$  constitutes an assignment  $A$ . A *mixed strategy*<sup>4</sup> is a probability distribution over the set of pure strategies  $\{\mathbb{N}, \mathbb{R}\}$ .

### 6.3.2 Mixed Strategies and Expected Costs

We first denote the probability that agent  $i$  choose to run on platform  $j$  with  $p_i^j = \mathbb{P}[A(i) = j]$ . Then the expected cost of platform  $j$  under the *strategy profile*  $P = \{p_i^j, i \in [n], j \in \{\mathbb{N}, \mathbb{R}\}\}$  is

$$\mathbb{E}[C_j] = \sum_{i \in [n]} p_i^j \left( \frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right). \quad (6.4)$$

For application  $i$ , its expected cost when selecting  $j$  is

$$\mathbb{E}[c_i^j] = \frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} + \sum_{\substack{k \in [n] \\ k \neq i}} p_k^j \left( \frac{w_k^d}{s_j^d} + \frac{w_k^b}{s_j^b} \right). \quad (6.5)$$

This together with (6.4), we have

$$\mathbb{E}[c_i^j] = \mathbb{E}[C_j] + (1 - p_i^j) \left( \frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) \quad (6.6)$$

---

<sup>4</sup>We consider mixed strategies rather than pure strategies because it is a better match to the mobile cloud computing scenario. First, in a game, there may be multiple (or none as in the rock-paper-scissors game) pure strategy equilibria, including the optimal assignment which we derive in Section 6.4. To reach a pure strategy equilibrium, the order in which each agent is given the right to make a strategy decision affects which pure strategy equilibrium the system would reach. In our mobile cloud scenario, the mobile OS does not explicitly define this order, and it also wouldn't be fair for the OS to do so without user consent. Second, beside the saving in execution time, hybrid applications can also provide the user with higher quality service when it is run on a remote cloud as seen in [24]. Therefore, the user may opt for a remote execution regardless. Therefore, only a probability of an application's pure strategy can be observed. Because of these reasons a pure strategy profile is not a stable representation of our offload game.

On the contrary, a mixed strategy profile only requires that each application is aware of the probability of others' offload decisions. The mobile OS has this information readily available from its network access log, and is able to share this with all applications.

which derives

$$p_i^j \left( \frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) = \mathbb{E}[C_j] - \mathbb{E}[c_i^j] + \left( \frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) \quad (6.7)$$

and further derives

$$p_i^j = \left( \mathbb{E}[C_j] - \mathbb{E}[c_i^j] + \left( \frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) \right) / \left( \frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) \quad (6.8)$$

which gives all applications' mixed strategies as a function of  $\mathbb{E}[C_j]$  and  $\mathbb{E}[c_i^j]$  and constitutes  $P$ .

### 6.3.3 Nash Equilibrium

We now describe the Nash equilibrium of this game. A game is said to be in Nash equilibrium when no agent (application  $i$ ) of the game, with complete knowledge of all other agents' strategies ( $P$ ), is able to make gains or reduce its cost by unilateral actions. Not all strategy profiles define a Nash equilibrium. In order to find the  $P$  which defines a Nash equilibrium, further constraints is to be added to (6.8).

First, in a Nash equilibrium, each application agent only assign non-zero probabilities to platform  $j$  if

$$\mathbb{E}[c_i] = \mathbb{E}[c_i^j] = \min_{j \in \{\mathbb{N}, \mathbb{R}\}} \mathbb{E}[c_i^j], \quad i \in [n]. \quad (6.9)$$

We define a support indicator

$$\alpha_i^j = \begin{cases} 1, & \text{if } p_i^j > 0 \\ 0, & \text{otherwise.} \end{cases} \quad (6.10)$$

Take (6.7) into (6.4) with the introduction of  $\alpha_i^j$  and (6.9), we get

$$\mathbb{E}[C_j] = \sum_{i \in [n]} \alpha_i^j \left( \mathbb{E}[C_j] - \mathbb{E}[c_i] + \left( \frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) \right) \quad (6.11)$$



for  $j \in \{\mathbb{N}, \mathbb{R}\}$ .

Second, each application  $i$  should distribute all of its weight completely, that is

$$\sum_{j \in \{\mathbb{N}, \mathbb{R}\}} p_i^j = 1, \quad i \in [n]. \quad (6.12)$$

Take (6.8) into (6.12) with the introduction of  $\alpha_i^j$  and we get

$$\sum_{j \in \{\mathbb{N}, \mathbb{R}\}} \alpha_i^j \left( \mathbb{E}[C_j] - \mathbb{E}[c_i] + \left( \frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) \right) = \left( \frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) \quad (6.13)$$

for  $i \in [n]$ .

Observe that (6.11) and (6.13) together have  $n + 2$  variables ( $\mathbb{E}[C_j]$  and  $\mathbb{E}[c_i]$ ) and  $n + 2$  equations, meaning that a unique solution is defined. Therefore, the strategy profile of the Nash equilibrium of our offload game is completely defined by (6.8), (6.11) and (6.13). We further give the solution of  $p_i^{\mathbb{R}}$  as

$$\begin{aligned} p_i^{\mathbb{R}} = & \left( \frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b} \right) / \left( \frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b} + \frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b} \right) \\ & + \left( C_{\mathbb{R}}^f - C_{\mathbb{N}}^f + \sum_{k \in [n]^{\mathbb{H}}} \left( \frac{w_k^d}{s_{\mathbb{R}}^d} + \frac{w_k^b}{s_{\mathbb{R}}^b} \right) - \left( \frac{w_k^d}{s_{\mathbb{N}}^d} + \frac{w_k^b}{s_{\mathbb{N}}^b} \right) \right) / \left( \left( 1 - |[n]^{\mathbb{H}}| \right) \left( \frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b} + \frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b} \right) \right) \end{aligned} \quad (6.14)$$

with  $C_{\mathbb{R}}^f$  and  $C_{\mathbb{N}}^f$  denote the cost from  $[n]^{\mathbb{R}}$  and  $[n]^{\mathbb{N}}$  respectively. The corresponding derivation is attached in Appendix C.

### 6.3.4 Social Cost and Price of Anarchy

So far we have been looking at the costs from each application's perspective. Indeed, because of the non-cooperative nature of the offload game, the derivation of  $P$  is driven by each application's expected  $c_i$ . However, from the device's user's perspective, the overall cost of the system is of greater importance. In game theory terms, this system cost is referred to as the *social cost* of the game system. In our offload game, we define the social cost to be the makespan of the system. We discuss the optimal social cost in Section 6.4 with the cooperative decision model. But first,

following our results of the Nash equilibrium strategy profile  $P$ , we derive the social cost of the system at Nash equilibrium.

Given a strategy profile  $P$  we derive the social cost (expected makespan) of the system at  $P$ , which we denote with  $\Theta_P$  as

$$\Theta_P = \sum_{A(1) \in \{\mathbb{N}, \mathbb{R}\}} \cdots \sum_{A(n) \in \{\mathbb{N}, \mathbb{R}\}} \prod_{i=1}^n p_i^{A(i)} \max_{j \in \{\mathbb{N}, \mathbb{R}\}} \mathbb{E}[C_j] \quad (6.15)$$

This quantity gives an indication of the system's performance at  $P$ . When strategy profile  $P$  defines an equilibrium, it is important to compare  $\Theta_P$  (*Nash social cost*) with the system's optimal performance (optimal social cost), denoted  $\Theta_{opt}$  which we discuss in Section 6.4. The ratio  $\Theta_P : \Theta_{opt}$  is referred to as the *price of anarchy* (also referred to as "coordination ratio" in [109]) of the game.

We study the price of anarchy of a system which is an indication of how much worse a system would perform if no control is applied on a system level. First introduced in [109], price of anarchy is a key concept often associated with the study of Nash equilibrium in game theory. A Nash equilibrium as we have shown is driven by the selfish behaviours of the agents of a system. Because each agent is only concerned with its own cost when making strategy decisions, without system level control, the overall performance of the system in anarchy becomes as a by-product of the competition between the agents. The distance between this by-product and the optimal performance is represented by the price of anarchy of the game.

We show in next section that the system cost can be minimised when system level control is applied. Then in Section 6.5.3 and Section 6.5.4 we further demonstrate how system performance is described by price of anarchy.

## 6.4 Cooperative Decision Model

The offload decision models we discussed in the previous two sections both assume non-cooperative behaviours within the system. In this third offload decision model, we assume the contrary where a global authority is in place to manage the offload / migration behaviour of the mobile cloud application ecosystem.

From a global perspective, recall that the cost of the system (also referred to as *social cost* in game theory terms) is defined to be the makespan, that is, the maximum schedule length between the two platforms. This naturally leads to a variation of the classic makespan scheduling problem. Recall that  $a_i^j$  indicates if  $i$  chooses to run on  $j$ , and that  $[n]^{\mathbb{N}}$  and  $[n]^{\mathbb{R}}$  denote the subsets of applications that are fixed to run on  $\mathbb{N}$  and  $\mathbb{R}$  respectively. With these we formulate the problem as an integer program:

$$\text{minimise } \Theta_{opt} = \max_{j \in \{\mathbb{R}, \mathbb{N}\}} \sum_{i=1}^n a_i^j \left( \frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) \quad (6.16)$$

$$\text{subject to } a_i^{\mathbb{R}} + a_i^{\mathbb{N}} = 1, \quad i \in [n] \quad (6.17)$$

$$a_i^j \in \{0, 1\}, \quad i \in [n], j \in \{\mathbb{R}, \mathbb{N}\} \quad (6.18)$$

$$a_i^{\mathbb{N}} = 1, \quad i \in [n]^{\mathbb{N}} \quad (6.19)$$

$$a_i^{\mathbb{R}} = 1, \quad i \in [n]^{\mathbb{R}}. \quad (6.20)$$

Note that our problem is different from the classic makespan scheduling problem in that the speed of each machine (platform) consists of two sub-speeds ( $s_j = \{s_j^d, s_j^b\}$ ). Therefore the machines in our problem can not be ordered by their speeds as in the classic makespan scheduling problem [110]. The complexity of this problem is at least NP-hard since it contains a special case, when  $\forall j \in \{\mathbb{R}, \mathbb{N}\} : s_j^d = s_j^b$ , which can be reduced to a classic makespan scheduling problem which is NP-hard even for two identical machines.

The solution of this integer program gives us the optimal assignment in terms of minimising the social cost of the system. However, besides the complexity, the solution also assumes that there is a global authority that enforces the assignment which is not the case in the current mobile cloud computing framework. Operating systems who manage the wireless data protocol on mobile devices does not schedule where applications are run. Techniques exist to exploit delay-tolerant property of some applications to reduce the tail energy overhead [111]. Pre-fetching is another technique used to improve the efficiency of the data link [111, 112]. Though in all cases, the operating system attempts to complete all requests from applications and does not proactively seek to offload any particular application.

Existing offload techniques in mobile cloud computing assumes exclusivity over the host device's data link. Offload decisions are made selfishly by the application. Therefore we next introduce a game theoretic framework to study the effect of the selfish behaviours in the ecosystem of mobile cloud applications.

## 6.5 Simulations, Comparisons and Discussion

### 6.5.1 Simulation Setup

In this section we demonstrate and visualise the behaviours of mobile cloud applications under different offload decision models, and the influence of such over the social cost of mobile cloud application ecosystems.

Each group of simulations is referred to in this chapter by a group ID which is given in the first column of Table 6.2. For instance, test group S1 has 40 test cycles. With each test cycle generates one simulation, S1 includes 40 simulations. Detailed parameters of these test groups are also given in this table. We define each application's data and computation weights to be the multiples of a unit weight, and each platform's processing data and computation speeds to be the number of unit weights it may process in one second. Therefore the social costs are also measured in seconds.

Note that in the results we present in Section 6.5, we use **red** to illustrate the results from the symmetrically incomplete information game, **blue** for systems in Nash equilibrium of the complete information game and **black** for results from the cooperative decision model.

Table 6.2: Simulation parameters

Test		Application Parameters				Platform Parameters	
Group	Cycles	$[[n]^{\mathbb{N}}],  [n]^{\mathbb{R}} ,  [n]^{\mathbb{H}} ]$	Support	$[w_i^d, w_i^b]$	Observed	$[w_k^d, w_k^b]$	$[s_{\mathbb{N}}^d, s_{\mathbb{N}}^b, s_{\mathbb{R}}^d, s_{\mathbb{R}}^b]$
S1	40	[0, 0, 10]	$i \in \{2, \dots, 10\}$	[50, 500]	$k \in \{1\}$	$[50, (+10)500^\dagger]$	[ <i>inf</i> , 200, 50, 800]
S1F	40	[0, 0, 10]	$i \in \{2, \dots, 10\}$	[50, 500]	$k \in \{1\}$	$[50, (+10)500]$	[ <i>inf</i> , 200, 50, 1600]
S2	40	[0, 0, 10]	$i \in \{2, \dots, 10\}$	[50, 500]	$k \in \{1\}$	$[50, 500(+10)]$	[ <i>inf</i> , 200, 20, 800]
S3	40	[0, 0, 10]	$i \in \{2, \dots, 10\}$	$[50, (+10)500]$	$k \in \{1\}$	[50, 500]	[ <i>inf</i> , 200, 50, 800]
S4	40	[0, 0, 10]	$i \in \{2, \dots, 10\}$	$[50, 500(+10)]$	$k \in \{1\}$	[50, 500]	[ <i>inf</i> , 200, 20, 800]
Y1	200	[1, 1, 15]	$i \in [n]$	$[50, Expo(500)]$	-	-	[ <i>inf</i> , 200, 50, 800]
Y2	200	[1, 1, 15]	$i \in [n]$	$[50, Pois(500)]$	-	-	[ <i>inf</i> , 200, 50, 800]
Y3	200	[1, 1, 15]	$i \in [n]$	$[50, Unif(0 : 1000)]$	-	-	[ <i>inf</i> , 200, 50, 800]
V1	100	[1, 1, 15]	$i \in [n]$	[100, 700]	-	-	[ <i>inf</i> , 100, 50, (400 : 3600)]
V2	100	[1, 1, 15]	$i \in [n]$	[100, 500]	-	-	[ <i>inf</i> , 100, 50, (400 : 3600)]
V3	100	[1, 1, 15]	$i \in [n]$	[50, 700]	-	-	[ <i>inf</i> , 100, 50, (400 : 3600)]
V4	100	[1, 1, 15]	$i \in [n]$	[100, 700]	-	-	[ <i>inf</i> , 200, (10 : 500), 800]
V5	100	[1, 1, 15]	$i \in [n]$	[100, 500]	-	-	[ <i>inf</i> , 200, (10 : 500), 800]
V6	100	[1, 1, 15]	$i \in [n]$	[50, 700]	-	-	[ <i>inf</i> , 200, (10 : 500), 800]

† - Increase by specified amount in every cycle. “(+10)500” means increase by 10 until 500 is reached, “500(+10)” means increase by 10 starting with 500.

### 6.5.2 Strategy Behaviour of Non-Cooperative Applications

In this group of experiments, we observe the behaviour of individual applications under different offload decision models.

#### Application with increasing weight

In this group of tests (S1 and S2), we assume a system of 10 hybrid applications. We increase the weight of one of the applications (observed) while keeping all other (support) applications' weights unchanged. In S1 and S1F, as shown in (a) and (b) of Fig. 6.3, we increase the computation weight of the observed application by 10 units until it reaches 500 at which point it has identical weights to the support applications. In S2, as shown in (c) and (d) of Fig. 6.3, we begin with a group of 10 identical applications and gradually increase the computation weight of the observed application. The applications' non-cooperative offload strategies towards remote execution are as shown in (a) and (c) of Fig. 6.3. The corresponding social costs are as shown in (b) and (d) of Fig. 6.3.

Recall that when offload decisions are made according to incomplete information, all applications assume exclusive usage of the device's data connection. Because the wireless bandwidth in S1 and S1F are sufficiently large ( $s_{\mathbb{R}}^d = 50$  for  $w_{\mathbb{R}}^d = 50$  takes 1 second), the delay caused by this communication task is small enough to not deter the support applications ( $i \in \{2, \dots, 10\}$ ) from remote execution. For the observed application ( $k = 1$ ), because its initial computation size is relatively small, unlike the applications in the support group, its benefit of remote execution is not sufficiently large enough to overcome the extra cost of data communication at early stages of S1 and S1F and prefers native execution.

On the contrary, when applications are given complete information of others' strategies, we see from Fig. 6.3 (a) that the observed application's preference on remote execution ( $p_1^{\mathbb{R}}$ ) is reduced as its computation weight increases.

This behaviour seems counterintuitive and counter-productive since it follows a completely opposite direction to that of the incomplete information scenario. Further reduction to (6.14) helps understand this strategy choice. We apply S1's application composition to (6.14) and

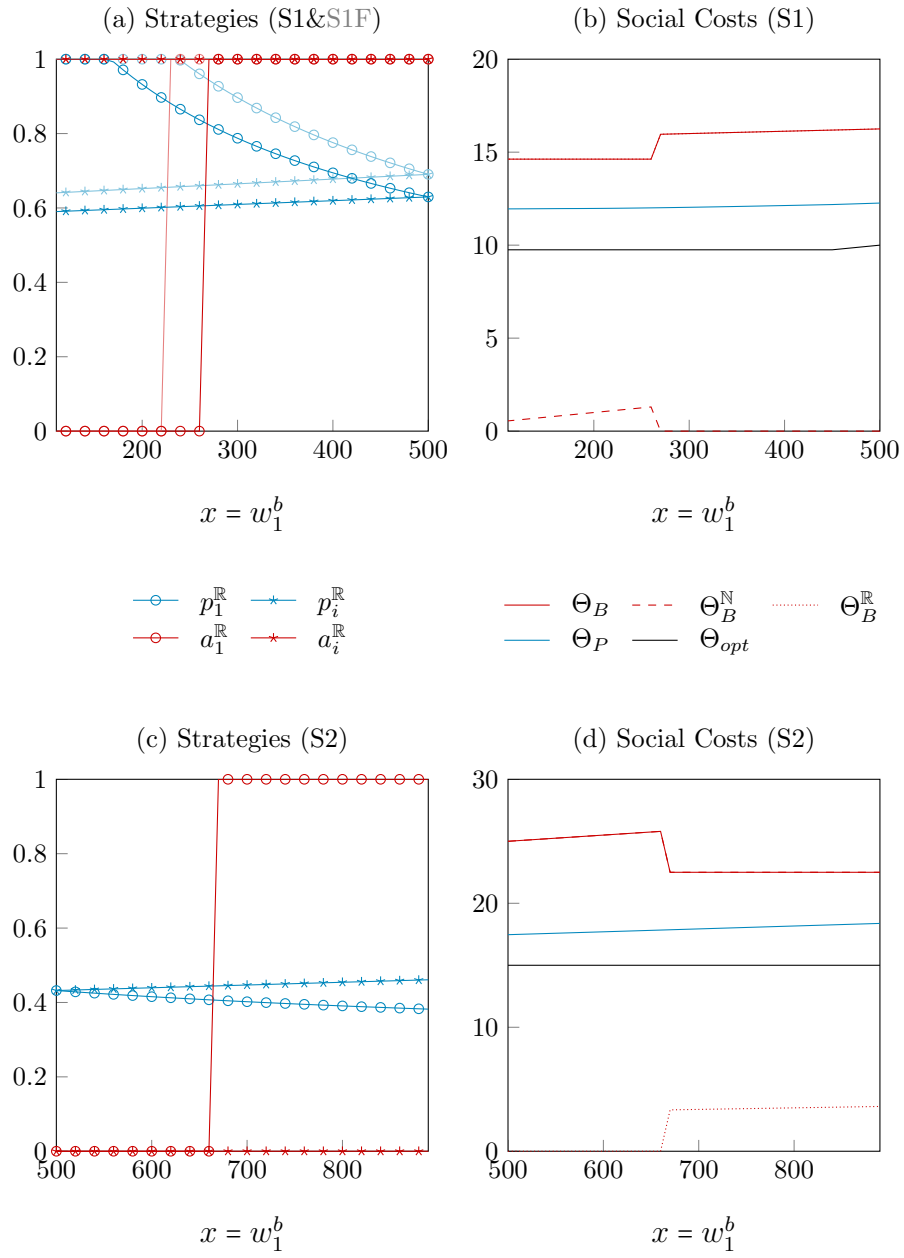


Figure 6.3: Results from S1, S1F and S2: Offload strategy behaviours of application with increasing weight, and the impact on social costs.

derives

$$p_1^{\mathbb{R}} = \frac{8\left(\frac{w_1^d}{s_{\mathbb{R}}^d} + \frac{w_1^b}{s_{\mathbb{R}}^b}\right) + \frac{w_1^b}{s_{\mathbb{N}}^b} + 9\left(\frac{w_i^b}{s_{\mathbb{N}}^b} - \left(\frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b}\right)\right)}{9\left(\frac{w_1^d}{s_{\mathbb{R}}^d} + \frac{w_1^b}{s_{\mathbb{R}}^b} + \frac{w_1^b}{s_{\mathbb{N}}^b}\right)} \quad (6.21)$$

In (6.21) we see that  $p_1^{\mathbb{R}}$  is dependent on both **internal** and **external** terms. When the platform parameters are fixed, the internal terms are influenced only by the weights of the observed application itself. The external term in (6.21) represents the *collective gain* that would have been obtained by other applications if they were to execute remotely.

In S1 and S2, the external term is a constant since the weights of the support applications are constants. When  $w_1^b$  increases, the second internal term always increase faster than the first, the reduction in  $p_1^{\mathbb{R}}$  as shown in (a) and (c) of Fig. 6.3 follows.

Note that in the first few test cycles, in S1, the external term dominates (6.21) and the observed application become a pure strategy agent with  $p_1^{\mathbb{R}} = 1$ .

### Application within Increasing weights

In S3 and S4, we fix the weight of the observed application and increase the support group's computation weight instead. In such cases, the external term in (6.21) become the variable. Because the increase in computation weights, the collective gain of the support group, i.e. the external term increases, and the increase in  $p_1^{\mathbb{R}}$  in (a) and (c) of Fig. 6.4 follows.

Also note that because of the switch of role between the observed application and the support group in terms of weight increase from S1 and S2 to S3 and S4, strategies of the observed application and the support group under incomplete information also swapped positions. In Fig. 6.3 (a) and (c) the observed application switched from native execution to remote execution as its weight increases, whereas in Fig. 6.4 (a) and (c), the same strategy is instead adopted by the support group.



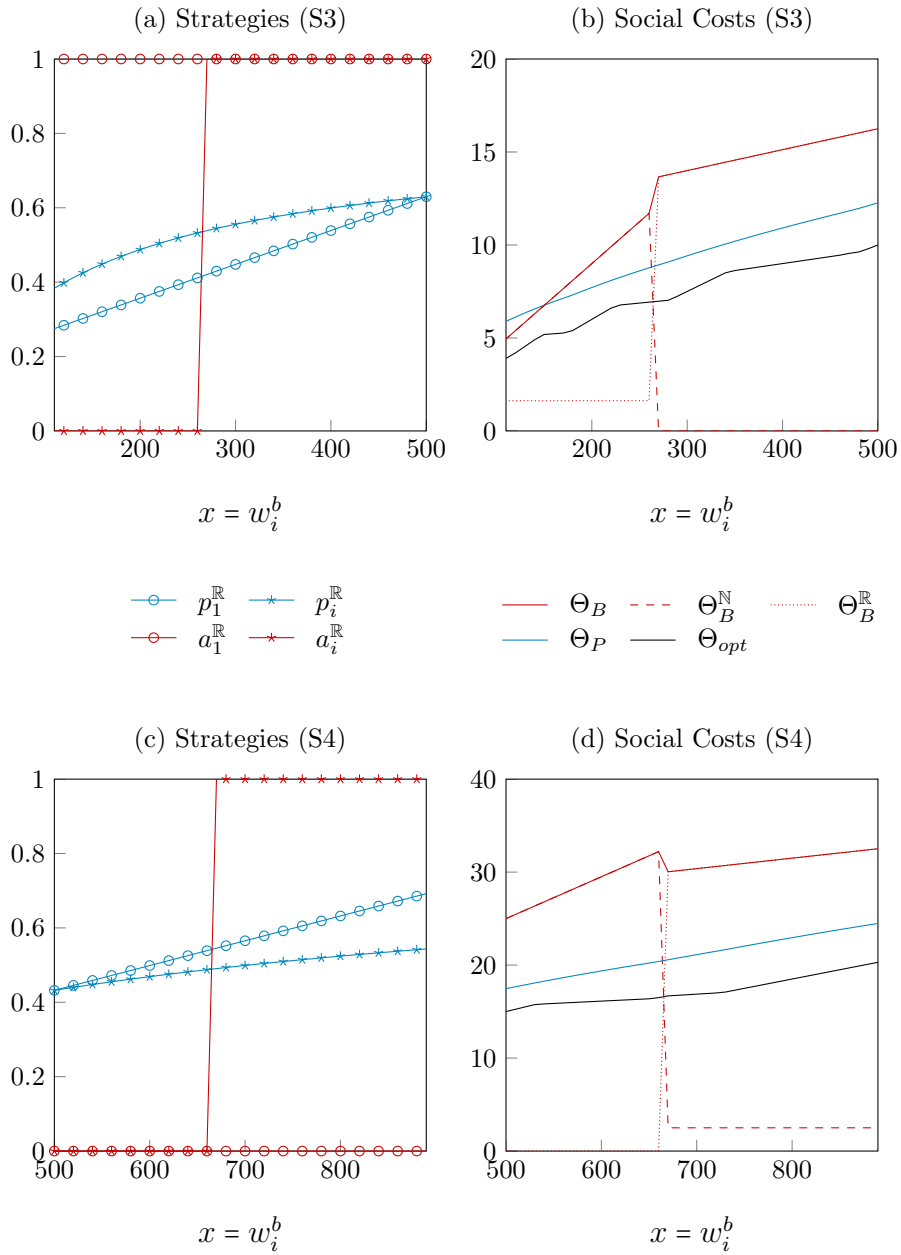


Figure 6.4: Results from S3 and S4: Offload strategy behaviours of application within increasing weight, and the impact on social costs.

### Change in platform parameters

Simulations in S1F is carried out as a comparison study to the results from S1. In S1F, we double the computation speed of the remote platform, therefore the remote platform become more attractive to all applications as compared to S1. The results shown in Fig. 6.3 (a) matches our expectation. In the incomplete information scenario, the observed application adopts remote execution ( $a_1^{\mathbb{R}}$ ) earlier than in S1. In the complete information offload game, all players shifted their strategy towards  $\mathbb{R}$ .

Also note that the external term dominated  $p_1^{\mathbb{R}}$  for more number of cycles at the beginning of S1F than in S1.

### 6.5.3 Social Costs

We now look at the social costs of different decision models of mobile cloud application ecosystems. As shown in (b) and (d) in both Fig. 6.3 and Fig. 6.4, in the incomplete information scenario, a step change is often observed because of the change of strategy by applications at certain thresholds. We plot the social cost ( $\Theta_B$ ) alongside the cost of  $\mathbb{N}$  ( $\Theta_B^{\mathbb{N}}$ ) and  $\mathbb{R}$  ( $\Theta_B^{\mathbb{R}}$ ) to illustrate the relations between the makespan and the costs of each platform.

Compared with the other two decision models, the incomplete information model produces systems with highest social costs. Systems that are in Nash equilibrium as defined by the complete information game have significant higher social costs ( $\Theta_P$ ) than the optimal solution ( $\Theta_{Opt}$ ). We further observe that the gap (price of anarchy) between the optimal social costs and Nash social costs in Fig. 6.3 (d) and Fig. 6.4 (d) increase while the gap between application computation weights increase. Therefore in the next group of tests, we investigate the relation between price of anarchy and the weight deviation in  $[n]$ .

### 6.5.4 Price of Anarchy

Recall that the price of anarchy of the complete information game is defined by the ratio between the Nash social cost ( $\Theta_P$ ) and the optimal social cost ( $\Theta_{Opt}$ ) of the system, which we denote with  $PoA_P$ . For comparison, we further define the price of anarchy in the symmetrically incomplete information game to be  $PoA_B = \Theta_B : \Theta_{Opt}$ . From S4 and S2, we observe slight increases in the

price of anarchy when the difference in weight increases in  $[n]$ . This leads us to the hypothesis that the price of anarchy is more significant when the weights in  $[n]$  have a high value of deviation.

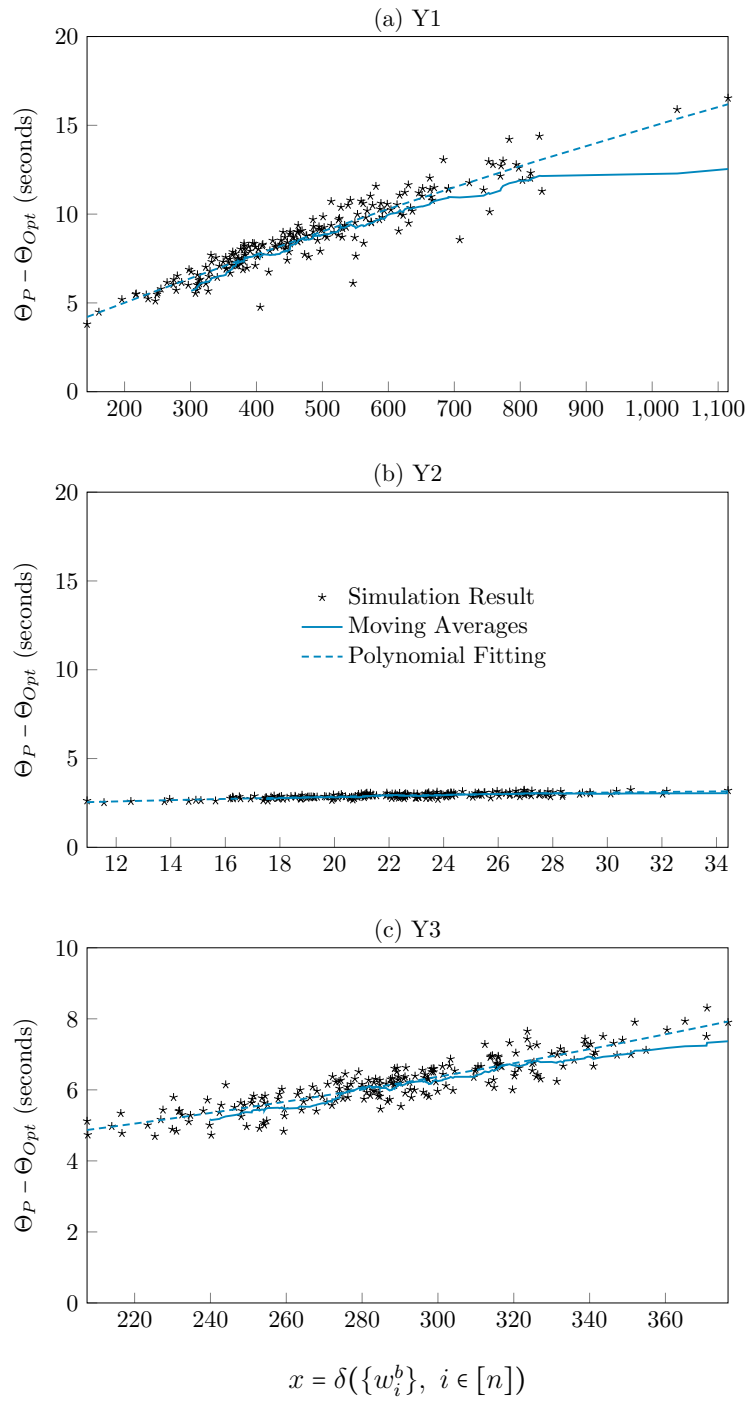
### Price of anarchy and application weight deviation

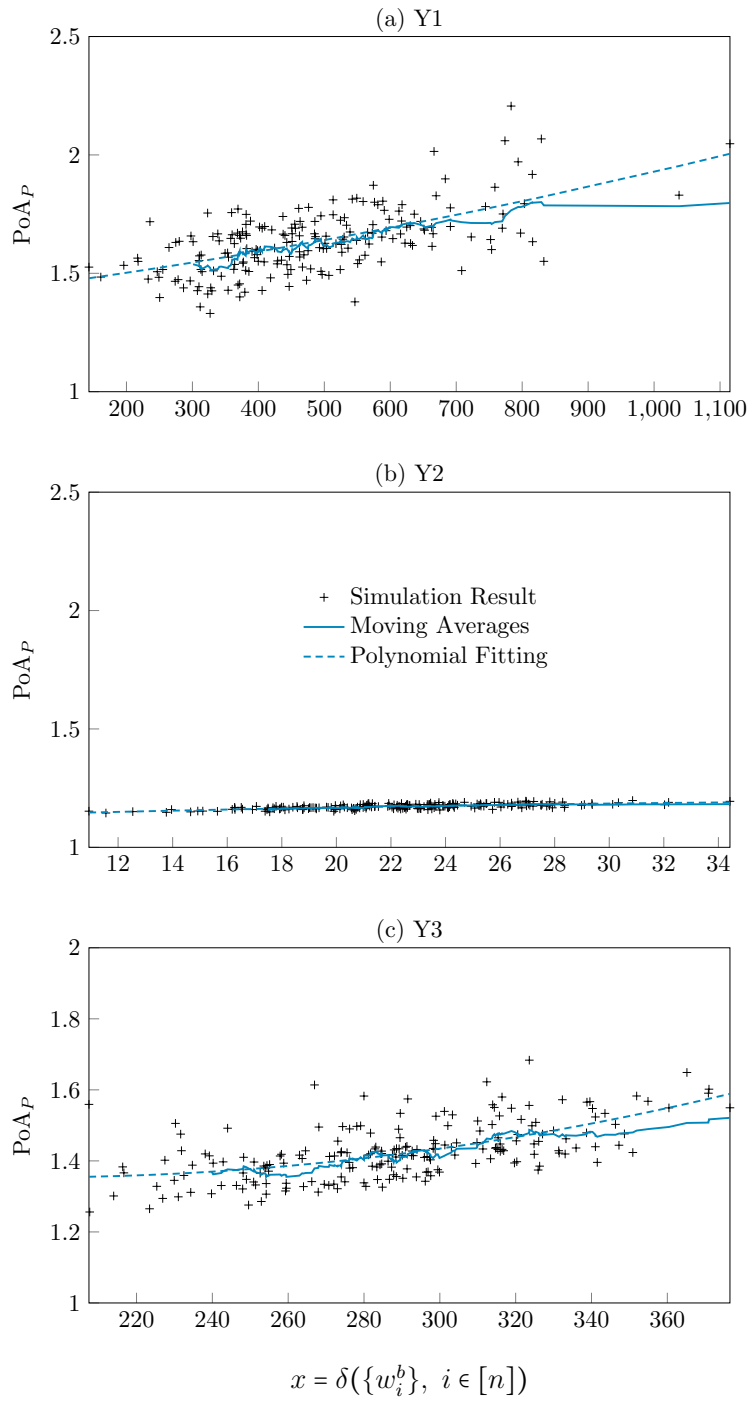
Following on the hypothesis, we conducted tests Y1, Y2 and Y3 the results from which are shown in Fig .6.5 to 6.9. In these three groups of experiments, we run each cycle of our simulation with the same parameters except the computation weights<sup>5</sup> of applications which is randomly drawn from three different distributions (exponential, Poisson and uniform) at each test cycle. We choose these three distributions not only because of their differences in range and variance, but also because each distribution may be suitable to simulate the workload pattern of particular mobile application ecosystems. For instance, a set of applications whose workload depends on the arrival time of different user requests may be more suited to the exponential distribution. Applications whose workload is pre-defined to be within a range with equal probability to pick within this range is more suited to the uniform distribution model.

We label the simulation generated by each test cycle with the standard deviation of the computation weights of all applications (i.e.  $\delta(\{w_i^b\}, i \in [n])$ ), and apply all three decision models to the system simulated in that cycle. With each of the two non-cooperative decision models, we record its social cost and compare it with the optimal social cost produced by the cooperative model. We plot five properties of the system against its deviation label in Fig .6.5 to 6.9. These properties includes  $\Theta_P - \Theta_{Opt}$  in Fig .6.5,  $PoA_P = \Theta_P : \Theta_{Opt}$  in Fig .6.6,  $\Theta_B - \Theta_{Opt}$  in Fig .6.7,  $\Theta_B : \Theta_{Opt}$  in Fig .6.8 and the social costs of the system in Fig .6.9. Each property of each simulation (generated in each test cycle) is plotted with its application weight deviation as x and the value of the property as y in each of the plots in Fig .6.5 to 6.9.

From (a) and (c) of Fig. 6.5, we observe that the increase in application weight deviation (along the x-axis) indeed increase the probability of bigger gaps between  $\Theta_P$  and  $\Theta_{Opt}$ . The same trend is also observed in (a) and (c) of Fig. 6.6 for the price of anarchy albeit with a smaller gradient. In contrast, as shown in (b) of Fig. 6.5 and (b) of Fig. 6.6, all simulations in Y2 have a similar and stable price of anarchy. This is because Poisson distribution generates application

<sup>5</sup>We also conducted experiments that randomised both data and computation weights. The results are similar to that of Y1, Y2 and Y3 and so are omitted for brevity.

Figure 6.5: Results from Y1, Y2 and Y3:  $\Theta_P - \Theta_{Opt}$ .

Figure 6.6: Results from Y1, Y2 and Y3:  $\text{PoA}_P = \Theta_P : \Theta_{Opt}$ .

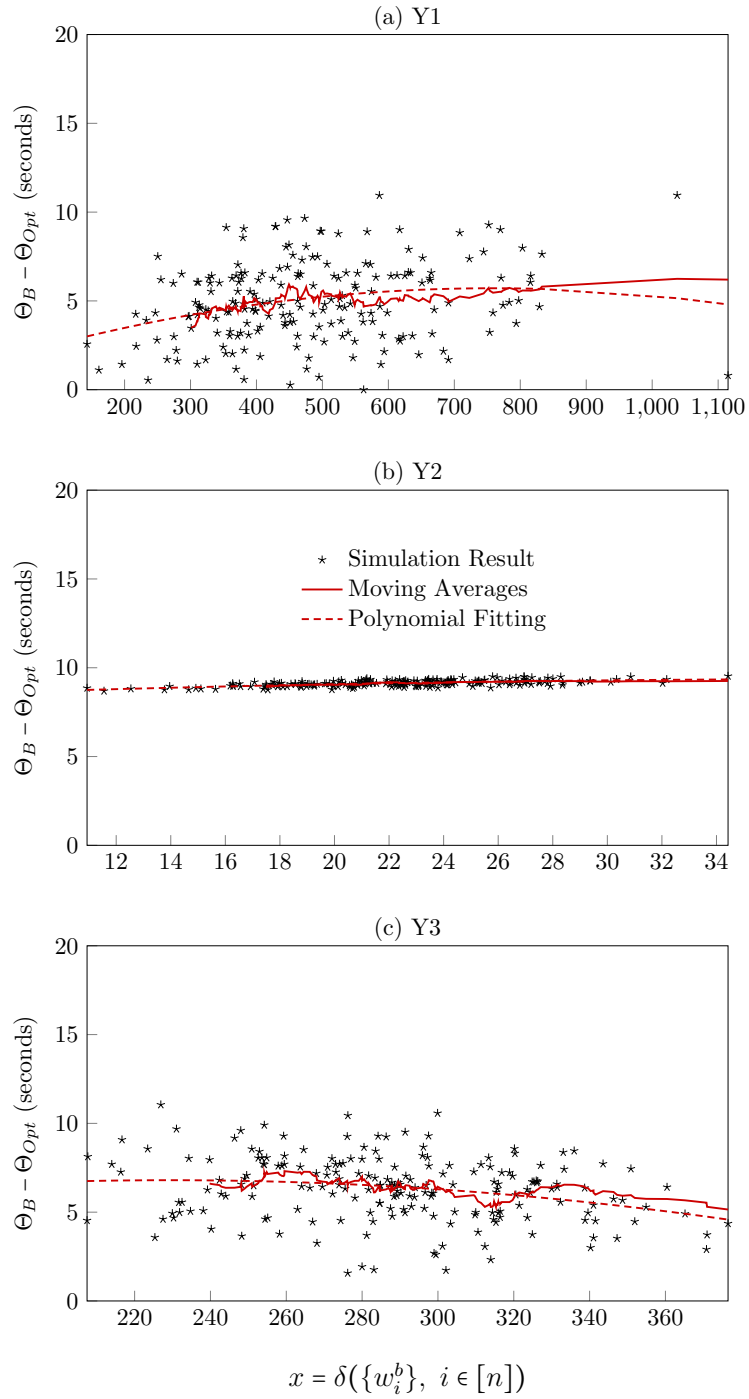
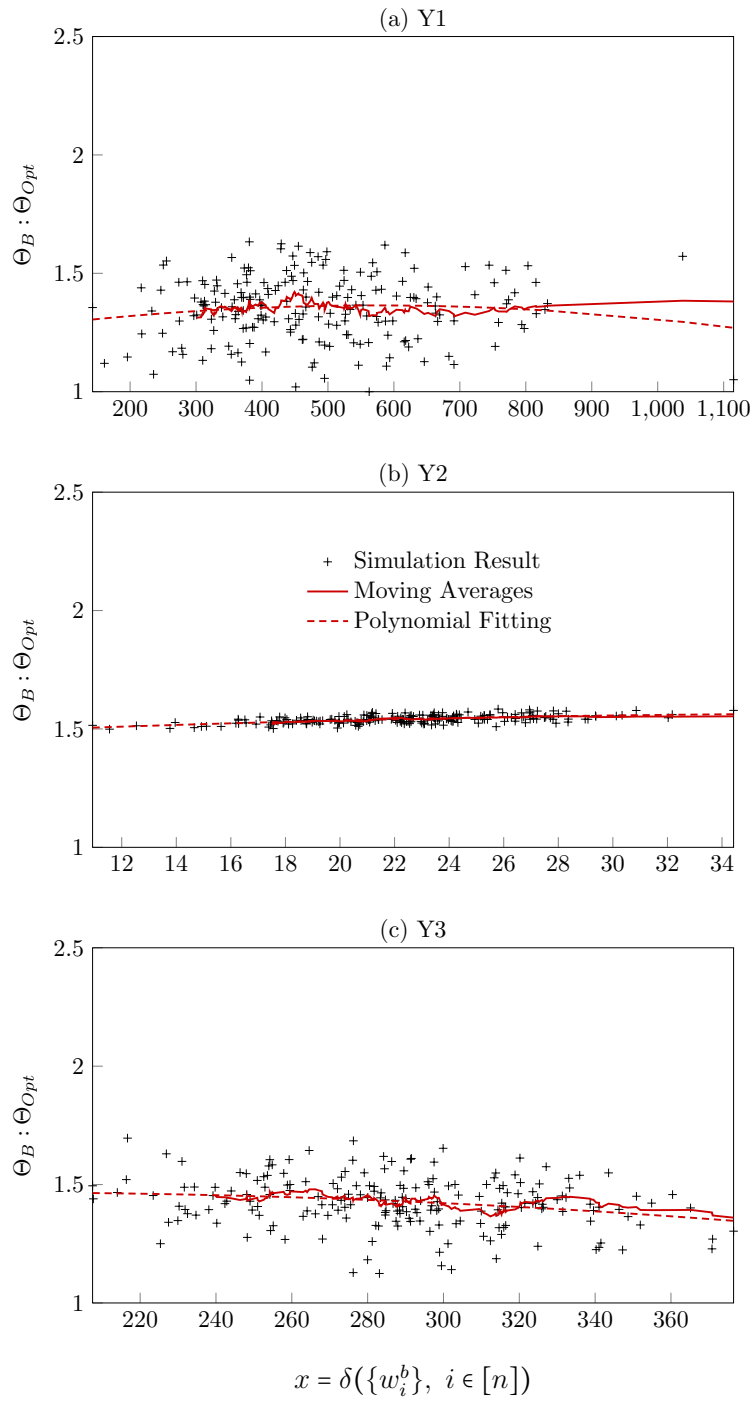


Figure 6.7: Results from Y1, Y2 and Y3:  $\Theta_B - \Theta_{Opt}$ .

Figure 6.8: Results from Y1, Y2 and Y3:  $\text{PoA}_B = \Theta_B : \Theta_{Opt}$ .

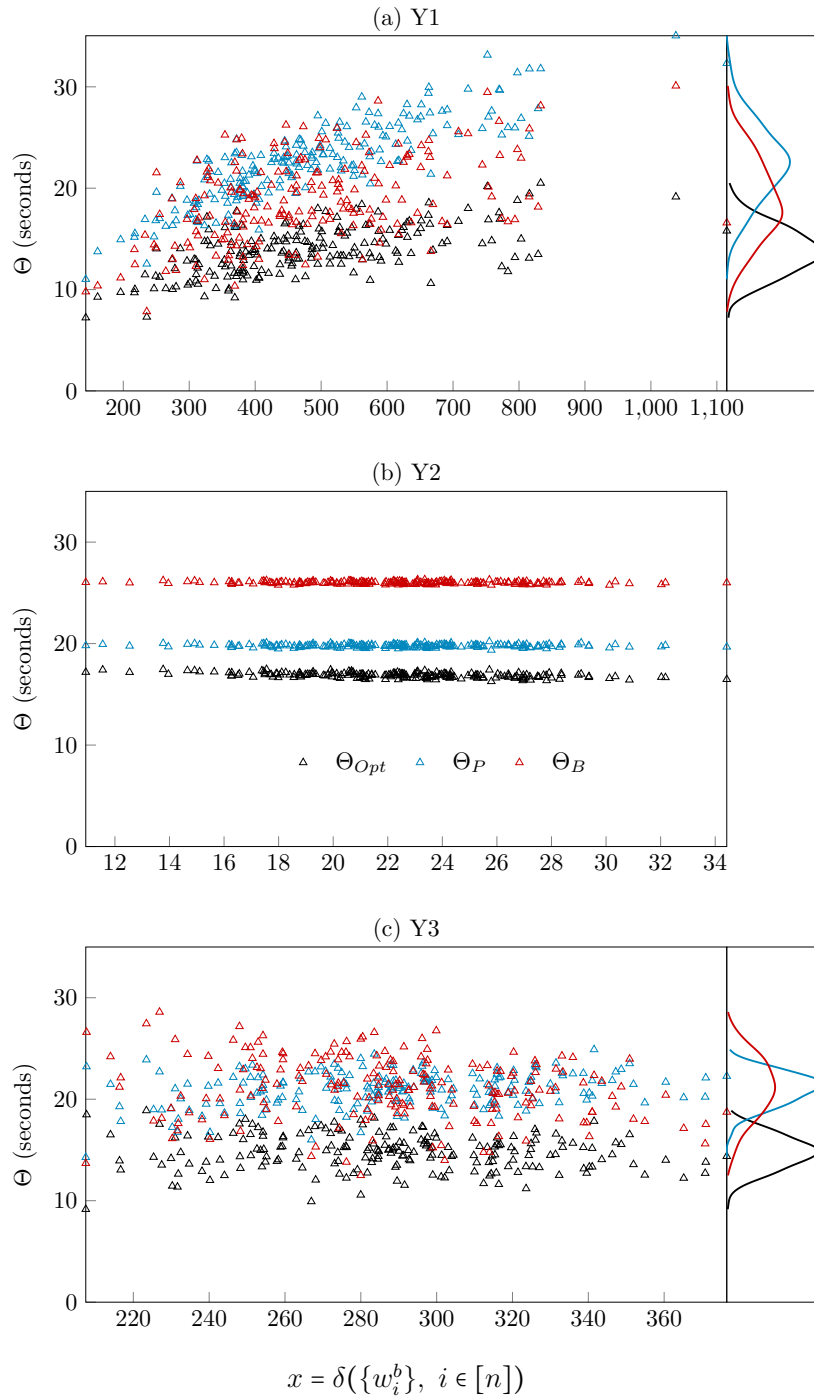


Figure 6.9: Results from Y1, Y2 and Y3: Social costs.



weights with small deviations (c.f. range of x-axis in (b) of Fig .6.5 to 6.9). Furthermore, because applications simulated in Y2 are very similar to each other, the social costs of all three decision models are bounded within three small region as shown in (b) of Fig .6.9.

Fig. 6.7 Fig. 6.8 illustrate the difference between  $\Theta_B$  and  $\Theta_{Opt}$ . While results from Y2 follow a similar pattern as in Fig. 6.5 Fig. 6.6, results from Y1 and Y3 are rather chaotic. This is due to the behaviour of the offload model based on incomplete information. Recall that the model predict an application's cost on both platforms based on incomplete information. This split is largely influenced by the device's bandwidth. When the bandwidth is given, this split is determined by the weights of the applications. When these weights are randomly chosen within a relatively big range as in Y1 and Y3, this split of applications is likely to produce randomly unbalanced groups. Compared to the optimal split produced by the cooperative model, it is predictable that the  $\Theta_B$  produced by this rather random behaviour has such random distance to  $\Theta_{Opt}$ . We also observe from (a) and (c) of Fig. 6.7 Fig. 6.8 that as well as having a big distance from  $\Theta_{Opt}$  (distance from the x-axis), it is also possible for the incomplete information model to produce near optimal results (near to the x-axis).

The actual system costs of Y1-Y3 are shown in Fig. 6.9. The increase in price of anarchy is most observable in (a) for it has the greatest x range.

### Price of anarchy and changes in platform parameters

To further observe the price of anarchy in the system, we also conducted V1-V3 in which  $s_{\mathbb{R}}^b$  is gradually increased in each test cycle, and V4-V6 in which  $s_{\mathbb{R}}^d$  is gradually increased in each test cycle. As shown in Fig. 6.10, the price of anarchy in these tests are significantly lower than that from Y1 and Y3 because all applications have similar weights.

The increase in either processing speed and wireless bandwidth reduces and then stabilises the price of anarchy. This is because once a speed term is greater than a certain value, the cost term it is related to tends to zero and no longer have any effect over the system cost. Note that the turning points in Fig. 6.10 are caused when the optimal cooperative strategy switches one of the application's allocation from  $\mathbb{N}$  to  $\mathbb{R}$  as  $\mathbb{R}$  becomes more and more attractive with its increasing computation speed (or wireless bandwidth).

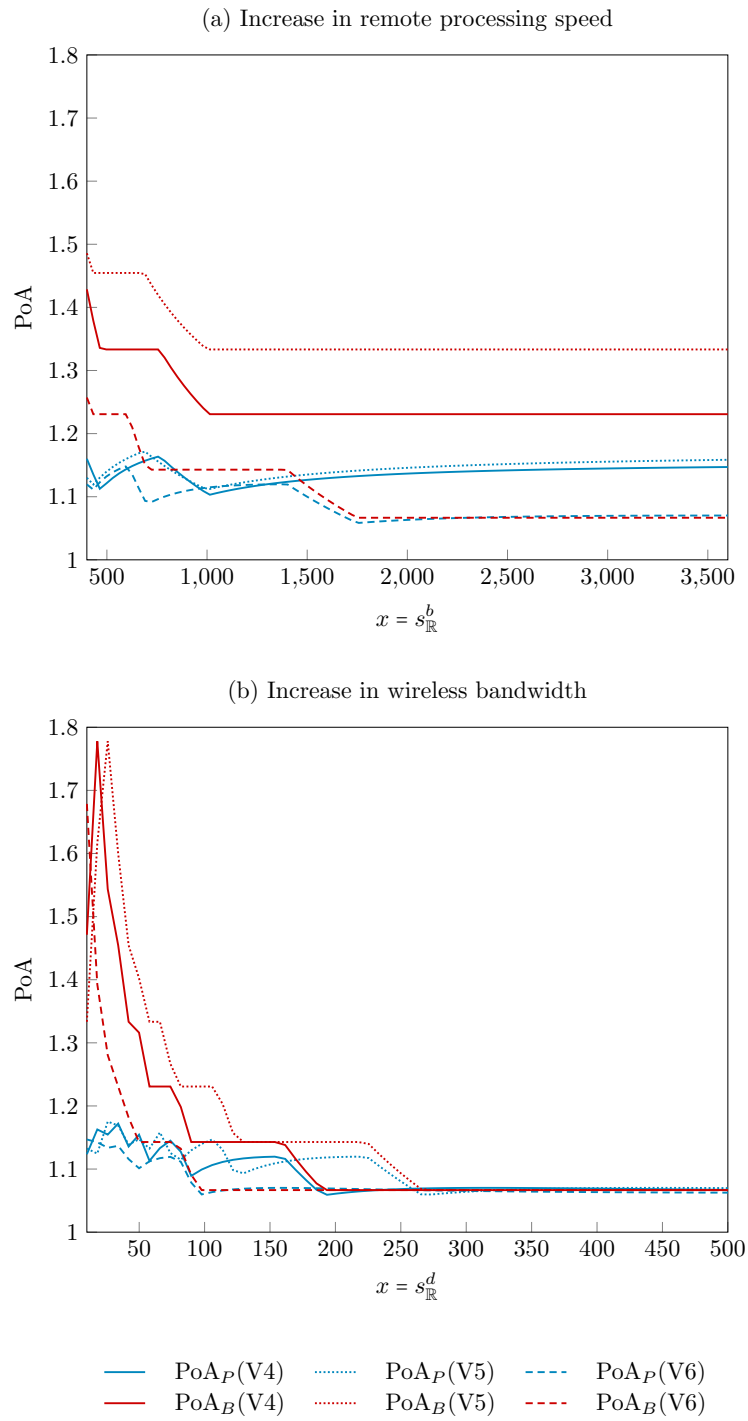


Figure 6.10: Price of anarchy following changes in platform parameters.

## 6.6 Summary

In this chapter, we investigated the efficiency of application offload in mobile cloud computing. We especially focus on the competition between mobile cloud applications residing on the same device which is overlooked by existing researches of mobile cloud computing.

Our main contribution is the game theoretic modelling of the non-cooperative offload game with complete information. This model is an extension to the classic load balancing game. We presented detailed derivation of the mixed-strategy Nash equilibrium of this game. To compare the system's performance at equilibrium with existing computation offload mechanisms, we also modelled existing offload decision processes as a non-cooperative offload game with symmetrically incomplete information. Furthermore, we propose a cooperative scenario and solve the offload decision problem as a min-max integer program to obtain optimal offload schedules.

We compare the performance of all three offload decision models with a series of simulation experiments. On an application level, we observe the counterintuitive strategy decisions made by applications in the complete information game which help understand application behaviours when no global control is applied. On a system level, we discuss the price of anarchy in non-cooperative scenarios. We show that significant reduction in social cost can be obtained in a cooperative setting. The dependencies between price of anarchy and various system parameters are also investigated. We show that high deviation in application weights encourages high price of anarchy in non-cooperative scenarios.

Our study demonstrates the importance of recognising the potential competition between mobile cloud computing applications, and provide a suite of modelling tools to simulate and solve the offload decision problem in ecosystems of mobile cloud computing applications.

## Chapter 7

# Conclusions and Further Work

The work described in this thesis has been concerned with improvements and extensions to the energy-aware workload offloading frameworks behind the latest development in mobile cloud computing. Recent years have seen significant growth in the size of the mobile computing market, and yet the rarest commodity in the world of mobile computing remains to be its battery power. Despite the moderately improved battery capacity on modern smart devices, user demands of applications with more complex functionalities continue to challenge the energy limit of mobile devices. Mobile cloud computing has emerged as a research topic which aim to overcome the limitations of the mobile platform by integrating cloud services onto the platform. One key technique applied in mobile cloud computing is computation workload offload. In this thesis, we extend existing scheduling and resource management framework of computation workload offload in mobile cloud computing.

The energy-aware task allocation problem were formulated for the mobile cloud platform. Two energy-aware objectives (MGECP and MMUP) were investigated. Two heuristics (SA and GAO) were proposed to approximate the solutions for both objectives. Offload strategies were developed taking into account both energy and time constraints. A heuristic algorithm (WGAO) were proposed to produce offload strategies and demonstrate the effect of different software and hardware characteristics. Bandwidth dependencies of mobile networks were modelled by a network I-O model. Cost-based and adaptive bandwidth allocation schemes were developed on

top of the network I-O model. Competition between applications that reside on the same device is highlighted.

Key contributions are summarised in the first four sections of this chapter. Further work is discussed in Section 7.5.

## 7.1 Energy-Aware Task Allocation

Task allocation is a key optimisation problem in the development of workflow management mechanisms of mobile cloud computing. For a mobile cloud platform to efficiently support the execution of many collaborative application workflows, solving the task allocation problem is a critical first step in ensuring the energy efficiency of the mobile cloud platform.

In Chapter 3, we started by looking at the energy-aware task allocation problem from the mobile cloud platform's point of view. We constructed a quadratic binary program to model the task allocation problem in a general mobile cloud computing platform. We investigated the task allocation problem for two energy-aware objectives: the overall energy cost of the platform which we refer to as the MGECP, and the longevity of the platform which we refer to as the MMUP.

In order to overcome the poor scalability of generic quadratic program solvers, we presented an implementation of the simulated annealing (SA) algorithm and also proposed a greedy autonomous offload (GAO) algorithm to approximate the optimal solution. Both heuristics are tailored to solve our task allocation problem efficiently. We verified and compared our algorithms against a commercial quadratic program solver in a series of simulations. Results show that both heuristics produce good solutions to the task allocation problem. Solutions provided by GAO is consistently close to optimal and can be obtained in a time efficient manor. The methodologies presented in this work are also applicable to other energy critical task allocation problems.

Readers who are familiar with the facility location problem may find a similar underlying structure in our formulation of the task allocation problem in Chapter 3. Our model extends a standard facility location problem in that multiple facilities may reside on the same device, some facilities are fixed or constrained within a set of locations and that not all locations have to be occupied.

## 7.2 Offloading Strategies for Time-Constrained Workflows

In Chapter 4, we investigated further into the energy-aware task allocation problem from a workflow's perspective. Offloading strategies were developed for mobile workflows. Compared to the general case of a mobile cloud computing platform discussed in Chapter 3, extra time constraint is applied when an individual workflow is concerned. Therefore, we model both energy and time constraints in our objective functions in this chapter. In order to develop offloading strategies accordingly, we apply the same design principles of GAO and further develop a workflow-oriented greedy autonomous offload (WGAO) algorithm to develop offload strategies for time-constrained mobile workflows. Simulation results illustrate how different hardware specifications affect the offload-abilities of the workflow and its efficiency. We also introduce a layer of computation offload platform referred to as cloudlets [38, 113] in our platform model of Chapter 4.

Note that in Chapter 3, we assume that the services that support the execution of workflow tasks are already deployed on compatible devices and therefore offloading a task from one device to another only requires modification in the workflow engine's task allocation scheme. In Chapter 4, we assume that the workflow is initially deployed only on the mobile devices, offloading a task occurs extra communication cost for uploading the task's binary from mobile to cloud.

## 7.3 Efficient Resource Allocation in Mobile Networks

Bandwidth is a key limiting factor in enabling workload offload in mobile cloud computing as we have shown in Chapter 3 and Chapter 4. Therefore, in Chapter 5 we looked at the resource management issues in mobile cloud computing, more specifically, the bandwidth allocation problem. In this chapter, we abstract the underlying network structure of a mobile cloud computing platform into a generic mobile service-oriented network (MSON) to that the approach we propose is applicable to general mobile networks rather than just for mobile cloud platforms.

In Chapter 5, we borrowed ideas from the Leontief I-O model in economy and present a network I-O model to formulate the bandwidth dependencies of an MSON. We take into account various factors such as interaction patterns among services, changing network conditions such as bandwidth and latency, arrival rate of service requests, service cost and so on in the model.

Based on the network I-O model, a cost-based bandwidth allocation scheme was proposed with the objective to maximise the benefit gained from completing service requests while taking into account bandwidth cost and penalties from QoS violations. Furthermore, we proposed a set of adaptive bandwidth allocation strategies also derived from our Network I-O model. When the bandwidths of mobile devices decrease (e.g., from WiFi to 3G), these adaptive strategies are able to adjust the bandwidth allocations for each service in the way that the overall impact on the service QoS is minimised. Simulation studies are presented which verify and demonstrate the effectiveness of the model.

## **7.4 Application Ecosystem and Offload Competition**

Mobile devices are shared between applications. Existing offload frameworks assume exclusive usage of the host device's resources like the bandwidth. In the scenarios where only a few applications are installed on the same device, and they are in sleep states most of the time, our assumption is close to reality. However, with the increasing popularity of mobile applications, and the emerging trend of intelligent mobile cloud applications, competition is likely to exist over the device's resources. Therefore offload decision models are to be adjusted accordingly.

In Chapter 6, we rethink the offload decision making processes of mobile cloud computing when applications deployed on the platform exhibit non-cooperative behaviours according to different level of knowledge they have in terms of the existence and strategies of each other. To this end, we extended the framework of the classic load balancing game and derive the mixed-strategy Nash equilibrium of the non-cooperative offload game. With this model we are able to derive system performance at equilibrium and compare it with that of a managed and cooperative environment.

We quantify the price of anarchy in non-cooperative settings and highlight the importance of a global offloading management mechanism to enforce cooperation in mobile cloud computing environments to maximise system performance. The equilibrium strategies we derived also help the decision making processes of individual applications when no global authority is in place.

## 7.5 Directions for Further Work

Two energy-aware objectives were studied in Chapter 3, namely MGECP and MMUP. As well as optimising toward each objective individually, it is also beneficial to join these two objectives in search of an energy-aware task allocation scheme. For instance, as shown in [4], adjustment to allocation schemes produced for MGECP may be adjusted towards the objective of MMUP. One issue related with such objective is how to control the balance between the two objectives. As we have shown in our work, allocation schemes produced for one of the objectives contradict with the goal of the other. Another issue related to this extension is the complexity of the quadratic program. Similar to the QCP of MMUP, many quadratic constraints are to be added to the program which dramatically increases the complexity of finding the exact solution.

The network I-O model proposed in Chapter 5 lays the foundation for further objective developments in other networked environments. For instance, we also apply the network I-O model to analyse the interaction between VMs of computation clusters in [7] and make resource management decisions in high performance computing infrastructures.

The game theoretical model we proposed in Chapter 6 extends the classic load balancing game which applicable to a wide spectrum of computing environments. Mixed-strategy Nash equilibriums for the classic load balancing game was developed for machines with only one processor which may be of different speed. Our extension to the model adds a second processor to the machine. This greatly expands the applicability of the model. For instance, in cluster computing, each node have two processors, a CPU and a GPU. Our model is ideal in modelling the Nash equilibrium in such high performance computing resources.

Finally, with the maturity of technologies like HTML5 and JavaScript, mobile application development frameworks like Apache Cordova dramatically reduce the complexity of developing mobile applications which is executable on both locally on device and remotely on cloud. We would like to propose further extension to such development environments like Apache Cordova to enable the offload mechanisms of mobile cloud computing implementing the theoretical frameworks we proposed in this thesis.



# Bibliography

- [1] B. Gao, L. He, and S. A. Jarvis, “Offload Decision Models and the Price of Anarchy in Mobile Cloud Application Ecosystems,” *IEEE Access, Special Section on Emerging Cloud-Based Wireless Communications and Networks*, vol. 3, pp. 3125–3137, 2016.
- [2] B. Gao, L. He, and C. Chen, “Modelling the Bandwidth Allocation Problem in Mobile Service-Oriented Networks,” in *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM’15)*, pp. 307–311, 2015.
- [3] B. Gao, L. He, X. Lu, C. Chang, K. Li, and K. Li, “Developing Energy-Aware Task Allocation Schemes in Cloud-Assisted Mobile Workflows,” in *Proceedings of IEEE International Conference on Ubiquitous Computing and Communications (IUCC’15)*, pp. 1266–1273, 2015.
- [4] B. Gao and L. He, “Modelling Energy-Aware Task Allocation in Mobile Workflows,” in *Proceedings of the 10th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous’13)*, vol. 131, pp. 89–101, 2013.
- [5] B. Gao, L. He, L. Liu, K. Li, and S. Jarvis, “From Mobiles to Clouds: Developing Energy-Aware Offloading Strategies for Workflows,” in *Proceedings of the 13th ACM/IEEE International Conference on Grid Computing (GRID’12)*, pp. 139–146, 2012.
- [6] H. Zhu, L. He, B. Gao, K. Li, and K. Li, “Modelling and Developing Co-Scheduling Strategies on Multicore Processors,” in *Proceedings of the 44th International Conference on Parallel Processing (ICPP’15)*, 2015.

- [7] C. Chen, L. He, and B. Gao, “Modelling and Optimizing Bandwidth Provision for Interacting Cloud Services,” in *Proceedings of the 13th International Conference on Service Oriented Computing (ICSOC’15)*, 2015.
- [8] S. Fu, L. He, X. Liao, C. Huang, K. Li, C. Chang, and B. Gao, “Cadros: The Cloud-Assisted Data Replication in Decentralized Online Social Networks,” in *Proceedings of the 11th IEEE International Conference on Services Computing (SCC’14)*, pp. 43–50, 2014.
- [9] S. Fu, L. He, X. Liao, C. Huang, K. Li, C. Chang, and B. Gao, “Modelling and Predicting the Data Availability in Decentralized Online Social Networks,” in *Proceedings of the 21st IEEE International Conference on Web Services (ICWS’14)*, pp. 161–168, 2014.
- [10] C. Chen, L. He, H. Chen, J. Sun, B. Gao, and S. A. Jarvis, “Developing Communication-aware Service Placement Frameworks in the Cloud Economy,” in *Proceedings of IEEE International Conference on Cluster Computing (CLUSTER’13)*, pp. 1–8, 2013.
- [11] K. Li, Z. Zhang, Y. Xu, B. Gao, and L. He, “Chemical Reaction Optimization for Heterogeneous Computing Environments,” in *Proceedings of the 10th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA’12)*, pp. 17–23, IEEE, 2012.
- [12] L. He, C. Huang, K. Li, H. Chen, J. Sun, B. Gao, K. Duan, and S. A. Jarvis, “Modelling and Analyzing the Authorization and Execution of Video Workflows,” in *Proceedings of the 18th International Conference on High Performance Computing (HiPC’11)*, pp. 1–10, 2011.
- [13] M. Gerla and L. Kleinrock, “Vehicular networks and the future of the mobile internet,” *Computer Networks*, vol. 55, no. 2, pp. 457–469, 2011.
- [14] D. Cuff, M. Hansen, and J. Kang, “Urban sensing: Out of the Woods,” *Communications of the ACM*, vol. 51, pp. 24–33, Mar. 2008.
- [15] Gartner Research, “Gartner Reveals Top Predictions for IT Organizations and Users for 2012 and Beyond,” 2011.

- [16] J. Paradiso and T. Starner, "Energy Scavenging for Mobile and Wireless Electronics," *IEEE Pervasive Computing*, vol. 4, pp. 18–27, Jan. 2005.
- [17] K. Pentikousis, "In Search of Energy-Efficient Mobile Networking," *IEEE Communications Magazine*, vol. 48, pp. 95–103, Jan. 2010.
- [18] M. Satyanarayanan, "Mobile computing: the Next Decade," in *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services Social Networks and Beyond - MCS '10*, pp. 1–6, 2010.
- [19] L. Pajunen and S. Chande, "Developing Workflow Engine for Mobile Devices," in *EDOC'07 11th IEEE International Enterprise Distributed Object Computing Conference*, Oct. 2007.
- [20] A. Mnaoue and A. Shekhar, "A Generic Framework for Rapid Application Development of Mobile Web Services with Dynamic Workflow Management," in *SCC'04 IEEE International Conference on Services Computing*, 2004.
- [21] L. Pajunen and A. Ruokonen, "Modeling and Generating Mobile Business Processes," in *IEEE International Conference on Web Services (ICWS 2007)*, pp. 920–927, July 2007.
- [22] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making Smartphones Last Longer with Code Offload," in *MobiSys'10 The 8th International Conference on Mobile Systems, Applications, and Services*, June 2010.
- [23] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems - EuroSys '11*, p. 301, 2011.
- [24] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo : a Computation Offloading Framework for Smartphones," in *MOBICASE 2010 IEEE Computer Society*, 2010.
- [25] D. Huang, X. Zhang, M. Kang, and J. Luo, "MobiCloud: Building Secure Cloud Framework for Mobile Computing and Communication," in *2010 Fifth IEEE International Symposium on Service Oriented System Engineering*, pp. 27–34, IEEE, June 2010.

- [26] S. Kosta, A. Aucinas, and R. Mortier, “ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading,” in *2012 Proceedings IEEE INFOCOM*, pp. 945–953, 2012.
- [27] U. Kremer, J. Hicks, and J. M. Rehg, “Compiler-directed remote task execution for power management,” *Workshop on Compilers and Operating Systems for Low Power (COLP’00)*, 2000.
- [28] Z. Li, C. Wang, and R. Xu, “Computation offloading to save energy on handheld devices,” in *Proceedings of the international conference on Compilers, architecture, and synthesis for embedded systems - CASES ’01*, p. 238, 2001.
- [29] C. Wang and Z. Li, “Parametric analysis for adaptive computation offloading,” in *Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation - PLDI ’04*, vol. 39, (New York, New York, USA), p. 119, June 2004.
- [30] S. Kim, H. Rim, and H. Han, “Distributed execution for resource-constrained mobile consumer devices,” *IEEE Transactions on Consumer Electronics*, vol. 55, pp. 376–384, May 2009.
- [31] J. Flinn and M. Satyanarayanan, “Balancing performance, energy, and quality in pervasive computing,” in *Proceedings 22nd International Conference on Distributed Computing Systems*, pp. 217–226, 2002.
- [32] R. K. Balan, M. Satyanarayanan, S. Y. Park, and T. Okoshi, “Tactics-based remote execution for mobile computing,” in *Proceedings of the 1st international conference on Mobile systems, applications and services - MobiSys ’03*, (New York, New York, USA), pp. 273–286, 2003.
- [33] J. Flinn, S. Sinnamohideen, N. Tolia, and M. Satyanaryanan, “Data Staging on Untrusted Surrogates,” in *USENIX Conference on file and storage technologies (2nd: 2003: San Francisco, CA)*, pp. 15–28, Mar. 2003.
- [34] Y.-Y. Su and J. Flinn, “Slingshot: Deploying Stateful Services in Wireless Hotspots,”

- in *Proceedings of the 3rd international conference on Mobile systems, applications, and services - MobiSys '05*, p. 79, June 2005.
- [35] B.-G. Chun and P. Maniatis, “Augmented smartphone applications through clone cloud execution,” p. 8, May 2009.
- [36] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, “Calling the cloud: enabling mobile phones as interfaces to cloud applications,” in *Middleware'09 Proceedings of the ACM/IFIP/USENIX 10th international conference on Middleware*, pp. 83–102, Nov. 2009.
- [37] R. Newton, S. Toledo, L. Girod, H. Balakrishnan, and S. Madden, “Wishbone: profile-based partitioning for sensornet applications,” in *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, pp. 395–408, Apr. 2009.
- [38] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, “The Case for VM-Based Cloudlets in Mobile Computing,” *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.
- [39] M. A. Khan, “A survey of computation offloading strategies for performance improvement of applications running on mobile devices,” *Journal of Network and Computer Applications*, vol. 56, pp. 28–40, 2015.
- [40] 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, 2012.
- [41] N. Fernando, S. W. Loke, and W. Rahayu, “Mobile cloud computing: A survey,” *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.
- [42] J. Balasooriya, J. Joshi, S. K. Prasad, and S. Navathe, “Distributed Coordination of Workflows over Web Services and Their Handheld-Based Execution,” in *ICDCN'08 Proceedings of the 9th International Conference on Distributed Computing and Networking*, pp. 39–53, Jan. 2008.
- [43] Y. Sun, Y. Li, X. Wen, and Z. Zhao, “Mobile P2P Content Distribution in Wireless Networks Environment,” in *2010 International Conference on E-Business and E-Government*, May 2010.

- [44] C. Chang, S. N. Srirama, and S. Ling, "An adaptive mediation framework for mobile p2p social content sharing," in *Proceedings of the 10th International Conference on Service-Oriented Computing, ICSOC'12*, Nov. 2012.
- [45] E. Philips, A. L. Carreton, N. Joncheere, W. De Meuter, and V. Jonckers, "Orchestrating Nomadic Mashups using Workflows," in *Mashups '09/'10 The 3rd and 4th International Workshop on Web APIs and Services Mashups*, Dec. 2010.
- [46] C.-M. Huang, T.-H. Hsu, and M.-F. Hsu, "Network-aware P2P file sharing over the wireless mobile networks," *IEEE Journal on Selected Areas in Communications*, vol. 25, pp. 204–210, Jan. 2007.
- [47] U. Lee, J.-S. Park, J. Yeh, G. Pau, and M. Gerla, "CodeTorrent: Content Distribution using Network Coding in VANET," in *MobiShare'06 1st International Workshop on Decentralized Resource Sharing in Mobile Computing and Networking*, p. 1, Sept. 2006.
- [48] Z. Zong, M. Nijim, A. Manzanares, and X. Qin, "Energy Efficient Scheduling for Parallel Applications on Mobile Clusters," *Cluster Computing*, vol. 11, pp. 91–113, Nov. 2007.
- [49] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A Scalable Application Placement Controller for Enterprise Data Centers," in *WWW'07 The 16th International Conference on World Wide Web*, May 2007.
- [50] H. Shachnai and T. Tamir, "On Two Class-Constrained Versions of the Multiple Knapsack Problem," *Algorithmica*, vol. 29, pp. 442–467, Mar. 2001.
- [51] J. Sharkey, "Coding for life - Battery Life, that is.," *Google IO Developer Conference 2009*, 2009.
- [52] J. H. Ahn and M. Potkonjak, "mHealthMon: toward energy-efficient and distributed mobile health monitoring using parallel offloading," *Journal of medical systems*, vol. 37, p. 9957, Oct. 2013.
- [53] C. Doukas, T. Pliakas, and I. Maglogiannis, "Mobile healthcare information management utilizing Cloud Computing and Android OS.," *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE*

- Engineering in Medicine and Biology Society. Annual Conference*, vol. 2010, pp. 1037–40, Jan. 2010.
- [54] P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, and M. Satyanarayanan, “Scalable crowdsourcing of video from mobile devices,” in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services - MobiSys '13*, p. 139, ACM Press, June 2013.
- [55] S. Wang and S. Dey, “Rendering Adaptation to Address Communication and Computation Constraints in Cloud Mobile Gaming,” in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pp. 1–6, Dec. 2010.
- [56] Key Lime 314 LLC, “KL Dartboard,” 2011.
- [57] N. Vallina-Rodriguez and J. Crowcroft, “Energy Management Techniques in Modern Mobile Handsets,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 179–198, 2013.
- [58] IBM, “IBM CPLEX Optimizer,” June 2015.
- [59] M. Dong and L. Zhong, “Self-Constructive High-Rate System Energy Modeling for Battery-Powered Mobile Systems,” in *MobiSys'11 The 9th International Conference on Mobile systems, Applications, and Services*, 2011.
- [60] A. Pathak, Y. C. Hu, and M. Zhang, “Where is the energy spent inside my app? Fine Grained Energy Accounting on Smartphones with Eprof,” in *EuroSys'12 7th ACM european conference on Computer Systems*, ACM Press, Apr. 2012.
- [61] L. Feeney and M. Nilsson, “Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment,” in *INFOCOM'01. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society*, 2001.
- [62] A. Rahmati and L. Zhong, “Context-for-wireless: Context-Sensitive Energy-Efficient Wireless Data Transfer,” in *Proceedings of the 5th international conference on Mobile systems, applications and services - MobiSys '07*, p. 165, June 2007.

- [63] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *CODES+ISSS'10 Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, 2010.
- [64] G. Ananthanarayanan and I. Stoica, "Blue-Fi: Enhancing Wi-Fi Performance using Bluetooth Signals," in *Proceedings of the 7th international conference on Mobile systems, applications, and services - Mobisys '09*, (New York, New York, USA), p. 249, 2009.
- [65] A. Shye, B. Scholbrock, and G. Memik, "Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture - Micro-42*, p. 168, ACM Press, Dec. 2009.
- [66] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks," in *Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys '12*, p. 225, 2012.
- [67] R. E. Burkard, L. S. Pitsoulis, J. Linearization, and Q. A. P. Polytopes, "The Quadratic Assignment Problem," in *Handbook of Combinatorial Optimization*, 1998.
- [68] A. Billionnet and S. Elloumi, "Using a Mixed Integer Quadratic Programming Solver for the Unconstrained Quadratic 0-1 Problem," *Mathematical Programming*, vol. 109, pp. 55–68, June 2006.
- [69] M. S. Bazaraa and H. D. Sherali, "On the Use of Exact and Heuristic Cutting Plane Methods for the Quadratic Assignment Problem," *Journal of the Operational Research Society*, vol. 33, pp. 991–1003, Nov. 1982.
- [70] C. A. Floudas and P. M. Pardalos, eds., *Encyclopedia of Optimization*. 2009.
- [71] R. Burkard and F. Rendl, "A thermodynamically motivated simulation procedure for combinatorial optimization problems," *European Journal of Operational Research*, vol. 17, pp. 169–174, Aug. 1984.



- [72] M. R. Wilhelm and T. L. Ward, "Solving Quadratic Assignment Problems by Simulated Annealing," *IIE Transactions*, vol. 19, pp. 107–119, Mar. 1987.
- [73] Ofcom, "Ofcom publishes 4G and 3G mobile broadband speeds research," 2014.
- [74] Y. Kun, O. Shumao, and C. Hsiao-Hwa, "On effective offloading services for resource-constrained mobile devices running heavier mobile Internet applications," *IEEE Communications Magazine*, vol. 46, pp. 56–63, Jan. 2008.
- [75] K. Kumar, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?," *Computer*, vol. 43, pp. 51–56, Apr. 2010.
- [76] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *HotCloud'10 Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, p. 4, June 2010.
- [77] A. Gupta and P. Mohapatra, "Energy Consumption and Conservation in WiFi Based Phones: A Measurement-Based Study," in *2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pp. 122–131, IEEE, June 2007.
- [78] G. P. Perrucci, F. H. Fitzek, G. Sasso, W. Kellerer, and J. Widmer, "On the impact of 2G and 3G network usage for mobile phones' battery life," in *2009 European Wireless Conference*, pp. 255–259, IEEE, May 2009.
- [79] K. Lee, I. Rhee, J. Lee, S. Chong, and Y. Yi, "Mobile Data Offloading: How Much Can WiFi Deliver?," in *Proceedings of the 6th International Conference on - Co-NEXT '10*, p. 1, Nov. 2010.
- [80] G. Chen, B.-T. Kang, M. Kandemir, N. Vijaykrishnan, M. Irwin, and R. Chandramouli, "Studying energy trade offs in offloading computation/compilation in Java-enabled mobile devices," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, pp. 795–809, Sept. 2004.

- [81] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao, "Energy-accuracy trade-off for continuous mobile device location," in *Proceedings of the 8th international conference on Mobile systems, applications, and services - MobiSys '10*, p. 285, June 2010.
- [82] E. Shih, P. Bahl, and M. J. Sinclair, "Wake on wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices," in *Proceedings of the 8th annual international conference on Mobile computing and networking - MobiCom '02*, p. 160, Sept. 2002.
- [83] Z. Duan, Z.-l. Zhang, and Y. T. Hou, "Service overlay networks: SLAs, QoS, and bandwidth provisioning," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 870–883, Dec. 2003.
- [84] N. B. Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, and V. Issarny, "QoS-aware service composition in dynamic service oriented environments," in *Lecture Notes in Computer Science SpringerLink*, vol. 5896, pp. 123–142, 2009.
- [85] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz, "OverQos: an overlay based architecture for enhancing internet Qos," in *Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation, NSDI'04*, pp. 6–20, Mar. 2004.
- [86] E. Park and H. Shin, "Reconfigurable service composition and categorization for power-aware mobile computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 11, pp. 1553–1564, 2008.
- [87] D. Niyato and E. Hossain, "A Cooperative Game Framework for Bandwidth Allocation in 4G Heterogeneous Wireless Networks," in *Proceedings of IEEE International Conference on Communications*, vol. 9, pp. 4357–4362, June 2006.
- [88] L. Xu, X. Shen, and J. W. Mark, "Dynamic bandwidth allocation with fair scheduling for WCDMA systems," *IEEE Wireless Communications*, vol. 9, pp. 26–32, Apr. 2002.
- [89] H. Heredia-Ureta, F. Cruz-Perez, and L. Ortigoza-Guerrero, "Capacity optimization in multiservice mobile wireless networks with multiple fractional channel reservation," *IEEE Transactions on Vehicular Technology*, vol. 52, pp. 1519–1539, Nov. 2003.

- [90] I. Katzela and M. Naghshineh, "Channel assignment schemes for cellular mobile telecommunication systems: a comprehensive survey," *IEEE Personal Communications*, vol. 3, pp. 10–31, June 1996.
- [91] T. Erl, *Service-oriented architecture: A field guide to integrating xml and web services*. Prentice Hall, 2004.
- [92] J. Teng, B. Zhang, X. Li, X. Bai, and D. Xuan, "E-Shadow: Lubricating social interaction using mobile phones," in *Proceedings of 31st International Conference on Distributed Computing Systems, ICDCS'11*, pp. 909–918, June 2011.
- [93] M. Ferreira, R. Fernandes, H. Conceição, W. Viriyasitavat, and O. K. Tonguz, "Self-organized traffic control," in *Proceedings of the seventh ACM international workshop on Vehicular InterNetworking, VANET'10*, p. 85, Sept. 2010.
- [94] J.-S. Park, U. Lee, S. Y. Oh, M. Gerla, and D. S. Lun, "Emergency related video streaming in VANET using network coding," in *Proceedings of the 3rd international workshop on Vehicular ad hoc networks, VANET '06*, ACM Press, Sept. 2006.
- [95] H. Viswanathan, E. K. Lee, and D. Pompili, "Enabling real-time in-situ processing of ubiquitous mobile-application workflows," in *Proceedings of IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems, MASS'13*, pp. 324–332, Oct. 2013.
- [96] Y. Natchetoi, H. Wu, and Y. Zheng, "Service-Oriented mobile applications for ad-hoc networks," in *Proceedings of IEEE International Conference on Services Computing, SCC'08*, July 2008.
- [97] C. Groba and S. Clarke, "Opportunistic composition of sequentially-connected services in mobile computing environments," in *Proceedings of IEEE International Conference on Web Services, ICWS'11*, pp. 17–24, July 2011.
- [98] S. Liu and A. D. Striegel, "Exploring the potential in practice for opportunistic networks amongst smart mobile devices," in *Proceedings of the 19th annual international conference on Mobile computing & networking - MobiCom'13*, Sept. 2013.

- [99] A. Gob, D. Schreiber, L. Hamdi, E. Aitenbichler, and M. Muhlhauser, “Reducing user perceived latency with a middleware for mobile SOA access,” in *Proceedings of IEEE International Conference on Web Services, ICWS’09*, July 2009.
- [100] T. Yu, Y. Zhang, and K.-J. Lin, “Efficient algorithms for web services selection with end-to-end QoS constraints,” *ACM Transactions on the Web*, vol. 1, May 2007.
- [101] D. Ardagna and B. Pernici, “Adaptive service composition in flexible processes,” *IEEE Transactions on Software Engineering*, vol. 33, pp. 369–384, June 2007.
- [102] M. Alrifai and T. Risse, “Combining global optimization with local selection for efficient QoS-aware service composition,” in *Proceedings of the 18th International Conference on World Wide Web, WWW’09*, Apr. 2009.
- [103] P. Bodík, I. Menache, M. Chowdhury, P. Mani, D. A. Maltz, and I. Stoica, “Surviving failures in bandwidth-constrained datacenters,” in *Proceedings of ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, SIGCOMM’12*, pp. 431–442, Aug. 2012.
- [104] Y. Chen, S. Jain, V. K. Adhikari, Z.-L. Zhang, and K. Xu, “A first look at inter-data center traffic characteristics via Yahoo! datasets,” in *Proceedings of IEEE INFOCOM, INFOCOM’11*, pp. 1620–1628, Apr. 2011.
- [105] M. Zbierski and P. Makosiej, “Bring the Cloud to Your Mobile: Transparent Offloading of HTML5 Web Workers,” in *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, pp. 198–203, IEEE, Dec. 2014.
- [106] Google, “Our Mobile Planet,” 2013.
- [107] Nielsen, “Smartphones: So many apps, so much time,” 2014.
- [108] C. Shin, J.-H. Hong, and A. K. Dey, “Understanding and prediction of mobile application usage for smart phones,” in *Proceedings of ACM Conference on Ubiquitous Computing, UbiComp’12*, pp. 173–182, 2012.

- [109] E. Koutsoupias and C. Papadimitriou, “Worst-case equilibria,” in *Proceedings of the 16th annual conference on Theoretical aspects of computer science, STACS’09*, pp. 404–413, 1999.
- [110] C. Chekuri and M. Bender, “An Efficient Approximation Algorithm for Minimizing Makespan on Uniformly Related Machines,” in *Proceedings of the 6th Conference on Integer Programming and Combinatorial Optimization*, 1998.
- [111] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, “Energy consumption in mobile phones,” in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference - IMC ’09*, p. 280, 2009.
- [112] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, “Optimal cache allocation for Content-Centric Networking,” in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pp. 1–10, Oct. 2013.
- [113] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, “Cloudlets: bringing the cloud to the mobile user,” in *Proceedings of the third ACM workshop on Mobile cloud computing and services - MCS ’12*, p. 29, June 2012.

## Appendix A

# MGECP Simulation Results

Table A.1: Comparison of algorithms for MGECP - S series - Solution optimality

S Series - Test Groups <sup>†</sup>			MEGCP Solution $\mathcal{E}^\psi$ in mAh: Cost, (Cost/Optimal Cost), [Standard Deviation].							
ID	$ P ( P^C )$	$ T / R $	Base	SA	SA-HT	SA-DC	GAO	SA+GAO	GAO+SA	Optimal <sup>‡</sup>
S0	10(2)	60/90	252.20	146.15	156.82	138.98	140.56	134.08	142.89	124.87
			(2.01)	(1.17)	(1.25)	(1.11)	(1.12)	(1.07)	(1.14)	(1.00)
			[18.04]	[12.03]	[12.79]	[11.56]	[12.71]	[11.70]	[11.93]	[10.67]
S1	10(2)	60/60	229.23	127.62	137.07	121.97	124.60	118.47	124.58	110.62
			(2.07)	(1.15)	(1.23)	(1.10)	(1.12)	(1.07)	(1.12)	(1.00)
			[18.50]	[12.01]	[12.53]	[11.66]	[13.20]	[11.93]	[11.84]	[10.89]
S2	10(2)	60/120	302.67	179.15	194.02	170.14	171.43	164.22	177.04	154.56
			(1.95)	(1.15)	(1.25)	(1.10)	(1.10)	(1.06)	(1.14)	(1.00)
			[20.49]	[14.30]	[15.25]	[13.80]	[15.03]	[13.93]	[14.30]	[12.94]
S3	10(4)	60/90	195.29	101.94	107.43	98.94	107.78	99.51	99.72	91.44
			(2.13)	(1.11)	(1.17)	(1.08)	(1.17)	(1.08)	(1.09)	(1.00)
			[19.84]	[11.80]	[12.25]	[11.47]	[13.04]	[11.59]	[11.53]	[10.67]
S4	10(2)	30/45	137.83	72.83	75.77	71.36	73.95	69.21	72.89	65.36
			(2.10)	(1.11)	(1.15)	(1.09)	(1.13)	(1.05)	(1.11)	(1.00)
			[10.18]	[ 7.19]	[ 7.36]	[ 7.09]	[ 7.86]	[ 7.16]	[ 7.24]	[ 6.65]

Series Summary:	■ 2.05	■ 1.14	■ 1.21	■ 1.09	■ 1.13	■ 1.07	■ 1.12	■ 1.00
-----------------	--------	--------	--------	--------	--------	--------	--------	--------









<sup>†</sup> - Each test group contains 100 simulation instances the averages of which is used to represent the performance of the group.

<sup>‡</sup> - The optimal solution is obtained from CPLEX's QP solver.

Table A.2: Comparison of algorithms for MGECP - M series - Solution optimality

M Series - Test Groups <sup>†</sup>			MEGCP Solution $\mathcal{E}^\psi$ in mAh: Cost, (Cost/Optimal Cost), [Standard Deviation].							
ID	$ P ( P^C )$	$ T / R $	Base	SA	SA-HT	SA-DC	GAO	SA+GAO	GAO+SA	Optimal <sup>‡</sup>
M0	20(2)	120/180	607.44	419.66	475.15	383.13	329.80	331.36	408.51	297.63
			(2.04)	(1.40)	(1.59)	(1.28)	(1.10)	(1.11)	(1.37)	(1.00)
			[21.80]	[15.84]	[17.98]	[14.83]	[15.20]	[14.76]	[15.39]	[12.94]
M1	20(2)	120/120	549.39	384.75	424.16	342.46	308.41	306.77	365.97	272.86
			(2.01)	(1.41)	(1.55)	(1.25)	(1.13)	(1.12)	(1.34)	(1.00)
			[26.61]	[19.36]	[20.81]	[17.67]	[19.66]	[18.81]	[17.96]	[15.92]
M2	20(2)	120/240	715.30	504.30	563.05	459.95	394.60	396.75	500.38	359.37
			(1.99)	(1.40)	(1.56)	(1.27)	(1.09)	(1.10)	(1.39)	(1.00)
			[22.00]	[17.41]	[19.18]	[16.20]	[16.34]	[16.01]	[17.03]	[14.15]
M3	20(4)	120/180	536.15	341.53	386.90	311.83	295.05	291.96	333.71	256.24
			(2.09)	(1.33)	(1.50)	(1.21)	(1.15)	(1.13)	(1.30)	(1.00)
			[23.03]	[15.75]	[17.54]	[15.10]	[15.78]	[15.09]	[15.64]	[13.02]
M4	20(2)	60/90	328.89	203.52	221.14	187.98	177.11	172.95	198.73	158.60
			(2.07)	(1.28)	(1.39)	(1.18)	(1.11)	(1.09)	(1.25)	(1.00)
			[11.72]	[ 9.17]	[ 9.87]	[ 8.68]	[ 9.40]	[ 8.94]	[ 9.04]	[ 7.95]

Series Summary:	 2.04	 1.36	 1.52	 1.24	 1.12	 1.11	 1.33	 1.00
-----------------	--	--	--	--	--	--	--	--

<sup>†</sup> - Each test group contains 100 simulation instances the averages of which is used to represent the performance of the group.

<sup>‡</sup> - The optimal solution is obtained from CPLEX's QP solver.



Table A.3: Comparison of algorithms for MGECP - L series - Solution optimality

L Series - Test Groups <sup>†</sup>			MEGCP Solution $\mathcal{E}^\psi$ in mAh: Cost, (Cost/Optimal Cost), [Standard Deviation].							
ID	$ P ( P^C )$	$ T / R $	Base	SA	SA-HT	SA-DC	GAO	SA+GAO	GAO+SA	Optimal <sup>‡</sup>
L0	30(4)	180/270	904.11	639.40	725.73	570.41	477.75	483.82	613.68	413.85
			(2.18)	(1.54)	(1.75)	(1.37)	(1.15)	(1.16)	(1.48)	(1.00)
			[26.93]	[19.30]	[21.25]	[17.57]	[17.78]	[17.39]	[17.90]	[14.33]
L1	30(4)	180/180	816.93	575.58	645.15	508.91	434.04	440.41	539.51	374.58
			(2.18)	(1.53)	(1.72)	(1.35)	(1.15)	(1.17)	(1.44)	(1.00)
			[33.07]	[22.50]	[25.08]	[20.44]	[21.73]	[21.15]	[20.51]	[17.00]
L2	30(4)	180/360	1058.71	759.86	862.95	672.71	563.38	566.80	733.70	489.61
			(2.16)	(1.55)	(1.76)	(1.37)	(1.15)	(1.15)	(1.49)	(1.00)
			[25.85]	[19.79]	[22.03]	[17.97]	[17.74]	[17.38]	[18.69]	[14.59]
L3	30(8)	180/270	773.90	490.82	561.49	443.28	420.63	415.25	475.44	350.30
			(2.20)	(1.40)	(1.60)	(1.26)	(1.20)	(1.18)	(1.35)	(1.00)
			[27.91]	[18.80]	[20.95]	[17.60]	[18.63]	[17.88]	[18.40]	[14.66]
L4	30(4)	90/135	480.17	294.68	328.45	265.74	249.94	246.05	288.69	215.55
			(2.22)	(1.36)	(1.52)	(1.23)	(1.15)	(1.14)	(1.33)	(1.00)
			[13.51]	[ 9.95]	[10.99]	[ 9.36]	[ 9.98]	[ 9.50]	[ 9.75]	[ 8.14]

Series Summary:	■ 2.19	■ 1.48	■ 1.67	■ 1.32	■ 1.16	■ 1.16	■ 1.42	■ 1.00
-----------------	--------	--------	--------	--------	--------	--------	--------	--------

<sup>†</sup> - Each test group contains 100 simulation instances the averages of which is used to represent the performance of the group.

<sup>‡</sup> - The optimal solution is obtained from CPLEX's QP solver.

Table A.4: Comparison of algorithms for MGECP - X series - Solution optimality

X Series - Test Groups <sup>†</sup>			MEGCP Solution $\mathcal{E}^\psi$ in mAh: Cost, (Cost/Optimal Cost), [Standard Deviation].							
ID	$ P ( P^C )$	$ T / R $	Base	SA	SA-HT	SA-DC	GAO	SA+GAO	GAO+SA	Optimal <sup>‡</sup>
X0	40(4)	240/360	1252.37	999.46	1086.62	887.60	679.18	698.92	924.78	591.77
			(2.11)	(1.68)	(1.83)	(1.49)	(1.14)	(1.18)	(1.56)	(1.00)
			[30.14]	[24.11]	[25.56]	[21.62]	[20.99]	[20.77]	[20.99]	[16.93]
X1	40(4)	240/240	1101.84	890.55	965.24	803.00	630.87	641.10	821.99	542.33
			(2.03)	(1.64)	(1.77)	(1.48)	(1.16)	(1.18)	(1.51)	(1.00)
			[36.41]	[28.89]	[31.11]	[26.33]	[27.48]	[26.63]	[25.35]	[21.29]
X2	40(4)	240/480	1473.55	1164.38	1271.99	1040.54	777.43	796.29	1088.53	679.46
			(2.16)	(1.71)	(1.87)	(1.53)	(1.14)	(1.17)	(1.60)	(1.00)
			[28.07]	[22.39]	[24.05]	[20.83]	[19.05]	[18.90]	[20.23]	[15.72]
X3	40(8)	240/360	1124.08	805.36	910.34	714.17	615.75	621.79	767.64	517.63
			(2.17)	(1.55)	(1.75)	(1.37)	(1.18)	(1.20)	(1.48)	(1.00)
			[30.08]	[22.00]	[24.23]	[20.16]	[21.00]	[20.29]	[20.87]	[16.27]
X4	40(4)	120/180	671.97	455.97	507.98	410.32	356.18	356.97	441.87	307.26
			(2.18)	(1.48)	(1.65)	(1.33)	(1.15)	(1.16)	(1.43)	(1.00)
			[15.59]	[11.98]	[13.14]	[11.10]	[11.71]	[11.38]	[11.30]	[ 9.36]

Series Summary:	■ 2.13	■ 1.61	■ 1.77	■ 1.44	■ 1.16	■ 1.17	■ 1.52	■ 1.00
-----------------	--------	--------	--------	--------	--------	--------	--------	--------

<sup>†</sup> - Each test group contains 100 simulation instances the averages of which is used to represent the performance of the group.

<sup>‡</sup> - The optimal solution is obtained from CPLEX's QP solver.

Table A.5: Comparison of algorithms for MGECP - Solution time


Test Groups	MGECP Solution Time in seconds, (Ratio to Optimal Solution)							
	ID	SA	SA-HT	SA-DC	GAO	SA+GAO	GAO+SA	Optimal <sup>†</sup>
	S0	0.46 (1.94)	0.24 (1.01)	0.90 (3.79)	0.03 (0.13)	0.46 (1.94)	0.49 (2.05)	0.23 (1)
	S1	0.48 (2.07)	0.25 (1.07)	0.95 (4.07)	0.03 (0.13)	0.48 (2.07)	0.50 (2.16)	0.23 (1)
	S2	0.45 (1.64)	0.23 (0.85)	0.91 (3.29)	0.02 (0.09)	0.45 (1.64)	0.48 (1.75)	0.27 (1)
	S3	0.44 (2.34)	0.22 (1.20)	0.87 (4.63)	0.02 (0.10)	0.44 (2.34)	0.46 (2.45)	0.18 (1)
	S4	0.41 (3.22)	0.21 (1.63)	0.81 (6.28)	0.01 (0.08)	0.41 (3.22)	0.42 (3.28)	0.13 (1)
	M0	0.69 (0.21)	0.37 (0.11)	1.33 (0.42)	0.10 (0.03)	0.69 (0.21)	0.75 (0.23)	3.16 (1)
	M1	0.66 (0.26)	0.35 (0.14)	1.28 (0.52)	0.09 (0.04)	0.66 (0.26)	0.73 (0.29)	2.47 (1)
	M2	0.71 (0.21)	0.39 (0.11)	1.37 (0.40)	0.11 (0.03)	0.71 (0.21)	0.78 (0.23)	3.36 (1)
	M3	0.66 (0.34)	0.35 (0.18)	1.30 (0.67)	0.09 (0.04)	0.66 (0.34)	0.73 (0.37)	1.93 (1)
	M4	0.53 (0.58)	0.28 (0.30)	1.04 (1.14)	0.04 (0.04)	0.53 (0.58)	0.56 (0.61)	0.91 (1)
	L0	1.25 (0.18)	0.74 (0.10)	2.30 (0.33)	0.42 (0.06)	1.25 (0.18)	1.47 (0.21)	6.89 (1)
	L1	1.14 (0.16)	0.65 (0.09)	2.14 (0.31)	0.35 (0.05)	1.14 (0.16)	1.35 (0.19)	6.87 (1)
	L2	1.32 (0.20)	0.76 (0.11)	2.46 (0.38)	0.41 (0.06)	1.32 (0.20)	1.55 (0.24)	6.47 (1)
	L3	1.18 (0.19)	0.68 (0.11)	2.21 (0.37)	0.33 (0.05)	1.18 (0.19)	1.36 (0.22)	5.97 (1)
	L4	0.80 (0.32)	0.45 (0.18)	1.52 (0.61)	0.15 (0.06)	0.80 (0.32)	0.88 (0.35)	2.48 (1)
	X0	2.74 (0.13)	1.66 (0.08)	4.92 (0.23)	0.97 (0.04)	2.74 (0.13)	3.15 (0.15)	20.55 (1)
	X1	2.54 (0.12)	1.52 (0.07)	4.58 (0.21)	0.91 (0.04)	2.54 (0.12)	2.95 (0.14)	21.06 (1)
	X2	2.89 (0.14)	1.76 (0.08)	5.15 (0.25)	1.03 (0.05)	2.89 (0.14)	3.28 (0.16)	20.11 (1)
	X3	2.57 (0.13)	1.54 (0.08)	4.65 (0.24)	0.86 (0.04)	2.57 (0.13)	2.94 (0.15)	19.16 (1)
	X4	1.33 (0.19)	0.82 (0.12)	2.37 (0.35)	0.42 (0.06)	1.33 (0.19)	1.46 (0.21)	6.72 (1)

<sup>†</sup> - The optimal solution is obtained from CPLEX's QP solver.

## Appendix B

# MMUP Simulation Results

Table B.1: Comparison of algorithms for MMUP - S series - Solution optimality

Test Groups	MMUP Solutions $\max\{\mathcal{U}_i^\psi\}$ in %: Algorithm, (Algorithm/Solver), [ $\sigma\{\mathcal{U}_i^\psi\}$ ]							
ID	Base	SA	SA-HT	SA-DC	GAO	SA+GAO	GAO+SA	QCP <sup>‡</sup>
S0	6.93	3.63	3.76	3.53	4.46	3.62	3.55	5.29
	(1.30)	(0.68)	(0.71)	(0.66)	(0.84)	(0.68)	(0.67)	(1)
	[2.23]	[1.35]	[1.38]	[1.31]	[1.53]	[1.34]	[1.31]	[1.79]
S1	7.17	3.70	3.89	3.62	4.87	3.70	3.67	5.55
	(1.29)	(0.66)	(0.70)	(0.65)	(0.87)	(0.66)	(0.66)	(1)
	[2.31]	[1.34]	[1.41]	[1.31]	[1.63]	[1.33]	[1.32]	[1.88]
S2	8.76	4.70	4.86	4.60	5.96	4.70	4.66	6.99
	(1.25)	(0.67)	(0.69)	(0.65)	(0.85)	(0.67)	(0.66)	(1)
	[2.77]	[1.75]	[1.84]	[1.73]	[2.00]	[1.74]	[1.75]	[2.34]
S3	6.75	3.49	3.61	3.45	4.42	3.49	3.46	5.21
	(1.29)	(0.66)	(0.69)	(0.66)	(0.84)	(0.67)	(0.66)	(1)
	[2.35]	[1.44]	[1.49]	[1.44]	[1.67]	[1.44]	[1.45]	[1.91]
S4	3.84	1.98	2.06	1.95	2.67	1.99	1.97	2.66
	(1.44)	(0.74)	(0.77)	(0.73)	(1.00)	(0.74)	(0.74)	(1)
	[1.25]	[0.74]	[0.77]	[0.73]	[0.90]	[0.74]	[0.74]	[0.94]
Series Summary <sup>†</sup> :	 6.69	3.50	3.64	3.43	4.48	3.50	3.46	5.14

<sup>†</sup> - These are the averages of each algorithm's  $\max\{\mathcal{U}_i^\psi\}$  over all groups in this series.

<sup>‡</sup> - The CPLEX QCP solver is time-limited to run within 5 times the execution time of SA.


Table B.2: Comparison of algorithms for MMUP - M series - Solution optimality

Test Groups	MMUP Solutions $\max\{\mathcal{U}_i^\psi\}$ in %: Algorithm, (Algorithm/Solver), [ $\sigma\{\mathcal{U}_i^\psi\}$ ]							
ID	Base	SA	SA-HT	SA-DC	GAO	SA+GAO	GAO+SA	QCP <sup>‡</sup>
M0	11.19	6.14	6.55	5.86	7.54	6.03	5.90	7.58
	(1.47)	(0.80)	(0.86)	(0.77)	(0.99)	(0.79)	(0.77)	(1)
	[2.79]	[1.82]	[1.96]	[1.75]	[2.01]	[1.79]	[1.76]	[1.95]
M1	12.89	7.47	7.88	7.11	9.61	7.33	7.19	9.41
	(1.37)	(0.79)	(0.83)	(0.75)	(1.02)	(0.77)	(0.76)	(1)
	[3.10]	[2.15]	[2.22]	[2.05]	[2.38]	[2.09]	[2.04]	[2.27]
M2	11.51	6.04	6.41	5.77	7.43	5.98	5.90	7.80
	(1.47)	(0.77)	(0.82)	(0.74)	(0.95)	(0.76)	(0.75)	(1)
	[2.88]	[1.82]	[1.93]	[1.78]	[2.05]	[1.80]	[1.81]	[2.04]
M3	10.39	5.58	5.95	5.38	6.89	5.56	5.44	7.09
	(1.46)	(0.78)	(0.83)	(0.75)	(0.97)	(0.78)	(0.76)	(1)
	[2.67]	[1.82]	[1.94]	[1.78]	[1.96]	[1.81]	[1.77]	[1.87]
M4	5.61	3.11	3.25	3.02	4.08	3.10	3.04	3.99
	(1.40)	(0.77)	(0.81)	(0.75)	(1.02)	(0.77)	(0.76)	(1)
	[1.41]	[0.97]	[1.00]	[0.95]	[1.08]	[0.96]	[0.93]	[1.06]
Series Summary <sup>†</sup> :	10.32	5.67	6.01	5.43	7.11	5.60	5.49	7.17

<sup>†</sup> - These are the averages of each algorithm's  $\max\{\mathcal{U}_i^\psi\}$  over all groups in this series.

<sup>‡</sup> - The CPLEX QCP solver is time-limited to run within 5 times the execution time of SA.


Table B.3: Comparison of algorithms for MMUP - L series - Solution optimality

Test Groups	MMUP Solutions $\max\{\mathcal{U}_i^\psi\}$ in %: Algorithm, (Algorithm/Solver), [ $\sigma\{\mathcal{U}_i^\psi\}$ ]								
ID	Base	SA	SA-HT	SA-DC	GAO	SA+GAO	GAO+SA	QCP <sup>‡</sup>	
L0	16.11	8.77	9.60	8.28	10.58	8.48	8.20	10.88	
	(1.48)	(0.80)	(0.88)	(0.76)	(0.97)	(0.77)	(0.75)	(1)	
	[3.32]	[2.32]	[2.47]	[2.22]	[2.36]	[2.24]	[2.14]	[2.34]	
L1	20.31	10.16	11.10	9.36	11.96	9.57	9.32	11.54	
	(1.75)	(0.88)	(0.96)	(0.81)	(1.03)	(0.82)	(0.80)	(1)	
	[4.02]	[2.51]	[2.69]	[2.36]	[2.55]	[2.37]	[2.31]	[2.42]	
L2	15.14	7.97	8.98	7.58	9.75	7.86	7.67	10.36	
	(1.46)	(0.76)	(0.86)	(0.73)	(0.94)	(0.75)	(0.74)	(1)	
	[3.39]	[2.31]	[2.58]	[2.21]	[2.50]	[2.28]	[2.26]	[2.38]	
L3	13.97	7.83	8.50	7.27	9.49	7.63	7.39	9.18	
	(1.52)	(0.85)	(0.92)	(0.79)	(1.03)	(0.83)	(0.80)	(1)	
	[3.10]	[2.29]	[2.42]	[2.18]	[2.35]	[2.22]	[2.20]	[2.07]	
L4	8.27	4.28	4.52	4.21	5.53	4.29	4.24	5.20	
	(1.59)	(0.82)	(0.87)	(0.80)	(1.06)	(0.82)	(0.81)	(1)	
	[1.78]	[1.25]	[1.27]	[1.21]	[1.30]	[1.24]	[1.21]	[1.19]	
Series Summary <sup>†</sup> :		12.76	7.80	8.54	7.34	9.46	7.56	7.37	9.43

<sup>†</sup> - These are the averages of each algorithm's  $\max\{\mathcal{U}_i^\psi\}$  over all groups in this series.

<sup>‡</sup> - The CPLEX QCP solver is time-limited to run within 5 times the execution time of SA.

Table B.4: Comparison of algorithms for MMUP - L series - Solution optimality

Test Groups	MMUP Solutions $\max\{\mathcal{U}_i^\psi\}$ in %: Algorithm, (Algorithm/Solver), [ $\sigma\{\mathcal{U}_i^\psi\}$ ]								
ID	Base	SA	SA-HT	SA-DC	GAO	SA+GAO	GAO+SA	QCP <sup>‡</sup>	
X0	18.28	10.78	11.98	10.43	12.22	10.08	9.66	11.83	
	(1.54)	(0.91)	(1.01)	(0.88)	(1.03)	(0.85)	(0.81)	(1)	
	[3.42]	[2.55]	[2.79]	[2.52]	[2.45]	[2.36]	[2.27]	[2.32]	
X1	23.59	14.42	16.61	13.60	17.09	13.86	13.52	16.97	
	(1.39)	(0.84)	(0.97)	(0.80)	(1.00)	(0.81)	(0.79)	(1)	
	[4.10]	[2.99]	[3.31]	[2.92]	[3.06]	[2.87]	[2.82]	[3.02]	
X2	17.64	9.26	10.39	8.59	10.70	8.86	8.57	11.32	
	(1.55)	(0.81)	(0.91)	(0.75)	(0.94)	(0.78)	(0.75)	(1)	
	[3.49]	[2.50]	[2.73]	[2.35]	[2.51]	[2.38]	[2.31]	[2.37]	
X3	19.27	11.45	12.32	10.60	13.49	10.99	10.61	13.18	
	(1.46)	(0.86)	(0.93)	(0.80)	(1.02)	(0.83)	(0.80)	(1)	
	[3.63]	[2.79]	[2.94]	[2.61]	[2.76]	[2.69]	[2.59]	[2.52]	
X4	9.78	5.66	6.06	5.44	7.23	5.60	5.50	7.02	
	(1.39)	(0.80)	(0.86)	(0.77)	(1.02)	(0.79)	(0.78)	(1)	
	[1.88]	[1.41]	[1.50]	[1.38]	[1.47]	[1.39]	[1.38]	[1.41]	
Series Summary <sup>†</sup> :		11.91	7.31	6.27	6.73	6.74	6.88	6.57	6.66

<sup>†</sup> - These are the averages of each algorithm's  $\max\{\mathcal{U}_i^\psi\}$  over all groups in this series.

<sup>‡</sup> - The CPLEX QCP solver is time-limited to run within 5 times the execution time of SA.



Table B.5: Comparison of algorithms for MMUP - Solution time

Test Groups	MMUP Solution Time in Seconds, (Ratio to QCP's Time)						
ID	SA	SA-HT	SA-DC	GAO	SA+GAO	GAO+SA	QCP <sup>†</sup>
S0	0.83 (0.27)	0.42 (0.14)	1.64 (0.54)	0.02 (0.00)	0.83 (0.28)	0.84 (0.28)	2.99 (1)
S1	0.83 (0.30)	0.41 (0.15)	1.61 (0.59)	0.01 (0.00)	0.83 (0.30)	0.83 (0.30)	2.74 (1)
S2	0.82 (0.25)	0.41 (0.12)	1.64 (0.50)	0.01 (0.00)	0.82 (0.25)	0.82 (0.25)	3.24 (1)
S3	0.80 (0.28)	0.40 (0.14)	1.58 (0.56)	0.01 (0.00)	0.80 (0.28)	0.80 (0.28)	2.81 (1)
S4	0.76 (0.27)	0.38 (0.13)	1.49 (0.53)	0.00 (0.00)	0.76 (0.27)	0.77 (0.27)	2.78 (1)
M0	2.09 (0.33)	1.08 (0.17)	4.11 (0.66)	0.06 (0.01)	2.10 (0.34)	2.09 (0.33)	6.17 (1)
M1	1.99 (0.39)	1.01 (0.20)	3.89 (0.76)	0.04 (0.00)	2.00 (0.39)	1.98 (0.39)	5.07 (1)
M2	2.07 (0.33)	1.05 (0.17)	4.03 (0.65)	0.07 (0.01)	2.08 (0.33)	2.06 (0.33)	6.20 (1)
M3	1.95 (0.40)	0.98 (0.20)	3.80 (0.79)	0.06 (0.01)	1.96 (0.40)	1.95 (0.40)	4.79 (1)
M4	1.54 (0.45)	0.79 (0.23)	3.06 (0.90)	0.02 (0.01)	1.54 (0.45)	1.55 (0.45)	3.38 (1)
L0	7.56 (0.33)	3.89 (0.17)	14.80 (0.66)	0.26 (0.01)	7.60 (0.33)	7.60 (0.33)	22.43 (1)
L1	7.01 (0.34)	3.58 (0.17)	13.64 (0.66)	0.23 (0.01)	7.06 (0.34)	7.01 (0.34)	20.55 (1)
L2	7.60 (0.35)	3.90 (0.18)	14.81 (0.68)	0.29 (0.01)	7.63 (0.35)	7.70 (0.35)	21.69 (1)
L3	6.78 (0.35)	3.44 (0.17)	13.32 (0.69)	0.20 (0.01)	6.81 (0.35)	6.85 (0.35)	19.24 (1)
L4	5.04 (0.42)	2.58 (0.21)	9.95 (0.84)	0.11 (0.00)	5.06 (0.42)	5.07 (0.42)	11.81 (1)
X0	15.46 (0.33)	8.14 (0.17)	30.29 (0.65)	0.86 (0.01)	15.58 (0.33)	15.73 (0.34)	45.99 (1)
X1	13.16 (0.36)	6.86 (0.18)	26.25 (0.72)	0.56 (0.01)	13.22 (0.36)	13.60 (0.37)	36.22 (1)
X2	14.56 (0.30)	7.62 (0.16)	28.87 (0.60)	0.74 (0.01)	14.64 (0.30)	14.96 (0.31)	47.44 (1)
X3	13.38 (0.34)	7.15 (0.18)	26.15 (0.67)	0.84 (0.02)	13.45 (0.34)	13.84 (0.35)	38.49 (1)
X4	9.09 (0.37)	4.71 (0.19)	17.69 (0.72)	0.39 (0.01)	9.12 (0.37)	9.06 (0.37)	24.36 (1)

† - The CPLEX QCP solver is time-limited to run within 5 times the execution time of SA.

## Appendix C

### Derivation of $p_i^{\mathbb{R}}$

When a game is in a state of mixed-strategy equilibrium, we have  $\mathbb{E}[c_i^{\mathbb{R}}] = \mathbb{E}[c_i^{\mathbb{N}}]$ . This with (6.6) we get

$$\begin{aligned} \mathbb{E}[C_{\mathbb{R}}] + (1 - p_i^{\mathbb{R}}) \left( \frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b} \right) &= \mathbb{E}[C_{\mathbb{N}}] + (1 - p_i^{\mathbb{N}}) \left( \frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b} \right) = \mathbb{E}[C_{\mathbb{N}}] + p_i^{\mathbb{R}} \left( \frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b} \right) \\ p_i^{\mathbb{R}} \left( \frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b} + \frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b} \right) - \left( \frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b} \right) &= \mathbb{E}[C_{\mathbb{R}}] - \mathbb{E}[C_{\mathbb{N}}] \end{aligned} \quad (\text{C.1})$$

For applications that are fixed to run on either  $\mathbb{N}$  or  $\mathbb{R}$ , i.e.  $i \in [n]^{\mathbb{N}} \cup [n]^{\mathbb{R}}$  we define

$$C_j^f = \sum_{i \in [n]^j} \left( \frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right), \quad j \in \{\mathbb{N}, \mathbb{R}\} \quad (\text{C.2})$$

Take this into (6.4) we have

$$\mathbb{E}[C_{\mathbb{N}}] = C_{\mathbb{N}}^f + \sum_{i \in [n] - [n]^{\mathbb{N}}} p_i^{\mathbb{N}} \left( \frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b} \right) \quad \text{and} \quad \mathbb{E}[C_{\mathbb{R}}] = C_{\mathbb{R}}^f + \sum_{i \in [n] - [n]^{\mathbb{R}}} p_i^{\mathbb{R}} \left( \frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b} \right) \quad (\text{C.3})$$

Take these into (6.11) we have

$$\mathbb{E}[C_{\mathbb{N}}] = C_{\mathbb{N}}^f + \sum_{i \in [n] - [n]^{\mathbb{N}}} a_i^{\mathbb{N}} \left( \mathbb{E}[C_{\mathbb{N}}] - \mathbb{E}[c_i^{\mathbb{N}}] + \left( \frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b} \right) \right) \quad (\text{C.4})$$

$$\mathbb{E}[C_{\mathbb{R}}] = C_{\mathbb{R}}^f + \sum_{i \in [n] - [n]^{\mathbb{R}}} a_i^{\mathbb{R}} \left( \mathbb{E}[C_{\mathbb{R}}] - \mathbb{E}[c_i^{\mathbb{R}}] + \left( \frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b} \right) \right) \quad (\text{C.5})$$

Take a difference between these two equations we have

$$\mathbb{E}[C_{\mathbb{R}}] - \mathbb{E}[C_{\mathbb{N}}] = C_{\mathbb{R}}^f - C_{\mathbb{N}}^f + |[n]^{\mathbb{H}}| (\mathbb{E}[C_{\mathbb{R}}] - \mathbb{E}[C_{\mathbb{N}}]) + \sum_{k \in [n]^{\mathbb{H}}} \left( \frac{w_k^d}{s_{\mathbb{R}}^d} + \frac{w_k^b}{s_{\mathbb{R}}^b} \right) - \left( \frac{w_k^d}{s_{\mathbb{N}}^d} + \frac{w_k^b}{s_{\mathbb{N}}^b} \right) \quad (\text{C.6})$$

$$\mathbb{E}[C_{\mathbb{R}}] - \mathbb{E}[C_{\mathbb{N}}] = \left( C_{\mathbb{R}}^f - C_{\mathbb{N}}^f + \sum_{k \in [n]^{\mathbb{H}}} \left( \frac{w_k^d}{s_{\mathbb{R}}^d} + \frac{w_k^b}{s_{\mathbb{R}}^b} \right) - \left( \frac{w_k^d}{s_{\mathbb{N}}^d} + \frac{w_k^b}{s_{\mathbb{N}}^b} \right) \right) / \left( 1 - |[n]^{\mathbb{H}}| \right) \quad (\text{C.7})$$

Finally, compare this with (C.1) we get

$$p_i^{\mathbb{R}} \left( \frac{w_k^d}{s_{\mathbb{N}}^d} + \frac{w_k^b}{s_{\mathbb{N}}^b} + \frac{w_k^d}{s_{\mathbb{R}}^d} + \frac{w_k^b}{s_{\mathbb{R}}^b} \right) - \left( \frac{w_k^d}{s_{\mathbb{R}}^d} + \frac{w_k^b}{s_{\mathbb{R}}^b} \right) = \left( C_{\mathbb{R}}^f - C_{\mathbb{N}}^f + \sum_{k \in [n]^{\mathbb{H}}} \left( \frac{w_k^d}{s_{\mathbb{R}}^d} + \frac{w_k^b}{s_{\mathbb{R}}^b} \right) - \left( \frac{w_k^d}{s_{\mathbb{N}}^d} + \frac{w_k^b}{s_{\mathbb{N}}^b} \right) \right) / \left( 1 - |[n]^{\mathbb{H}}| \right) \quad (\text{C.8})$$

$$p_i^{\mathbb{R}} = \left( \frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b} \right) / \left( \frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b} + \frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b} \right) + \left( C_{\mathbb{R}}^f - C_{\mathbb{N}}^f + \sum_{k \in [n]^{\mathbb{H}}} \left( \frac{w_k^d}{s_{\mathbb{R}}^d} + \frac{w_k^b}{s_{\mathbb{R}}^b} \right) - \left( \frac{w_k^d}{s_{\mathbb{N}}^d} + \frac{w_k^b}{s_{\mathbb{N}}^b} \right) \right) / \left( \left( 1 - |[n]^{\mathbb{H}}| \right) \left( \frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b} + \frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b} \right) \right) \quad (\text{C.9})$$