Control Theory for Computing Systems: Application to big-data cloud services & location privacy protection

Sophie Cerf

To cite this version:

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

**DOCTEUR DE LA
COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES**

Spécialité : **AP - Automatique-Productique**

Arrêté ministériel : 25 mai 2016

Présentée par

Sophie Cerf

Thèse dirigée par **Nicolas Marchand**, DR CNRS
et codirigée par **Bogdan Robu**, CR CNRS

préparée au sein du **GIPSA-lab**
et de l' **École doctorale « Électronique, Électrotechnique, Automatique et Traitement du Signal »**

Control Theory for Computing Systems: Application to big-data cloud services & location privacy protection.

Thèse soutenue publiquement le **16 mai 2019**
Jury composé de :

Madame Vana Kalogeraki
Associate Professor, Athens University, Rapporteur
Monsieur Karl-Erik Arzen
Professor, Lund University, Rapporteur
Monsieur Eric Kerrigan
Reader, Imperial College London, Examinateur
Madame, Sara Bouchenak
Professor, INSA Lyon, Présidente
Monsieur Nicolas Marchand
Directeur de recherche, CNRS délégation Alpes, Directeur de thèse
Monsieur Bogdan Robu
Maître de Conférences, Université Grenoble-Alpes, Co-Directeur de thèse

Contents

Contents i

List of Figures v

List of Tables viii

I Generalities 3

1 Introduction 5
- 1.1 Context, Motivation and Applications . . . 5
- 1.2 Main Results and Collaborations . . . 7
  - 1.2.1 Publications . . . 7
  - 1.2.2 Collaborations . . . 8
  - 1.2.3 Technical contributions . . . 9
- 1.3 Thesis Outline . . . 9
- 1.4 Reading Roadmap . . . 11

2 Background and Motivation 13
- 2.1 Basics on Control Theory . . . 13
  - 2.1.1 Assumptions and Representations . . . 13
  - 2.1.2 Objectives . . . 14
  - 2.1.3 Tools of a Control Engineer . . . 14
  - 2.1.4 Applications . . . 17
- 2.2 On the use of Control Theory for Computing Systems . . . 17
  - 2.2.1 Eligible Computing Systems . . . 18
  - 2.2.2 Promises of control for software adaptation . . . 18
  - 2.2.3 Benefits . . . 19
  - 2.2.4 Challenges . . . 19

3 Related Work 21
- 3.1 Monitoring Techniques for Software Adaptation . . . 21
  - 3.1.1 Observations-based rules . . . 21
  - 3.1.2 MAPE-K . . . 22
  - 3.1.3 Queuing Theory . . . 22
  - 3.1.4 Game Theory . . . 23
  - 3.1.5 Machine Learning . . . 24
  - 3.1.6 Discrete Event Systems . . . 25
- 3.2 On the Use of Control Theory for Computing Systems . . . 26

3.2.1 Motivation for using Control Theory 26
3.2.2 Application fields 26
3.2.3 Objectives 27
3.2.4 Sensors and performance metrics 27
3.2.5 Actuators and control signal 27
3.2.6 Modeling 27
3.2.7 Control 28
3.2.8 Evaluation 28
3.2.9 Publications trends 28
3.2.10 Limitations and open challenges 28

4 Objectives and Contributions 31
4.1 Privacy and Utility for Mobility Data 31
4.2 Performance and Reliability of Cloud Services 33
4.3 Contributions of the Thesis 35
4.3.1 Automatic Choice and Configuration of Location Privacy Protection Mechanisms 35
4.3.2 Dynamic Control of Utility and Location Privacy 35
4.3.3 Adaptive Control for Cloud Service Performance Robust to Plant and Environment Changes 36
4.3.4 Cost-aware Optimal Control of Performance and Availability of Cloud Services 36

II Privacy and Utility Aware Control of Users' Mobility Data 37

5 Location Privacy Background and Related Work 41
5.1 Mobility traces and datasets 41
5.2 Notions of Location Privacy 43
5.2.1 Threats 43
5.2.2 Definitions 44
5.3 Location Privacy Protection Mechanisms 44
5.4 Evaluation of LPPMs 47
5.4.1 Privacy metric 48
5.4.2 Utility metric 49
5.4.3 Illustration of Privacy and Utility Metrics with LPPMs 51
5.5 LPPM Configuration 52

6 PULP: Privacy and Utility through LPPMs Parametrization 55
6.1 Introduction 55
6.1.1 Description 55
6.1.2 Problem Statement 56
6.1.3 Proposed Approach 57
6.2 PULP Framework 57
6.2.1 Profiler 58
6.2.2 Modeler 58
6.2.3 Configurator 59
6.3 Configuration Laws 60
6.3.1 *PULP* 's Ratio-Based Configuration Law 60
6.3.2 $\mathscr{P}$-thld Law: Privacy Above a Minimum Threshold 61

6.3.3 $\mathscr{U}$-thld Law: Utility Above a Minimum Threshold . . . 62
6.3.4 $\mathscr{PU}$-thld : Privacy and Utility Above Minimum Thresholds . . . 62
6.4 *PULP* Evaluation . . . 63
6.4.1 Experimental Setup . . . 63
6.4.2 Evaluation of the *PULP* Modeler . . . 64
6.4.3 Evaluation of the *PULP* Configurator . . . 66
6.4.4 Comparison with State of the Art . . . 68
6.5 Conclusion . . . 69

7 dynULP: dynamic control of Utility and Location Privacy 75
7.1 Introduction . . . 75
7.1.1 Context, Hypothesis and Motivation . . . 75
7.1.2 Proposed Approach . . . 76
7.2 Control Problem Formulation . . . 77
7.2.1 Overview . . . 77
7.2.2 Formalization . . . 77
7.2.3 Illustrated motivation . . . 80
7.3 Privacy Modeling . . . 82
7.3.1 Overview . . . 82
7.3.2 Inputs Scenario . . . 82
7.3.3 Static Characterization . . . 82
7.3.4 Dynamic modeling . . . 84
7.4 Control Strategy . . . 85
7.4.1 Objectives . . . 85
7.4.2 Linear Control Problem Formulation . . . 86
7.4.3 PI Formulation . . . 86
7.4.4 Anti-windup . . . 86
7.5 dynULP Evaluation . . . 87
7.5.1 Methodology . . . 87
7.5.2 Modeling . . . 89
7.5.3 Control . . . 94
7.6 Conclusion . . . 100

8 Conclusions on Location Privacy 101

IIIPerformance and Reliability of Hadoop Cloud Services 103

9 BigData Cloud Services: Background and Related Works 107
9.1 Cloud Services . . . 107
9.2 A BigData Processing Framework: Hadoop/MapReduce . . . 108
9.3 Problem Statement . . . 109
9.4 Related Works . . . 111
9.4.1 Cloud Performance Monitoring . . . 111
9.4.2 MapReduce Performance Monitoring . . . 111
9.4.3 Benchmarking and Platforms . . . 112
9.5 Background on Control Theory applied to MapReduce . . . 113
9.5.1 Input and output signals . . . 113
9.5.2 Modeling . . . 114

9.5.3 Control 115
9.5.4 Limitations 116

10 Adaptive Control for Robust Cloud Services 117
10.1 Introduction 117
10.1.1 Context and Approach 117
10.1.2 Illustrated Motivation 118
10.2 Control Strategy 120
10.2.1 Preliminary Formulation 120
10.2.2 Going beyond hypotheses 121
10.3 Adaptive Controller Evaluation 123
10.3.1 Evaluation Setup 123
10.3.2 Simulation Results 124
10.3.3 Experimental Results 127
10.4 Conclusion 130

11 Cost-aware Control of Cloud Services 131
11.1 Introduction 131
11.2 Preliminaries 134
11.3 Control Strategy 136
11.3.1 The Lyapunov based triggering strategy 136
11.3.2 Stability of the proposed scheme 136
11.4 Evaluation 137
11.4.1 Experimental conditions and methodology 137
11.4.2 Cost based event triggering mechanism validation 138
11.4.3 Input-constrained control 138
11.4.4 Evaluation using a real MapReduce workload 141
11.5 Conclusion 143

12 Conclusions on MapReduce Control 145

IV Conclusions and Perspectives 147

V Résumé en français 151

Bibliography 159

VI Appendix: Machine learning algorithms as systems to control 173

A Duo Learning for Classifications with Noisy Labels 175

B Robust Anomaly Detection on Unreliable Data 181

C Feedback Control for Online Training of Neural Networks 191

List of Figures

2.1 Block diagram: control representation of a system 13
2.2 High-level representation of a controlled system 15
2.3 Simplest controlled system: a Proportional Controller 15

3.1 MAPE-K autonomic manager [98]. 22
3.2 Queuing basic model [89]. 22
3.3 Game theory payoff matrix, basic example. 24
3.4 Machine learning simple example: k-nearest neighbor classifier [177]. 25
3.5 Discrete Event System Control Schema[156]. 26

4.1 Location Based Services 32
4.2 Usages and threats of mobility data when using a LBS. 32
4.3 Location Privacy Protection Mechanism 32
4.4 Common Cloud Services 33
4.5 News breaks showing that unmastered resource leads to service unavailability. All outages were caused by unusual workload, where some where however expected to be of huge amplitude. 34

5.1 GPS data are dynamic. Mobility data of a user from Cabspotting dataset illustration: (a) on a map and (b) trough speed over time of various activities (a stop and two movements). 43
5.2 Synthetic mobility trace scenario. 43
5.3 Obfuscation of a mobility trace using Geo-I. Illustration on a Cabspotting user for various configurations. (a) user raw trace, (b) obfuscation with $\varepsilon = 10^{-2}$, (c) obfuscation with $\varepsilon = 10^{-2.5}$. 46
5.4 Obfuscation of a mobility trace using Promesse. Illustration on a Cabspotting user for various configurations. (a) user raw trace, (b) obfuscation with $\varepsilon = 400\,\text{m}$, (c) with $\varepsilon = 1000\,\text{m}$. 47
5.5 Privacy computation: schematic examples of how POI retrieval is computed for a single user when using Geo-I and Promesse. 51
5.6 Utility computation: schematic examples of how cell coverage is computed for a single user when using Geo-I and Promesse. 52

6.1 The same LPPM can result in different privacy and utility metrics: examples from 4 users using Promesse with $\varepsilon = 100\,\text{m}$ and Geo-I with two different configurations: $\varepsilon 1 = 0.01\,\text{m}^{-1}$ and $\varepsilon 2 = 0.005\,\text{m}^{-1}$. 56
6.2 *PULP* framework 57
6.3 Impact of LPPMs configuration on a user's privacy and utility metrics – Real system vs. modeled system (for one cabspotting user used as an example). 59
6.4 Automatic configuration laws in *PULP* 59
6.5 Illustration of $\mathscr{PU}$-ratio configuration law for three schematic LPPMs with $w_{pr/ut} = 2$. . 61

6.6 *PULP* modeler accuracy. Cumulative distribution function (cdf) among all users. 65
6.11 Execution time comparison, Geolife dataset 68
6.7 $\mathscr{PU}$-ratio configuration law evaluation. (a) Recommended LPPM and its configuration (b) for Geo-I, (c) for Promesse. Achieved (d) level of privacy and (e) utility when users are protected according to *PULP* recommendations, and the corresponding (f) privacy to utility ratio. Four objective ratios $w_{pr/ut}$ are illustrated: 0.5, 1, 2 and 3. 70
6.8 $\mathscr{PU}$-thld configuration law evaluation. (a) Recommended LPPM and its configuration (b) for Geo-I, (c) for Promesse. Achieved (d) level of privacy and (e) utility when users are protected according to *PULP* recommendations, and the corresponding (f) privacy to utility ratio. Five objective couples of constraints on privacy and utility are illustrated. . . 71
6.9 $\mathscr{P}$-thld configuration law evaluation. (a) Recommended LPPM and its configuration (b) for Geo-I, (c) for Promesse. Achieved (d) level of privacy and (e) utility when users are protected according to *PULP* recommendations, and the corresponding (f) privacy to utility ratio. Five objective constraints on privacy Pr_{min} are illustrated: 0.3, 0.5, 0.7, 0.8, 0.9. 72
6.10 $\mathscr{U}$-thld configuration law evaluation. (a) Recommended LPPM and its configuration (b) for Geo-I, (c) for Promesse. Achieved (d) level of privacy and (e) utility when users are protected according to *PULP* recommendations, and the corresponding (f) privacy to utility ratio. Five objective constraints on utility Ut_{min} are illustrated: 0.3, 0.5, 0.7, 0.8, 0.9. 73

7.1 Working scenario: location data are sent continuously and the service is received constantly 76
7.2 Classic control schema applied for privacy protection of a location based service user. . . 77
7.3 Privacy metric computation on a simple mobility trace. 80
7.4 Privacy and utility variations over time for various static LPPM configurations. Data from a cabspotting user. 81
7.5 Static characteristics of the ε to privacy function, for various disturbance scenario. 83
7.6 System step response. Input from $\varepsilon = 10^{-1}\,\mathrm{m}^{-1}$ to $\varepsilon = 10^{-2}\,\mathrm{m}^{-1}$, during a stop (constant disturbance). 84
7.7 Feedback loop with signal and transfer functions notations 86
7.8 Privacy dynamic model validation with control variable steps. 90
7.9 Privacy dynamic model validation with user's speed steps. 90
7.10 Privacy dynamic model validation on substantial synthetic data, constant control signal. . 92
7.11 Privacy dynamic model validation on substantial synthetic data and control signal. 92
7.12 Privacy dynamic model validation on a real mobility trace, with constant control signal. . 93
7.13 Privacy dynamic model validation on a real mobility trace with a realistic control signal. 93
7.14 Dynamical privacy control of a stopped user with step privacy specifications. 94
7.15 Dynamical privacy control with constant privacy specifications, with user's speed steps. . 95
7.16 Dynamical privacy control with constant privacy specifications on substantial synthetic data. 96
7.17 Dynamical privacy control with substantial synthetic data and privacy specifications. . . 97
7.18 Dynamical privacy control on a real mobility trace, with constant privacy specifications. . 98
7.19 Dynamical privacy control, real mobility trace with a privacy specifications scenario. . . 99

9.1 Cloud IaaS platform adoption in 2017, percentage of applications deployed. [9] 108
9.2 MapReduce step-wise functional scheme. [60] 109
9.3 MapReduce Service Time model. 114
9.4 MapReduce Service Time and Availability model. 115

10.1 MapReduce State of the Art Modeling. [25] 118

10.2 MapReduce State of the Art Feedback and Feedforward Control experimental performance for a comprehensive client disturbance scenario. 119
10.3 MapReduce adaptive control schema. 120
10.4 Client disturbance Scenario 123
10.5 Bode diagram of the transfer functions given in eq. (10.7) and (10.14) a) without PI b) with PI for $\hat{a}_N = a_N = -0.92$, $\hat{a}_N = -0.84$ and $\hat{a}_N = -0.96$ 124
10.6 Controllers performance comparison with overestimated model gain. 125
10.7 Controller performance with characteristic time variation. 125
10.8 Controller performance for various plant gain scenarios, with and without PI 126
10.9 Experimental performance in Open Loop (no control). 127
10.10Experimental performance with the adaptive feedforward and feedback controller. 128
10.11Experimental performance of the adaptive controller for a comprehensive workload. . . . 129

11.1 On-line shopping website - MapReduce workload on a 2,000-node cluster [155] 132
11.2 Comparison of time-based, error-based and cost-based controller performance without constraints on the number of allowed reconfigurations 139
11.3 Comparison of cost-based controller performance with and without constraints on the control changing time 140
11.4 Cost-based event function 141
11.5 Evaluation of cost-based controllers on a real 1-day MapReduce workload scaled to our cluster size. 142

List of Tables

5.1 30-days Mobility trace datasets 42

5.2 Overview of Location-Based Services and the location properties used ('+' for main purpose and '-' for accessory need) to satisfy the end-user. 49

6.1 *PULP* output for selected users 66

7.1 dynULP parameters sum-up. 88

10.1 Experimental controllers performance comparison on the simple workload scenario. Best results are in bold. 130

11.1 Cost comparison of different solutions based on a 1-day real workload 142

Acknowledgements

Science, in its research form, aims at escaping from subjectivity and biases, but in the end is always carried out by human beings.

A PhD thesis is known to be the apotheosis of an individual's educational path, but in the end, it exists only thanks to fruitful and benevolent interactions.

State of the art contributions seems to bring the society further, but in the end, the tools go beyond their originator and are as righteous as the people using them.

I dedicate this chapter for acknowledging all those who participated in this work, in all the possible ways; and to those who will use it as a humble stone for building their own project, may they do it consciously and serenely.

In the first place, I would like to thank my advisors Bogdan, Sara and Nicolas for their directions, advises, ideas, support, encouragements, devotion and fellowship. During this four-year experience, they taught me how to independently carry out researches while taking advantage of any collaboration. Aside from the scientific quality of their work, they are devoted teachers.

I had the great opportunity to collaborate with researchers from diverse horizon, I thank Sonia Ben Mokhtar, Antoine Boutet, Lydia Y. Chen, Robert Birke and Ioan D. Landau. My special thanks go also to the (former) students with whom I directly worked with, Mihaly Berekmeri, Vincent Primault, Mohamed Maouche and Zilong Zhao. They were very dedicated in explaining and sharing their works, they contributed a lot in this thesis.

This work would not have been qualified as a PhD contribution without careful reviews and examinations. I thank the jury members Vana Kalogeraki, Karl-Erik Arzen and Eric Kerrigan for the time and expertise they gave during this process.

The research environment is as important as all of the above mentioned contributors of the manuscript. Thanks to my colleagues from LIRIS and Gipsa-lab, the fruitful discussions with my fellow researcher and the great help of technical and administrative services, which importance is way too often forgotten.

Aside from the scientific aspect, a PhD is also psychological challenge. All my gratefulness goes to my relatives that helped me in this experience. To my dear friends from the lab that always had a beer and a card game at the rightful time. All the cheerful friends with whom I share my daily, my weekends and my reflections. À ma famille, pour leur soutien inconditionnel. And to the one that fits in all those categories.

Eventually, thanks to you, reader of this piece of work, to hopefully make it contribute to the world's knowledge and expertise, and not let it as a great but vain experience.

Part I

Generalities

Chapter 1

Introduction

This first chapter presents the motivation of the approach taken in this thesis of using control theory as a tool for software adaptation, and its application to two use-cases. Beforehand, some useful context is detailed on those key notions. The present manuscript's outlines are given, and reading advises are provided as a concluding section.

1.1 Context, Motivation and Applications

Computing systems, either for personal use or in the industrial world, are constantly growing in amount and complexity. Smartphones provide user-based services such as personalized recommendations or adapted roaming. Companies have access to growing amounts of information about their clients and/or their environment and use them to develop business-intelligence based analytics and services. These two simple examples highlight that computing tools are used, directly or indirectly, with various degrees of consciousness and apprehension, by most people in their everyday life.

The last decade has seen the culmination of differentiation between software and hardware. Cloud services have emerged as the new computing paradigms up to a point where companies that have failed following this trend are now facing major difficulties. Its principle is to provide access to hardware infrastructures or services and take care of its maintenance so that clients can deploy their own application without having to deal with hardware or operating systems issues. Cloud providers own datacenters all around the globe and their clients can access those resources distantly. Customers and contents of the cloud are extremely diverse, ranging from a simple person using Gmail or iCloud Storage to international companies or organizations running large data analytics. This example shows the interconnectivity and concurrency of softwares sharing the same resources and networks. Another every-day life illustration of such dependencies - this time outside of the cloud world - is the battle for computing and energy resources lead by applications and operating system on a smartphone. Data, tasks and resources need to be managed to maximize the quality of services.

Softwares are highly dependent on the environment in which they are running, and this environment is constantly changing [98, 144]. Whether it is their workload or the concurrent applications, the running environment of softwares are highly dynamic. A recent example, particularly timely for researchers, is the outage of the latex cloud service Overleaf that happened synchronously to a famous machine learning conference [145]. At another scale, the hardware itself on which the software is running can change, sometimes even constantly and unpredictably if the service is run on the cloud [93]. It is also quite often that softwares have to run on hardware that did not even exist when it has been designed, and that sometimes implies disrupting technologies.

Users and clients are asking for continuous availability, optimal performance and reliability; and they are always more demanding. Moreover, they are asking for guarantees of those properties, often embodied in service-level agreements (SLAs). Meeting SLA is a key challenge for all service providers. Eventually, the clients requirements are sometimes changing during runtime of the software, requiring system adaptation [164].

All those changes of context and requirements are enforced on the software even though this later has incomplete knowledge about its environment (specifically in the case of cloud-based services). Its working conditions are most of the time uncertain but also unpredictable, e.g. workload estimation.

Another important point is the energy consumption of large or small computing systems and, to a larger extent, their running costs. Each non optimal decisions either on the software runtime strategy or on the resources usage leads in useless extra costs. Given the scale of datacenters, the spread of mobile devices and the always increasing usage of computing systems, wise and sparing planning of resources need to be achieved.

Eventually, given the increasing complexity of computing tools, the question of their practical usage, either by technical experts or simple users, raises. State-of-the art technologies mean always more specialized experts to install them, configure them and monitor their behavior in real time, at a point where these tasked are hardly feasible by humans anymore. On the other side of the pipeline, end-user are more and more non-experts that hardly benefited from education on computing and software usage. The usage of software tools should be adaptable to all kind of users.

A solution to all these challenges is software adaptation. It is a discipline of software engineering which aims at modifying the software's parameters/mode/resources in reaction to changes of its own behavior or of its environment. Many different tools can be used to realize adaptation in practice, such as queuing theory, machine learning or control theory. In this work, we choose to focus on this last technique. Control theory is an engineering field that aims at monitoring dynamic systems, through the use of feedback loops. It started to be well established since the early 1900's, for its applications to industrial systems, notably in aeronautics. Due to its history for systems ruled by physics' laws, it is only since the late 2000's that its application to computing systems has been investigated. However, its strong mathematical background and its formal guarantees on the results makes control theory a prevalent solution for software adaptation. This is well illustrated by the significant rise in the number of papers about this topic [162].

In order to illustrate the potential of the use of control theory for software adaptation, two complementary computing systems will be studied. The first one deals with one side of the computing world, namely the user of smartphone. More specifically, we investigate the protection of people's privacy when sharing their localization through applications. Particular attention is paid on the quality of the geo-localized service received. The literature on the use of control theory for privacy issues is almost non-existing, which makes this use-case a real playground where everything is to be built, from the definition of user's objectives to the privacy-ensuring setup. Cyber-privacy challenges are of prior importance for the people but still lack of solutions. Moreover, the specific properties of such new formulation of control problems has a lot to bring to the research community.

The second use-case is the monitoring of performance and reliability of cloud-based BigData services, through resource allocation and admission control. This application deals with the other side of the computing ecosystem, namely the data scientists and experts. This flourishing area of research has already produced many works and solutions, the efforts now focus on always more optimal and reliable outcomes. In this context, we advocated to use state of the art control theory tools to deal with this well established challenge. Indeed, highly complex systems can benefit from all the experience and maturity of control theory. On the other hand, a new application field means new objectives, new challenges and new formulations that will make control theory grow.

Those two applications that will be further developed in this manuscript are complementary in two ways. First, they are from two area of the vast computing world which reflects the complexity and diversity of challenges, from providing an accessible and useful tool for a large population, to building up the state of the art tools for data scientists. Second, most of the vast control theory domain is illustrated, from its very beginning where problem has to be formulated and where simple tools can be applied to the most complex non-linear time varying system for which recently developed techniques are required.

A third use-case has been studied alongside and following this thesis work. The computing systems under consideration are machine learning algorithms, such as the well known k-means, decision trees or the popular neural networks. Their control has been investigated in two complementary ways. First, the challenge of robustness of the learning process regarding noise in the dataset was raised. Second, research has been focused on the parametrization of the algorithms, with the introduction of feedback action.

1.2 Main Results and Collaborations

1.2.1 Publications

The work developed in this thesis has lead to several contributions which have been published in various venues, both the control and computing systems communities:

International Journals

- Sophie Cerf, Sara Bouchenak, Bogdan Robu, Nicolas Marchand, Vincent Primault, Sonia Ben Mokhtar, Antoine Boutet, Lydia Y. Chen. **Automatic Privacy and Utility Preservation for Mobility Data: A Nonlinear Model-Based Approach.** IEEE Transaction on Dependable and Secure Computing (*TDSC*) [45].
- *Submitted:* Sophie Cerf, Bogdan Robu, Nicolas Marchand, Sara Bouchenak. **Utility-Aware Modeling and Control of Location Privacy.** IEEE Transaction on Automatic Control (*TAC*), special issue: Security and Privacy of Distributed Algorithms and Network Systems.

International Conferences with Proceedings

- Sophie Cerf, Mihaly Berekmeri, Nicolas Marchand, Sara Bouchenak, Bogdan Robu. **Adaptive Modelling and Control in Distributed Systems**. PhD Forum 34th International Symposium on Reliable Distributed Systems (*SRDS* 2015), Sep 2015, Montreal, Canada [39].
- Sophie Cerf, Bogdan Robu, Nicolas Marchand, Antoine Boutet, Vincent Primault, Sonia Ben Mokhtar, Sara Bouchenak. **Toward an Easy Configuration of Location Privacy Protection Mechanisms**. *Poster* at ACM/IFIP/USENIX *Middleware* conference, Dec 2016, Trente, Italy [48].
- Sophie Cerf, Mihaly Berekmeri, Bogdan Robu, Nicolas Marchand, Sara Bouchenak. **Towards Control of MapReduce Performance and Availability**. Fast Abstract in the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (*DSN* 2016), Jun 2016, Toulouse, France [42].
- Sophie Cerf, Mihaly Berekmeri, Bogdan Robu, Nicolas Marchand, Sara Bouchenak. **Adaptive Optimal Control of MapReduce Performance, Availability and Costs**. 11th International Workshop on *Feedback Computing* co-located with the 13th IEEE International Conference on Autonomic Computing (ICAC 2016), Jul 2016, Wurzburg, Germany [40].
- Sophie Cerf, Mihaly Berekmeri, Bogdan Robu, Nicolas Marchand, Sara Bouchenak. **Cost Function based Event Triggered Model Predictive Controllers - Application to Big Data Cloud Services.** 55th IEEE Conference on Decision and Control (*CDC 2016*), Dec 2016, Las Vegas, United States [41].

- Sophie Cerf, Vincent Primault, Antoine Boutet, Sonia Ben Mokhtar, Sara Bouchenak, Bogdan Robu, Nicolas Marchand. **Données de mobilité : protection de la vie privée vs. utilité des données**. Conférence francophone d'informatique en parallélisme, architecture et système *ComPAS*, Jun 2017, Sophia Antipolis, France [46].

- Sophie Cerf, Mihaly Berekmeri, Bogdan Robu, Nicolas Marchand, Sara Bouchenak, Ioan Landau. **Adaptive Feedforward and Feedback Control for Cloud Services.** *IFAC World Congress*, Jul 2017, Toulouse, France [43].
- Sophie Cerf, Sonia Ben Mokhtar, Sara Bouchenak, Nicolas Marchand, Bogdan Robu. **Dynamic Modeling of Location Privacy Protection Mechanisms.** 18th IFIP International Conference on Distributed Applications and Interoperable Systems (*DAIS 2018*), June 2018, Madrid, Spain [38].
- Sophie Cerf, Bogdan Robu, Nicolas Marchand, Sonia Ben Mokhtar, Sara Bouchenak. **A Control Theoretic Approach for Location Privacy in Mobile Applications.** 2nd IEEE Conference on Control Technology and Applications (*CCTA 2018*), Aug. 2018, Copenhagen, Denmark [49].
- Sophie Cerf, Robert Birke, Lydia Y. Chen. **Duo Learning for Classifications with Noisy Labels.** Continual Learning Workshop, Neural Information Processing Systems (*NIPS 2018*), Dec. 2018, Montréal, Canada [44].
- Zilong Zhao, Sophie Cerf, Robert Birke, Bogdan Robu, Sara Bouchenak, Sonia Ben Mokhtar, Lydia Y. Chen. **Robust Anomaly Detection on Unreliable Data.** 49th IEEE/IFIP International Conference on Dependable Systems and Networks (*DSN 2019*), June 2019, Portland, Oregon, USA [187].
- Zilong Zhao, Sophie Cerf, Bogdan Robu, Nicolas Marchand. **Feedback Control for Online Training of Neural Networks.** 3rd IEEE Conference On Control Technology And Applications (*CCTA 2019*), Aug. 2019, Hong Kong, China [188].

1.2.2 Collaborations

Those works has been conducted thanks to fruitful collaborations:

- Pr. Sara Bouchenak (LIRIS lab, INSA-Lyon) on all the contributions of this manuscript,
- Dr. Sonia Ben Mokhtar (LIRIS lab, INSA-Lyon) on Location Privacy,
- Dr. Lydia Y. Chen (TU Delft) on data analytics and privacy,
- Pr. Ioan D. Landau (Gipsa-lab, Univ. Grenoble-Alpes) on adaptive control of Clouds,
- Dr. Antoine Boutet (Privatics, INRIA Lyon) on Location Privacy,
- Dr. Mihaly Berekmeri (Equifax UK) on modeling and control of Hadoop,
- Dr. Robert Birke (ABB Research) on data analytics and privacy,
- Dr. Vincent Primault (University College London) on Location Privacy.

I also had the great opportunity to work 5 months at IBM Research Center, Zürich, Switzerland from July to November 2018 in the form of an intership. I developed fruitful collaboration with Dr. Lydia Y. Chen and Dr. Robert Birke (former researchers at IBM) that expanded my research interest in the direction of machine learning, see the corresponding publication in Appendices A B and C.

1.2.3 Technical contributions

To foster the theoretical contributions presented in the above mentioned publications, technical advances were made. For the control of MapReduce over a cloud environment, the experimental environment developed in the Gipsa-lab by Mihaly Berekmeri [25] was re-used, maintained and extended to run more controllers and under more sophisticated workload conditions. The setup consists in Matlab-based control laws running locally on a computer while Lunix bash scripts are used for the interface between the remote cluster and the local computer.

For location privacy experimentations, the JSON-based setup developed in the LIRIS laboratory by Vincent Primault [149] was used. For the dynamical control of privacy, an independent Matlab experimental setup was built, allowing easy development and evaluation of modeling and control tools.

1.3 Thesis Outline

This thesis is organized in six main parts. The first part is a general overview on control of computing systems and the place of this work in the corresponding state of the art. The second part is dedicated to the control of privacy and utility of geolocalized data in the context of personal mobile device use. The third part details the contributions made on the area of performance and dependability guarantees for cloud-based BigData services. The fourth part draws conclusions on this work and provides insights of future works that would worth investigating on. The fifth part is a extended summary in French of this manuscript and eventually the sixth part contains the appendices that gathers the publications on the third application use-case, the control of learning algorithms. These contributions are only presented as annex of this manuscript, as they were initiated as a side project during an internship in the company IBM. However, they evolved as being the continuity of the application of control theory for computing systems during the very last months of this thesis work, thus their are included to this manuscript through papers in appendix.

Each part is divided in chapters, which contents are briefly described in the following.

Part 1 - Generalities.

Chapter 1 - Introduction. The current chapter presents the context of this work, the approach and applications taken and the reasons that motivated their choice. Main results, in terms of publications and codes, and collaborations are presented and a reading roadmap is provided.

Chapter 2 - Background and Motivation. The second chapter provides background on control theory for those who are not familiar with this domain. The description is made in a pedagogical way, by giving the objectives, describing the major tools and concepts such as sensors, actuators, models and controllers. Examples of applications are also given. Basics are provided regarding the use of control theory for computing systems, starting with the condition of this duo and explaining the advantages of the combination for both domains, concluding with the challenges it represents.

Chapter 3 - Related Work is then detailed. First, an overview of the different tools and theories used for achieving software adaptation, namely observation-based rules, MAPE-K, queuing theory, game theory, machine learning and discrete-event systems. For each of those techniques, background on the theory is given, followed by examples of its application in our context and eventually limitations of the approach are given. A second section is dedicated to the analysis of the state of the art on control of computing systems. Different aspects are reviewed, such as the analysis of the works' motivations and their application domains. Control-specific aspects are then detailed: objectives, sensors, actuators, models, controllers and evaluation. Notes on publications trends are given. Eventually, limitations and the corresponding challenges of control of computing systems are listed.

Chapter 4 - Objectives and Contributions. Concluding the first part, the fourth chapter sums-up the objectives of this works following the state of the art limitations highlighted beforehand. These objectives are put in perspective of two uses cases: location privacy and utility; and cloud performance and dependability. Contributions of this thesis are presented in four points with mention of the related publications, each contribution being a distinct chapter in the next two parts - after an introduction chapter for each.

Part 2 - Privacy and Utility Aware Control of Users' Mobility Data.

Chapter 5 - Location Privacy Background and Related Works. After the general introduction on location privacy given in the previous chapter, more details are given. The mobility data are described, the notion of privacy is defined both formally and through threats examples. The state of the art mechanisms of privacy protection specially designed for location-related issues are then presented. Methods to evaluate the so called Location Privacy Protection Mechanisms (LPPMs) are overviewed, with special focus on privacy and utility metrics and on their practical illustration. Eventually, we review the related works aiming at configuring LPPMs, independently of the technique used.

Chapter 6 - PULP: Privacy and Utility through LPPMs Parametrization. This first contribution chapter presents PULP, a framework that realize user-level LPPMs choice and configuration to meet privacy and utility objectives, in the case of already collected databases needing privacy enhancement. An introduction on this scenario and on our proposed approach is first given. The following section describes PULP's framework and its three components: the profiler, the modeler and the configurator. PULP enables four objectives formulations, all combining differently privacy and utility aspects, each of them and their corresponding configuration law are presented. Evaluation is eventually performed using mobility data collected on people in the wild.

Chapter 7 - dynULP: dynamical control of Utility and Location Privacy. The second contribution on location privacy is presented in the seventh chapter. The scenario here is a mobile device user using geolocalized services through time and willing to protect her privacy, while still benefiting of the service. After discussions on this context and on the proposed approach, dynULP solution is presented in three steps: problem formulation, modeling and control law. Each time, the approach is visualized and validated on synthetically generated mobility data. Afterwards, global evaluation is carried out on real location data. An analysis of the work and on its perspective is then discussed.

Chapter 8 - Conclusions on Location Privacy. The last chapter of Part 2 summarizes the contributions on the protection of users' location privacy and the preservation of their service utility.

Part 3 - Performance and Reliability of Hadoop Cloud Services.

Chapter 9 - BigData Cloud Services: Background and Related Works. The eighth chapter gives context to the second area of contribution of this manuscript: the control of a cloud-based bigdata framework: Hadoop. After presenting the above mentioned service, related work on the state of the art of cloud control is reviewed. The control formulation on Hadoop configuration, taken from the state of the art, is then described and will serve as a basis for the development of the two contributions detailed in the next sections.

Chapter 10 - Adaptive Control for Robust Cloud Services. State of the art techniques of control are used in this section to improve Hadoop robustness to unpredicted fluctuations in its workload and unmeasured ones for its environment. This specific issue is put in context and motivated, and the approach we took is presented. The adaptation-based control strategy is then described, in principle and in details. Evaluation is performed on a real platform deployed on a nation-wide grid with over 50 computing resources.

Chapter 11 - Cost-aware Control of Cloud Services. The last contribution of this thesis is given in the ninth chapter. Here three contradictory objectives are formulated: performance in terms of execution time of jobs, availability which is the rate of accepted requests, and costs and energy efficiency. For this scenario, an optimal model-predictive controller with a designed event-triggered approach is developed. This control strategy is presented with mathematical preliminaries. Evaluation is carried out using a real MapReduce workload.

Chapter 12 - Conclusions on MapReduce Control. Eventually, the contributions of this thesis on the control of cloud services are summarized.

Part 4 - Conclusion and Perspectives.

Conclusions on this thesis contributions and broadly on the control of computing system is given. Future works are put in perspective of the two applications use-cases presented: location privacy and cloud services; and on the on-going work on control of learning algorithms.

Part 5 - Résumé en français.

This thesis work has been funded and conducted in the frame of the French communities of universities, and is thus briefly presented in French.

Part 6 - Appendix - Machine learning algorithms as systems to control.

This ending thesis, the most recent works on control of computing systems through the use-case of machine learning is given in appendix.

1.4 Reading Roadmap

This manuscript can interest various types of readers in their background and interests. This section aims at directing the reader to relevant chapters.

For the reader with a control theory background and interested about the contributions of this manuscript in this field. Chapter 2 Section 2.1 on control theory background can be eluded, while the next Section 2.2 gathers the interesting challenges of control of computing systems. Chapter 4 can be used to get familiar and possibly chose the application of interest, if not both. For location privacy, Chapter 5 gives relevant background while Chapter 7 is the core control contribution. For Hadoop performance and reliability monitoring, Chapter 9 sets the useful computing background while the next two Chapters 10 and 11 are the contributions. The reader can also be interested in looking at Appendix C, which presents the use of a feedback algorithm to control the training phase of a neural network, with application to image classification.

For the reader with interest in location privacy. If not familiar with control theory, please start with Chapter 2. Then all the Part II is of relevant interest. Chapter 5 should be read first, the two other Chapters 7 and 6 can be looked at independently.

For the reader with interest in Hadoop cloud-service performance and reliability control. If not familiar with control theory, please refer to Chapter 2 for relevant background. The Part III gathers all the work on this topic. Chapter 9 detailing background and state of the art should be read first, the two other Chapters 10 and 11 can be read without order.

For the reader with interest in machine learning. Chapter 2 should be read first in case background on control theory is needed. The Part VI gathers all the works on the machine learning topic. No general introduction is given, one can successively read Appendix A and then Appendix B on the topic of robustness to noise. The Appendix C on the use of a feedback controller can be read independently.

Chapter 2

Background and Motivation

This chapter gives a general overview on Control Theory and on its use for Computing applications. First, basic concepts of control are explained as an attempt to depict the global picture of the field. Then, the association of control and software adaptation is motivated. Readers having notions on control can skip the first section.

2.1 Basics on Control Theory

This section is for readers willing to have a first approach of control theory. More details and formal formulations about relevant points will be given in due course in dedicated chapters. It is organized as follows. First, the assumptions and hypothesis that define the scope of control systems, as well as dedicated representation, are presented. Second an overview of the goals one can achieve with control theory is given. Then, the most common tools used by control engineers and researchers are presented in simple terms. Some insights on the possible application areas of control conclude this overview of the field. For a complete introduction to control theory for computing systems, refer to Hellerstein et al.'s book [91].

2.1.1 Assumptions and Representations

Control theory is to be applied on dynamical systems: that is its main requirements. The process under study should evolve over time and be causal (only past and present events impact the future). It is also assumed that temporal signals can be measured or observed from the system, those later will be called output signals, or only *outputs*. The system (often referred to as *plant*) should also have some tuning knobs or parameters than will influence its behavior, and indeed the output signals. The evolution of the plant configuration is characterized by so called input signals (or *inputs*). There can also be other signals influencing the plant's behavior but that cannot be tuned, these are categorized as exogenous input signals. Depending on the number of signals considered, a system can be either SISO (Single-Input, Single-Output) or MIMO (Multi-Inputs, Multi-Outputs).

To sum-up, a system eligible for control should be dynamic, causal and be configurable by at least one input signal, and observable with at least one output signal [86].

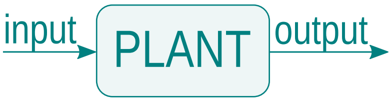


Figure 2.1 – Block diagram: control representation of a system

The usual way to illustrate a control system is to use block schemas, such as the one of Figure 2.1. In those representations, the arrows represent signals evolving with time. Block are systems that can be seen as transformations functions on the signals. The classic representation of a system eligible for control is thus a block called plant with an arrow entering on the left hand side representing the input and an arrow on the right hand side pointing outside representing the measured output. The arrow shapes are here to remind that signals are evolving over time and the plant causality.

2.1.2 Objectives

Given such a system, three complementary goals can be achieved when using control theory:

- **Stability.** The basic idea behind the term stability is that, given a bounded input signal, the output signal should also be bounded. As such, the system is stable and do not diverge to infinity. While some systems are inherently stable (e.g. a heating device), some other are instable in absence of control strategy (e.g. a inverted pendulum). Control can be used to stabilize an unstable system, but most of all it should ensure to keep stable an originally stable one.

- **Bring the system to a set point.** Once the system is ensured to be stable, one may want to decide on *how* the plant should behave. This translates to objectives values on the output signal, such as the temperature of a heating device. The main notions of performance regarding output reference values in this context are the *precision* of the achieved set point compared to the desired one, the *settling time* taken to achieve the set point, and the *overshoot* above the achieved set-point. This objective is often referred to as a tracking problem.

- **Disturbance Rejection.** As no system evolves in a perfectly mastered environment, a control strategy should be able to deal with exogenous influences. Whether these disturbances can be measured or not, modeled or not; control theory provides ways to reject them, i.e. minimize their impact on the plant outputs. This aspect of control is called a regulation problem.

The three objectives: stability, tracking and regulation are the main reasons of using the control theory. In order to provide solutions to achieve these goals, some specific tools have been developed or re-used from other disciplines. The major ones are presented in the next subsection. The use of these tools have open new possibilities, making some people use them to achieve new goals than what they where designed for. These complementary control objectives are dependent on the tool itself, and thus will be explained after the tools description. Examples of such side objectives are modeling of systems or estimation of unmeasurable characteristics of the plant.

2.1.3 Tools of a Control Engineer

An overview of the methods used to solve control problems is given next. First, the idea of feedback loops is depicted. Second, insights on modeling tools are provided. Then, the control usage of frequency domain is shortly presented. Eventually, last paragraphs explains the usage of applied mathematic tools by control scientists.

2.1.3.1 Feedback loops and controllers

The idea behind control theory is the use of feedback loops [62]. The configuration of the plant is computed based on the measurement of its state: the output signal is fed back to decide on the input signal. The mathematical relation that links the output to the input is called *controller*. Going back to the block schemas, the controller is thus a box that expressed the link between the output signal and the system input, see Figure 2.2. When controlled, the input of the plant is usually called control signal.

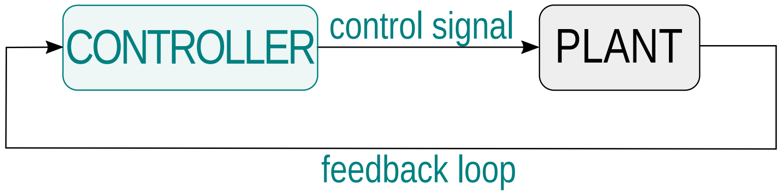


Figure 2.2 – High-level representation of a controlled system

In its simplest form, the controller is called Proportional, see Figure 2.3. The output signal is compared to its reference value, to which the plant should tend. The generated error is leveraged by a gain K_P, and this signal is set as system input. Thus, if in steady state the measured state of the plant is equal to the reference value, then error equals to zero and so no action is taken any more on the plant. If the system output and the objective set point are different, then action is taken on the plant, and the larger the error, the bigger the action. The system is ensured to move in the right direction by choosing the sign of the gain.

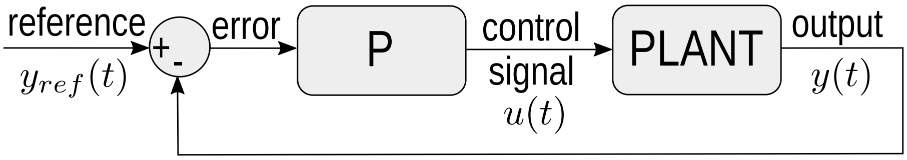


Figure 2.3 – Simplest controlled system: a Proportional Controller

Usually, the control signal is noted $u(t)$, the system output $y(t)$ and the desired output signal $y_{ref}(t)$. The Proportional controller formulation is thus:

$$u(t) = K_P.(y_{ref}(t) - y(t))$$

Indeed, the relation between the tracking error (difference between reference and output signal) and the control signal is often more advanced than a simple proportional gain. Most of the controllers used in industry falls into the PID controller category. P stands for proportional, I for integral and D for derivative. As it can be assumed by their name, PID controllers also take into account the integral and derivative of the error signal, once again weighted by a gain. With its variants only derivative (PD) or only integral (PI), many control problems can be solved.

Even if in practice PID controllers are by far the most used ones, many more advanced control techniques exists, that aim at dealing with either more complex systems that cannot be controlled by PIDs (for instance multi-input and multi-output systems) or more restrictive objectives that a simple non-optimal controller cannot achieve. These controllers usually take as input the measured states of the plant, their reference value and possibly other signals such as measurement of the environment of the plant, often called disturbances. The idea of using measures of the disturbances that impact the system to control it (i.e. generate its input signal(s)) is called feedforward. The mathematical link between all these variables can be of many kinds. The main categories of controllers are optimal ones (in the formal mathematical meaning of the term), or robust, adaptive, non-linear etc.

Controllers are parametrized algorithms, see for instance the value of the gain K_P of the proportional. Their tuning enables to leverage the controller effect: speed up or slow down their action, refine the precision in reaching the desired state, reduce the use of resources, etc. The parametrization of a controller also depends on the system's dynamics and behavior. For instance, a thermal system is way slower than a electric engine, thus the same controller with the same parameters applied for both systems is not likely to be efficient (neither stable) in all cases. This motivates the use of modeling.

2.1.3.2 Modeling

Models in control theory are mathematical equations that predict the current output(s) of a system, knowing its previous ones and the input signal(s) applied. Other signals can also be taken into account, such as the disturbances of the environment on the plant. Models can have various memory size (called order, the number of past values of the signals needed to predict the current one), and of course parametrization.

The mathematical tools to represent the models are quite specific to control theory. They use the Laplace transform, which idea is to transform derivative and integral functions in products and divisions. The SISO systems are often modeled by a transfer function, which is a link between the input and the output of the system. It is more formally the ratio of the output over the input, both being in Laplace transform.

Multi-variable (MIMO) systems use more advanced modeling techniques called state-space representation, linking the states, inputs and disturbances to the derivative of the *states*. These states are all signals that are needed to describe the system. Some of them are measurable and thus called outputs (sometimes subject to a transformation), other are not, and thus are mostly intermediate of calculation. As a concrete example, let's take consider a pendulum. To fully describe its behavior, both the position of the mass and its speed are needed. However, most of the time only the position is measured. The pendulum two states are thus the mass position and speed, but only position is called system output. Note that there exist an infinity of state-space representations of a system.

In order to find the order and parameters of a system's model, two main techniques exist. The first one consist in studying the system from a physical point of view, i.e. computing the behavioral laws that links the input(s) to the output(s). This is for instance what is done if one wants to model a electrical engine. The rotation speed can be linked to the engine input voltage though other physical quantities. This solution has the advantage of being accurate, but can become complex or even impossible to apply depending on the system to be controlled. In the case that interest us in this thesis, computing systems such as softwares do not depend on such kind of laws. The other tool of control theory for modeling is called identification, or black-box approach. The idea behind it is to vary the input(s) of the plant and observe the output(s), and try to fit the observed behavior with a classical model, which order and parameters can be varied. Controlled systems being always more complex and large, identification is the most used technique. One of its advantages is being able to leverage the model complexity (and thus its precision) in order to have a computationally practical model. It is worth noticing that, for a good behavior is closed loop, the model does not need to be perfect.

Models are usually computed to derive a proper controller that will enable to reach the objectives. However, the development of advanced modeling tools for dynamical systems are sometimes used as an end-goal, for instance is the biological domain, when trying to model complex dynamical systems.

2.1.3.3 Time and Frequency Domains

As in signal processing, the systems can be considered with two complementary points of view: the time or the frequency domains. The use of time domain is linked to the nature of the control systems, that have to be dynamic. However, the use of the frequency domain helps the analysis and even sometimes the design of controllers. This will be the case in this thesis when designing robust controllers (see Chapter 10).

Some requirements can also be expressed in the frequency domain, for instance when working in the acoustic field where specific frequencies have to be attenuated. In a general way, the study of control systems in the frequency domain helps for analyzing stability, performance, etc. in all possible scenarios that the system can encounter.

2.1.3.4 Filters, optimization and other applied mathematics tools

Even though the large usage of control in industry can be sum up to PID controllers, state-of-the-art companies and indeed researchers are working with advanced control problems, being close to applied mathematics. Optimization techniques are largely used for instance to find the best input signal(s) that fits the performance objectives, to minimize the use of resources, and so on. As a main example of the contribution of control theory to the broad work of mathematics users is the work of Kalman on filters. From a control perspective, Kalman filters are used as so called *observers*, that estimate the states of Multi-Input Multi-Output systems that cannot be measured. Similar to modeling tools, estimation techniques initially developed with the idea of deriving a controller are now widely used as such for other applications, such as image processing to mention only one.

2.1.4 Applications

Control being a tool with few constraints regarding the application system, it has been used in an very large variety of fields [86]. To still try to give a overview, one can start by explaining the historical domains that have motivated the emergence of control, and thus move to specific domains that can now benefit of control theory in its most advanced forms.

The most basic applications of control are for mechanic and thermodynamic systems. Temperature regulation of a room or an oven, speed rotation of an engine, or water levels in tanks are simple everyday examples. Control is now widely used in automotive, aeronautics, drones industries and all kind of plants such as chemistry, nuclear or hydraulic ones. Event if the terminology of *application* may not be appropriated, the large and common contributions of control to mathematics is to be stated.

Control is also used in new and promising fields of application. Biological systems are nowadays studied with the point of view of control to better understand the natural regulation systems that exists for instance in human bodies. Large scale and distributed systems such as electric grids and networks are also a state-of-the-art application of control, particularly when considering issues of connection of local renewable sources. Eventually, computing systems are a promising field to be explored by control theorists and engineers. Section 2.2 is dedicated to the motivation of this association, while Section 3.2 of Chapter 3 overviews its state-of-the-art works. Eventually, Chapter 4 presents the two application cases this thesis will focus on.

2.2 On the use of Control Theory for Computing Systems

Using the basic notions overviewed given in Section 2.1, the use of control for software adaptation is considered in this section. First, a precision is made on the notion of *computing system* that can benefit from control, corresponding to the assumptions required. Second, the achievements a control approach can provide is investigated from the point of view of a computer scientist. Then, the advantages of the association is highlighted, from the point of view of both fields. Finally, the challenges one has to face when trying this approach are depicted, justifying the interest of researchers on this question.

2.2.1 Eligible Computing Systems

Computing systems are omnipresent in our society, and even though they emerged quite recently, their diversity is huge. So what is meant by computing systems in this context? A first categorization can be done between hardware and software. Both are eligible to benefit from control theory. However, this thesis focuses on software applications.

Coming back to assumptions, a controlled system first has to be dynamic, i.e. evolve with time. In computing terms, that means that the process should be *on-line*, or in *real time*. However, this temporal dimension is often scale-dependent. Let us take the example of a query request. From a high scale perspective, this can be seen as an event that, when successively repeated, consist in a dynamical system. At a smaller scale, all the processes involved in the request (network, computing, etc.) can also be seen as a dynamic system on its own. Consequently, the problem formulation plays an important role in the control approach.

The second main requirement is the possibility of tuning the system in an on-line fashion. Indeed, one need to be able to adjust the behavior of the system over time to control its performance. For instance, if an application is launched with a given amount of computing resources and there is no way to leverage them while running, it will not be possible to control the application run time.

Third, the system willing to be controlled should be observable, which means that the desired characteristic should be measured or estimated, possibly indirectly. If the response time of queries cannot be evaluated in real time, it will not be possible to decide on a control strategy to accelerate them.

These last two constraints (tunable input and measurable output of the system) are usually issues that can be overpassed by the development of system add-ons if it is not present by default, if dealing with softwares. However, sometimes the constraints on the system prevent from doing so, for instance when working with certified codes or embedded systems with limited resources. Once again, this brings us to the challenge of problem formulation, where actuators (i.e. tuning mechanisms) and sensors (i.e. measure of the system state) have to be designed and implemented.

2.2.2 Promises of control for software adaptation

Computing systems are most of the time stable. The contribution of control is thus not considered there, even though one main contribution of the control is to offer guarantees about the stability of complex systems. Performance monitoring and robustness to external factors are thus the contributions of control that mostly interest computer scientists.

In software adaptation, it is often an easy task to find ways to tune the system. Indeed, they have a multitude of parameters that can be fixed but this task is very complex. Finding appropriate ways to chose these parameters is precisely the purpose of control. After defining objectives such as response time or resource consumption, controllers tune the system to reach these requirements. This often non trivial dynamic relation between desired outputs and controllable inputs are dealt by the controller.

Another contribution of control is disturbance rejection, which in computing terms means handling the impact of the changing environment, or external factors, on the system. For instance, the steep raise in the number of people visiting a website requires the augmentation of dedicated resources to avoid a crash. Control algorithms can detect such circumstances and adapt the system accordingly.

On top of all this, the mature mathematical foundations of control enable to provide analytical guarantees about all the above mentioned contributions.

2.2.3 Benefits

The advantages of using control on computing applications are for both community. A quick overview is given in the next two paragraphs.

2.2.3.1 For the computing world

Computer scientists using control can rely on tools developed decades ago in well known, efficient and complete formulation. Having a feedback, i.e. a closed loop, enables to monitor the states (data, tasks, resources, etc.) of the systems to better understand its current behavior and better regulate it. Control framework also enable to clearly formulate one's objectives, e.g. quality of service, and also to leverage them if needed, through the use of cost functions. Thus, all aspects of the application can be taken into account, such as energetic or financial costs.

Feedforward loops (or other advanced controllers) take into account the environment of the system to decide on its later configuration. This is a significant advantage in computer science as most of the systems to control are not isolated: they often are subsystems in a bigger framework. Adaptation algorithms are able to react to the changes in the plant itself, for instance dealing with application having multiple steps or stages. A side advantage of control approached is that they provide an estimation or even prediction of the unmeasured metrics, for instant workload or completion rate of tasks.

Last but not least, control theory provides mathematical guarantees of its performance and robustness. This means that service providers can engage themselves to their clients and be aware of the risks they are taking, even sometimes in the case of faults or attacks. From a juridic point of view it can also be very interesting, this means that results obligation can be realized and checked in practice.

2.2.3.2 For the control community

Computer science is not yet another application field of control, whereas, it has a lot to offer. Control has been developed and largely used for physics-rules systems. It has brought many constraints on the systems that, when dealing with computing systems do not exists any more. Thus, many new possibilities are open for control theory to expand, new control laws to be built and so on.

Another particularity of computing systems is their scale and complexity. Control theory started recently dealing with such systems, for instance country-size electrical grids including local intermittent sources. Advances in control of computing systems will also help the communities of other control application fields.

2.2.4 Challenges

Even though the use of control for computing system has been proved possible and beneficial for both fields, this approach is still at its premises. The main challenges to face are regarding the problem formulation, i.e. finding the appropriate tunable input and performance output signal for the software. The terminologies used is both fields are different and even sometimes contradictory. A simple example is the notion of *response time*, which for computing people means the time for a request to be answered, while for control people it is the time it takes for a objective to be reached. One can also think of the principle of *adaptation*, which in computer science means reacting to changes in the plant or in the environment (which is what a controller do), while in control theory the adaptation is one step higher, the control law itself is changing over time. Consequently, defining the bounds of the system, its inputs and outputs that satisfy control conditions is a major part of the work.

Additional complexity comes from the properties of the systems themselves [99]. They have a variety of time dynamics, power consumption and cost scales that require specific tuning of each control approach to the problem considered. Systems often involve multiple metrics with various criticalities, each signal being able to vary over several orders of magnitude, requiring the control solution to be scalable. Most computing systems also present hybrid behavior depending of the scale at which they are considered, mixing discrete-events dynamics (for instance between the various functions of a software), discrete-time (sequential execution of a program) or even continuous-time (for instance if the heat dissipation is looked at for cost-related issues). All those challenges are what motivates researchers to find appropriate formulations and tools to overcome them.

Chapter 3

Related Work

This chapter reviews the state of the art works on the control of computing systems. It is divided in two sections, the first surveys the variety of theories and techniques used, the second one focuses on the use of control theory and on the various applications that already benefited from it. Given the vastness of the targeted fields, this review is not meant to be exhaustive but rather to give the reader a global picture of the methods and applications.

3.1 Monitoring Techniques for Software Adaptation

This section gives a classification of the state of the art solutions that realize software adaptation. Discarding hardware, we focus here on the software systems. The execution of these later are highly dependent on their environment, for instance through available computing resources and concurrency on them, or because of unexpected workload. Mechanisms that realize adaptation are thus needed, and the diversity of applications and objectives has led to a variety of solutions. The remaining of the section gives a non-exhaustive but hopefully representative overview of the literature on software adaptation. For each technique, we first sum-up the principle (which is often a field on its own), then give examples of applications and eventually conclude on the limitations.

3.1.1 Observations-based rules

This category gathers the simplest forms of adaptation, based on IF-THEN rules. Even if most solutions are based on observations and end up with rules, we elude here all methods that use advanced processing on measurements such as feedback. A simple example of such mechanism is for instance:

IF runtime $>$ 10s
THEN #-resources $=$ 2

which set the number of resources dedicated to a job depending of the its tasks' runtime.

These simple solutions are already implemented by cloud provider, see for instance Amazon EC2 Auto Scaling [13] of AWS Auto Scaling [12]. Another example is given by *AutoScale* [76], a framework that scales up or down the number of server used in a datacenter depending on the current workload. Other works monitor performance of web services using shared architecture [51]. They measure the Quality of Service goal and adapt resources accordingly (accept queue and CPU). Their main advantage is being able to deal with several web services classes and workload. More advanced works have used the fuzzy rules theory to improve their approach. One can see as examples the works of Jamshidi in the field of cloud resource provisioning [94, 95].

These approaches have the advantage to be simple to implement as they do not require modeling of the system nor complex algorithms to run. However, deciding on the thresholds is often a hard task, even for experts, and could require another automatization. Eventually, those solutions are inherently reactive, while it has been shown that predictive action of the adaptation show better results [162].

3.1.2 MAPE-K

A famous approach to autonomic computing is MAPE-K [98], which brings knowledge in the decision process. MAPE-K is a iterative approach that performs sequentially: Monitoring of the system, Analysis of sensed data, Planning of actions to take and their Execution, all of this based on Knowledge of the system under study, see Figure 3.1.

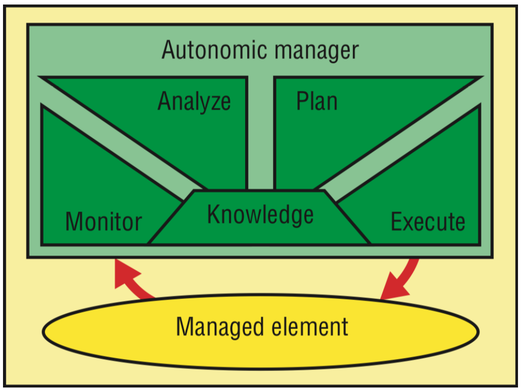


Figure 3.1 – MAPE-K autonomic manager [98].

MAPE-K approach is widely known and used, including in the industry [108, 58]. Its formulation is really close to the control theory one as has been shown in various studies [156, 71]. Indeed, the monitoring can be seen as the sensor and the execute part as the actuator. The analysis is the states estimations, the planing is the control, and the knowledge is embedded in the system model. Consequently in the following of this work, we will discard the MAPE-K formulation and stick with the control theory perspective.

3.1.3 Queuing Theory

Along with control, queuing theory is the most used approach for software adaptation [89]. A queuing model consists in two main parts: the service, called *server*, and the incoming *clients* willing to benefit from the service. A classic representation is given in Figure 3.2, where the server is the circle on the right denoted μ and the queue of arriving clients is the left part λ. Queuing theory based adaptation approaches model the system to predict the control signals, for instance resource allocation or admission control, in order to monitor the delay of clients waiting for the server.

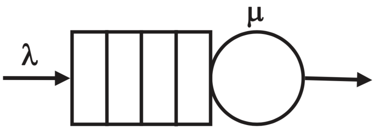


Figure 3.2 – Queuing basic model [89].

The probability distribution of the client arrival process and the one of the duration of the server processing (also called *service time*) are two main elements of the queuing model. Two of the most common distributions are the Poisson one where the inter-arrival times are exponentially distributed or the general one for which arrivals are independent.

The Kendall notation is often used to define queuing models. It comes in the form of three letters separated by slashes, for instance $M/G/1$. The first letter represents the distribution law of the clients arrival λ and the second the distribution of the server μ. As there is mostly two distributions, the letters are usually M (which stands for Markovian or memory-less, and is the Poisson distribution) or G (general, independent, also noted GI sometimes). The third element of the Kendall notation is the number of servers.

The Kendall notation reflects the parametrization of queuing models, but one can also play with its architecture. Basic modeling elements (Figure 3.2) can be combined in series or parallel. The basic model is a black-box approach at the system-level. Combining several allows to consider a model detailed at the component-level, for instance by having a basic model for the processor and another for the disk; or one for the web service and another for the database. Another queuing model knob is the classes. The previously described models are single-class as the distribution of the clients are the same for all of them. One can also consider multi-class models where several types of clients are considered with different arrival rates and service distribution. Eventually, the previously mentioned models are called open (regarding the clients), and one could consider there closed version where the clients are limited and go back into the waiting queue after being processed (with some delay).

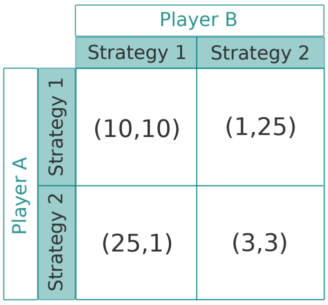
The literature that use queuing theory for software adaptation take advantage of all the possibilities in the modeling, i.e. there is not a commonly accepted architecture and parametrization. For e-commerce workload application, Ranjan *et al.* [154] used a open $G/G/N$ while Liu *et al.* [117] considered a multi-class closed model. Open $M/G/1$ as been used to monitor algorithm execution time [10]. Resource allocation in shared web servers has been addressed using mutli-class open models [50] taking as evaluation workload the *World Cup Soccer'98*. Eventually, the load-balancing and service time control challenge for cloud application has also been tackled using Poisson-based queuing theory [143], however combined with a discrete controller.

The main limitation of queuing theory is that they mostly guarantee the steady-state monitoring but dot not enable to handle transient phases [89]. Transient behavior is essential to monitor as it can get the system through states that are detrimental or even dangerous for the system (e.g. too high CPU or memory usage). Moreover, queuing models are not suitable to model systems with too complex client arrival and server distribution. Nowadays software applications are become more and more complex, which makes this limitation of prior importance.

In order to deal with those limitations, some approaches have proposed to combine queuing theory and control [8, 116].

3.1.4 Game Theory

Game theory has initially been developed in the field of economic in the beginning of the 20th century and has been excessively applied in the context of computing systems, notably with the particularly relevant notion of rational agents. In the game theoretic formulation [174], a system is composed of agents, or players, who can realize a set of possible strategies or decisions. Each of the players have a utility function that defines their relative advantages and profits. Game theory provides strategies to players in order to maximize their profits and reach equilibrium, such as the well known Nash equilibrium [138]. Representation of game theory simplest models can be modeled with decision trees or payoff matrices. An example of such matrix is given in Figure 3.3 in which two agents have both two strategies available, and the quantified gains of each strategies combinations is given for each player $(gain_A, gain_B)$. Many variants of this simple example can be drawn with for instance non-symmetric options between players, sequential or simultaneous choices, zero or non-zero gain sum, complete or partial information disposal, or eventually cooperative or non-cooperative agents (i.e. no alliances).



		Player B	
		Strategy 1	Strategy 2
Player A	Strategy 1	(10,10)	(1,25)
	Strategy 2	(25,1)	(3,3)

Figure 3.3 – Game theory payoff matrix, basic example.

This approach as been applied in various domains of software adaptation such as radio resource allocation [163] or load balancing in grid computing [166]. The most common solution used is the Nash equilibrium, such as by Guo *et al.* with a Nash cooperative bargain formulation for network bandwidth sharing in datacenters [90] or by Grosu *et al.* in a non-cooperative formulation for load balancing [87]. More complex formulations has also been studies, such as two steps models [180] or parametrizable agents [122]. Combination of game and control theories have also been studies, for instance to the network bandwidth allocation challenge for self-adaptive cameras [160].

The main limitations of game theory for software adaptations are that it is hard to distinguish between positive and detrimental dynamic strategies [121] and that no self-adaptation is possible in the sense that the trial-and-error process is not applicable [68]. Moreover, the learning of the solution should be done online in the case of asymmetric formulation, as they depend on the initial state, which can be computing intensive. Eventually, the formal solution gets more and more complex to retrieve as the number of agents grows [68].

3.1.5 Machine Learning

Machine learning is the science that retrieves statistical models from datasets. Its popularity has widely grown in the last decades, particularly with the advances in computing capacities, leading to a really vast field of research regarding as well the diversity of its techniques and algorithms as their application field. The basic stone of machine learning is data. Each record of the system under study is called *instance*, and is characterized by *features*, a set of values that describe the data instance. The learning algorithm is then trained on a set a previously known instances in order to adapt its parameters to best fit the system under study. Afterwards, the learned model is used to provide a prediction on new unknown instances. In the case of classification problems, the output of the model is the category of the instance (for instance "short" or "medium" or "long" in case of service time prediction) whereas for regression problems, a continuous value is retrieved (e.g. "150s"). Eventually, a distinction can be done between supervised learning, for which the ground truth of the training set is supposed to be known (i.e. the classes of each instance), and unsupervised learning. Figure 3.4 illustrates the k-nearest neighbor supervised classifier, that predicts the class of an new sample by taking the majority class between its k neighbors from the training set. In this example, the feature space is two dimensional.

All the variety of algorithm from the machine learning community has been applied in the context of software adaptation. Wang *et al.* proposed the use of k-nearest neighbors for resource allocation [177] in order to find the closest past scenario and apply its offline (near-)optimal solution. Decision trees has been used to perform load balancing for IoT fog computing [11]. Most advanced techniques such as time series has been studied for resource demand prediction [173] combined with an allocation algorithm; and in the context of self-healing software [5]. Famous neural networks have also been

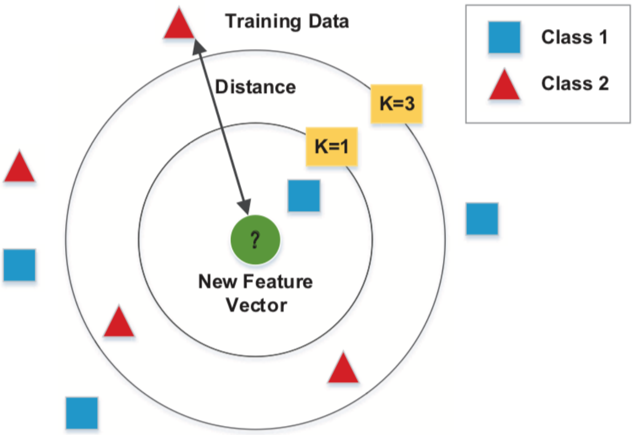


Figure 3.4 – Machine learning simple example: *k*-nearest neighbor classifier [177].

applied to predict power usage effectiveness in datacenters [77] or in their deep reinforcement learning version for performing resource management [125].

Machine learning techniques are inherently limited by the quantity and the quality of available data [77]. Moreover, they are modeling tools, able to achieve extremely good fitting to data, but which are not designed together with a decision processes. In the context of software adaptation, the objective is always to be able to achieve a suitable action on the system depending on the environment changes. This objective does not exactly overlap the one of maximizing modeling accuracy, which limits the use of machine learning for decision making.

3.1.6 Discrete Event Systems

The theory of discrete event systems is a part of the global control theory. However it has specific hypothesis regarding the system's behavior that will make us consider it apart in this literature review. A discrete event system must have the following properties: be describable with states that take only discrete values (integer, Boolean, etc.), it should be dynamic, that is evolving through time, but transitions between states should be event-driven, not necessary at regular time intervals [37]. Such systems are modeled using automata, also called finite state machines or transition systems, and are mostly defined by a memory of the states and a transition function [156]. Petri nets are a variant of automata in which states definition is based on token triggered by transitions. Such models are used for two main objectives, being model checking and discrete controller synthesis. Only the last one is of interest for us in the context of software adaptation. Discrete controller synthesis (DCS) is a formal approach for automated controller design, with the specificity of being compliant by design to the control objectives [153]. The specifications are usually deadlock avoidance and invariance of a subset of the state space (which can be seen as a kind of reference tracking), with the constraint of being maximally permissive for all that is not stated in the objectives. DCS uses some classic tools of control theory, such as the notions of feedback and controllable/uncontrollable inputs. A schematic representation of a close-loop control of a discrete event system in given in Figure 3.5.

DCS in the context of software adaptation have mostly focus on fault tolerant systems [84, 178]. Other works have addressed more exotic systems such as IoT deployment and smart buildings [168] or dynamically reconfigurable architectures such as FPGAs [17].

Discrete event systems control is inherently limited by its assumptions (discrete states and event-driven dynamics) and thus cannot be generally applied for all software adaptation problems. Moreover, discrete controller synthesis are facing scaling issues regarding the size of the automata [156].

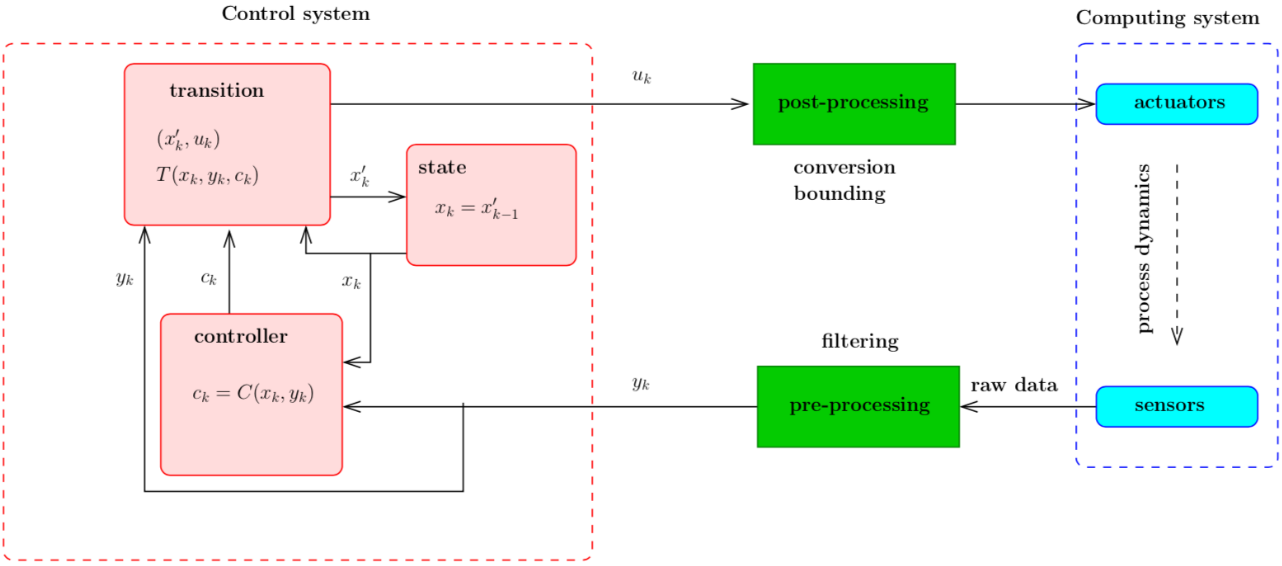


Figure 3.5 – Discrete Event System Control Schema[156].

3.2 On the Use of Control Theory for Computing Systems

After describing the various techniques that can be used for software adaptation, we now review the literature that used control theory. An overview of the issues addressed and the techniques used is given. For readers who are not familiar with control theory, please refer to the terminology presented in Chapter 2. This section is widely based on results given in Shevtsov *et al.* literature review on control-theoretical software adaptation [162] and in Patikirikorala *et al.* survey [147], which respectively studied 42 papers from 1998 to 2016 and 158 papers from 2000 to 2011.

The remaining of this section is organized as follows. First the motivations that lead to the use of control theory are reviewed. The application domains are presented, as well as the objectives of the authors, that are then expressed as performance metrics. Next, we present the control knobs that have been used. Modeling and control techniques are presented, and eventually we review the evaluation methods used. Finally, we analyze the publication trends and conclude with the limitations raises in those works.

3.2.1 Motivation for using Control Theory

When stated, the main reason given by the authors to use control is the formal guarantees that it provides [162]. The second one is the maturity of the technique that has solid and formal foundations and provides a systematic approach to find solutions. This highlights that the need for automation and reliability is of first level importance for software systems. Note that another motivation is the inefficiency of other techniques, which can let us think that control is usually not used in the first attempt to solve a new challenge. However, more than a fourth of the literature papers do not give insights about the justification of the choice of using control for software adaptation [162].

3.2.2 Application fields

Two categories have to be distinguished: the application system and the application use-case. The first categorization has been proposed by Patikirikorala *et al.* [147]. A third of the paper they studied focused on middleware systems, others on real-time systems and and data center management (around 20% each), the rest were dealing with Virtual Machines or data storage. Given that this study stops at the early 2010's, one can expect that the proportion of virtual machine and cloud configuration is significantly higher nowadays. However we did not found a more recent global study to highlight this point.

Regarding the use-case, most of the papers focus on web application (i.e. e-commerce) or on image or video processing [162]. Some works also focus on service-based systems and search engines. One can notice also that few works have dealt with software testing and development [34, 96].

3.2.3 Objectives

The three main objectives of control theory are performance, reliability and energy consumption. Some minor aspects are also taken into account, such as security or usability. They can be expressed though three formulations: setpoint tracking, with eventually confidence intervals, disturbance rejection and minimization or maximization of a quantity (with the use of a cost function if some goals are conflicting). According to Shevtsov *et al.* [162], 85% of the works are motivated by changes in the environment (disturbance rejection), 70% by changes in the requirements (i.e. set point) and fewer (15%) are also driven by changes in the software itself.

3.2.4 Sensors and performance metrics

Metrics measured by sensors can be distinguished in three main categories: software utility, resource utilization and software inefficiency [162]. The utility is measured with variables such as response time (in around 40% of the papers [147]), throughput, latency or video quality and processing speed. Resource utilization gathers processor/CPU/memory utilization, power consumption and bandwidth. Eventually, software inefficiency is measured with metrics such as miss ratio or failure rate.

Note that only 15% of the papers describe the sensor and its implementation, while the vast majority only describe the metrics used [162].

3.2.5 Actuators and control signal

The actuators being highly application dependent, their categorization is not a easy task. Three types of actuators have been identified [162]: parametric, component or mode adaptation. Parametric adaptations are the modification of the parameters of the software system, for instance the degree of quality of a filter. Component adaptation intervene a degree of abstraction higher, for example the load of a service or the number of service instances. Eventually, mode change can be for instance a switch to the preference given to each service.

The particularity of software adaptation regarding actuator is that it is rather easy to create and remove some [71]. However, one should be specifically careful when designing an actuator (the same can apply to sensors) that this later is consistent with further modifications of the software under study.

3.2.6 Modeling

The most common model used for software applications is a linear, time-invariant black-box discrete model [162, 147]. In this case, the link between the control signal and the performance metric is linear (i.e. not quadratic, exponential, etc.), the parameters do not depends on the time variable, and this later is considered at regular time intervals (not in a continuous way). The parameters of the model are found using system identification techniques, that subtly solicit the system, record its behavior and fit it with a model. The use of black-box is particularly spread because its an easy process that can deal with highly complex systems and still enable to derive performance guarantees. Even though many studies state that their system is non-linear, linear modeling is preferred, maybe because of lack of tools as simple and practical in the non-linear case.

However, when linear black-box is not used, models tends to be non-linear and analytical [162], even though it represents a smaller portion of the literature. Most of the combination in terms of linearity, continuity, time-variance and model type (analytical of black-box) can still be found in the state of the art, even though only the two above mentioned combinations are popular.

As for the dimensions of the model, 60% are MIMO (multi-inputs multi-outputs) while 40% are SISO (single input and output) [147].

3.2.7 Control

Almost all the controllers used were including feedback, and a small portion of them where also using feedforward to deal with workload rates [147]. The most used controller is by far the PID one (50% of the reviewed papers [162]), with a preference for its PI declination. They are mainly used in SISO scenarios, for regulation and tracking objectives, and are most of the time adaptive to compensate for the linear modeling of a non-linear system. The second main trend is the use of Model Predictive Controllers, mostly with MIMO discrete time models, in a non adaptive formulation, to deal with a minimization or maximization objective.

Some specific structures of controller can also be found, such as hierarchical control, switching control or cascaded control.

3.2.8 Evaluation

Three kind of evaluation methods are used: formal analyze, simulations and real-case experimental study. All three are widely used, simulations being however quite limited, and the quality and the generalizability of experimental use-cases are limited. Some benchmark and workloads are commonly used for some works such as [147]. The main qualities that are looked at are indeed performance, efficiency, costs and reliability. The metrics used to guarantee those later properties in control theory are stability, settling time, overshoot, steady state error, robustness, optimality and control signal usage [162]. All of them are commonly used for evaluation, stability and settling time being the most spread ones.

3.2.9 Publications trends

Aside from the technical content of papers, two interesting metadata to look at are the publication year of the papers as well as their venue. Both review papers [147, 162] concluded that the number of publication per year is significantly increasing, showing a growing interest of control theory for software adaptation. They also reported that the source of publication of such contributions are largely spread among venues and communities. There is however a higher number of paper published in the software engineering community, showing a higher interest in using control theory as a tool for solving software issues as to use computing systems adaptation as a playground for control scientists, at least from the point of view of the global communities.

3.2.10 Limitations and open challenges

Each design step previously detailed is subject to limitations. Using control theory for software require the use of new sensors and actuators that were not included in the design phase, such as the evaluation of the cost of adaptation. Some others are crucial notions that however still lack of commonly accepted metrics, such a security, resilience or privacy. From an architectural/implementation point of view, actuators and sensors need to be consistent and evolve with the different versions of the software.

The most common type of modeling used (linear time invariant discrete model) is still ineffective when dealing with sharp and drastic disturbances. More research need to be done regarding the properties of the models and how to chose them (linearity, continuity, etc.). The same conclusion can be drawn about controllers. Even though various types of controller are used, few research has been done to derive a systematic procedure regarding how to chose the best controller type. A recent study has highlighted the promises of using more advanced modeling and control techniques such as closed-loop metrics or real-time cooperative model predictive controllers [99].

Eventually, while the formal background of control theory is often cited at the number one reason to choose this technique compared to another, the analysis of these guarantees in the evaluation process is many times missing.

Chapter 4

Objectives and Contributions

In this context of association of control theory and computing systems, this thesis take the approach of deeply exploring two use cases. In this work, these applications have been approached with a large variety of complementary techniques. As such, diverse aspects of the combination of the two fields have been explored. In this chapter, the two application scenario are presented. The first one focuses on the protection of mobility-related privacy of mobile devices' users, while ensuring the usability of geo-localized services. The second one deals with BigData cloud services and how to ensure their performance and reliability in their extremely variable environment. These two use-cases cover different and complementary aspects of software systems, services running on mobile devices for everyday users and large scale data processing, mostly business oriented. The last section of this chapter summarize the contributions this thesis has brought to these two objective scenario. The third use-case, seeing machine learning algorithms as a control systems, is not introduced here as a contribution of this thesis work. One can however refer to Part VI for the corresponding publications.

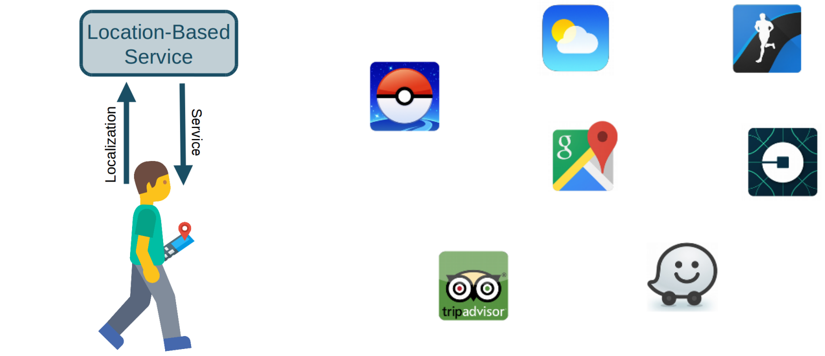
First, these problematics will be presented in computer-science words, not only understandable by experts but also by the everyday users that the reader can be. These scenario will be reformulated with a more control-oriented vocabulary in due time, as introduction of contribution chapters.

4.1 Privacy and Utility for Mobility Data

The democratization of mobile devices has fostered the development of applications using the owners' location data to provide or improve a service. Such applications are called Location Based Services (LBS), their operating principle is illustrated in Figure 4.1(a). Examples of LBS are navigation applications, recommendation systems or fitness tracking apps. Some of the most used geo-localized applications are Google Map, Uber, TripAdvisor or weather applications, see Figure 4.1(b).

A large number of mobility data are generated and collected, which are currently used by companies and researchers. Indeed, the processing of mobility data can reveal valuable information that may be used for a broad range of applications, e.g., traffic congestion management, urban development, etc. See Figure 4.2 for a schematic representation. Those applications and the corresponding collected data thus benefit to all parties.

LBSs provide users with always more personalized and convenient services but at the cost of personal data publishing. Service providers, or any third party attackers, take advantage of these data to derive always more informations about users. The attacks can happen at every processing step, as pictured in Figure 4.2. The most common threats for users are (i) reidentification attacks where the identity of an anonymous user is guessed based on previously recorded data [73, 126], (ii) mobility prediction that anticipates users' next moves based on their habits [184, 75], (iii) extraction of user's



(a) Operating principle (b) Some illustrated examples

Figure 4.1 – Location Based Services

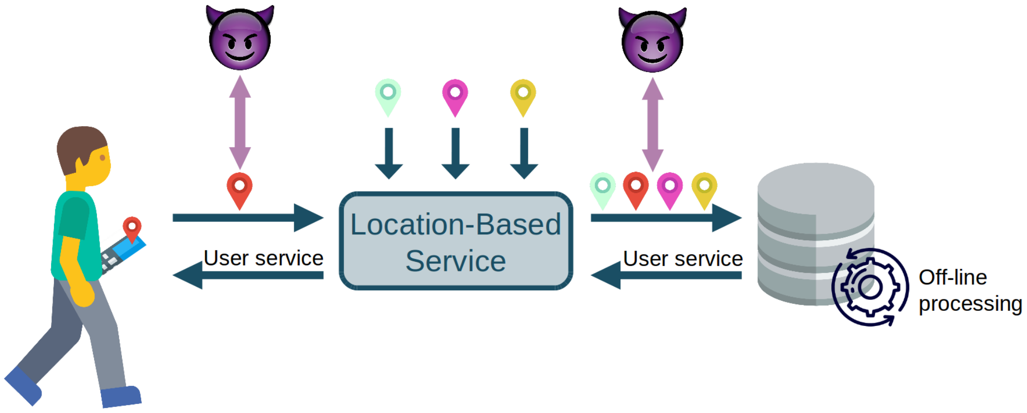


Figure 4.2 – Usages and threats of mobility data when using a LBS.

places of interest (home, workplace [74], place of worship [72], etc.) and (iv) inference of social relationships (partners, coworkers, etc.) [29].

In order to provide ways to protect users' privacy, Location Privacy Protection Mechanisms (LPPMs) have been developed. This terminology gathers all algorithms that modify location data in order to improve the users' privacy, see Figure 4.3.

Simple examples of LPPMs are adding noise to the data of the user, reduce the precision of the

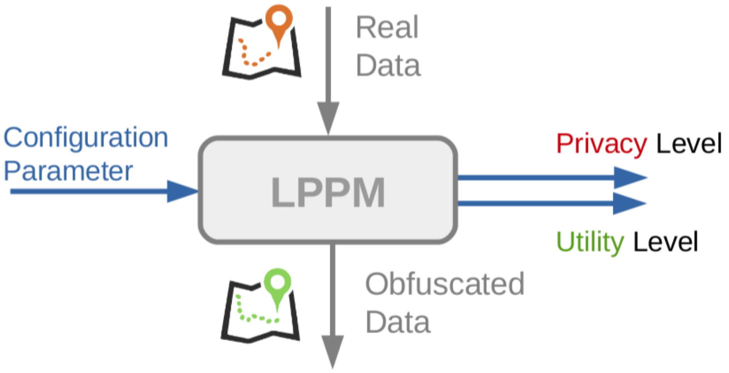


Figure 4.3 – Location Privacy Protection Mechanism

data or merging close users' mobility information. There is a high diversity among LPPMs: some are at the user level, other require trusted servers; some are on-line mechanisms, others can only be applied on a dataset; etc.

Usually LPPMs are parametrized algorithms, for instance the amount of noise added to the data can be varied. The tuning of those parameters enable to leverage their action, i.e. to enforce more or less privacy on the data. This property is highly valuable considering that privacy often comes at the cost of a reduction of the service utility. Hence, a configurable LPPM enable to tune the privacy to utility trade-off. However, the fine tuning of these parameters may require sophisticated knowledge and experience. The choice of these configurations significantly impacts the level of protection of the obfuscated data. The obfuscation should be carried out carefully in order to make sure that the utility of the protected data is preserved. Indeed, the mobility data are directly used by the LBS to provide the user with a service, and when taking the LBS point of view, those data are jointly processed to retrieve some high level information (road usage, means of transportation, etc.). Dealing with both privacy and utility at the same time is not straightforward given the natural trade-off that exists between the two. As privacy enhancing mechanisms consist in deforming the original data to hide information, their usability is by definition decreased.

Moreover, every user mobility is highly dynamic, with varying speeds and frequencies of move. Thus, the application of a protection mechanism may result in various levels of privacy and utility depending on the properties of the user's mobility.

Objectives. To sum up, the objective is to guarantee given levels of privacy and utility both from the user perspective and the data analytics one. Moreover, given the variability of users' moves and his or her environment, the guarantees of privacy and utility should be robust.

4.2 Performance and Reliability of Cloud Services

In the last decades, the world has faced a steep surge in the number of produced data. The previously presented case of people mobility data collection is a perfect example of how everyday activities generates huge amounts of data: web pages requests, financial transactions, etc.. These Big Data bring novel challenges for their storage and analysis.

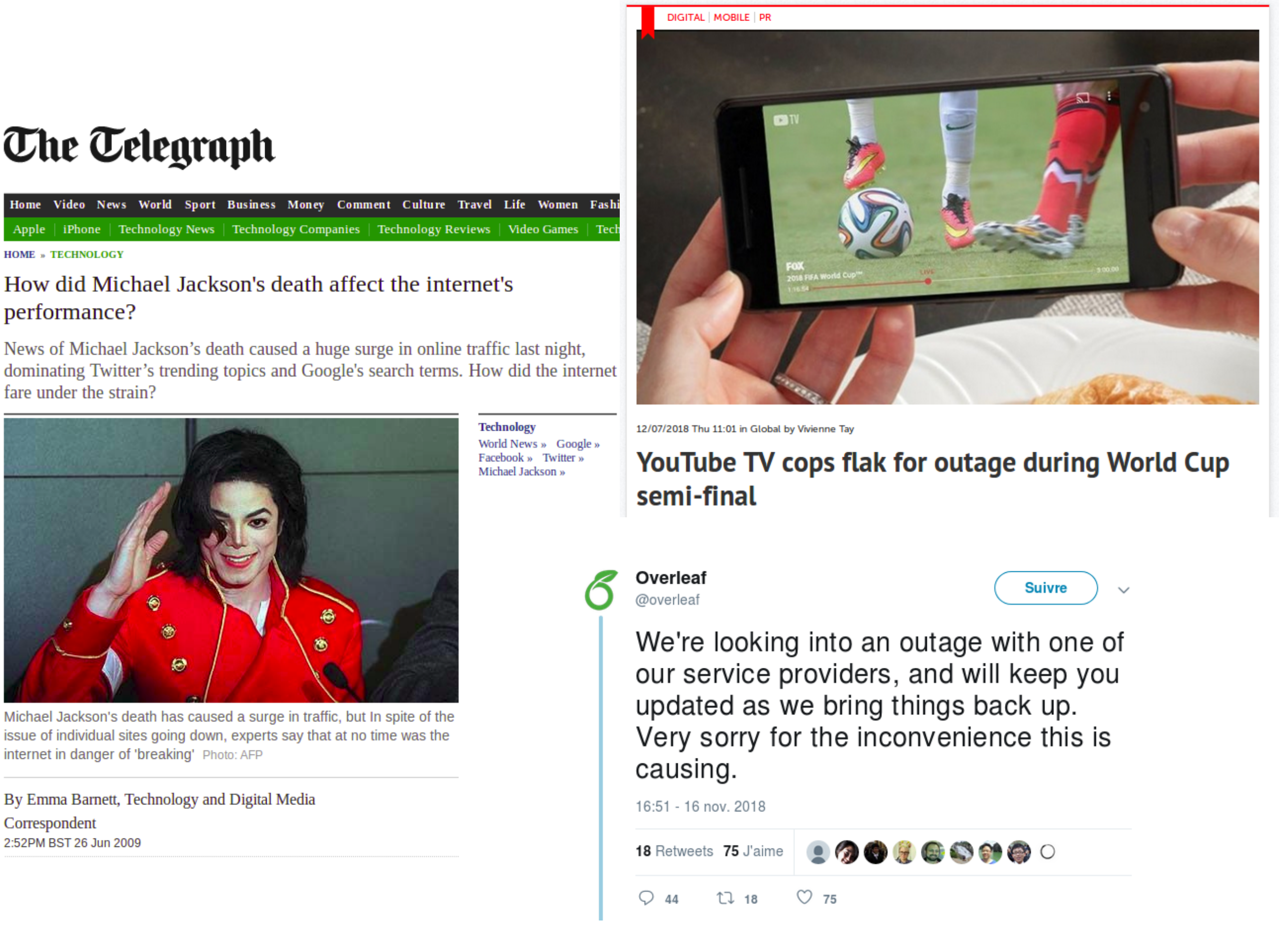
New programming paradigms have emerged to face Big Data specificities, such as MapReduce, Hadoop or Spark. Cloud computing, with its promise of barely unlimited storage and processing capabilities, is becoming an increasingly attractive solution, see Figure 4.4 for on overview of some of the most famous cloud services [142]. The use of clouds enables to share with others the physical resources, as most applications do not require their full time usage.

Figure 4.4 – Common Cloud Services

One of the biggest appeals of cloud computing is the *on-demand* assigning of shared hardware resources to software applications, called elastic resource provisioning. This enables to provide more resources to applications that need it *when* they need it, and thus try to optimize the resource usage. Therefore, there is a growing need for frameworks that efficiently deal with the dynamics of cluster resources [119].

Current autonomic resource provisioning approaches that are deployed in public clouds still don't work optimally for real time applications. In most clouds, if an application has to meet run time criteria, a reactive algorithm scales the resources to the demand. However, it requires high level of expertise to decide on how much to intervene, as it depends on many factors that are not necessary directly accessible. The difficulty of the task is increased by the fact that the optimal configuration can vary due to the shared use of hardware resources or workloads fluctuations over time [20], among many other factors [103]. Therefore, there is a need for fully automatized and robust cluster scaling algorithms able to deal with these environment disturbances.

Ensuring service performance is essential as missing deadlines result in consequent financial losses. It is estimated that an on-line brokerage industry service unavailability costs around 100.000$ per minute. For on-line shopping companies, page load-time should not be more than 2 seconds or users tend to give up their shopping or switch to another seller, resulting in revenue loss for companies [137]. Figure 4.5 gives examples of unmastered resource usage that led to significant consequences for companies. Cloud providers have to guarantee service performance and availability in order to win users' loyalty and avoid to huge financial impacts.



The Telegraph

Home Video News World Sport Business Money Comment Culture Travel Life Women Fashi
Apple | iPhone | Technology News | Technology Companies | Technology Reviews | Video Games | Tech
HOME » TECHNOLOGY

How did Michael Jackson's death affect the internet's performance?

News of Michael Jackson's death caused a huge surge in online traffic last night, dominating Twitter's trending topics and Google's search terms. How did the internet fare under the strain?

Technology
World News » Google »
Facebook » Twitter »
Michael Jackson »

Michael Jackson's death has caused a surge in traffic, but In spite of the issue of individual sites going down, experts say that at no time was the internet in danger of 'breaking' Photo: AFP

By Emma Barnett, Technology and Digital Media Correspondent
2:52PM BST 26 Jun 2009

12/07/2018 Thu 11:01 in Global by Vivienne Tay

YouTube TV cops flak for outage during World Cup semi-final

Overleaf
@overleaf

We're looking into an outage with one of our service providers, and will keep you updated as we bring things back up. Very sorry for the inconvenience this is causing.

16:51 - 16 nov. 2018

18 Retweets 75 J'aime

Figure 4.5 – News breaks showing that unmastered resource leads to service unavailability. All outages were caused by unusual workload, where some where however expected to be of huge amplitude.

Ensuring the performance of cloud services is a highly complex challenge. A cloud service behavior varies over time due to the dynamic of its environment (hardware, network, etc.). As cloud providers desire to maximize the utilization of their clusters, they have mechanisms for the dynamic reallocation of unused resources in the cluster, which further add to the variability of system performance. Hence, even with the same workload and the same resource amount, an application performance may vary depending on how noisy neighboring applications are. Moreover, cloud services are highly complex paradigms that run on top of multiple software stacks, making their behavior regarding resource provisioning highly non linear.

Objectives. To sum up, the objectives in this context are to ensure desired performance and availability of cloud services while being robust to the changes in the application and its environment; in a financial and energetic cost-aware manner.

4.3 Contributions of the Thesis

The challenges of the two uses cases presented in the beginning of this chapter have been overcome in this thesis by four major contributions. They are presented in the next subsections, where an overview is given on the scenario considered, the approach used and the publications that resulted from these works.

4.3.1 Automatic Choice and Configuration of Location Privacy Protection Mechanisms

Mobility datasets collected by companies or researchers need to be protective regarding users' privacy and still enable to extract useful aggregated information. Location Privacy Protection Mechanisms (LPPMs) shall be used. However, the appropriated LPPM and its configuration to use may vary for each user. Moreover, some guarantees regarding the levels of protection and utility of resulting dataset is needed.

PULP framework (standing for Privacy and Utility through LPPMs Parametrization) realizes user-level objective-driven recommendation of the best LPPM to use and how to configure it. It works in three steps: first the profiling of the impact of LPPM on individual users, then the non-linear modeling of the LPPM behavior, and eventually the recommendation of a properly tuned LPPM for each user. The recommendation is based on objectives, such as the trade-off between privacy of data and their utility.

This work has been presented in an international conference [47] and is further extended in a journal version [45].

4.3.2 Dynamic Control of Utility and Location Privacy

The dynULP framework (standing for dynamic and Useful Location Privacy) takes the perspective of an individual using a mobile device to enjoy the service of a LBS. The data sent should be protected while the service received should be useful. In this dynamic and user-centric scenario, the need for a rightful configuration of the LPPM also rises. Moreover, as the mobility of the user is changing over time, the protection mechanism should also evolve to ensure a constant level of privacy.

Control theory is applied to this use-case. Privacy is considered as a measurable signal to control, while the configuration of the LPPM enables to leverage it. Contributions have been made on the problem formulation, i.e. finding proper ways to measure privacy and utility of data to enable controlling it. Then, modeling tools and control law have been applied to ensure privacy reference tracking and robustness regarding user's mobility.

The work can be found in two international conferences [38, 49] and a journal submission is under review.

4.3.3 Adaptive Control for Cloud Service Performance Robust to Plant and Environment Changes

The cost of cloud services is continuously decreasing, hence service performance is becoming a key differentiator between providers. Solutions that aim to guarantee Service Level Objectives (SLO) in term of performance by controlling cluster size are already used by cloud providers. However most of these control solutions are based on static *if-then* rules, they are therefore inefficient in handling the highly varying service dynamics of cloud environments.

In this thesis, a novel adaptive control approach realizing resource allocation is presented that is robust to these phenomena. It consists of PI and feedforward controller which parameters are adapted online. The use of adaptation enables to improve control efficiency and robustness with respect to variations in the dynamic of the system.

Results of this approach have been presented in two international conferences [43, 110].

4.3.4 Cost-aware Optimal Control of Performance and Availability of Cloud Services

High rate cluster reconfigurations is a costly issue in Big Data Cloud services. Current control solutions manage to scale the cluster according to the workload, however they do not try to minimize the number of system reconfigurations.

This thesis presents a novel event-triggered optimal control approach. The triggering mechanism relies on a Model Predictive Controller and is defined upon the value of the optimal cost function, to reduce the number of control changes but also ensures a very reactive behavior to changes of exogenous inputs.

This work has lead to a publication in the control community [41].

Part II

Privacy and Utility Aware Control of Users' Mobility Data

This part tackles the issue of location privacy, i.e. the protection of people's privacy for whom their mobility information is shared with untrusted parties. Two scenario appear: (i) the locations are broadcasted in exchange of a service delivered to the mobile device user, e.g. a geo-located weather forecast, or (ii) datasets of many users are used to performs analytics, e.g. finding the most crowded streets in a city. In both cases, two main notions can be identified: knowing the privacy protection of mobile devices holders and the utility or quality of the service.

The tools of modeling, configuration and control for this use-case is presented here. Chapter 6 presents PULP (Privacy and Utility through LPPM Parametrization), a framework that address the static scenario where a mobility database has been recorded and privacy protection needs to be performed, wisely in order to ensure some accuracy of the performed analytics on it. Such scenario is particularly pervading since the European regulation GPDR effective since May 2018 [79]. PULP finds the best Location Privacy Protection Mechanism and configures it at a user granularity to satisfy both privacy and utility objectives. Chapter 7 presents dynULP (dynamic Utility-preserving Location Privacy) the user-side version of PULP, which performs online protection of location data before sending it to a third party while keeping a eye on the returned service quality. In dynULP, the action of the protection mechanism is leveraged on line to cope with changes in users' mobility and environment, and its objectives are driven in a way that enables the user to choose high utility when she needs it, and enforced privacy (i.e. relaxed utility) otherwise. Beforehand, Chapter 5 gives all the context and background needed to fully understand the data and mechanisms involved, and present the related works from the state of the art. Eventually, Chapter 8 concludes on the contributions of those works regarding location privacy, and gives perspectives for future works.

Chapter 5

Location Privacy Background and Related Work

First, the key concepts of Location Privacy are presented in this Chapter: the location data itself and the properties of a mobility dataset; the notion of privacy related to location data, both practically and theoretically; and the state of the art solutions that aim at protecting privacy, with illustrated examples. Then we focus on evaluating the protection mechanisms through the definition of state of the art privacy and utility metrics. Finally, an overview of the state of the art on adaptation or configuration of protection mechanisms is given.

5.1 Mobility traces and datasets

Mobility data are records of locations over time. The location is often the one of a person using a handled device, but it can also be records of some transportation modes (car, bikes, etc.). The databases can reflect two scenarios: either a user[1] is collecting her locations and sending it from time to time to an external service (e.g. crowdsensing applications) possibly with some remuneration; or a person is using a location-based service that requires the user's location continuously through time.

Formally, mobility databases are GPS points (latitude, longitude) labeled with timestamps. In the following, a mobility trace will be referred to with the notation M, as mobility trace, which is mathematically a matrix with three columns: latitude, longitude, time. When the user that realized this mobility is specified, it is with the subscript i: M_i. A mobility trace is basically two signals evolving through time, with most of the time a varying sampling rate. Many processing can be done to those signals to characterize them. We will consider two complementary axis of analysis: (i) the time scale and (ii) the derivation degree, as explained in the following. The time scale can be instantaneous t, a small variation Δt or a time window T. The derivation can be of order 0 (position through time), order 1 (speed) or order 2 (acceleration). By looking at the different signals (position, speed, acceleration) in the various time properties enables to characterize the mobility, e.g. small variations of position gives the changes in the user direction, the speed of the user at a given time, etc. It exists some redundancy in this formulation, e.g. small variations of speed are indeed acceleration, without being detrimental to the analysis. These properties of the mobility trace are what LBS use for improving their services, it will be discussed more in details in the Utility Metric Section 5.4.2.

In this work, two types of mobility data have been used: datasets collected on the field, by mobile devices carried by users or synthetic data that we generated in order to emphasis some properties. The first dataset has the advantage of being realistic while the second enable to control and cover

[1] Note that in the Privacy community, the user is by convention referred to as a female. This usage will be kept for the rest of this manuscript.

all possible scenarios. Hence, synthetic data will be mostly used for analysis and modeling while real-life datasets will enable validation and evaluation of our approaches.

For the field data, four datasets have been considered: Cabspotting [148], Privamov [31], Geolife [191, 189, 190] and Mobile Data Challenge [102, 112]. They differ in their number of user sensed, duration of the collection campaign, area covered, sampling time, number of records, etc.

Cabspotting dataset [148] gathered the mobility records of 536 taxi cabs during their service in San Francisco Bay Area, USA, over one month, in May to June 2008. The sampling frequency is of the order of magnitude of the tenth of seconds. Privamov dataset [31] records the location traces of 100 student and academic staff smartphone users in the region of Lyon, France. Data collection lasted from October 2014 to January 2016 (15 months), with an average sampling period of 10 seconds. The Mobile Data Challenge dataset (MDC) [102, 112] involves 185 users around the Lake Geneva region, Switzerland. Data were collected between October 2008 and March 2011 (3.5 years), with a sampling period of around 10 seconds. The last dataset considered is GeoLife [191, 189, 190], collected by Microsoft Research Asia in Beijing region from April 2007 till August 2012 (5 years) upon 182 users, with a higher sampling frequency that the other datasets: around a second. This dataset has the advantage of have been labeled by the users regarding their transportation mode.

For fair comparison, we align the length period of the four datasets to that of the shortest one (i.e. Cabspotting which has 30 days of mobility data). Hence, we extracted the most active period of 30 days from Privamov, GeoLife, and MDC data collections. Details on the restricted datasets can be found in Table 5.1.

Dataset	**Location**	**#users**
Cabspotting	San Francisco, USA	536
GeoLife	Beijing, CN	42
MDC	Geneva Region, CH	142
Privamov	Lyon, FR	48

Table 5.1 – 30-days Mobility trace datasets

An extract of a mobility trace from a Cabspotting user can be found in Figure 5.1, the location points on the map of San Francisco on the left and the variation of the user's speed through time on the right. Some sections of this trace have been highlighted to show mobility diversity. First, the cab driver is stopped at a cab station (yellow dots). Later on, he drives with high speed on a highway from the airport to the city (blue crosses). Eventually, he drives in the city from a hotel to the piers with relatively slow speed (green circles).

The synthetic data we generated is a variation on all the above mentioned properties of a mobility trace. Its main characteristics are the speed of the user and the period between two changes of the speed, that are illustrated in Figure 5.2. Speed can be high (e.g. in a train), medium (is a car or a bus), low (walking or by bike) or null (the user is stopped). Frequencies of changes ranges from a short stop at a traffic light (less than a minute) to a longer stop in a shop (a tenth of minutes) or even longer stops (hour-long). In the first two hours, the direction of the movement is varied. Around hours 8 to 9, the acceleration of the user is considered. From hour 11, the user do not really stop, but has low speed in a restricted area (e.g. walking around a mall). Those late properties are however hardly visible on the speed over time plot of Figure 5.2. Sampling frequency is set constant at 10 seconds.

To sum-up, the user mobility is unpredictably changing with various amplitude, frequencies and sampling time. Robustness against such behaviors is a real challenge.

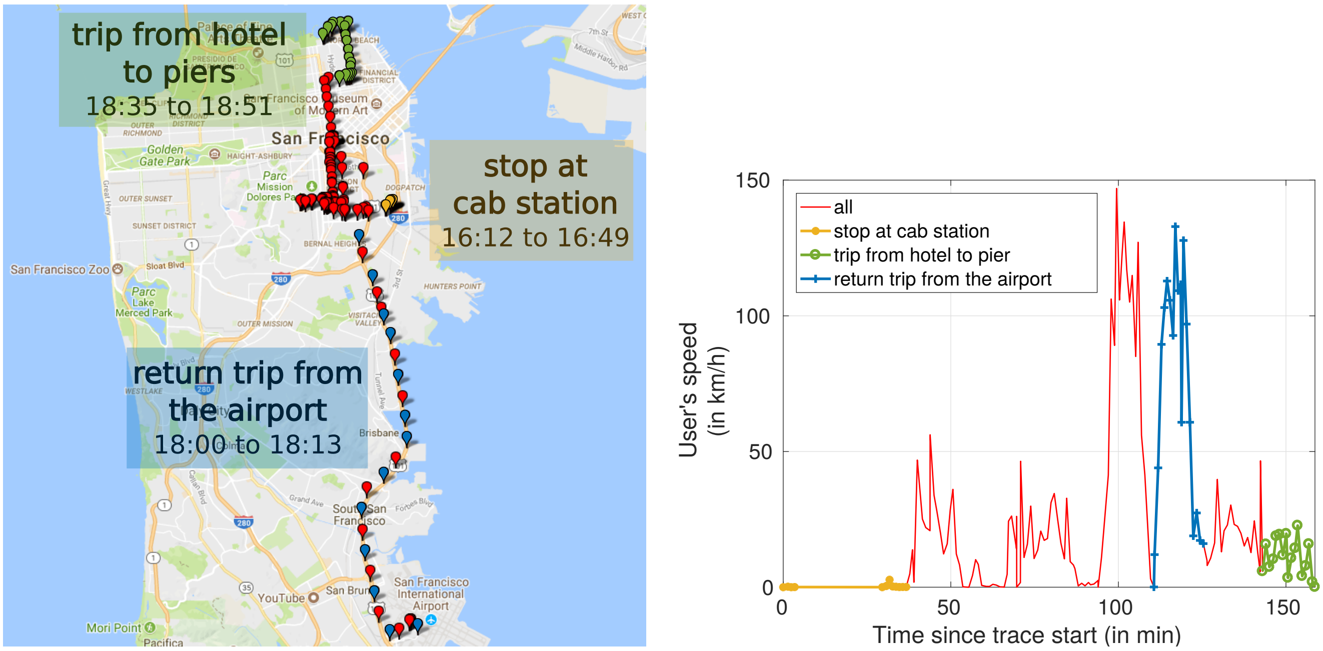


Figure 5.1 – GPS data are dynamic. Mobility data of a user from Cabspotting dataset illustration: (a) on a map and (b) trough speed over time of various activities (a stop and two movements).

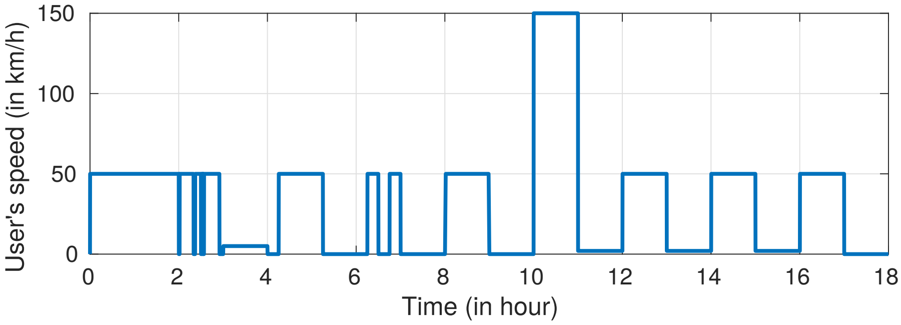


Figure 5.2 – Synthetic mobility trace scenario.

5.2 Notions of Location Privacy

The processing of location data also comes with threats on the privacy of the recorded users. As a motivation for privacy protection among many others, one can cite the publication of a mobility dataset by Strava in 2018 that revealed the locations of unknown US military bases [88]; or the new regulations that are enforced by the governments, such as the European GDPR [79].

5.2.1 Threats

Many threats exist regarding privacy, which are coming from various processing of the data [69, 33]. The most common ones are:

- Re-identification attacks. The attacker has access to prior knowledge about the user, such as publicly available information or former mobility records, and tries to guess the identity of an anonymous user [73, 126]. These approaches have shown that simple anonymisation, or pseudo anonymisation that just remove the name of users is not enough to guarantee their privacy.

- Mobility prediction. The attacker tries to anticipate users' next moves based on their habits [184, 75]. Such knowledge could be used for commercial purposes (such as a cab company suggesting you a ride) or even personal interception.
- Social relationship inference. By processing one's mobility pattern, the nature of relationship between people can be found such a partners, co-workers, etc.

5.2.2 Definitions

Defining privacy is not a easy task, but there are flourishing attempts in the literature. Nevertheless, one can differentiate between two main categories: the formal definitions and the practical ones.

A well-known privacy guarantee is k-anonymity [167], which states that a user is k-anonymous if it is hidden among k-1 other users sharing similar properties. In the context of location privacy, it means that, instead of reporting their exact location, users report their position inside cloaking areas containing at least k users. This method has been successfully implemented using a trusted third party to compute cloaking areas (e.g., CliqueCloak [81]) as well as in distributed systems relying on peer-to-peer communication between users (e.g., PRIVÉ [82]). Another popular privacy guarantee is differential privacy [65], which ensures that the presence or absence of a single user from a dataset should not significantly affect the outcome of any query on this dataset. Differential privacy has been applied as such in [97] and [18], where a controlled amount of noise was added to each location of a mobility trace. Those two concepts of k-anonymity and differential privacy are not specific to location privacy and are used in many other fields. However, they are the one that transpose the best to the location specific challenges.

Practically, there are a lot of location privacy metrics, see [152] for a overview. A key concept of privacy is the notion of Points Of Interest (POI), which are the significant places where a user spends his time, such as home, workplace [74], place of worship [72]. POI are a particular case of the data density property, which has been identified as a privacy thread indicator [33]. The retrieval of POI is a crucial knowledge as it can reveal many information about the user. Indeed, it is often the very first step for performing the above mention attacks, see [150] for re-identification, and [75] for mobility prediction.

Hence, this work considers the obfuscation of Points Of Interests as being the practical definition of location privacy protection. We however acknowledge that some works of the literature consider the POIs retrieval as being on the utility side [185, 55, 128].

5.3 Location Privacy Protection Mechanisms

The denomination LPPM (for Location Privacy Protection Mechanisms) groups all the processes or algorithms that, by manipulating location data, aim at improving the privacy protection of the people involved in the mobility. LPPMs attempt to enhance location privacy of users willing to interact with location-based services.

Roughly speaking, state-of-art LPPMs alter the spatial information of user mobility data and/or the temporal information of the data. They take as input a mobility data and outputs another mobility data, hopefully a more private one. The input data is called raw data, or original one; while the output is called obfuscated data. While a raw mobility trace (i.e. collection of records) is referred to at M, or M_i where the subscript i refers to the user; the obfuscated trace is noted M', or M'_{ij} when the user and the LPPM used for obfuscation is specified. A LPPM transforms M into M', but the mechanism applied can be of many kinds. It may require as input a single location data or a sequence of locations (i.e. a trace) from a single user or from several. It may also use external knowledge, such a semantics of the location or of its demography. The transformation to the data itself is performed through a removal of location, a modification or even as an addition of new data.

Although our work does not consist in designing a new LPPM, we briefly present here some prominent privacy protection schemes. An interesting way to categorize LPPMs is to look at the general principle of their protection mechanism. Therefore, according to [152], LPPMs can be classified in four categories:

- Mix-zones. This online approach creates zones in which users exchange their identifiers in order to fool the service, and redistribute the results to the right users. This follows the *k*-anonymity concept. It has been developed by [28] and [27]. A version has been developed where trajectories are exchanges based on their entropy to achieve equiprobability regarding data sensitivity [32]. One issue that arise with this approach is to find the right delimitation of those zones and their size (see [115]). The main limitation of this approach is that it requires many users using it in a restricted area, thus it may not be always usable in practice.

- Generalization. A user location has usually a really good accuracy from few meters for classic GPS systems [139], to a tenth of a meter for more accurate ones, using WiFi spots for instance [104]. Generalization-based mechanisms tends to reduce this accuracy by reporting the location of the user with location zones instead of a precise point. Such areas are called spatial cloaking zones. The location based service knows the location of the user only with a fixed, deteriorated, accuracy that depends of the size of the cloaking area.
  An implementation of such principle requiring a trusted third party server has been developed by [136], with in addition *k*-anonymity guarantees. This approach has hence two parameters: the number of people between which you are hidden (*k*) and the size of the cloaking zone. A peer to peer version has been presented in [56]. Another approach consist in providing differential privacy guarantees on top of cloaking areas [140]. Eventually, offline versions of the generalization based mechanisms have been presented by [1] and [2], where location databases are modified (both spacial and temporal data) to achieve *k*-anonymity.
  The main limitation of such approach are that the LBS is required to work with spacial zone. If not, one can imagine reporting only the center of the cloaking zone, but then the guarantees are not exactly the same, and the mechanism is close to perturbation-based one (see fourth point). For some implementations a large number of people using the LPPM are required [136], and might have a high computing complexity.

- Dummies. The idea behind those protection mechanisms is to hide the user between fake ones. Extra locations or even users are generated, which are hopefully realistic enough to make the user indistinguishable. The first online implementation of such principle has been provided by [100], and has been extended by [165] with a historic considerations that enable to ensure *k*-anonymity. The offline version of the dummies principle has been developed by SybilQuery [161] which create false traces that still preserve some properties of the original trace like its length or some semantic information of the places visited.
  Depending on the quality of the fake locations generators, it can be quite easy with some processing to distinguish them from real one, for instance by looking at the instantaneous speed or by mapping the movements with a street map. Moreover, such mechanisms are often highly demanding in communication data and in computing.

- Perturbation. More or less all the LPPMs that do not fit in the previous categories fall into the perturbation one, which make this latter really diverse. The common point of those mechanisms is the modification of the mobility traces with the addition of some kind of noise. A first

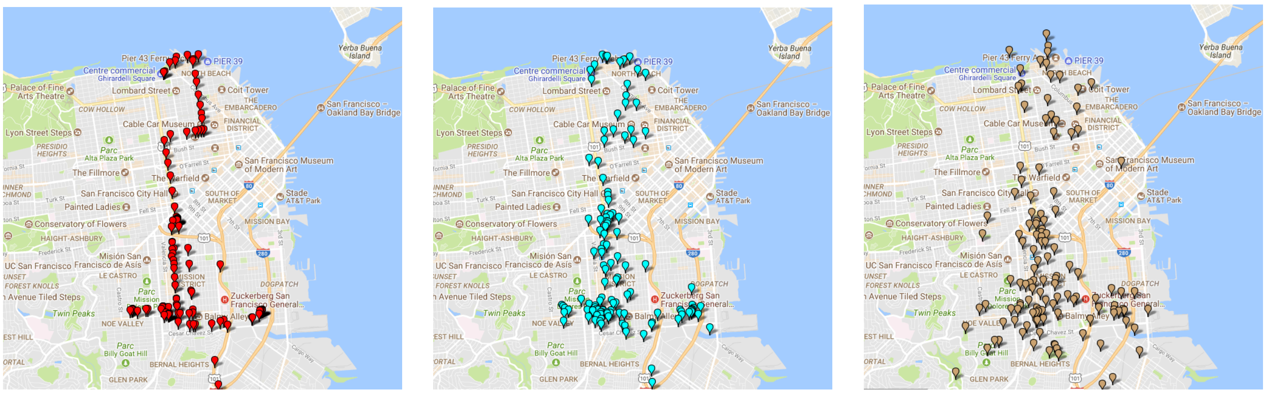
> example of such LPPM that works online and alter the spatial dimension of the data is Geo-Indistinguishability [18], that ensure differential-privacy guarantees. However, this process is quite expensive in privacy budget ε, so the authors developed a extension that uses prediction to overcome this issue [52]. An approach using the temporal correlation between the locations of the users are used in [182] to enable once again differential privacy but with higher utility and less budget. For the offline approach of perturbation based protection mechanisms, some differential privacy have also been applied [97] while other methods advocate for path confusion between various users [92]. Eventually, some mechanisms took the approach of generating completely fake traces but that preserve the information needed by the LBS to provide its service [134], [30]. Those approaches differ from the dummies ones as they consist in sending only a single fake trace, not hiding the true one between some synthetic ones.

Eventually, some protection mechanisms are not location specific but can still be used to ensure location privacy, such as LBS with privacy-by-design guarantees or private query engines. In the first one, the LBS itself is changed to ensure privacy guarantee. Those approach are however out of the scope of this thesis.

In the following, we present in detail two examples of LPPMs in detail that will be used as application cases of the proposed control approach. The two LPPM are complementary as the first one Geo-Indistinguishability focuses on spatial distortion of user mobility data and can work in an online fashion, while Promesse adds temporal disturbance to a location database.

Geo-I. Geo-Indistinguishability protects user's location data by adding spatial noise drawn from a Laplace distribution to the actual user location of each record in the mobility trace [18]. Geo-I has a configuration parameter ε, expressed in inverse of meters varying in $\mathbb{R}^+$, which quantifies the amount of noise to add to raw data. The lower the ε is, the more noise is added. Geo-I is a state of the art LPPM that follows the differential privacy model [65].

Figure 5.3 illustrates the application of Geo-I on the mobility trace of a user from Cabspotting for two values of the parameter ε (b) and (c), and the raw trace as a comparison (a). The noisier the data are, the more the user privacy is preserved: less information can be inferred from the trace. However, the less accurate the service will be as the reported location data are far from the actual one. Tuning Geo-I parameter enables to leverage the privacy protection and also the utility of the data sent to the LBS.



(a) no obfuscation

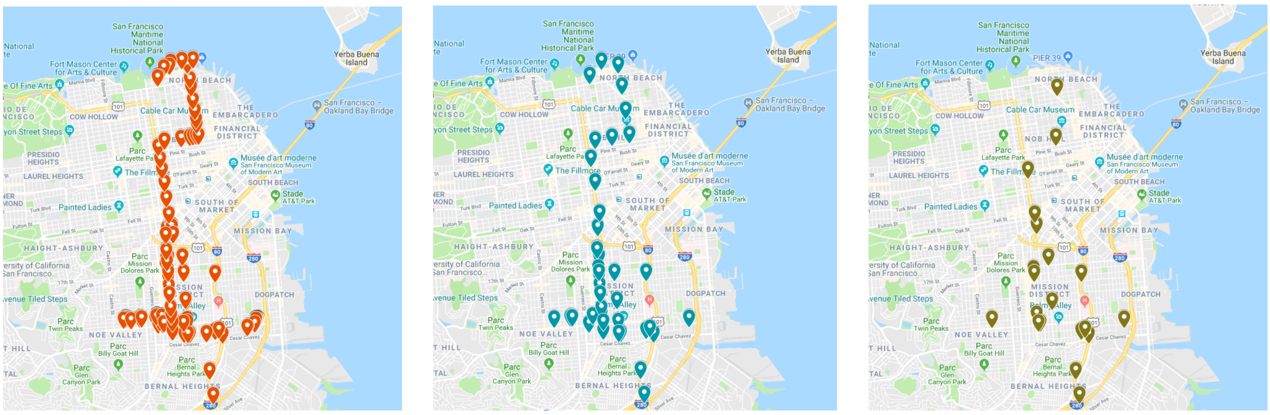
(b) obfuscated with Geo-I. ($\varepsilon = 10^{-2}$)

(c) obfuscated with Geo-I. ($\varepsilon = 10^{-2.5}$)

Figure 5.3 – Obfuscation of a mobility trace using Geo-I. Illustration on a Cabspotting user for various configurations. (a) user raw trace, (b) obfuscation with $\varepsilon = 10^{-2}$, (c) obfuscation with $\varepsilon = 10^{-2.5}$.

Promesse. The LPPM Promesse has been developed in order to prevent the extraction of Points-Of-Interest (users' stop places) while maintaining a good spatial accuracy [150]. Its principle is to distort timestamps of location traces as well as remove and insert records in a user's trace in order to keep a constant distance between two events of the trace (tunable with a ε parameter in meters). One can see its behavior as adding temporal noise to a trace instead of spatial noise as in Geo-I. However, as timestamps of location data are modified, this LPPM can only be applied offline to a whole database or semi-online, using batches of data.

Figure 5.4 illustrate Promesse obfuscation on the same Cabspotting user. No obfuscation is done on the left map, while the middle one has Promesse applied with a medium value parameter ($\varepsilon = 400\,\text{m}$) and the right plot has a larger ε, $1000\,\text{m}$. The distance between two consecutive points is fixed, but the points do not seem to be on a grid, as the user is moving around. Privacy protection is higher with the larger ε, however utility of the trace is also reduced.



(a) no obfuscation

(b) obfuscated with Promesse. ($\varepsilon = 400\,\text{m}$)

(c) obfuscated with Promesse. ($\varepsilon = 1000\,\text{m}$)

Figure 5.4 – Obfuscation of a mobility trace using Promesse. Illustration on a Cabspotting user for various configurations. (a) user raw trace, (b) obfuscation with $\varepsilon = 400\,\text{m}$, (c) with $\varepsilon = 1000\,\text{m}$.

5.4 Evaluation of LPPMs

Many LPPM flourish in the literature, and a big challenge that we are facing currently is to be able to evaluate them. Indeed, even though some rely on a strong theoretical basis, some are not, and in both cases their practical performance need to be assessed to allow a fair comparison of the various mechanisms. A contribution in this sense has been done in Primault's thesis [149]. Three aspects must be evaluated: the privacy protection level, the quality or utility of the service answer and the performance in terms of time or computing. While the last aspect is quite straightforward and unanimous, the privacy and utility require more explanations and justifications. There is no standard way of assessing these two complementary dimensions associated to LPPMs at a user level. We advocate to define privacy by looking at a user's POI (i.e. significant stops) protection [74], and utility by evaluation the spatial accuracy of revealed locations [53]. Both metrics evaluate the gain in privacy and the loss of utility of the obfuscated data compared to the raw data. The proposed metrics have parameters that enable the notion of privacy and utility to be adjusted to the considered LBS and to the user requirements.

In the following, we review standard privacy and utility metrics for a user mobility data and present the one we opted for, before illustrating them when applying Geo-I and Promesse LPPMs.

5.4.1 Privacy metric

5.4.1.1 Overview

Notions of privacy have already been presented in Section 5.2, we now discuss how to evaluate the location privacy protection of a user. We will not try to give an exhaustive list of the metrics that have been used in the literature but only sum up two surveys that give a good overview of those methods [107, 152]. Privacy can be evaluated using (i) formal guarantees such as k-anonymity or ε differential privacy, (ii) data distortion such as entropy of the data or (iii) attack correctness, which consist in simulating a malicious behavior and see what can be retrieved from the data.

This work will consider a privacy metric of this third category, i.e. the robustness to POI retrieval. We do not claim that privacy can be reduced to this metric, however it is the one we found the most relevant for location privacy due to its intermediate scale between data point and global knowledge, and it has been significantly mentioned and used in the literature [75, 150].

5.4.1.2 Definition

In the following, we formally define the metric used, starting with the notion of Point of Interest. Note that this definition only apply for evaluating the privacy of a mobility dataset.

A POI (point of interest) is a meaningful geographical area where a user made a significant stop. A POI is defined by the position of a centroid of a given diameter d where the user stayed for at least t minutes. We define $poi(M)$ as the set of POIs retrieved from the mobility trace M.

Using the concept of POI and $poi(\cdot)$ set, we aim to quantify a user's privacy level by how POIs retrieved from the obfuscated data (under LPPM j) successfully match the POIs retrieved from the non-obfuscated data, i.e., comparison between set of $poi(M_i)$ and $poi(M'_{ij})$. We define the function $\text{Matched}(poi(M'_{ij}),poi(M_i))$ that, given two sets of POIs, derive the subset of $poi(M'_{ij})$ containing the POIs that match with POIs in the second set $poi(M_i)$. Two POIs are considered as *matched* if they are sufficiently close to one another (d_{max} being the maximal distance threshold). To formally define privacy, one can either use the measurement of precision ($P_{pr}(i,j)$) which defines the ratio between the number of obfuscated trace's POIs successfully matched with real POIs and the number of obfuscated POIs,

$$P_{pr}(i,j) = \frac{|\text{Matched}(poi(M'_{ij}),poi(M_i))|}{|poi(M'_{ij})|},$$

or recall ($R_{pr}(i,j)$) which defines the ratio between the number of obfuscated trace's POIs successfully matched with real POIs and the number of real POIs,

$$R_{pr}(i,j) = \frac{|\text{Matched}(poi(M'_{ij}),poi(M_i))|}{|poi(M_i)|}.$$

The precision function assesses the accuracy of the matching while the recall function evaluates the completeness. To cater to both measurements, we advocate using F-score (see eq. (5.1)) to reconcile both the precision and recall.

We formally write the privacy metric, showing the normalized percentage of successfully hidden (non-matched) POIs, after applying LPPM j on user i as:

$$Pr(i,j) = 1 - \frac{2 \cdot P_{pr}(i,j) \cdot R_{pr}(i,j)}{P_{pr}(i,j) + R_{pr}(i,j)}. \tag{5.1}$$

This privacy metric is defined in the range [0,1] where a higher value reflects a better protection.

Leveraging the POI diameter d, its minimal stay time t and the matching threshold d_{max} enables a user's conception of her privacy to be clearly defined. For instance, a user wanting to hide her home and work place with a high accuracy should choose a large t, and small d and d_{max}. However, if a user wants to hide most of the places she goes to, in order to dissimulate her hobbies, she should set a small t and a rather large d_{max}.

5.4.2 Utility metric

5.4.2.1 Overview

The notion of utility, or quality of the data or service, is not easier to define than privacy. In the beginning of location privacy research, it was even absent from the main considerations. In Krumm's survey of 2009 [107], only the last paragraph mention the quality of service and the importance of the privacy over quality trade-off, but no definition nor overview is given. In Primault's 2018 survey [152], utility metrics are categorized as either *data distortion*, which is spatial (local or area-based) and temporal precision, or *task oriented*, referring to data mining or analytics queries.

This work introduce another dimension to the utility metrics, driven by control theory. Utility is linked to the use-case scenario (online or offline), as well as to the recipient of the location-based query. We will thus differentiate two cases: (i) the LBS user-oriented utility and (ii) the off-line analytics utility.

An non-exhaustive overview of LBS categories and examples are given in Table 5.2, which are extensively inspired by [107, 152, 33]. For each of those categories, the property of the utility data used to perform the service is indicated. The properties are selected as explained in Section 5.1: instantaneous position, direction of the movement, instantaneous speed, instantaneous acceleration, average speed in the past instants. Two kind of applications appear: (i) some only require the current position while (ii) the others are using all the features of the mobility trace to provide their service. The direction and navigation apps can be studied more specifically, as it is one of the most common used LBS. In order to compute and keep update the trip to the destination, only the instantaneous position is

Location-Based Service		**Used trace properties**				
Category	**Example**	**current position**	**move direction**	**current speed**	**mean speed**	**current acceleration**
Direction, navigation	Waze	+	-	-	-	
Weather	Weather Channel	+				
Social games	PokemonGo	+	-	-	-	-
Venue finder	Trip Advisor	+				
Crowd-sensing	apisense	+	-	-	-	-
Public transportation	Mobile CFF	+				
Nearby friends	Facebook	+				
Movie times	Cinemap	+				
Discount coupons	RetailMeNot	+				
Local information	France Bleu	+				
Image Sharing	Instagram	+				
Fitness	Strava	+	-	+	+	+
Daylight	f.lux	+				
Meeting	Tinder	+				
Sky maps	Sky Walk	+	-			

Table 5.2 – Overview of Location-Based Services and the location properties used ('+' for main purpose and '-' for accessory need) to satisfy the end-user.

needed. Those services are often linked to venue finders (e.g. to find the closest gas station), that also use only the current location. However, some additional services can be provided, such as orientation help if the direction of the move is computed, speed limit control with the instantaneous speed or timing until end of the trip with average speed sensing. Consequently, sub-services of navigation apps require the accuracy of many of the trace properties, however the main service only need instantaneous location precision.

The other use-case is the knowledge inference from a mobility database. In this case, the utility is oriented toward the one performing the analytics, most of the time companies or sometimes researchers. Examples of such processing are transportation mean extraction, count of the transit at a given point (for infrastructure management for instance) or path matching to an external database (e.g. to compute the exposition of user to noise of pollution). In comparison to LBS user oriented utility, the notion of temporarily appears here, as well as the shift from local precision accuracy to more area based one.

While the notion of online user utility will be further developed in Chapter 7, the database, service-level utility measure is presented in the following.

5.4.2.2 Definition

To evaluate data utility, we resort to the comparison between the area coverage of the original mobility traces and the one of the obfuscated data, as it covers most of the use-cases presented precedent. The granularity of the utility is given at the user level.

Formally, we define the area coverage by the concept of cells. The size of the cell reflects the granularity of the considered spatial dimension, ranging from the size of a house to an entire city. A cell is said visited or covered by a user, if the mobility trace of the user contains at least one record with coordinates in this cell. We first define $cell(M_i)$ and $cell(M'_{ij})$ as the sets of cells visited by the mobility trace of user i, respectively before and after applying the LPPM. One can think of $cell(\cdot)$ as a set containing cells that are visited by a user. To enable the comparison of cell coverage across a user's trace, we use the measurement of precision and recall to describe the percentage of cells that are correctly covered by M'_{ij}, relative to the original cell sets from M'_{ij} and M_i, respectively. We formally write the precision and recall of correct recovered cells for user i as

$$P_{ut}(i,j) = \frac{|cell(M_i) \cap cell(M'_{ij})|}{|cell(M'_{ij})|},$$

$$R_{ut}(i,j) = \frac{|cell(M_i) \cap cell(M'_{ij})|}{|cell(M_i)|}.$$

Similar to privacy the metric, we finally define the utility metric of user i obfuscated with LPPM j, $Ut(i,j)$, by the Fscore reconciling the precision and recall of cell coverage

$$Ut(i,j) = \frac{2 \cdot P_{ut}(i,j) \cdot R_{ut}(i,j)}{P_{ut}(i,j) + R_{ut}(i,j)}. \tag{5.2}$$

This utility metric is defined in the range of [0,1] where a higher value reflects a better utility, meaning a better spacial accuracy of the LBS results.

Playing with the cells' size enables adaptation to the analytics performed. Some services require a really good spatial accuracy such as route matching and some are less demanding, such as weather related information extraction. For the first category, cells' size should be small (tens of centimeters) while for the other, the size can be much larger (more than a kilometer).

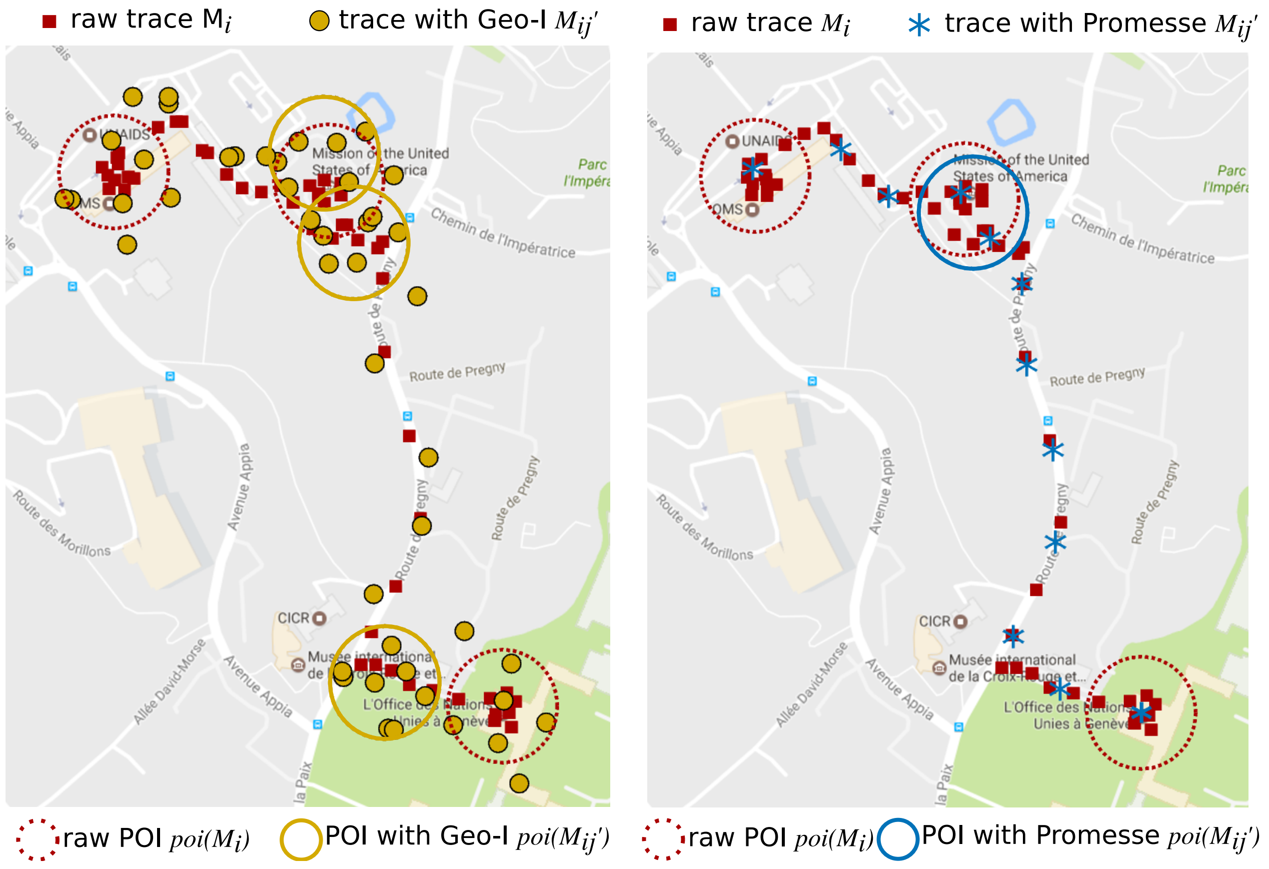
Note that the level of privacy and utility of a user depends not only on the LPPM used to protect her data but also of its configuration ε. However, for the sake of readability, we did not introduce ε here in our notations.

5.4.3 Illustration of Privacy and Utility Metrics with LPPMs

To better illustrate the definition of privacy and utility, we use a schematic example by applying Geo-I and Promesse on a synthetic user's trace, see Figure 5.5 and 5.6.

Computing privacy metric In Figure 5.5 (a), the raw mobility trace of the user M_i is represented with the small squares, each square being a location record. We overdraw the mobility trace of the user after using Geo-I (M'_{ij}), configured with a high ε i.e. low noise (small dots). We clearly see that the trace obfuscated with Geo-I corresponds to the original one but with some noise. For those two traces, we illustrate their Points-of-Interest (POIs) with large circles. The set of POIs of the original trace $poi(M_i)$ are the dashed circles, while POIs of the obfuscated trace $poi(M'_{ij})$ are the continuous ones. Based on those sets, we can compute the number of obfuscated POIs that match the real ones (here the two top ones $\text{Matched}(poi(M'_{ij}),poi(M_i)) = 2$). Then our privacy metrics can than be computed using the precision of the matching of POIs and its recall, that both are $2/3$. Then, the level of privacy is $1 - 2.\dfrac{2/3 * 2/3}{2/3 + 2/3} = 0.33$.

Figure 5.5 (b) is similar to Figure 5.5 (a) but here the considered LPPM is Promesse. In this case, the obfuscated data M'_{ij} (the small stars) are spatially regularly distributed (time-stamps are modified). In this illustration, all obfuscated POIs correspond to the real ones, the privacy precision is 1 and its recall is $1/3$. The resulting privacy value is then $2.\dfrac{1 * 1/3}{1 + 1/3} = 0.5$.

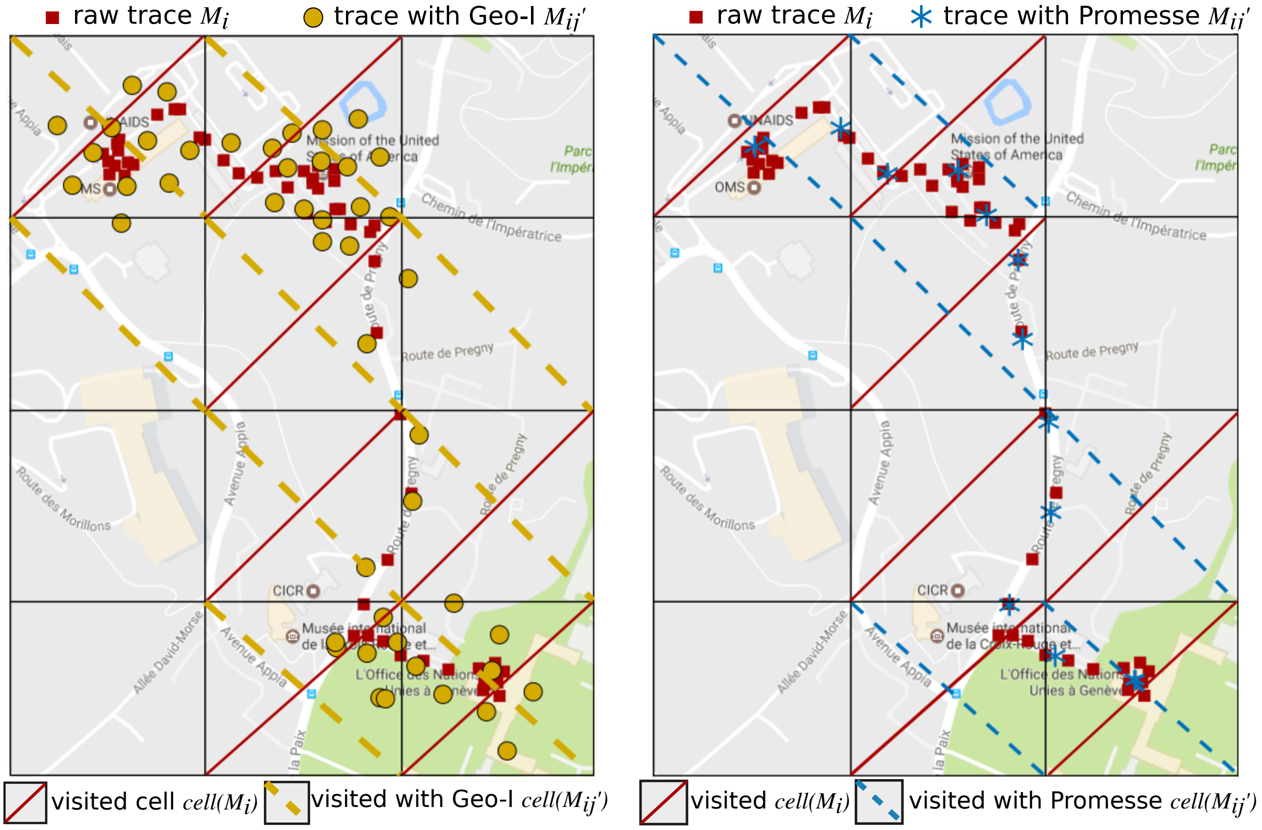


■ raw trace M$_i$
● trace with Geo-I M_{ij}'
■ raw trace M_i
✳ trace with Promesse M_{ij}'
UNAIDS
OMS
Mission of the United States of America
Parc l'Impéra
Chemin de l'Impératrice
Route de Pregny
Avenue Appia
Route des Morillons
CICR
Musée international de la Croix-Rouge et...
L'Office des Nations Unies à Genève
Allée David-Morse
la Paix
raw POI $poi(M_i)$
POI with Geo-I $poi(M_{ij}')$
raw POI $poi(M_i)$
POI with Promesse $poi(M_{ij}')$

(a) Geo-I

(b) Promesse

Figure 5.5 – Privacy computation: schematic examples of how POI retrieval is computed for a single user when using Geo-I and Promesse.


raw trace M_i
trace with Geo-I M_{ij}'
raw trace M_i
trace with Promesse M_{ij}'
visited cell $cell(M_i)$
visited with Geo-I $cell(M_{ij}')$
visited $cell(M_i)$
visited with Promesse $cell(M_{ij}')$
UNAIDS
OMS
Mission of the United States of America
Parc l'Impér
Chemin de l'Impératrice
Route de Pregny
Avenue Appia
Route des Morillons
CICR
Musée international de la Croix-Rouge et...
L'Office des Nations Unies à Genève
Allée David-Morse
la Paix

(a) Geo-I

(b) Promesse

Figure 5.6 – Utility computation: schematic examples of how cell coverage is computed for a single user when using Geo-I and Promesse.

Computing utility level Utility metric is illustrated in Figure 5.6 (a) for Geo-I and in (b) for Promesse. In each case, the set $cell(M_i)$ is illustrated by the cells with the right diagonal (7 in total) while the sets $cell(M'_{ij})$ are the ones with left dashed diagonals.

For Geo-I, the obfuscated trace covers 9 cells, the utility precision is $7/9$ and the recall 1, thus the utility level is 0.86.

From Promesse, the obfuscated trace covers only 6 cells, the precision and recall are respectively 1 and $6/7$, hence a utility of 0.92.

5.5 LPPM Configuration

What makes LPPMs difficult to use in practice is that they rely on a set of configuration parameters. For instance, the ε parameter of differentially private protection mechanisms is a sensitive parameter that has a great impact on the resulting data privacy and utility, see Figure 5.3 for an illustration.

In [106], the author showed that defeating a well-performing privacy attack would require adding so much noise that it would make the resulting data unusable by any LBS, and hence useless. With this inherent trade-off between privacy and utility, it is a difficult task to set LPPM configuration parameters to an appropriate value.

Here again, two approaches can be found in the literature, either formal or practical, implemented.

A formal framework for trading location privacy to the resulting quality of service has been first attempted by Duckham [63] in 2005. This paper present a negotiation algorithm to enforce a privacy to utility trade-off, abstract from any implementation, only taking the assumption of a perturbation-based LPPM. However, this approach do not get rid of the challenge of the LPPM configuration, and only consider a static scenario, unlike in our approaches.

A few works have been proposed to help a user choose a LPPM configuration that fits his actual needs. Agir et. al [3] proposed an adaptive mechanism that dynamically computes the size of the cloaking area the user will be hidden within. More specifically, starting at a given parametrization of the LPPM, they iteratively modify the configuration in a way that strengthen the privacy until a minimum privacy level, fixed by the user, is met. However, their privacy estimation routine has a complexity of $O(L^2)$, L being the maximum number of locations that a cloaked area can be formed of. This routine is further repeated until required privacy level is met or at most λ times.

Theodorakopoulos et al. [170] presented new LPPMs that adapt to the user mobility pattern. Knowing that a user trace is highly repetitive and correlated, it makes the user especially vulnerable and in the need for personalized protection. The attacked is considered as a playing a Bayesian Stackelberg game, in which he knows the details of the LPPM used and the profile of the user. Configuration parameter of the LPPMs are optimally computed.

L2P2 is a solution that enable to refine the size of the cloaking area of a user LPPM, based on her location-dependent privacy requirements [179]. This paper is one of the very few that address the issue of a real-time LPPM configuration, knowing that users' privacy objective vary depending on both their location and their need.

Chatzikokolakis et. al introduced an extension of Geo-I that uses contextual information to adapt the effective privacy level [53]. Specifically, the amount of noise effectively added to locations depends whether the user is located in a dense urban area or in the countryside. This qualification is done by looking at the density of venues (e.g., restaurants, monuments, amenities) in the vicinity. It is expected that the number of venues is higher in urban environments and will better hide the user's interests in the area than if located outside of a city. However, this approach still requires some parametrization from the user side, which is not objective-driven. Koufogiannis et al. adopts a similar approach, by combining differential privacy and the density of the surrounding areas, and prove this translate into a locally Lipschitz constraint [105].

Primault et al. presented *ALP*, a system that configures a LPPM depending on users objectives [151]. This solution relies on a greedy approach that iteratively evaluates the privacy and utility for refining configuration parameters. Evaluating privacy and utility has a complexity depending on the objectives under consideration, varying between $O(L)$ and $O(L^2)$. Moreover, the convergence is not ensured and consequently there is no guarantee that the objectives are actually met. However, this work is the closest to our approach, given its user-level objective formulation both in privacy and utility, defined in an applied way.

Chapter 6

PULP: Privacy and Utility through LPPMs Parametrization

This chapter presents *PULP* , a framework that automatically chooses between protection mechanisms and configure the chosen one in order to meet privacy protection objectives on a mobility dataset while guaranteeing utility on its processing. *PULP* performs profiling runs on the dataset and then uses nonlinear models to capture the impact of each LPPM on data privacy and utility levels. Eventually, *PULP* uses those models to find the suitable protection mechanism and automatically configures it for each user in order to achieve objectives in terms of both privacy and utility. Before describing and evaluation *PULP* framework, some background is given regarding the working scenario, problem statement and motivation.

6.1 Introduction

6.1.1 Description

Geolocation data is increasingly used to improve the quality of online services. Hence, a large number of mobility data is generated and currently used by companies and researchers. Indeed, the processing of mobility data can reveal valuable information that may be used for a broad range of applications, e.g., traffic congestion management, urban development, etc. The work presented in this section focuses on these outbreaking mobility databases, and will not consider real-time online mobility data. However, the processing of location data also comes with threats to the privacy of the recorded users. As a motivation for privacy protection on mobility databases, one can cite the publication of the mobility dataset by Strava in 2018 that revealed the maps of unknown US military bases [88]; or the new regulations that are enforced by governments, such as the European GDPR [79].

To overcome these privacy issues, many efforts in the literature aim to develop protection mechanisms. The protection efforts are not only motivated by cautious companies and researchers but are more and more forced by national and international governments and organizations. The so-called Location Privacy Protection Mechanisms (LPPM) modify the location information of users to improve their privacy level. An overview of the literature on the protections mechanisms has been provided in Chapter 5, Section 5.3. LPPMs need fine tuning of their parameters which may require sophisticated knowledge and experience. The choice of these configurations (e.g. the amount of noise and data granularity) significantly impacts the level of protection of the obfuscated data. The obfuscation should be carried out carefully to make sure that the utility of the protected data is preserved. Indeed, the processing of the mobility data aims to retrieve some high level information (e.g. road usage and means of transportation). Dealing with both privacy and utility simultaneously is not straightforward given the natural trade-off that exists between the two. As privacy enhancing mechanisms alter the original datasets to hide information, the data usability by definition decreases. Using a LPPM may result in various levels of privacy and utility depending on the properties of the user's mobility.

In order to enable the feasibility and practicability of these protection mechanisms for end-users, some configuration mechanisms have been proposed in the literature. An overview is given in Chapter 5, Section 5.5. They do not always explicitly take into account the usability of the protected data and lack the objective-driven formulation that enables a user to define her privacy and utility requirements. Moreover, these works only focus on specific protection mechanisms, hindering their applicability to compare across mechanisms.

6.1.2 Problem Statement

We present a motivating example showing that applying LPPMs in an ad-hoc fashion can result in very different privacy and utility values for individual users. Particularly, we choose four users (selected among all datasets to show diversity) and apply both Geo-I with $\varepsilon 1 = 0.01\,\mathrm{m}^{-1}$ and $\varepsilon 2 = 0.005\,\mathrm{m}^{-1}$, and Promesse with $\varepsilon = 100\,\mathrm{m}$ for all of them. Following definitions of eq. (5.1) and (5.2), we obtain the privacy and utility metrics for all combinations of LPPMs, configurations, and users in Figure 6.1. Let us first analyze those metrics from the perspective of individual users. Both utility and privacy values of user 4 (triangles of all colors) differ when applying Geo-I or Promesse, showing the importance of LPPM choice. Such an observation can also be made for user 1, 2 and 3, with varying degrees of differences. Taking the perspective fixed LPPM, either Geo-I or Promesse, one can see that they offer different levels of privacy protection and utility preservation to different users (symbols of the same color). Figure 6.1 also illustrates that using one LPPM but with various configurations can lead to a totally different privacy protection and service utility. In other words, it is impossible to find a single (configuration) solution that fits all users' privacy and utility objectives. All these observations highlight the complex interplay among privacy/utility metrics, the LPPM and its configuration. Moreover, to ensure the fulfillment of privacy and utility objectives for every user, it is deemed important and necessary to consider the impact of LPPMs and their configurations at the level of individual users.

There is a strong need for a solution that enables choosing between LPPMs and configuring the chosen one in order to meet user-defined objectives in terms of privacy and utility.

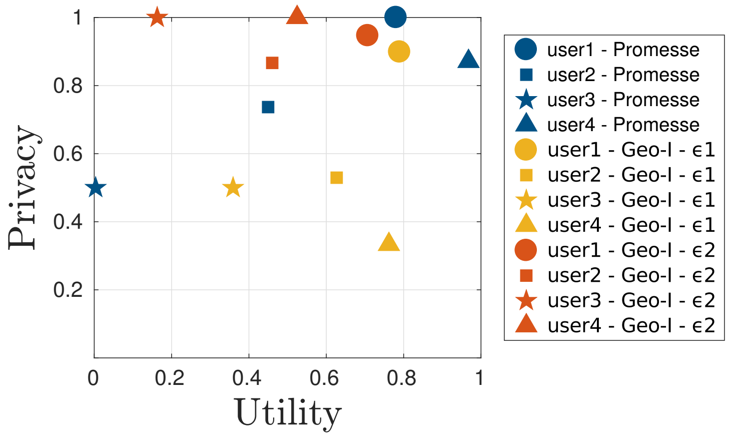


Figure 6.1 – The same LPPM can result in different privacy and utility metrics: examples from 4 users using Promesse with $\varepsilon = 100\,\mathrm{m}$ and Geo-I with two different configurations: $\varepsilon 1 = 0.01\,\mathrm{m}^{-1}$ and $\varepsilon 2 = 0.005\,\mathrm{m}^{-1}$.

6.1.3 Proposed Approach

In this chapter we present *PULP* , standing for Privacy and Utility through LPPM Parametrization. *PULP* is a framework which automatically selects a LPPM from different ones, and determines the best configuration of the LPPM based on each user's objective. The core of *PULP* is user-specific modeling that captures the impact of every considered LPPM on the privacy and utility level of the obfuscated data. A model is built by measuring the privacy and utility levels of obfuscated data after a few profiling runs of applying the LPPM with a set of configuration parameters. Based on each user's objectives, the behavioral model is used to choose and configure a LPPM for each user. Four objective formulations are considered, for various combinations of objectives in terms of privacy and utility: (i) ensure a given ratio between privacy and utility, (ii) guarantee minimal levels of privacy and utility, (iii) keep privacy above a given level while increasing utility as much as possible, and (iv) guarantee a minimal utility level while improving privacy as much as possible.

The specific contributions of this chapter are:

- Accurate, robust and adaptive modeling of LPPMs with different configuration parameters,
- Computing-efficient objective-based recommendation and configuration laws of protection mechanisms.

The rest of this chapter is organized as follows. First *PULP* is described in Section 6.2, first by giving an overview of the framework and then by detailing its three parts. Section 6.3 describe *PULP* 's automatic LPPM configuration laws. Experimental validation and analysis are carried out in Section 6.4. Conclusion and perspectives end the chapter.

6.2 PULP Framework

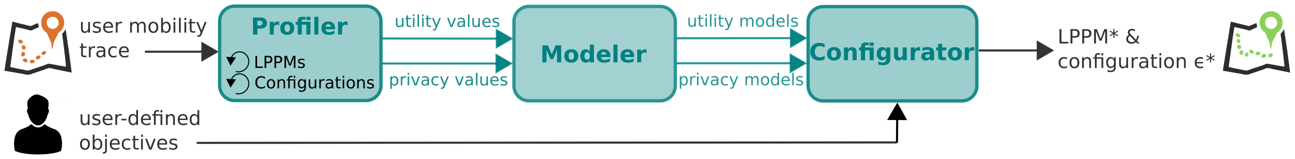


Figure 6.2 – *PULP* framework

This section describes the methodology and design principles of *PULP* , a framework that efficiently chooses and configures LPPMs according to each users' privacy and utility objectives. *PULP* leverages a nonlinear modeling approach and provides several variants of automatic LPPM configuration laws. The key components of the *PULP* framework are illustrated in Figure 6.2.

Although we consider two specific LPPMs (Geo-I and Promesse), the proposed methodology in the following sections is general for any LPPM that considers every user independently and that has a single configuration parameter (or a main one among many others). For some LPPMs, the computation of obfuscated trace is done accordingly the obfuscation of other users, k-anonymity for instance. *PULP* works only for LPPM for which the obfuscation for one user depends only on this user.

The profiler conducts off-line experiments to build users' privacy and utility profiles, with respect to the LPPMs considered and a set of values of their configuration parameter. For each user, the modeler bases on its off-line profile and extrapolates the privacy models and utility models which are non-linear functions in the LPPM configuration parameter (one privacy model and one utility model for each LPPM). According to users' objectives and privacy & utility models, the configurator suggests the suitable LPPM and its configuration.

PULP is effective for processing databases already collected, with protection mechanisms that work at the user level, and one main configuration parameter. Indeed, *PULP* is not suitable for online processes, as it does not include temporal aspects in the decision making.

The rest of this section presents each component of *PULP* . Then, Section 6.3 describes *PULP* 's automatic LPPM configuration laws.

6.2.1 Profiler

The aim of the profiler is to obtain the values of privacy and utility of individual users under a given LPPM and its configuration parameter set of values. The profiler takes as input a user's mobility trace M_i and loops on all LPPMs and on a set of their possible configurations. The outputs are the resulting list of privacy and utility metrics values for all cases. Specifically, the profiler considers two LPPMs, Geo-I with $\varepsilon \in [10^{-4}, 1]$ in $meter^{-1}$ and Promesse with $\varepsilon \in [50, 10^4]$ in $meter$ where range values are chosen according to the LPPMs' conceptors recommendations. The necessary number of configuration values is driven by the fitting accuracy of the modeler. The set of configuration values to run and its size is chosen such that a certain accuracy of the model is reached. The number of values required depends on the accuracy target as well as the functional form of models. Suggestions on how to choose them are given in Section 6.4.2.3.

6.2.2 Modeler

The aim of the modeler is to derive the functional relation between privacy/utility metrics and the configuration parameter of a given LPPM, i.e., $Pr(i,j) = F_{pr}^{i,j}(\varepsilon)$ and $Ut(i,j) = F_{ut}^{i,j}(\varepsilon)$.

To search for the most suitable and general function, we conduct numerous data fitting schemes on our datasets. Figure 6.3 depicts commonly seen dependency between privacy/utility and ε, via an example of applying Geo-I and Promesse on a cabspotting user (continuous lines). Experimental conditions are further detailed in Section 6.4.1. The curves' shape can be explained by the limited ranges of privacy and utility metrics in $[0, 1]$ and the insensitiveness of metrics to extreme values of ε. These observations lead us to choose the arctan function as our base model, instead of general polynomial functions. The general shape of our observations causes us to use $ln(\varepsilon)$ to fit the arctan model of $F_{pr}^{i,j}$ and $F_{ut}^{i,j}$, instead of ε directly.

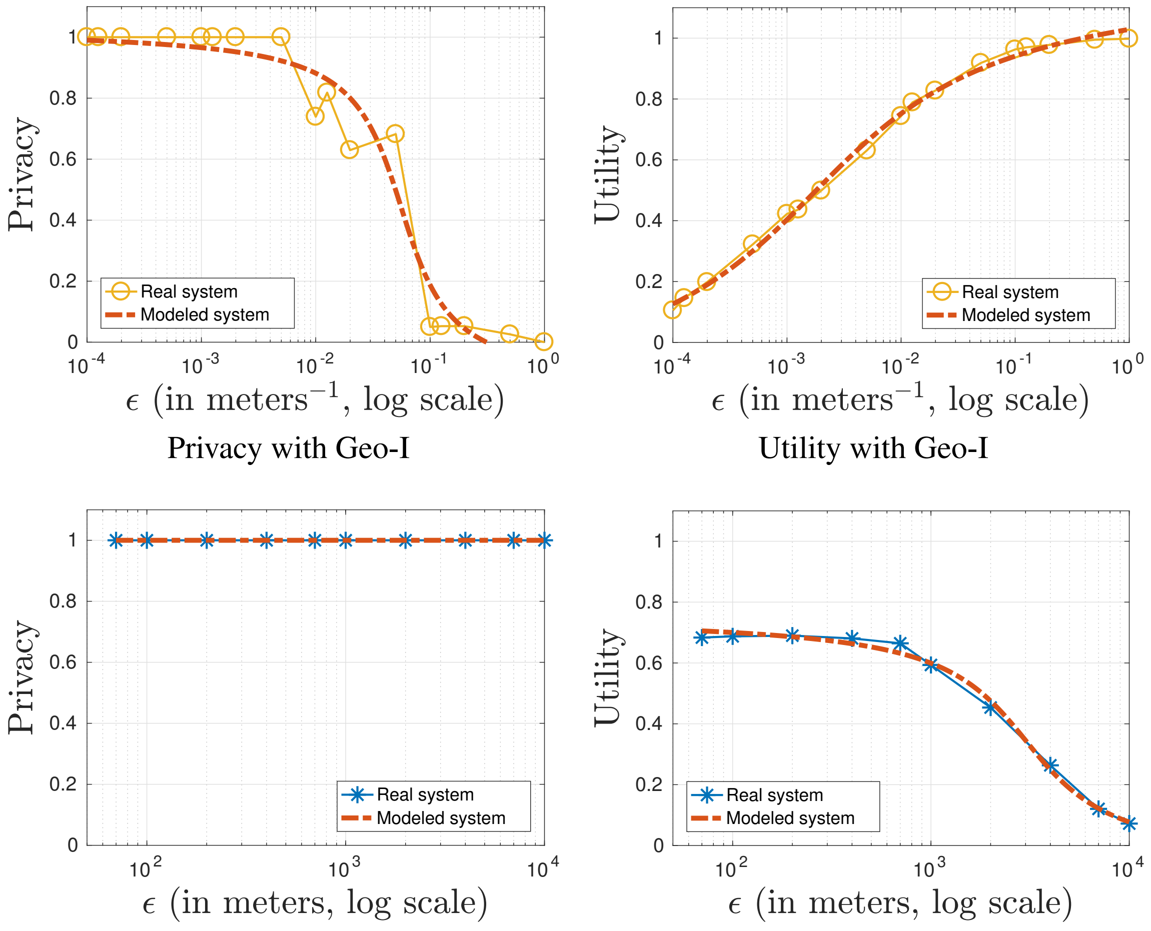
Now, we formally introduce the utility and privacy models with four coefficients each. For sake of simplicity, index i of the user and j of LPPM are not used in the following notation even if there is one privacy model and one utility model per user and per LPPM.

$$F_{pr}(\varepsilon) = a_{pr}.\tan^{-1}\left(b_{pr}(\ln(\varepsilon) - c_{pr})\right) + d_{pr}, \tag{6.1}$$

$$F_{ut}(\varepsilon) = a_{ut}.\tan^{-1}\left(b_{ut}(\ln(\varepsilon) - c_{ut})\right) + d_{ut}. \tag{6.2}$$

Illustrations of model shapes are given in Figure 6.3, in dashed lines.

The physical meaning of model parameters in both F_{pr} and F_{ut} are: a and d represent the two saturation levels, a models their amplitude and d their offset. b characterizes the transition speed between the saturation levels while parameter c corresponds to the ε value that results in the median privacy (or utility) value. Specific values of parameters in F_{ut} and F_{pr} need to be learned from each combination of user i and LPPM j. The proposed models have the computational advantage that there are only four coefficients to be learned.



Privacy with Geo-I

Utility with Geo-I

(c) Privacy with Promesse

(d) Utility with Promesse

Figure 6.3 – Impact of LPPMs configuration on a user's privacy and utility metrics – Real system vs. modeled system (for one cabspotting user used as an example).

6.2.3 Configurator

The aim of the *PULP* configurator is to select and configure a LPPM from the available LPPM set so as to satisfy the user defined objectives. These objectives are related to the user privacy (the proportion of her POIs to be hidden) and to the data utility (the proportion of correct map cells coverage). We consider four types of objective formulation, which combine privacy and utility differently, see Figure 6.4.

- $\mathscr{PU}$-ratio : keeping both privacy and utility as high as possible, with a given ratio between privacy and utility, e.g., privacy is twice as important as utility;
- $\mathscr{P}$-thld : guaranteeing that privacy is above a given threshold, while keeping utility as high as possible;
- $\mathscr{U}$-thld : guaranteeing that utility is above a threshold, while keeping privacy as high as possible;
- $\mathscr{PU}$-thld : keeping both privacy and utility as high as possible, above given thresholds.

		Privacy (Pr)	
		Pr as high as possible	Pr ≥ Pr_{min}
Utility (Ut)	Ut as high as possible	$\mathcal{PU}$-ratio	$\mathcal{P}$-thld
	Ut ≥ Ut_{min}	$\mathcal{U}$-thld	$\mathcal{PU}$-thld

Figure 6.4 – Automatic configuration laws in *PULP*

Using the models that link each LPPM configuration parameter to privacy and utility values, one can reformulate the objectives on privacy and utility as requirements on LPPMs' configuration. Then, in the case where several LPPMs are able to fulfill the objectives, *PULP* selects the most efficient one to achieve the specified objectives. *PULP* 's output is then the recommended $LPPM^*$ and its configuration ε^*.

6.3 Configuration Laws

In the following, we first present *PULP* 's ratio-based configuration law $\mathscr{PU}$-ratio and then describe *PULP* 's threshold-based configuration laws $\mathscr{P}$-thld , $\mathscr{U}$-thld and $\mathscr{PU}$-thld .

6.3.1 *PULP* 's Ratio-Based Configuration Law

The first configuration law proposed by *PULP* is $\mathscr{PU}$-ratio . Its objectives are as follows:

($O1$) Privacy-to-utility ratio is fixed:
$Pr = w_{pr/ut} \cdot Ut$, and

($O2$) Privacy and utility are as high as possible.

For example, when a user specifies $w_{pr/ut} = 0.5$, that means that utility is twice as important for her than privacy. On the contrary, $w_{pr/ut} = 2$ implies that a user thinks preserving the privacy is twice as important as contributing to the LBS accuracy. We now detail the solving procedure, in two steps, to find $LPPM^*$ and its configuration parameter ε^*, based on a relative trade-off $w_{pr/ut}$ provided by the user.

The first step consists of finding, for each $LPPM_j$, its configuration ε_j^* that satisfies objectives $(O1)$ and $(O2)$. To achieve the trade-off ratio of $w_{pr/ut}$ between the privacy and utility of objective $(O1)$, we need to find configuration ε_j^* such that $Pr = w_{pr/ut} \cdot Ut$. Applying the model of Eq. (6.1) and (6.2), we obtain ε_j^* by solving

$$F_{pr}(\varepsilon_j^*) = w_{pr/ut} \cdot F_{ut}(\varepsilon_j^*).$$

Due to the complexity of this equation, we do not derive closed-form solution for ε_j^*. Instead, we numerically solve it as the minimization problem of the absolute difference between F_{ut} and F_{pr}

$$\varepsilon_j^* = \arg\min_{\varepsilon_j} |F_{pr}(\varepsilon_j) - w_{pr/ut} \cdot F_{ut}(\varepsilon_j)|. \tag{6.3}$$

The convergence of the solution is ensured by the convexity of the function to minimize in Eq. (6.3). However, when the resulting configuration parameter value does not fall into its legitimate range (which depends on the LPPM), we then consider $LPPM_j$ as an infeasible LPPM to provide the target trade-off between privacy and utility. Thus, this first step results in the set of values $\{\varepsilon_j^*$ s.t. Eq. (6.3) is minimized for a feasible $LPPM_j\}$ that fulfill objective $(O1)$.

To better understand this step, we schematically illustrate in Figure 6.5 the behavioral models of three LPPMs. Here, the model equations of each $LPPM_j$ are represented, each point of the curve of a $LPPM_j$ represents one of $LPPM_j$'s configuration. In this example, objective $(O1)$ specifies a privacy twice as important as utility, i.e., $w_{pr/ut} = 2$. Thus, the result of the first step of *PULP* ratio-based Configurator $\mathscr{PU}$-ratio is the set of values of $\{\varepsilon_1^*, \varepsilon_2^*, \varepsilon_3^*\}$, the configuration of each feasible LPPM that fulfills objective $(O1)$.

Among the subset of LPPMs that can achieve the target trade-off with a valid configuration parameter, the $\mathscr{PU}$-ratio Configurator then selects the LPPM that maximizes the weighted sum of the

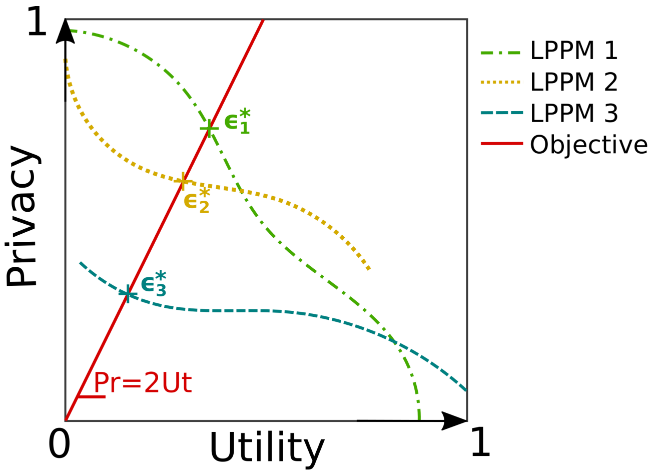


Figure 6.5 – Illustration of $\mathscr{PU}$-ratio configuration law for three schematic LPPMs with $w_{pr/ut} = 2$.

resulting privacy and utility, to keep privacy and utility as high as possible, c.f., objective $(O2)$. Thus, the resulting $LPPM^*$ and its configuration ε^* for a user are

$$LPPM^* = \arg\max_j (F_{pr}(\varepsilon_j^*) + w_{pr/ut} \cdot F_{ut}(\varepsilon_j^*)). \tag{6.4}$$

From the example in Figure 6.5, the LPPM that best achieves objective $(O2)$ is the one crossing the objective $(O1)$ line at the upper point. Here, *PULP* ratio-based Configurator $\mathscr{PU}$-ratio returns $(LPPM_1, \varepsilon_1^*)$.

6.3.2 $\mathscr{P}$-thld Law: Privacy Above a Minimum Threshold

Another possible set of objectives is to guarantee a minimum privacy level while keeping utility as high as possible:

$(O3)$ Privacy is higher or equal to a minimum privacy value: $Pr \geq Pr_{min}$, and

$(O4)$ Utility Ut is as high as possible.

For each LPPM, we define ε_{pr} as the configuration parameter satisfying the equation $F_{pr}(\varepsilon_{pr}) = Pr_{min}$. Using eq. (6.1), we can express ε_{pr} as

$$\varepsilon_{pr} = \exp\left(\frac{1}{b_{pr}} \tan\left(\frac{Pr_{min} - d_{pr}}{a_{pr}}\right) + c_{pr}\right). \tag{6.5}$$

Due to the trade-off between utility and privacy, the higher the utility is, the lower the privacy will be. Then for objective $(O4)$, utility can be increased until the privacy reaches its lower bound specified in objective $(O3)$. Thus for each $LPPM_j$, the configuration ε_j^* that achieves objectives $(O3)$ and $(O4)$ for that LPPM is:

$$\varepsilon_j^* = \varepsilon_{pr} \;\; for \;\; LPPM_j. \tag{6.6}$$

Finally, as the privacy level is achieved for each combination $(LPPM_j, \varepsilon_j^*)$, the overall $LPPM^*$ is the one that maximizes utility:

$$LPPM^* = \arg\max_j (F_{ut}(\varepsilon_j^*)). \tag{6.7}$$

6.3.3 $\mathscr{U}$-thld Law: Utility Above a Minimum Threshold

Similarly, one can set the constraint on a minimal level of utility while keeping privacy as high as possible:

$(O5)$ Utility is not below a given minimum utility threshold $Ut \geq Ut_{min}$,

$(O6)$ With the highest data privacy Pr.

For each LPPM, we define ε_{ut} such that $F_{ut}(\varepsilon_{ut}) = Ut_{min}$:

$$\varepsilon_{ut} = \exp\left(\frac{1}{b_{ut}}\tan\left(\frac{Ut_{min} - d_{ut}}{a_{ut}}\right) + c_{ut}\right). \tag{6.8}$$

Here, objective $(O6)$ is ensured given objective $(O5)$ iff ε converges to the highest value that guarantees $F_{ut}(\varepsilon) \geq Ut_{min}$, due to the trade-off between the two. Thus, the configuration ε_j^* for $LPPM_j$ is

$$\varepsilon_j^* = \varepsilon_{ut} \;\; for \;\; LPPM_j. \tag{6.9}$$

In order to elect the protection mechanism $LPPM^*$, we compare the values of privacy of the obfuscated data and choose the following:

$$LPPM^* = \arg\max_j (F_{pr}(\varepsilon_j^*)). \tag{6.10}$$

6.3.4 $\mathscr{PU}$-thld : Privacy and Utility Above Minimum Thresholds

This configuration law aims at guaranteeing that the level of privacy *and* the level of utility of the obfuscated data are above given thresholds:

$(O7)$ Privacy is higher than or equal to a given minimum threshold: $Pr \geq Pr_{min}$, and

$(O8)$ Utility is higher than or equal to a given minimum threshold: $Ut \geq Ut_{min}$.

The trade-off between privacy and utility described in Section 6.1.2 shows that the utility function $F_{ut}(\varepsilon)$ and privacy function $F_{pr}(\varepsilon)$ have opposite directions of variation, both functions being monotonous. Let us first make the hypothesis that the privacy function is decreasing and utility function increasing (which is the case for Geo-I for example). Then the objective $(O7)$ of a threshold value on privacy $F_{pr}(\varepsilon) \geq Pr_{min}$ can be written as:

$$\varepsilon \leq \varepsilon_{pr}$$

And $(O8)$ $F_{ut}(\varepsilon) \geq Ut_{min}$ as

$$\varepsilon \geq \varepsilon_{ut}.$$

Then the two objectives can be combined in one condition with regard to the value of the configuration parameter:

$$\varepsilon_{ut} \leq \varepsilon \leq \varepsilon_{pr}.$$

Then for users for whom $\varepsilon_{ut} \leq \varepsilon_{pr}$, all the parameters in the range $[\varepsilon_{ut}; \varepsilon_{pr}]$ satisfy the objectives, and *PULP* returns the mean value of the range:

$$\varepsilon^* = \frac{\varepsilon_{pr} + \varepsilon_{ut}}{2}. \tag{6.11}$$

Otherwise if $\varepsilon_{ut} \geq \varepsilon_{pr}$, there is no solution to both objectives $(O7)$ and $(O8)$ for this particular combination of LPPM and user.

Similarly, if the utility function on a LPPM decreases while the privacy function increases (as for Promesse for instance), the objectives can be written as:

$$\varepsilon_{pr} \leq \varepsilon \leq \varepsilon_{ut}$$

and the condition of the existence of a solution is thus $\varepsilon_{ut} \geq \varepsilon_{pr}$. However, in the case where a solution exists, *PULP* returns the same solution as eq. (6.11).

Once the configuration ε_j^* of each *LPPM* j is found, if any, the protection mechanism $LPPM^*$ is selected as follows: the values of privacy and utility are compared by computing their weighted sum, after using each $LPPM_j$ in its previously calculated configuration :

$$LPPM^* = \arg\max_j (Pr_{min} \cdot F_{pr}(\varepsilon_j^*) + Ut_{min} \cdot F_{ut}(\varepsilon_j^*)). \tag{6.12}$$

6.4 *PULP* Evaluation

PULP 's validation is carried out in three steps: first, an analysis of the modeler with an emphasis on the accuracy of the derived models and on their robustness; second, the configurator evaluation that illustrates its effectiveness in choosing a suitable LPPM to achieve different user's objectives; and eventually a comparison with the state of the art. Prior to those core results, the experimental setup is depicted.

6.4.1 Experimental Setup

For the experimental validation of *PULP* , two different machines were used. The profiler was executed on a machine running Ubuntu 14.04 and equipped with 50Gb of RAM and 12 cores clocked at 1.2 GHz. We run the profiler using the 30-days datasets. The modeler and the configurator use Matlab R2016b on a Ubuntu 14.04 equipped with 3.7Gb of RAM and 4 cores clocked at 2.5 GHz.

The number of configuration of each LPPM to be tested by the profiler has been set at first to 17 for Geo-I and 10 for Promesse, corresponding to 4 values per decade of the definition range, uniformly distributed. The modeler searches for each user's model by fitting the experimental data using *fminunc*, and the $\mathscr{PU}$-ratio configuration law uses *min* (both are Matlab functions).

The metrics of privacy and utility used have first been parametrized to correspond to our datasets collected in dense-cities. For measuring privacy, we consider POIs of a maximum diameter of $d =$ 200 m and a minimal stay time of $t =$ 15 min. In order to calculate intersections between sets of POIs, we consider that two POIs matched if their centroids are within $d_{max} =$ 100 m from each other. Google's S2 geometry library [157] is used for cell extraction when computing utility. The size of the cells is highly related to the nature of the LBS. Indeed, a navigation application needs a spatial accuracy at a really fine level while a recommendation system needs accuracy at a neighborhood level. We consider cells at level 15, which corresponds to areas having the size of around 300 meters (city block or neighborhood).

As initial values for the models' parameters of eq. (6.1) and (6.2), we choose the following:

a The metric amplitude. The arctan function varies between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. Our metrics have been defined to vary between 0 and 1. Moreover, we expect Geo-I utility function and Promesse privacy function to be increasing and Geo-I privacy function and Promesse utility function to be decreasing. Consequently, we set $a = \frac{1}{\pi}$ for Geo-I utility and Promesse privacy and $a = -\frac{1}{\pi}$ for Geo-I privacy and Promesse utility.

b The transition speed between the saturations. It should be non-null and positive, the value $b = 1$ was chosen.

c The offset - configuration parameter value. This parameter should be the default value of the configuration parameter defined by the authors of the LPPM: $c = ln(10^{-2})$ for Geo-I and $c = ln(200)$ for Promesse.

d The offset - metric value. As metrics vary between 0 and 1 (or 1 and 0), the offset was set to $d = 0.5$.

When considering a new LPPM, all that is needed for configuring the modeler is a standard value of its parameter (update of c initial value only).

6.4.2 Evaluation of the *PULP* Modeler

This section evaluates the ability of the *PULP* modeler to capture the behavior of privacy and utility metrics when the LPPM configuration varies. We focus particularly on the accuracy of the modeling, its robustness regarding the amount of input data and its adaptability to model any privacy and utility metric.

6.4.2.1 The Modeler's theoretic guarantees

The working hypothesis regarding LPPMs are their configuration by a single parameter, influencing both privacy and utility metrics. Thus, the variation of those metrics will be stable or monotonic (at least on average in the case of stochastic LPPM), varying at most between 0 and 1 (by definition of the metrics), with eventual saturated levels for high and low values of the parameter. The arctan shape of the model allows us to capture this behavior. The accuracy of the modeling is thus given by the relevance of the parameters of the model, output by the *fminunc* function. The tolerance for stopping the iterative search for the best parameters has been set to 10^{-6}, nonetheless with a maximum number of iterations set to 400.

6.4.2.2 The Modeler's performance

As a preliminary analysis, one can take a look at Figure 6.3. Experimental data (continuous lines with circles for Geo-I and stars for Promesse) is compared to their model (dotted lines) for both the utility and privacy metrics. The closer the model curves are to the real data, the better the model fitting is for that user. For our cabs user example of Figure 6.3, the modeler accuracy is good for Geo-I and Promesse utility and Promesse privacy; however for Geo-I privacy the modeler is less accurate but still relevant as it avoids overfitting the experimental data.

In order to ensure that *PULP* modeler is accurate *for every user*, we compute the variance of the fitting error (difference between experimental data and model prediction), which is a relevant indicator for non-linear modeling. Results are shown in Figure 6.6, in the form of a cumulative distribution function where low values of error variance show a high accuracy of the modeling. For all metrics and all LPPMs, the median modeling accuracy is less than $5 \cdot 10^{-2}$, which, when put into the perspective of our metrics varying between 0 and 1, is a really good fit. Promesse privacy is by far the better modeled data, 95% of users have an accuracy of less than 10^{-4}. This can be easily explained as many users have a privacy of 100% no matter the configuration of Promesse, as is the case for the user illustrated in Figure 6.3. From Figure 6.6, one can also notice that the modeler still has a high accuracy when dealing with outliers, as the 99^{th} percentile of the error variance is smaller than 2.10^{-1} for all metrics and all LPPMs.

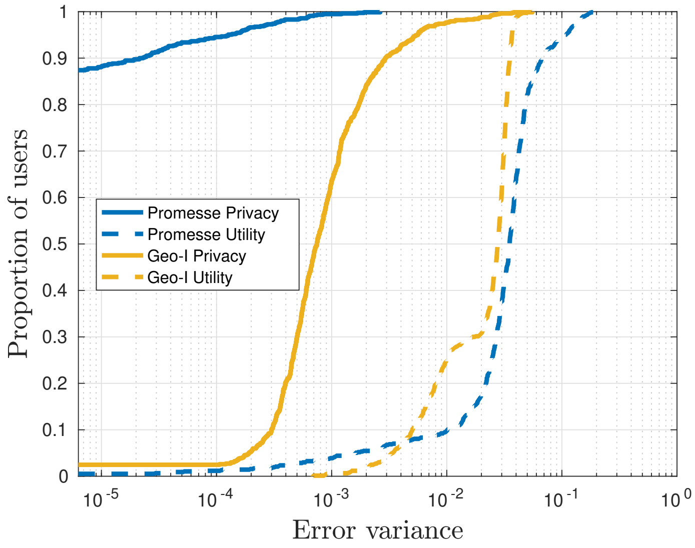


Figure 6.6 – *PULP* modeler accuracy. Cumulative distribution function (cdf) among all users.

6.4.2.3 The Modeler's robustness and adaptability

In the next paragraphs, we comment on the robustness of the modeler regarding both its sensibility to input data and its adaptation to metrics parametrization.

First, we study the impact of the amount of profile data needed for accurate modeling. We vary the number of values of ε taken for the profiling phase, from 1 value per decade up to 4. Results show that the modeling accuracy varies depending on the LPPM. In all cases, the more data are used, the better the modeling is. However the improvement is negligible when modeling Geo-I, which leads us to recommend using only a few experiments for the profiling phase in order to limit computing. When modeling Promesse, only 4 values per decade enable us to properly capture the behavior of users.

The metrics described in Chapter 5 are parametrized. The privacy metric depends on the diameter and duration of a POI as well as on the maximum distance between two POIs to consider they match. As for the utility metric, one can vary the size of the cells that discretize the map. When varying these parameters, the metrics reflect several notions of privacy and utility. To ensure the performance of the modeler even with these other notions of privacy and utility, we varied the four metrics' parameters and again computed the modeler accuracy. In a general way, the modeler is able to keep a good accuracy: around 10^{-3} for the median value and 10^{-2} for the 99^{th} percentile (excluding extreme cases).

We now detail the impact of each metric parameter on modeling performance. We varied the duration of a POI between 5 and 120 minutes. For Promesse, the longer the POI, the better the modeling. For Geo-I however, medium duration POIs (around 15 to 30 minutes) are well modeled while extreme ones have error variance close to 10^{-1}. When looking at the impact of the POI diameter (from 100m to 1000m) on the modeling accuracy, we found none on Geo-I (all metrics are well modeled), while for Promesse the smaller the POI is, the better the modeling is. As for the maximum distance between two matched POIs (ranging from 25% of the POI diameter to 250%), we obtained similar results: no impact for Geo-I modeling and the smaller the distance is, the better the modeling is for Promesse. When looking at the size of the cells in the utility metric computation, we found that the larger the cells are, the better Promesse behavior on users' traces is captured. However, Geo-I modeling is more accurate for large or small cells, and a sightly worse for medium-size cells.

6.4.3 Evaluation of the *PULP* Configurator

We now analyze the configurator's ability to fulfill users' objectives. To do so, we ran the four versions of *PULP* configuration laws, each of them with several objectives.

PULP outputs for a set of users (selected to show diversity) and a few objectives can be found in Table 6.1. Results show that for each user, a LPPM is recommended with a configuration value, except when no LPPM can fulfill the objectives, which is the case of user 3 with a $\mathscr{PU}$-thld objective. As a preliminary analysis, we can see that users have different recommendations even when having the same objectives. Moreover, a single user gets various recommendations depending on her objectives.

The next sections (and corresponding Figures 6.7 to 6.10) detail the results for each law, by looking at four indicators: the LPPM selected for each user (a) and its associated configuration parameter (b and c), the privacy and utility of users' data when obfuscating with *PULP* recommendation (d and e), and the corresponding trade-off between privacy and utility (f).

Config. Law Objectives	$\mathscr{PU}$-ratio $w_{pr/ut} = 2$		$\mathscr{PU}$-thld $Pr_{min} = 0.6$ $Ut_{min} = 0.7$		$\mathscr{P}$-thld $Pr_{min} = 0.7$		$\mathscr{U}$-thld $Ut_{min} = 0.5$	
PULP Output	LPPM*	ε^*	LPPM*	ε^*	LPPM*	ε^*	LPPM*	ε^*
User 1	Promesse	694	Geo-I	0.014	Promesse	69	Geo-I	0.004
User 2	Geo-I	0.001	Promesse	244	Promesse	197	Geo-I	0.0034
User 3	Promesse	173	NaN	NaN	Geo-I	0.0097	Geo-I	0.0074

Table 6.1 – *PULP* output for selected users

6.4.3.1 Evaluation of the $\mathscr{PU}$-ratio Configuration Law

In this variant of the configurator, the objective is to achieve a given trade-off between privacy and utility. For its evaluation, we run *PULP* on all users with various objective ratios $w_{pr/ut}$ ranging from 0.5 (utility is twice as important as privacy) to 3 (privacy is three times more important than utility). Results are shown in Figure 6.7. We computed the actual privacy to utility ratio after applying the LPPM selected with its right configuration. Results illustrated in Figure 6.7 (f) show that at least 95% of the users have a resulting ratio in a range of +/- 1% of user specified values.

From Figure 6.7 (a) we can see that all users ended with a recommended LPPM. For a given objective, the LPPM chosen by *PULP* varies depending on the user, and the distribution changes according to the objective, meaning that the adequate LPPM of a single user may vary depending on the objective. There is no a priori relation between the objective $w_{pr/ut}$ and the distribution of selected LPPM.

When analyzing the *PULP* choice of LPPM configuration parameters from Figure 6.7 (b) and (c), we make two observations: (i) users need different configurations to fulfill the same objective and (ii) different objectives lead to various configurations' distribution. When looking at users for whom Geo-I is chosen, the higher the objective ratio is, the lower the recommended ε value is. Whereas for users with Promesse recommended, the higher the objective ratio is, the larger the suitable ε value is and the higher diversity there is in the recommended value. For instance with $w_{pr/ut} = 2$, users have ε from 100m to 2km.

When using the appropriate LPPM configured in a suitable way, users can maintain privacy and utility levels that jointly respect the objective trade-off. In terms of absolute values, when looking at the utility, one can notice that most users have the same level of utility (Figure 6.7 (e)). The lower the objective ratio is (i.e. the more utility matters), the higher the utility is and the more diversity there is in the utility values. For privacy, the same trend is observed: most users have the same privacy but the lower the objective ratio is, the more diversity there is in the privacy values (see Figure 6.7 (d)).

With an objective expressed as a trade-off between privacy and utility, PULP finds a suitable LPPM for all users and guarantees a high utility and privacy to almost all of them.

6.4.3.2 Evaluation of the $\mathscr{PU}$-thld Configuration Law

In this section we evaluate the *PULP* $\mathscr{PU}$-thld configuration law, aiming to achieve privacy and utility at levels higher than some minimal thresholds. The evaluation, reported in Figure 6.8, has been carried out with five couples of objectives.

First, it is important to notice that some users do not have any LPPM recommended, as can be seen in Figure 6.8 (a). High utility constraints seem to hamper the feasibility of recommending suitable LPPMs. Recommendations range from less than 10% of all users ($Pr_{min} = 0.5$,$Ut_{min} = 0.9$) to more than 95% ($Pr_{min} = 0.3$,$Ut_{min} = 0.5$). Most of the recommendations are Geo-I. The higher the utility constraint, the more Geo-I is recommended. When looking at values of the LPPM configuration parameter, for Geo-I the general trend is that ε is lower when privacy constraint in high, except in the extreme case where $Ut_{min} = 0.9$. For Promesse, the higher the utility constraint, the smaller ε is.

The privacy criteria is always satisfied, and almost no user gets the limit privacy Pr_{min}, see Figure 6.8 (d). However, when the utility constraint is high, most users tend to have a privacy close to its bound. All users have their utility criteria fulfilled (see Figure 6.8 (e)). The utility of most users is really high: 80% of them have a utility above 0.8 (0.6 for $Pr_{min} = 0.9$,$Ut_{min} = 0.4$). As for the privacy to utility ratio, within a set of objectives most users (70-90%) have the same privacy to utility ratio (Figure 6.8 (f)).

PULP is not able to find a suitable LPPM for all users using this objective formulation; however, when it can, the privacy and utility are, most of the time, way above the minimum values required.

6.4.3.3 Evaluation of the $\mathscr{P}$-thld Configuration Law

Here the objective is to guarantee that the privacy is above a given level. Results are given in Figure 6.9. *PULP* can recommend a suitable LPPM and fulfill all users' objectives considered here. Around 10 to 20% of users can even achieve higher privacy levels than the requested threshold (even 80% for the objective $Pr_{min} = 0.9$), see Figure 6.9 (d). These proportions correspond to users to which Promesse is recommended, see correspondence with Figure 6.9 (a).

For those Promesse users, the utility is quite low but for the other 80% of users, utility is more than 0.5 (Figure 6.9 (e)). Moreover, the higher the privacy limit, the lower the utility. Looking at the privacy to utility ratio (Figure 6.9 (f)), for 80% of users (those with Geo-I recommended), the lower the privacy limit, the higher the ratio (allowing more utility to the data) and all users have the same privacy to utility ratio. However, for 20% of them (Promesse users) the ratio is always the same no matter what the objective.

As for the parameter values, for users with Geo-I recommended, the higher the objective is, the smaller ε is. Users with Promesse recommended always seem to have the same ε recommended no matter what the value of the objective, except for $Pr_{min} = 0.9$ where ε is at its upper bound for most users.

When PULP guarantees a minimal privacy level, but two distinctive types of users are observed. Some have are recommended Geo-I, a limited privacy and a high utility; while the others use Promesse with a high privacy but a low utility.

6.4.3.4 Evaluation of the $\mathscr{U}$-thld Configuration Law

The results of the configuration law guaranteeing a minimum level of utility are illustrated in Figure 6.10. They show similar patterns than those of the $\mathscr{P}$-thld configuration law.

All users had a LPPM recommended, and the general trend is that the lower the utility limit, the more Promesse is recommended. Hence, Promesse tends to protect better than Geo-I but results in lower utility, see Figure 6.10 (a).

All users have exactly the minimum utility they wanted, no matter the value of the limit (see Figure 6.10 (e)). The lower the utility limit, the higher the privacy. Most users (almost 70%) have good privacy levels, i.e. more than 0.7, except with $Ut_{min} = 0.9$ (see Figure 6.10 (d)). Therefore, the lower the utility limit, the higher the ratio, allowing more privacy preservation in the data. Within a set of objectives, most users have the same privacy to utility ratio (Figure 6.10 (f)).

For users with Geo-I recommended, the higher the objective is, the higher ε is. However, for users with Promesse recommended, the higher the limit is, the smaller ε is.

In this objective formulation, PULP always sets utility to its minimum value, ensuring a good privacy to users especially for those with Promesse recommended.

6.4.4 Comparison with State of the Art

As *PULP* works with few profiling experiments, its execution time is significantly shorter compared to the state of the art. Indeed, all LPPMs configuration mechanisms that we are aware of use greedy processes that need to run many experiments in order to converge to a suitable configuration (if ever they converge). We compare our framework *PULP* to the closest work from the state of the art, the configurator ALP from [151]. ALP is a framework that iteratively looks for a LPPM configuration that satisfies high level objectives such as *maximizing privacy and utility*. We consider only one LPPM in *PULP* , Geo-I, and set our objective to $w_{pr/ut} = 1$ to be as close as possible to the ALP working conditions. The execution time of *PULP* in these conditions is of the order of the minute for the Geolife dataset while ALP requires around ten hours to converge, as illustrated in Figure 6.11. This makes a difference of 3 orders of magnitude. The execution time of *PULP* is almost all spent on the profiling phase. Indeed the modeler and configurator execution times are of a few milliseconds. This enables a user to change her objective and easily find again the new adequate LPPM and its configuration.

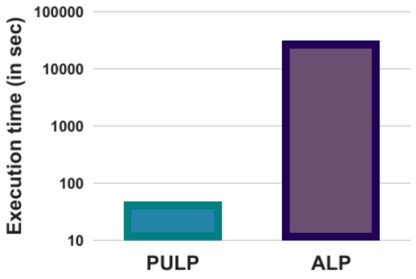


Figure 6.11 – Execution time comparison, Geolife dataset

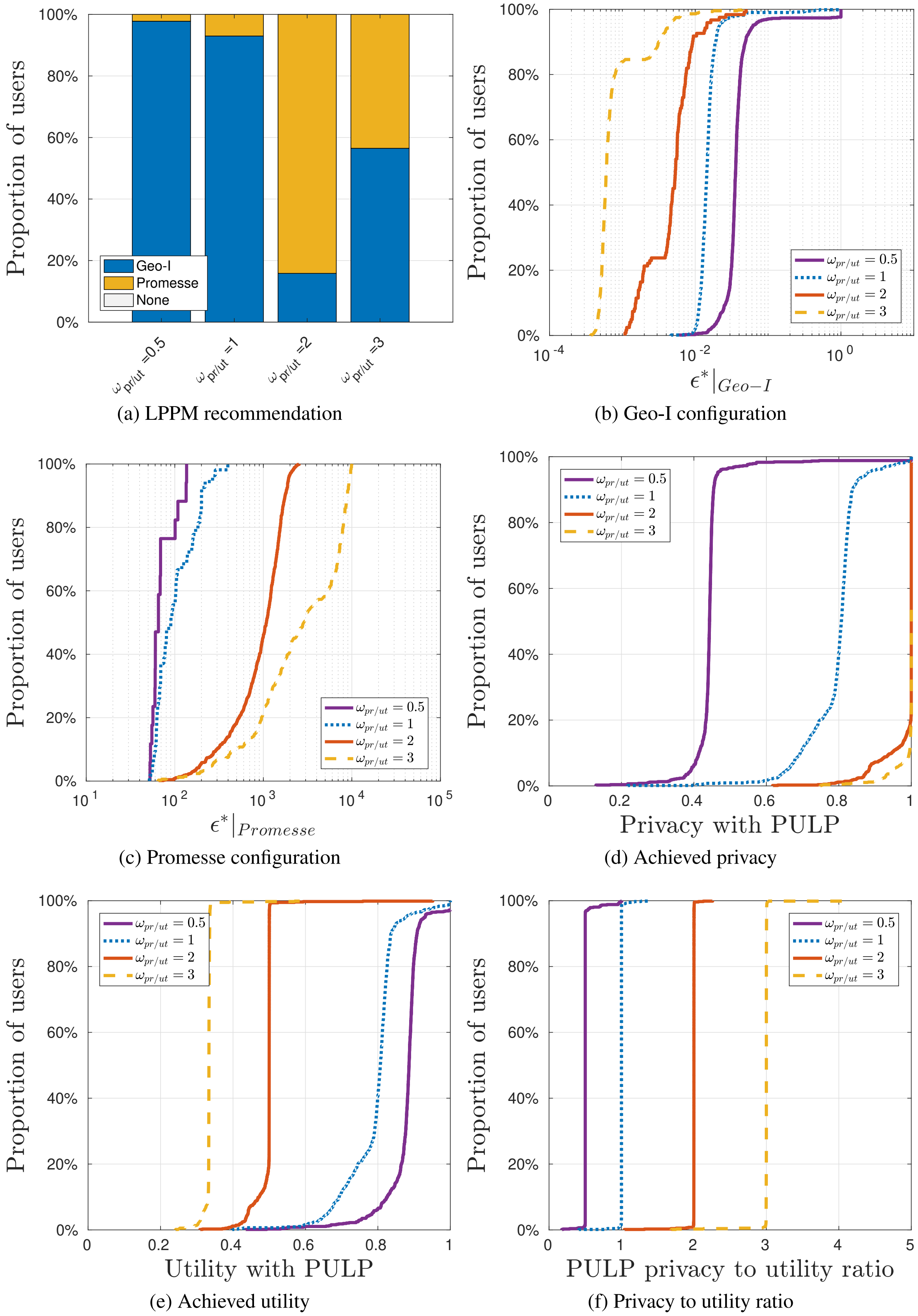
ALP only considers the configuration challenge and does not allow a choice between several LPPM. Thus, to compare the accuracy of *PULP* with regards to the state of the art, the focus will be on the users' privacy and utility preservation after using the frameworks. While the objective given to ALP is to maximize both utility and privacy (no preference is given to one or the other), the ratio between the two after running ALP is almost always greater than 1, meaning that more importance is given to privacy than to utility [151]. With *PULP* , the ratio is almost always 1, see Figure 6.7 (f). Moreover, with *PULP* , 80% of the users have a utility and privacy higher than 0.7, while with ALP 90% of the users have a privacy higher than 0.8, while 80% of them have their utility only between 0.4 and 0.7 [151]. The low utility with ALP comes from a small configuration parameter (less that 5.10^{-2} for 70% of the users). Hence, the objectives are more evenly treated when using *PULP* , and enable a better utility to be achieved.

6.5 Conclusion

This chapter presents *PULP* , a framework that ensures users' objectives regarding privacy and utility for mobility databases by automatically choosing and configuring LPPMs. Our notion of privacy relies on the hiding of users' points of interest, and the utility of the services is measured by looking at the spatial proximity of obfuscated data to the original ones. *PULP* realizes an in-depth analysis of the considered LPPMs applied at a user scale in order to capture the formal relationship between the configuration parameters of the LPPMs and both privacy and utility metrics. Then *PULP* leverages the models derived to identify the adequate LPPM and its configuration that enables the objectives to be achieved. The considered objectives aim at maximizing privacy and utility with various constraints regarding minimal levels or the ratio between the two metrics.

We illustrated the ability of our system *PULP* to efficiently protect a user while keeping utility of her service using two LPPMs from the state of the art: Geo-I and Promesse. Evaluation has been done for several objectives and using data from four real mobility datasets containing 770 users in total. *PULP* can accurately model the behavior of LPPMs on individual users and thus successfully achieve privacy and utility objectives at the same time in an automated way. Moreover, when comparing with the state of the art, we proved *PULP* to be 3 orders of magnitude faster (minutes versus hours) and more robust to achieve user specified privacy and utility objectives.

Regarding the future directions of research, let us put them in perspective of the assumptions taken. The users' mobility data is supposed to have been recorded beforehand, but the on-line obfuscation scenario could be considered too and it is the problem addressed in Chapter 7. The privacy metric used is based on POI protection, and the processing oriented utility measure is based on timeless spacial accuracy. More metrics could be added, such as predictability of users' movements for privacy and time distortion of traces for utility. This improvement would be achievable with *PULP* current working principle; however it would add more constraints on the configurator that would challenge even more the existence of a solution. In order to deal with this latter issue, we advocate to investigate the use of other LPPMs that have more than one configuration parameter, such as [1]. On top of enlarging the capabilities of *PULP* , it would transform the system in a multi-input (several configuration signals) multi-output (various privacy and utility metrics) problem, which is feasible since backed-up with a strong theoretical background in control theory.



(a) LPPM recommendation

(b) Geo-I configuration

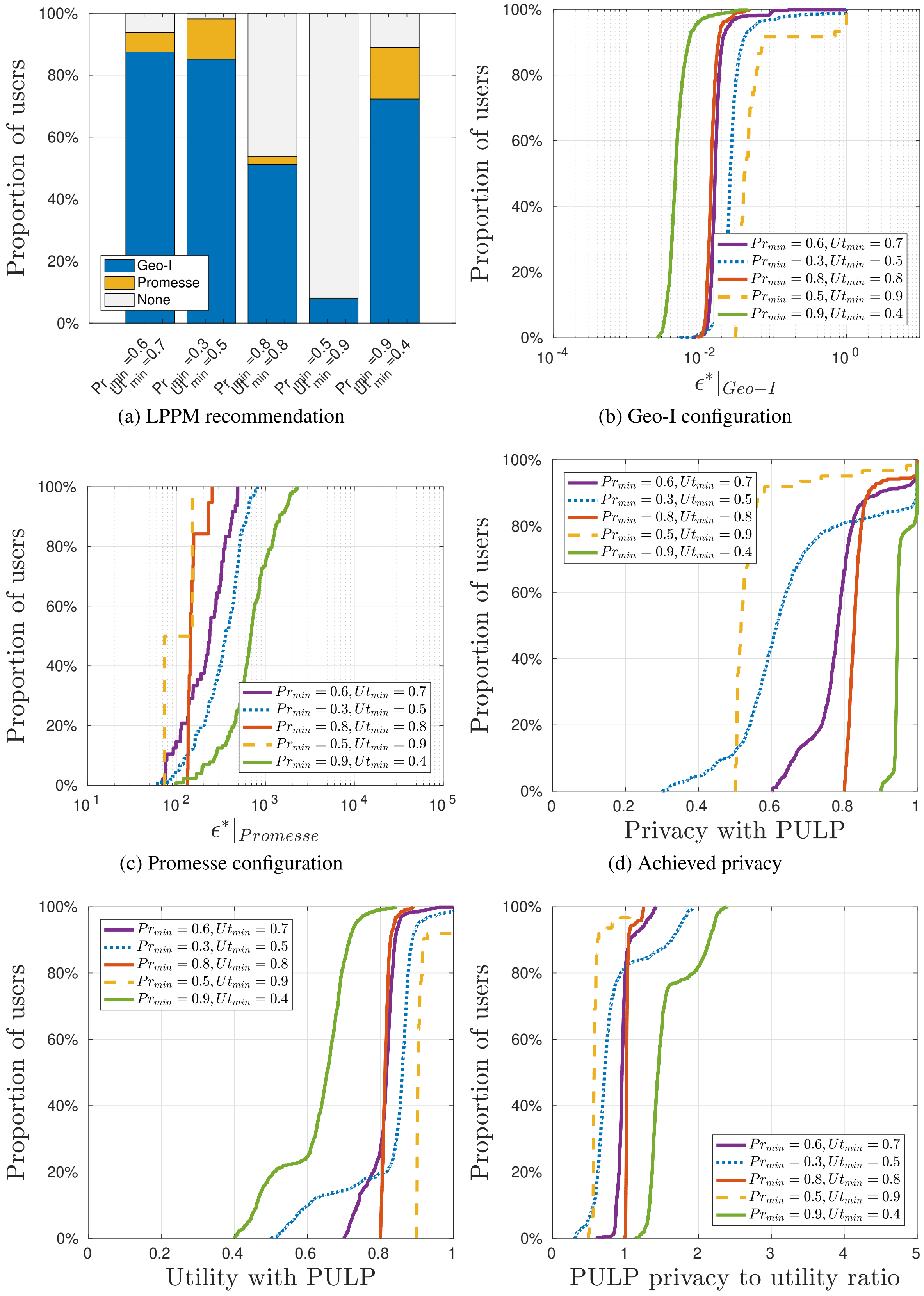
(c) Promesse configuration

(d) Achieved privacy

(e) Achieved utility

(f) Privacy to utility ratio

Figure 6.7 – $\mathscr{PU}$-ratio configuration law evaluation. (a) Recommended LPPM and its configuration (b) for Geo-I, (c) for Promesse. Achieved (d) level of privacy and (e) utility when users are protected according to *PULP* recommendations, and the corresponding (f) privacy to utility ratio. Four objective ratios $w_{pr/ut}$ are illustrated: 0.5, 1, 2 and 3.



(a) LPPM recommendation

(b) Geo-I configuration

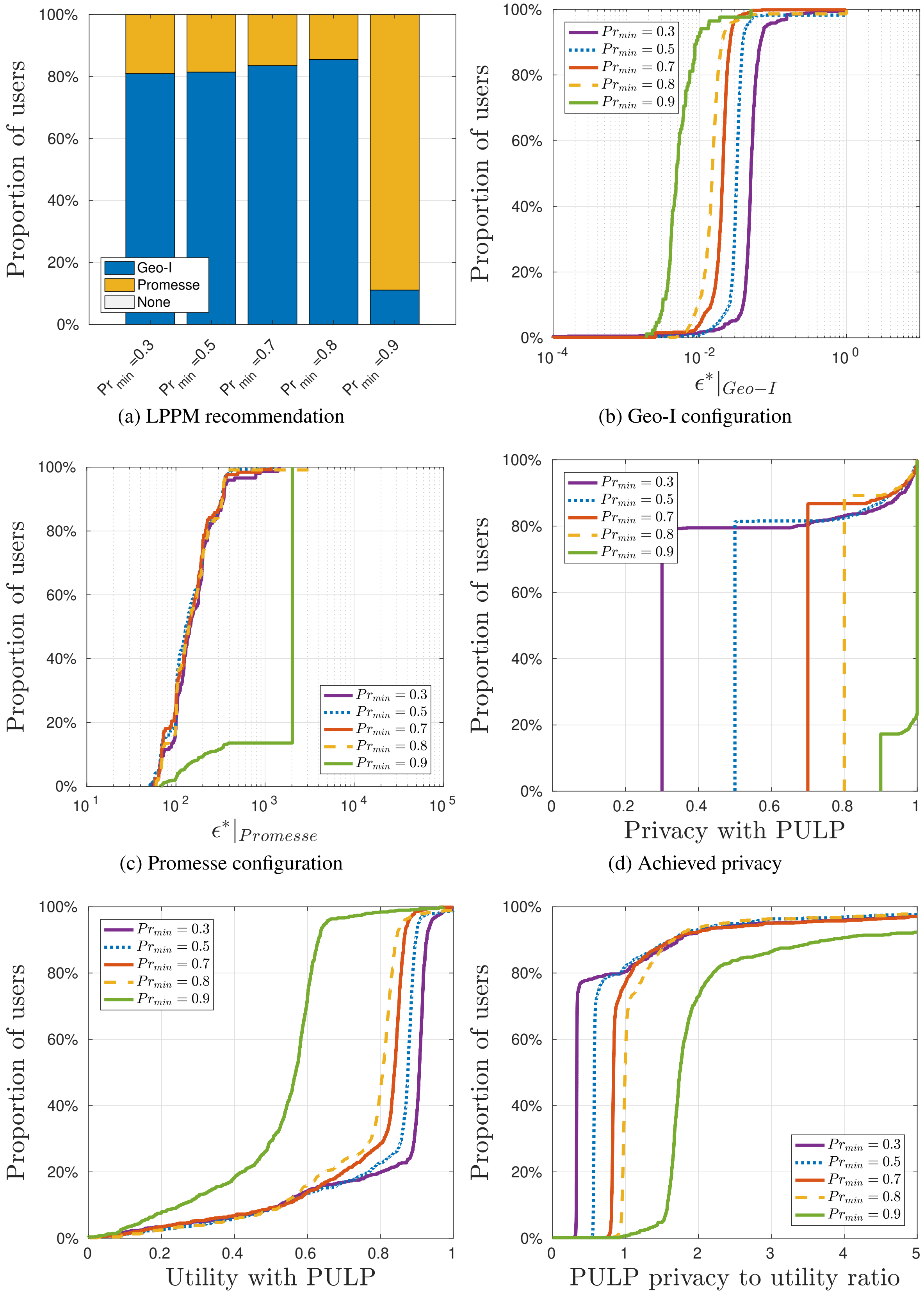
(c) Promesse configuration

(d) Achieved privacy

(e) Achieved utility

(f) Privacy to utility ratio

Figure 6.8 – $\mathscr{PU}$-thld configuration law evaluation. (a) Recommended LPPM and its configuration (b) for Geo-I, (c) for Promesse. Achieved (d) level of privacy and (e) utility when users are protected according to *PULP* recommendations, and the corresponding (f) privacy to utility ratio. Five objective couples of constraints on privacy and utility are illustrated.



(a) LPPM recommendation

(b) Geo-I configuration

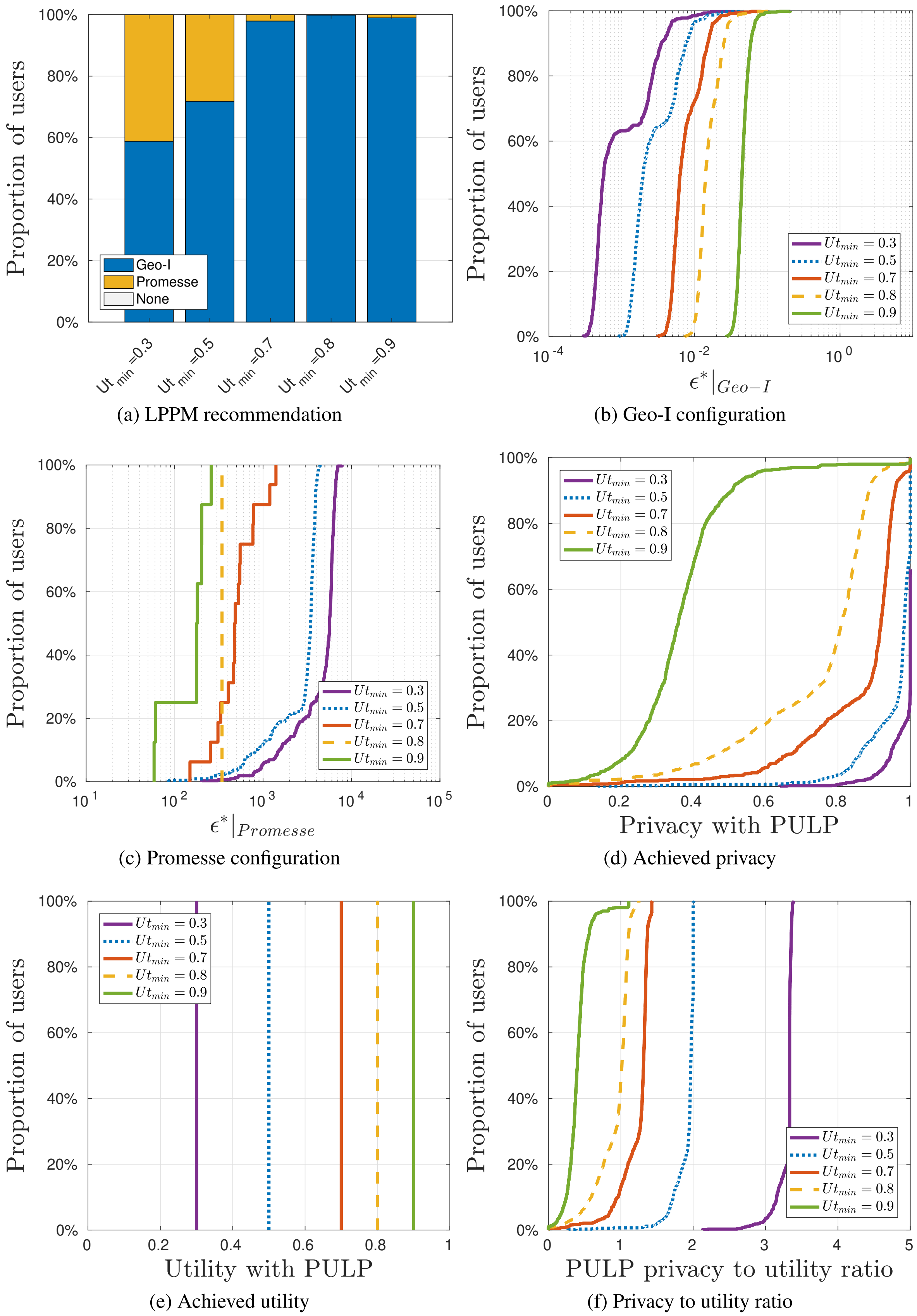
(c) Promesse configuration

(d) Achieved privacy

(e) Achieved utility

(f) Privacy to utility ratio

Figure 6.9 – $\mathscr{P}$-thld configuration law evaluation. (a) Recommended LPPM and its configuration (b) for Geo-I, (c) for Promesse. Achieved (d) level of privacy and (e) utility when users are protected according to *PULP* recommendations, and the corresponding (f) privacy to utility ratio. Five objective constraints on privacy Pr_{min} are illustrated: 0.3, 0.5, 0.7, 0.8, 0.9.



(a) LPPM recommendation

(b) Geo-I configuration

(c) Promesse configuration

(d) Achieved privacy

(e) Achieved utility

(f) Privacy to utility ratio

Figure 6.10 – $\mathscr{U}$-thld configuration law evaluation. (a) Recommended LPPM and its configuration (b) for Geo-I, (c) for Promesse. Achieved (d) level of privacy and (e) utility when users are protected according to *PULP* recommendations, and the corresponding (f) privacy to utility ratio. Five objective constraints on utility Ut_{min} are illustrated: 0.3, 0.5, 0.7, 0.8, 0.9.

Chapter 7

dynULP: dynamic control of Utility and Location Privacy

In this chapter, we are interested in the online control of privacy. A mobile device user can constantly send her current location and benefit from geo-localized services. She indeed wants to protect her privacy while still enjoying a reasonable quality of service. Compared to PULP's approach presented in Chapter 6, dynULP is at the user level, locally on the mobile device, thus the service utility considered here is the quality of the LBS answers. We advocate to use a control-theoretic approach for two main reasons, being (i) the end-user do not want to bother with the understanding and configuration of a LPPM and (ii) the problem is by definition dynamic and needs fine adaptation through time since the user is mobile in a dynamic environment.

A simple controller is presented, that ensures privacy according to the user's requirements, while being robust to her movement. To help the initial tuning of the controller, a control-oriented model of the protection mechanism is derived. For each modeling and control steps, the methods used are illustrated and validated on synthetic mobility data, which enable the full understanding of the approaches. Afterwards, an evaluation using real users mobility records is also provided. dynULP approach shows good results both for achieving a desired privacy level and for keeping it despite disturbances. The quality of the resulting service is also maintained at a reasonable level.

The outlines of this Chapter are as follows. After a deeper introduction and motivation of dynULP scenario (Section 7.1), the formulation of the problem in control words is given, with a emphasis on the output metric sensing: real-time privacy (Section 7.2). Then the two-step modeling approach (static then dynamic) is carried out (Section 7.3). Then the control strategy is presented (Section 7.4). Eventually, an evaluation of the modeling and controller consisting in a proof of concept of the control approach is given (Section 7.5) before a conclusion that end this chapter (Section 7.6).

7.1 Introduction

7.1.1 Context, Hypothesis and Motivation

The democratization of mobile devices has fostered the development of services using the users' location data to provide or improve a service. Everyday examples of Location Based Services (LBS) are navigation applications, recommendation systems or fitness tracking apps. Some of those services are one-shot, they need one location point to provide their service e.g. a weather app. Some others need dynamic records, for instance navigation or gaming apps. We advocate in this work to consider them all as dynamic, for two main reasons. First, people using "one-shot location" application do not necessary turn on and off instantaneously the GPS option on their phone, thus the records are still sent to the service constantly. Second, the dynamic problem is more challenging than the static one as it has more complexity, the attacker has access to more information. Thus, without loss of generality, we can focus here on the user-level dynamic, online scenario as illustration on Figure 7.1.

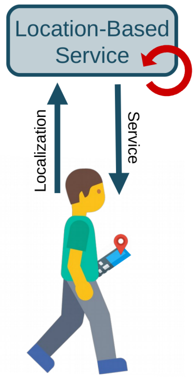


Figure 7.1 – Working scenario: location data are sent continuously and the service is received constantly

LBSs provide users with always more personalized and convenient services but at the cost of personal data publishing. Service providers, or any third party attackers, take advantage of these data to derive always more information about users. In order to provide ways to protect users' privacy, Location Privacy Protection Mechanisms (LPPMs) have been developed. In the dynamic scenario considered here, only the LPPMs able to work in a real-time fashion, i.e. they only modify the current position and are causal, are considered.

However, nowadays LPPMs are facing some limitations. On one hand, the notion of privacy is often addressed with high level, theoretical principles that might lack of practical meaning for average user of mobile devices. It is thus challenging to assess the impact of an LPPM on one's privacy for a non expert user. On the other hand, the parametrization of LPPMs makes them tricky to use as the user is not always able to predict what will be the impact of a given parametrization on her privacy. Moreover, location data are highly dynamic, meaning that the user may be in a sensitive place at a given time while she can move to a place of none interest for her few minutes later. Similarly, if a user starts to obfuscate her data at a given point, it may take some time before she is actually protected, due to memory of the potential attacker. Thus the measures and reactions must be real time processes.

To sum up, LPPMs are usually statically tuned, by experts. The problem needs to be shifted from choosing the configuration parameter of a LPPM to automatically setting a desired privacy level, all this in a robust and automated fashion.

7.1.2 Proposed Approach

This thesis presents a control-theoretic approach to solve these challenges. The challenge of how to automatically leverage a Protection Mechanism action *to meet users objectives*, can be considered as a reference tracking problem. When it comes to automatically leverage a Protection Mechanism action *to be efficient whatever the mobility pattern the user is having*, a regulation approach that aims at rejecting the so called disturbances is used. Prior to those control considerations, some works have to be done on the problem formulation, particularly on having continuous measures of performance indicators (privacy and utility). Then identification is performed to compute a model of the system and eventually the controller itself can be designed. Those three steps form the next sections.

7.2 Control Problem Formulation

7.2.1 Overview

The system under study is a Location Privacy Protection Mechanism (LPPM), that obfuscates a user location record before sending it to the Location Based Service (LBS). This section details and formulates in a control-theoretic way the following concepts: the protection mechanism (the plant), the LPPM parameter (controllable input), the raw location data (latitude and longitude over time, seen as an uncontrollable input) and the privacy and utility of the resulting obfuscated data (outputs). These concepts are schematically represented in Figure 7.2 in a closed loop box-diagram form.

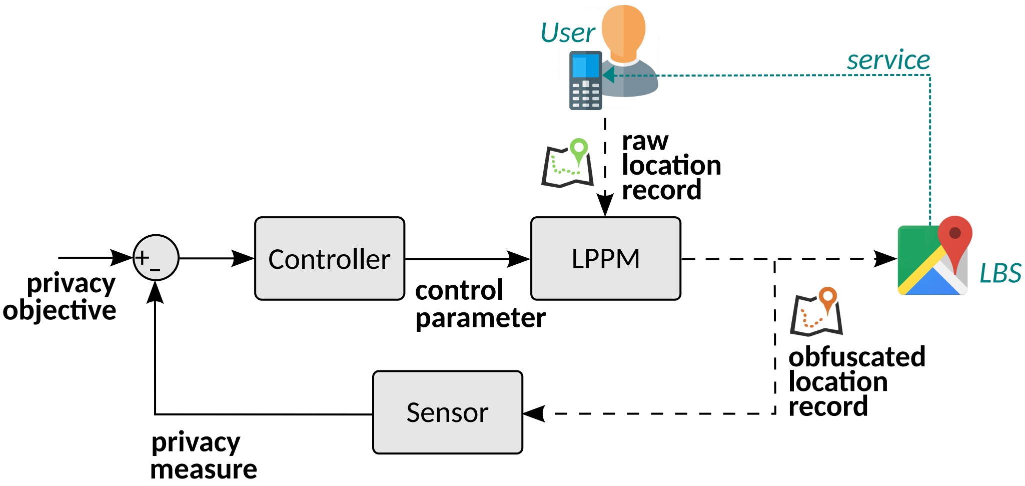


Figure 7.2 – Classic control schema applied for privacy protection of a location based service user.

7.2.2 Formalization

7.2.2.1 Process

Location Privacy Protection Mechanisms (LPPMs) are algorithms that aim at increasing privacy of location data. They are parametrized algorithms that take as input raw mobility data and output the obfuscated ones. For a control approach, only LPPMs that can work in real-time and enable on-line tuning of their parameters to adapt privacy level of data can be considered. Such example of LPPMs are perturbation based mechanisms such as cloaking algorithms [80, 131, 136, 146, 183].

A schematic example of the LPPM in its scenario is given in Figure 7.2. The user of a mobile device send her location to a Location-Based Service through a LPPM, that performs the obfuscation of the current location record before sending it to the LBS. The service is thus returned to the user, and all this process is done in an online fashion.

Geo-Indistinguishability [18] (Geo-I for short), the chosen LPPM for this approach, is a well known LPPM from the state of the art that can work on-line. As a reminder, its simple yet efficient principle is to add spatial noise to the location data. Each new obfuscated location is computed using a Laplacian distribution centered in the raw location record with parameterizable variance. Geo-I's parameter is noted ε and is inversely proportional to the amount of noise added: the lower ε is, the more noise is added and the better the location is obfuscated.

Geo-I has been initially developed for an offline usage, meaning that it aimed at obfuscating a location dataset in order to mathematically guarantee its differential privacy at a given level ε. Given the sparsity of the literature to provide consistent real-time LPPMs, we advocate to use Geo-I, or more broadly the principle of adding spatial noise to a location record, knowing that the differential privacy guarantee do not apply the same way for the online scenario.

To sum up, the methodology presented in this chapter can apply for LPPMs satisfying the following requirements:

- being an on-line process, every location is individually obfuscated in real time,
- being tunable by a single parameter, such as the ε of Geo-I,
- being user centric: the obfuscation should not depend on other people's location or other properties such as the density of the area.

The example of Geo-I will be used as illustration for the rest of this chapter. However, the core of this work is to take a level of abstraction higher than the LPPM and consider that only the privacy and utility achievements matter, the LPPM in itself being only a tool. In that sense, if the objectives are met, there is no need to explore the use of other protection mechanisms.

7.2.2.2 Input signals

The control-oriented presentation of Geo-I's two inputs (the location record of the user and its own parameter ε) is provided in the next paragraphs.

Control Variable. Geo-I's parameter ε can be changed at each iteration and impacts the POI-oriented privacy and the utility of the obfuscated data. Indeed, as can been seen in Figure 5.3, the lower ε is, the more noise is added to the data and the more difficult it becomes to extract the real points of interest of the user - or even to realize that the user has stopped. ε usually takes its values in the range $10^{-4}\,\mathrm{m}^{-1}$ to $1\,\mathrm{m}^{-1}$. A low ε (i.e. high obfuscation) also alter utility of the data. When the location record sent to the service is far from the real one, the quality of service can be significantly reduced. Then, ε is chosen as the control variable.

Disturbance. Another factor impacting the level of privacy, even when using a LPPM, is the properties of the raw mobility data to be obfuscated. For instance if a user is in a train, the continuous move naturally prevents from extracting any POI. Whereas if she is home, the location data require obfuscation to protect the extractable POIs. This highlights the dependency of the privacy on the raw mobility data itself. Indeed some cases are not as trivial regarding whether it requires obfuscation or not (and how much), for instance when driving a car and stopping at traffic lights or to grab groceries at self-driving shops. In the following, raw data will be considered as an uncontrollable but measurable input.

For a quick illustration of the impact of those two parameters on the system outputs, one can refer to Section 7.2.3. However, prior to this, the output metrics need to be defined.

7.2.2.3 Output signals

In the context of location privacy, there are two goals to be achieved: the protection of a user's privacy and the guarantee of the usability of the revealed data. As a consequence, two performance signals are considered, that will be called privacy and utility. The next sections formally define those output signals. Note that those definitions differ form the previously used ones in their real time characteristic. Indeed, at the record of every new location, the measure of privacy and utility has to be updated. The metrics defined in Chapter 5, Section 5.4 require the complete dataset to be computed and do not capture the metrics' dynamics. Thus, new formulations of those metrics are given, however still based on the notions of POIs for privacy and spacial distortion for utility.

Privacy. This work takes as assumption that the objective of a user in terms of privacy is to prevent an attacker from retrieving her points of interest. For the addressed problem, one should have an *online* measure of privacy. The privacy signal should thus represent how likely the user is to reveal a POI, i.e. if she is spending a significant time in a restricted area. Regarding the control-theoretic approach, the privacy signal should also enable a control as simple as possible, for instance by ensuring its linearity. Consequently, the following requirements for the privacy signal are used: (i) reflect a user stop, (ii) being controllable.

The privacy definition is thus based on the spatial dispersion of the obfuscated data over a past time window. Indeed a small dispersion represents a concentration in space (but also in time due to the time window calculation) of location records, which matches with the definition of a POI. The data density property of a data sharing has already been identified as a privacy thread indicator [33], however without formal definition nor use as a privacy signal. Formally, the privacy signal is defined as being twice the median distance between the location records of the time window and the centroid of those points. The location $l(k)$ is considered as being the vector of the latitude and longitude of the user at time k. Then, the centroid $l_c(k)$ of the locations over the past window of length T is defined by eq. (7.1)

$$l_c(k) = \frac{1}{T} \sum_{t=k-T}^{k} l(t) \tag{7.1}$$

and the privacy signal is defined as

$$priv(k) = 2 \times median\left(dist[l(t), l_c(k)]\right), \quad t \in [k-T; k] \tag{7.2}$$

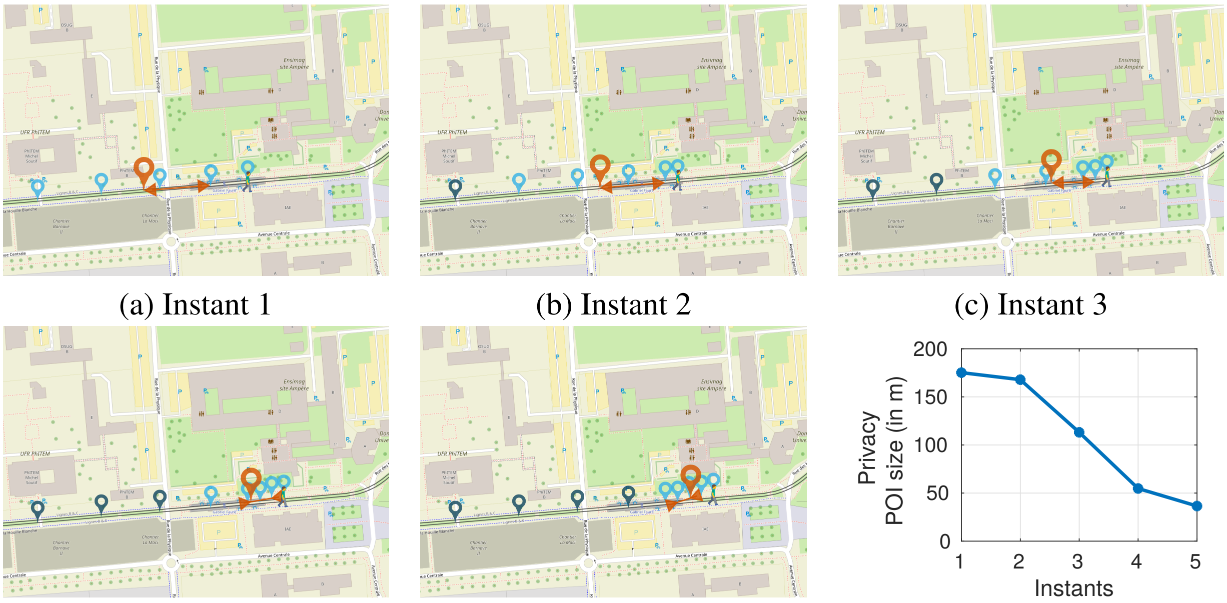
with $dist[x,y]$ being the euclidean distance between two points x and y at the surface of the earth.

The privacy signal is expressed in meters and is to be related with the POI size. This formulation with a factor 2 enables to have a meaningful measure, that is the radius of the smallest POI currently extractable from the trace. The choice of the median rather that other aggregation methods such as the mean or the maximum is to enhance the robustness of the metric regarding outliers, and thus enable a more stable measure for the control perspective. Given its relation to POIs, the privacy signal reference value is to be set by the user as previously explained for the offline case. The length of the time window T is again chosen by the user to fit her conception of privacy.

An illustration of the metric computation is given in Figure 7.3. Each subfigure (a) to (e) is the privacy computation at a given instant $k = 1 \dots 5$. Records of the user position are the small location points, to be considered from left to right. The user was in a tram (most distant darker points revealing high speed), went out and started walking (points are close one to another). The lighter points are the ones in the moving time window T considered at the instant of the picture. For instance on Figure 7.3 (a), the privacy is computed for the fifth point. The duration of the window T is of five samples. Figure 7.3 (b) illustrates the privacy at the instant of the sixth point, and so on. The centroid l_c of the selected points is the large location position. The privacy metric is then two times the median distance between the centroid and any point of the time window (light points), the median is illustrated by the arrow. In this illustration, as the user is slowing down and is likely to arrive in an important place for her, the privacy metric decreases, as reported in the graph Figure 7.3 (f).

To sum up, the privacy metric is the radius of the smallest POI extricable on the last time window. The higher the privacy value is, the better the user is protected

Utility. In this work, utility is considered as being instantaneous (the service mainly need the user's current location) and spatial (the closer the location sent to the LBS is to the user's real one, the better the service will be). This choice has been motivated in Chapter 5, Section 5.4.2, where a summary of the most used properties of location data are used by the LBSs is given (Table 5.2).



(a) Instant 1 (b) Instant 2 (c) Instant 3

(d) Instant 4 (e) Instant 5 (f) Privacy over time

Figure 7.3 – Privacy metric computation on a simple mobility trace.

The notion of utility in this chapter is the distance between the obfuscated data and the original one. This information is partially given by Geo-I's parameter ε, as it is the parameter of the distribution from which the noise is drawn. Utility here only measures the realization of this distribution, actual distance between the real location of the user and the obfuscated one communicated to the service:

$$util(k) = dist[l(t),l'(k)] \tag{7.3}$$

Then, the objective of maximizing utility translates into minimizing ε (which for reminder is inversely proportional to the amount of noise added). We will thus focus out control on the privacy variable and on the spare use of the control variable budget ε.

We also cover the case where more location data are used to provide the service, however with a higher constraint in our formulation as the leveraging of utility loss between the various records is not possible. Some limitations exists however to this formulation. For instance, some applications are using the speed of the user, which is not preserved in our formulation of utility.

7.2.3 Illustrated motivation

Now that the evaluation metrics have be formally defined, we illustrate the impact of both time and the control variable on the user's privacy, for a real trace of a cabspotting mobility dataset user (see Figure 5.1 for an overview of the trace on a map and its speed over time.). Figure 7.4 shows the privacy and utility measures through time, for various scenario: (i) the mobility data has not been obfuscated, it is the raw one, (ii) the user's trace has been obfuscated using Geo-I, parametrized with $\varepsilon = 10^{-2}\,\text{m}^{-1}$, and (iii) $\varepsilon = 10^{-2.5}\,\text{m}^{-1}$. Figure 5.3 illustrates those three traces on a map. Note that for the utility plot, the curve for the raw trace - constantly equal to 0, no distortion - is not visible due to the logarithmic scale. Also, as the sampling period of the trace is not constant, there is no record between 5 and 30 min.

Figure 7.4 illustrates that privacy varies through time, depending on the user's mobility behavior. A high value means a good protection. The lowest privacy is close to 1m, meaning that the user stayed in a 2 m diameter region during the previous $T = 15\,\text{min}$ (or less if records do not cover the exact fifteen past minutes). Knowing with such accuracy the location of a POI is indeed a important privacy breach. The highest privacy level of that user on this record is 7 km, corresponding to the moment when the user is driving along the highway. With time, the privacy metric variation goes up to four orders of magnitude difference, which illustrates the need of a time-varying adaptation of the protection mechanism. Moreover, the privacy curves of the obfuscated traces show that (i) using Geo-I increases privacy and that (ii) the LPPM parameter ε impacts the privacy level. The utility measures vary through time due to the stochasticity of Geo-I process. However, the mean utility is constant and only depends on the LPPM parametrization ε. With the smallest value, the spatial noise added to the user trace is higher (see Figure 5.3), hence the spatial distortion is in average higher, meaning a worst accuracy. Records around 120 min illustrate that there are situations for which having a lower ε do not induce more privacy as the user is inherently protected by its movement, but only damage the utility of data.

Consequently, there is a need for a regulation of privacy *through time* that can be achieved by wisely *tuning* Geo-I's parameter, which will also lead to *utility gains*.

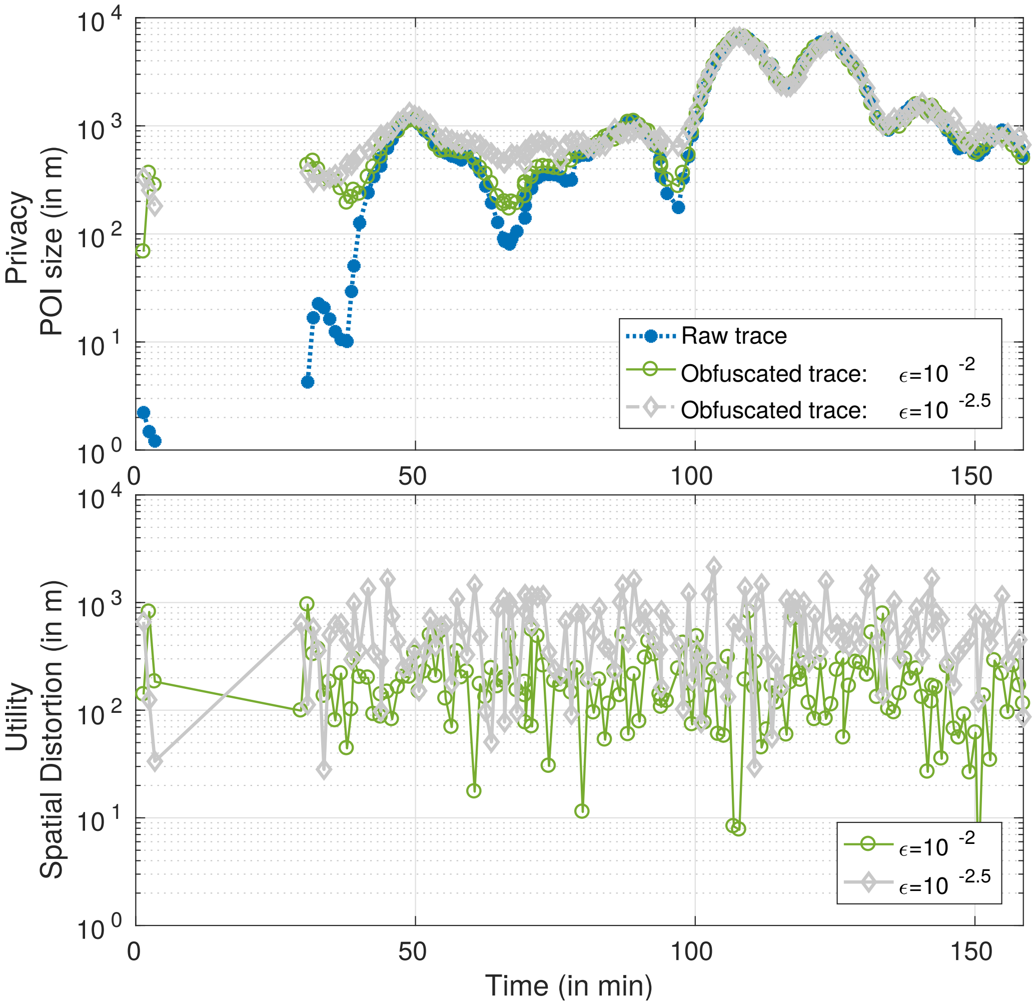


Figure 7.4 – Privacy and utility variations over time for various static LPPM configurations. Data from a cabspotting user.

7.3 Privacy Modeling

7.3.1 Overview

The next step before being able to control privacy is to have a mathematical model describing and quantifying the LPPM behavior, that links the inputs signals to the output one. No utility model is identified as we consider the minimization of the control signal usage to be our utility constraint. Identification in control theory consists in linking the input(s) and output(s) with a mathematical formula, based on experiments. Various input scenario are applied to the system and the output(s) that were generated are measured. The observed behavior is then matched to a set of already known equations with varying properties (in terms of linearity, order, etc.) and parameters, and the best one is kept. This black-box characterization can be done in two steps. First, a general overview of the input to output link is drawn by overlooking the transitory dynamical behavior. A constant input signal is applied and the stabilized output is measured. This process is repeated several times, each time with a different input value. In case of multiple input signals, all combinations are tried. This step is called static characterization. In the second step, the input signal is no longer constant but varies through time. The induced variations of the output are measured. This is the dynamic modeling. In case of multiple input signals, an hypothesis of linearity is often taken, thus only one input is varied at a time and the others are fixed. Those two steps are detailed in the following subsections. Beforehand, details are given regarding the values of our input signals taken during those processes.

7.3.2 Inputs Scenario

The control signal (i.e. LPPM parameter ε) is first iteratively set at different values taken in its whole definition range (from $10^{-4}\,\mathrm{m}^{-1}$ to $1\,\mathrm{m}^{-1}$), for the static characterization. In a second time, a step, i.e. a continuous signal with a sudden change, will be applied, in which the values of the initial and final level will be chosen in the linear working range of the system, found with the static characterization. The dynamics of the system, i.e. its variation through time, is expected to come only from the time window calculation of the privacy signal, which motivates the time analysis over the frequency one. Indeed, the use of a step enable to solicit all frequencies at once.

The second input we have is the movement of the user. To study its impact on the LPPM performance, we first stimulate the system with a synthetic movement that we manually created, in order the master perfectly its properties. The disturbance scenario, i.e. the mobility data, is characterized by one main parameter: the user's speed. The synthetic mobility scenario that we created for the preliminary analysis of the plant has thus various speeds. This parameter is discretized: speed can be high (50 km/h), low (5 km/h) or null (the user is stopped).

A movement, i.e. an evolution of the user latitude and longitude through time, can indeed be characterized with various other metrics, such as the direction of the movement and its changes over time, the acceleration of the user, etc. Even though the model will be derived using this simplified movement scenario, we will validate it with real user data in the evaluation section.

7.3.3 Static Characterization

For the static characterization step, the LPPM parameter and the user movement are fixed, and the privacy is measured. Those measures are repeated for each value of the parameter ε and each movement.

Results are reported in Figure 7.5, x-axis being the control parameter, y-axis the privacy measure, and each curve is a different disturbance condition. For both axis, a logarithmic scale is used. The following statements can be formulated: (i) the logarithm of privacy is linear with respect to the logarithm of Geo-I parameter for low values of ε (high noise) and (ii) for high values of ε (low noise) there is a saturation, and the level of this saturation depends on the disturbance (i.e. speed of the user).

The saturation reflects that there are some conditions, for instance if the user is moving fast, for which adding a few noise has no impact on the privacy as the user is already protected (i.e. only POI with large diameters can be extracted from her raw trace). The linear part of the curve means that, at some point, the more noise is added to the data, the larger the diameter of the extracted POI is.

The linear part of the static characteristic has the same equation in all cases:

$$log(priv) = a\ log(\varepsilon) + b \tag{7.4}$$

with $a = -1$ and $b = 0.2$ (parameters are found using Matlab regression tool).

The saturation level corresponds to the privacy of the mobility data when $\varepsilon \to +\infty$, i.e. no noise is added. It is the privacy of the raw trace, that can be measured in real time, locally on the user's mobile device. This value is denoted $priv_{raw}$.

The static gain A links the LPPM parameter to the privacy level, still logarithmically, see eq. (7.5).

$$log(priv(t \to \infty)) = Alog(\varepsilon(t \to \infty)). \tag{7.5}$$

According to the static characterisation, we have the following formula for the static gain:

$$A = \begin{cases} a & \text{if } a\ log(\varepsilon) + b > log(priv_{raw}) & : \text{linear zone} \\ 0 & \text{otherwise} & : \text{saturated zone.} \end{cases}$$

In this formulation, no smooth transition between the saturation level and the linear behavior is modeled.

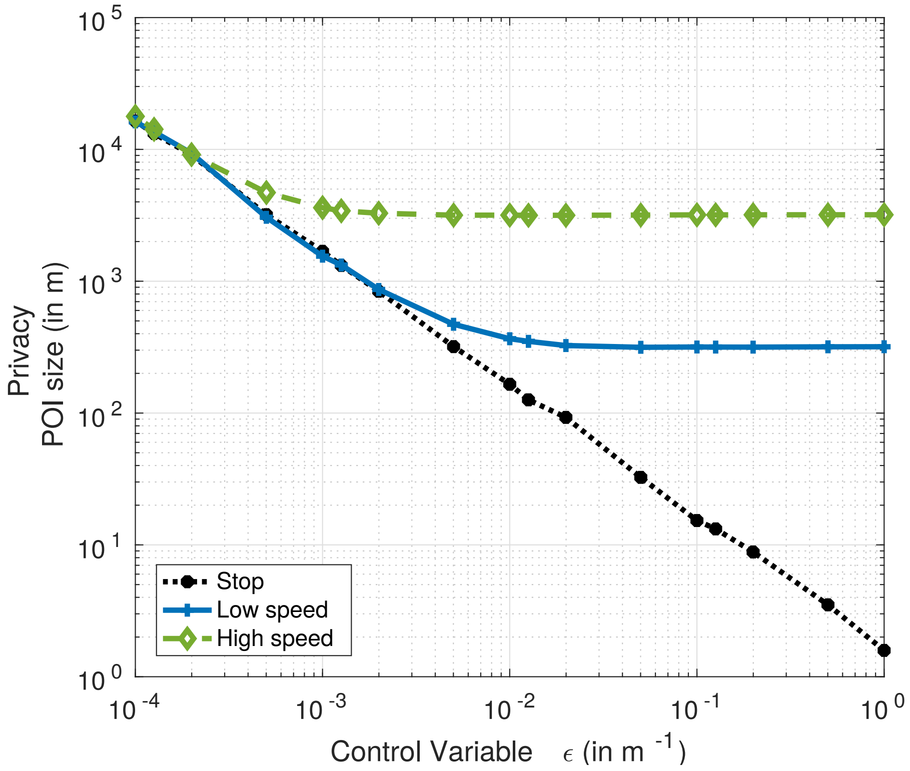


Figure 7.5 – Static characteristics of the ε to privacy function, for various disturbance scenario.

Linearization In order to match the control theory formulation and due to the presence of the offset b, the model is linearized around a working point ε_0 and its corresponding privacy level $priv_0$ (computed using eq. (7.5)). Thus the control signal is defined as

$$\Delta\varepsilon = log(\varepsilon) - log(\varepsilon_0), \tag{7.6}$$

and the performance signal as

$$\Delta Priv = log(priv) - log(priv_0). \tag{7.7}$$

The linearization point is chosen as the value of ε in the middle of its definition range: $\varepsilon_0 = 10^{-2}$.

7.3.4 Dynamic modeling

The system is linear for low values of the control signal while for higher values, the control signal do not impact the output signal. Therefore, the system will be dynamically characterized only in this linear zone, where a single step response can be used for modeling. The control signal step goes from $\varepsilon = 10^{-1}\,\mathrm{m}^{-1}$ to $\varepsilon = 10^{-2}\,\mathrm{m}^{-1}$ and the disturbance has constant null speed. The two values of the control signal are in the linear zone of the system given the user is stopped. The evolution of privacy through time is measured. Results are reported in Figure 7.6.

The shape of the step response can be approximated by a first order transfer function given its exponential form with non null tangent at the origin. The transfer function relating the LPPM parameter to privacy is then:

$$H(s) = \frac{\Delta Priv(s)}{\Delta\varepsilon(s)} = \frac{A}{1+\tau s} \tag{7.8}$$

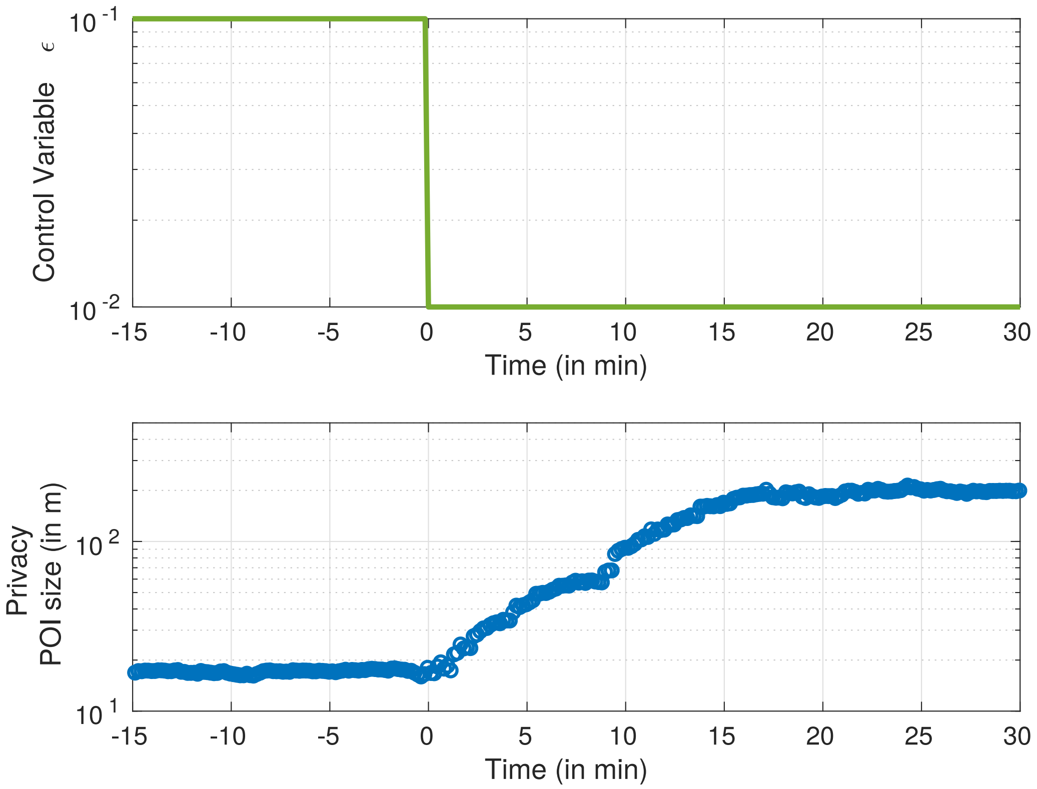


Figure 7.6 – System step response. Input from $\varepsilon = 10^{-1}\,\mathrm{m}^{-1}$ to $\varepsilon = 10^{-2}\,\mathrm{m}^{-1}$, during a stop (constant disturbance).

with τ =5 min by considering a rise time of around 15 min (i.e. the length of the privacy metric time window T) and $A = a = -1$ (linear zone), with s the Laplace variable.

If written in the recursive form, eq. (7.8) becomes:

$$log(priv(t)) = \alpha \cdot log(priv(t-1)) + \beta \cdot log(\varepsilon(t)) + \gamma. \tag{7.9}$$

When time goes to infinity, eq. (7.9) should fit eq. (7.4) as it corresponds to the steady state value of privacy. This creates the following constraints:

$$a = \frac{\beta}{1-\alpha}, \;\; b = \frac{\gamma}{1-\alpha}. \tag{7.10}$$

These two constraints let one degree of liberty in eq (7.9). This enables to tune the time dynamics of the response τ, i.e. the time the privacy signal takes to reach its steady state.

The resulting model for privacy prediction, combining both static and dynamic studies is the following:

$$priv(t) = \begin{cases} 10^{\alpha \cdot log(priv(t-1)) + \beta \cdot log(\varepsilon(t)) + \gamma} & \text{if } \varepsilon < \varepsilon_0 \\ priv_{raw}(t) & \text{otherwise.} \end{cases}$$

This equation enables to predict, for each time instant, the value of privacy knowing the obfuscation level (ε), the past value of privacy ($priv(t-1)$) and the raw trace properties (ε_0 and $priv_{raw}$).

7.4 Control Strategy

7.4.1 Objectives

This section presents the Location Privacy controller. The user's objective is defined as a minimal threshold on the privacy value no matter the mobility pattern. Indeed as privacy and utility are contradictory objectives, in order to maximize utility the control signal should be high (i.e. low noise), but still enabling the reference privacy to be met. Regarding the rejection time of disturbances, it can vary according to the user's requirements. In our case, we chose a value of 5 min. It corresponds to a constrained objective where every small stops (in a shop, at a bus stop, etc.) and all larger ones should be hidden.

To sum up, the closed-loop specifications to assure are:

- follow the privacy reference,
- reject the perturbation with zero steady state error, small overshoot and in approximately 5 min,
- increase utility whenever privacy constraints are met.

Similarly as in [49], the control is meant to be simple, as its objective is not only to achieve the above mentioned performance but also to be easily implementable on mobile devices. Hence, a PI controller with anti-windup is selected. Its integral action enables zero steady state and its tuning allows to avoid overshoot and reach a desired response time. The anti-windup action is added to cope with the control signal saturations.

7.4.2 Linear Control Problem Formulation

We now reformulate the problem to linearize it and take into account the logarithmic formulation. The schema of Figure 7.7 illustrates the new notations on the control loop. $\Delta Priv_{sp}$ is the desired privacy level, linearized:

$$\Delta Priv_{sp} = log(priv_{sp}) - log(priv_0). \tag{7.11}$$

The user's objective $priv_{sp}$ is for instance 100 m, meaning that the user does not want POIs of 100 m radius or smaller to be extractable from their mobility record.

The plant model that we are interested to derive in this section gathers the LPPM and the sensor processes. In the static case studied just before, we had $H = A$. We now study the time dynamic scenario, to link the control signal $\Delta\varepsilon$ to the output privacy signal $\Delta Priv$

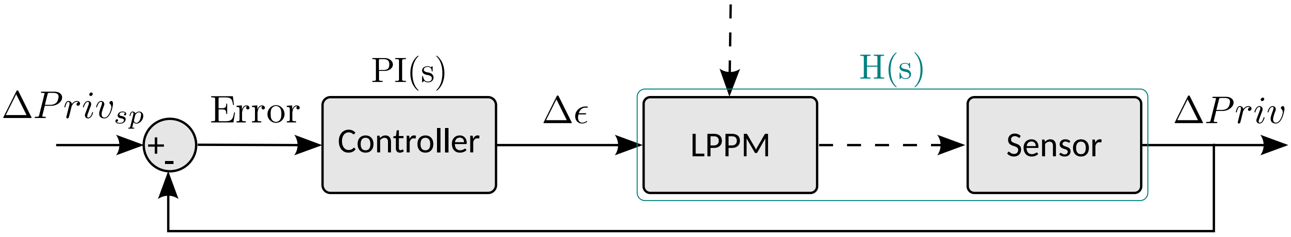


Figure 7.7 – Feedback loop with signal and transfer functions notations

7.4.3 PI Formulation

A classic PI controller has the following expression in the s-domain, after a Laplace transform:

$$PI(s) = \frac{\Delta\varepsilon(s)}{\Delta Priv_{sp}(s) - \Delta Priv(s)} = \frac{K_I}{s} + K_P. \tag{7.12}$$

In the time recursive form, the PI becomes:

$$\begin{aligned} log(\varepsilon_{PI}(t_i)) = \quad & log(\varepsilon(t_{i-1})) - K_P[log(priv_{sp}(t_{i-1})) - log(priv(t_{i-1}))] \\ & + [K_I(t_i - t_{i-1}) + K_P](log(priv_{sp}(t_i)) - log(priv(t_i))) \end{aligned} \tag{7.13}$$

In our case the parameters are tuned using pole placement as detailed in [23]: $K_I = \dfrac{\tau}{A.\tau_{sp}}$ and $K_P = \dfrac{1}{A.\tau_{sp}}$, with τ_{sp} the pole of the objective closed loop, fixed by the desired response time. After computation, with $\tau_{sp} = 5\,\text{min}$, the parameters of the controller become $K_I = \dfrac{-1}{300}$, $K_P = -1$.

7.4.4 Anti-windup

In this control formulation, the gain K is assumed to be linear, always equal to a. As such, the controller is not aware of the saturated effect on privacy (flat zones of Figure 7.5). If the privacy objective is naturally overshooted thanks to the user mobility pattern, the controller will keep decreasing ε as it will not see any impact on privacy. This is a behavior that interest us, as decreasing the control signal without impacting privacy actually means decreasing utility loss without damage on privacy. However, if the user stops or slows down, the controller should not take too much time to react and decrease the control signal. In order to ensure such behavior, an anti wind-up strategy is added, in the form of an actuator saturation (see [175]):

$$\varepsilon(t_i) = min(max(\varepsilon_{PI}(t_i),\varepsilon_{min}),\varepsilon_{max}). \tag{7.14}$$

where threshold are chosen according to Geo-I's common range of variations $[\varepsilon_{min}, \varepsilon_{max}] = [10^{-4},1]\ \mathrm{m}^{-1}$.

7.5 dynULP Evaluation

Both the modeling and the control are evaluated in this section. We first give an overview of the methodology: the scenario considered and the performance indicators used for the evaluation.

7.5.1 Methodology

Results presented in this section are the outputs of single runs. No averaging is done on several similar experiments in order (i) to preserve the explainability of the impacts of the signals on each other, (ii) to fit as much as possible to the online scenario with only one-shot opportunities for the controller, and (iii) not to hide the difficulty of controlling a stochastic system by showing only its average control.

7.5.1.1 Inputs scenarios

The evaluation aims at illustrating three points: (i) the understanding and **explainability** of the modeling and control, (ii) the **generality** of their performance in front of a diversity of situations and (iii) the **applicability** of the solution on a real-life mobility scenario. The inputs of the model (mobility trace and ε signal) and the ones of the controller (mobility trace and the reference signal) are set in order to reflect those three points, as explained in the following:

- **Mobility trace.** First a simple scenario is used, in which the user is considered either stopped, or moving with a constant speed (low or medium), with step variations between those situations. The mobility trace used for generality validation is the synthetic one presented in Chapter 5, Section 5.1, which gathers more complex moves such as turns, acceleration, micro-stops, etc. Eventually, the applicability in practice is tested using a real-life mobility record, the one of a Cabspotting user, also presented in Chapter 5, Section 5.1.

- **LPPM parameter ε.** It is set by the controller in order to meet the reference signal. However, in the modeling phase it is a free signal that should be set in order to stimulate the plant. In its simplest form, ε is constant at the median value of its definition range, 10^{-2}. A step signal - instantaneous change from one value to another - is also used. Eventually, we generate a more complex signal, with various amplitudes ($10^{-4}\,\mathrm{m}^{-1}$ to $1\,\mathrm{m}^{-1}$) and periods (from one 10 s, i.e. about a sampling period of the mobility trace, to 30 min).

- **Privacy specification** $priv_{sp}$**.** When needed, e.g. for the control evaluation, the reference value is either (i) constant at 500 m (POI of max 1 km of diameter), (ii) a step, or (iii) a random signal with varying amplitude (from 1 m to 10 km) and period (from 5 min to 5 h). Changes in the specifications are made by the user herself, thus are not as frequent as the changes in ε, that are driven by the controller. Regarding the values of amplitude values, they range from very low, i.e. the stop place can be determined with the precision of a meter, to really large, the stop location cannot be more precise than several kilometers.

For the explianability scenario, the inputs are considered in their simplest form, i.e. the steps, and one input varies at a time. For the generality and applicability scenarios, the robustness is first considered, meaning that only the mobility of the user varies through time while ε is fixed for modeling and the reference is fixed for control. Then, the complex varying scenarios of those last two signals are also considered, still with the complex mobility behavior, in order to insure realism.

Even though the numerical values of each parameters and variables have been given and justified in due time, we give in Table 7.1 a sum-up. Some of the variables in this table are sometime varied, only their nominal value is reported.

Notation	Definition	Nominal value
T	Privacy metric time window.	15 min
a b	Static characteristic parameters.	-1 0.2
A τ	Transfer function gain, characteristic time.	-1 5 min
ε_0	Linearizion point.	10^{-2} m^{-1}
τ_{sp}	Desired rejection time.	5 min
K_P K_I	Controller parameters.	-1 -0.033
$[\varepsilon_{min}, \varepsilon_{max}]$	Anti-windup parameters.	[10^{-4},1] m^{-1}
$priv_{sp}$	Privacy reference value.	500 m

Table 7.1 – dynULP parameters sum-up.

7.5.1.2 Performance indicators

In order to evaluate modeling and control performance, both qualitative and quantitative aspects will be considered. The qualitative one puts in perspective this work in its context, going back to the signification of the metrics and of their orders of magnitude. It also allows to take examples for illustration. A quantitative evaluation is mainly meaningful when comparing two approaches, which is not the case in this work. It will however be used to give a first idea on performance (even if no reference on the values are available) and to allow comparison with future works. Moreover, performance indicators will be useful to compare accuracy of the model and controller on real user's data compared to synthetic ones.

Modeling accuracy. The mean normalized square error of privacy is used, for which the definition is given in eq. (7.15), where K is the set of all samples. Is represents the difference between the model prediction and the measured privacy, with a quadratic importance of the difference, normalized with regard to the measured privacy and averaged over all sampling instants. A logarithm function is first applied on the privacy signals in order to give equal importance to a error made on small values of privacy and on large ones. Like this, the performance indicator gives an idea of the errors in terms of *orders of magnitude*.

$$Err_{model}^{priv} = \frac{1}{|K|} \sum_{k \in K} \frac{(log(priv_{model}(k)) - log(priv_{measure}(k)))^2}{log(priv_{measure}(k))} \tag{7.15}$$

Privacy regulation. The controller's performance in terms of privacy is reflected by its ability to keep the measured privacy *at least* at the required level no matter the disturbances. Indeed, a privacy higher than the required level is not considered as detrimental. Based on this fact, for the simple reference tracking validation, only the maximum overshoot *below* the reference is reported, as well as the time spent below 95% of the reference. Steady state error, rejection time and maximal amplification of the disturbance are also measured, still taking into account only the samples for which privacy is lower that specified.

For the scenarios with elaborated inputs, we again compute the mean normalized squared logarithmic error but taking into account only the error when privacy is lower than its reference. The privacy evaluation indicator is calculated as follows:

$$Err_{control}^{priv} = \frac{1}{|K|} \sum_{k \in K} \frac{(max[log(priv_{sp}(k)) - log(priv_{measure}(k)),0])^2}{log(priv_{sp}(k))}. \tag{7.16}$$

Here again the error is computed using the logarithm of the privacy signals, to weight the importance of all orders of magnitude.

Utility preservation. In order to evaluate the ability of the control law to preserve utility for the user through time, two indicators are used: (i) the median utility over the experiment and (ii) its 99^{th} percentile. The first metric gives a global overview of the quality of the service, robust to the spread of the metric across orders of magnitude, while the second ensures no major malfunction was done to the service.

7.5.2 Modeling

This section evaluates the modeling, i.e. the ability of the model to capture the privacy level of a user through time, knowing the LPPM parametrization ε and the user movement. We start the evaluation with a simple step scenario for those two inputs, then we use a comprehensive synthetic trace generated to overview most use cases and we conclude on its performance on a real-life data.

7.5.2.1 Validation on simple scenarios

We first fix the user movement (the user is stopped, 0 km/h) and vary the control variable ε from 10^{-1} m^{-1} to 10^{-2} m^{-1} and then back to 10^{-1} m^{-1}, to experiment both a step up and a step down, see Figure 7.8. The top plot is the user's speed trough time, the middle one is the privacy measured and its modeling and the bottom plot is the LPPM parametrization, all signal vary in time. This layout is the same for Figures 7.8 to 7.13. At time 0, when ε decreases, the noise amount increases and so does privacy, both its measure and modeling. Conversely when ε increases. The modeling presents no steady state errors and no overshoots, but there are some errors during the transient phases. For both steps, the model is slightly faster than the measure. The global modeling error is 1.1%.

Then we fix the control variable to $\varepsilon = 10^{-2}$ m^{-1} and vary the mobility, from moving at 50 km/h to 2 km/h and then back to 50 km/h, see Figure 7.9. Here the privacy starts by decreasing, stabilizes and then climbs back up to the same initial value. The model captures perfectly the dynamics, but has a small steady state error on the step down while none on the step up. However, the steady state error is limited, only 1.7% at 52 min if we take the logarithm of the privacy values, 8.3% otherwise. The global modeling error is 0.1%, showing a better modeling of the impact of the user's speed (disturbance) on privacy that of the one of the control variable, the LPPM parameter.

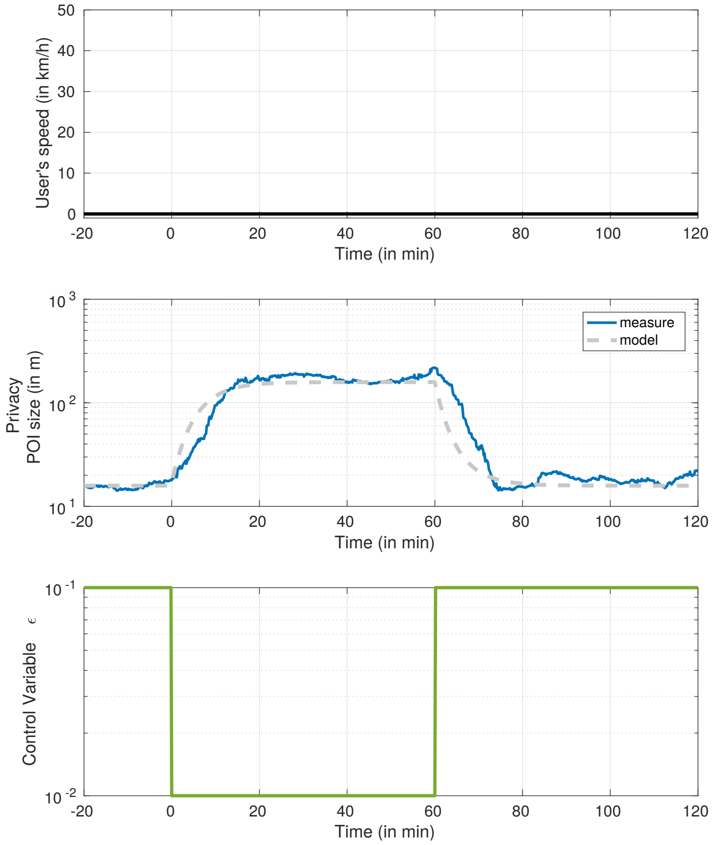


Figure 7.8 – Privacy dynamic model validation with control variable steps.

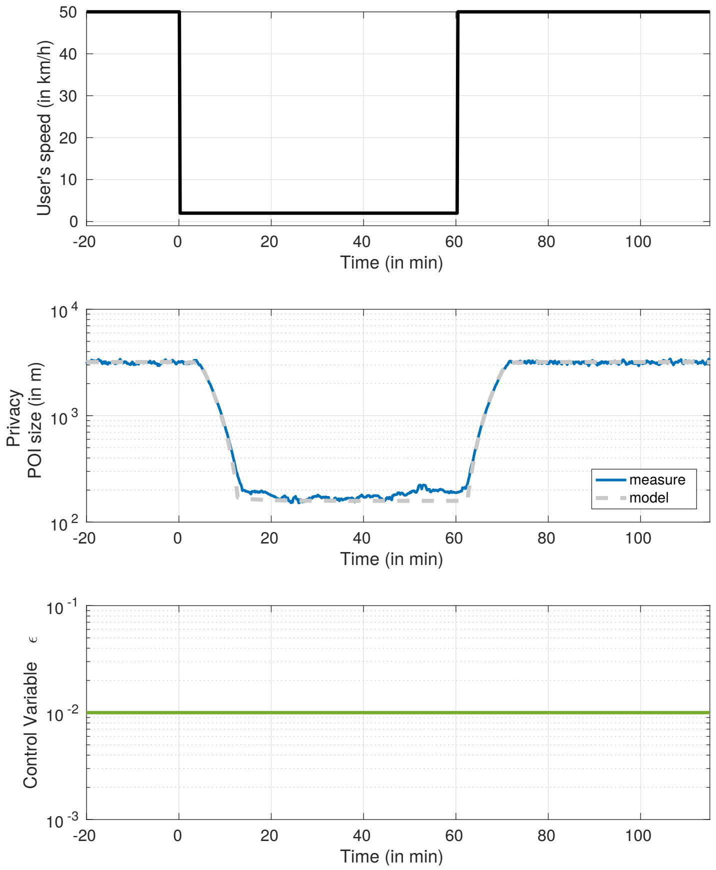


Figure 7.9 – Privacy dynamic model validation with user's speed steps.

7.5.2.2 Evaluation on substantial synthetic data

The accuracy of the privacy model is now investigated on a complete synthetic trace, see Figures 7.10 and 7.11, with the disturbance scenario represented by the user's speed each time on their top plot.

In Figure 7.10, the control variable ε is kept constant, and only the modeling of the user's movement is evaluated. Once again, the accuracy is very good, transients are perfectly captured and the steady states are precisely modeled when privacy is high (above 10^3 m) but quite imprecise for low privacy. However, the metric is most of the time *underestimated*, which in our privacy context results in being more conservative, which is, even if non-optimal, not detrimental.

The global modeling error is here again 0.1%.

In Figure 7.11, the complex scenario for the control variable is used, see bottom plot. The values taken by ε cover all its definition range, the periods between two changes also vary, to stimulate the system with various frequencies. The privacy's measure and model curves are almost overlaid, indicating a good model accuracy most of the time. The steady state values are always correct, unless at times 380, 405 and 840 min. At these moments, ε raises so the model starts decreasing the privacy. However, slightly after the user which was stopped starts moving, which make the privacy level rise again. As the model is too fast when ε varies, the predicted privacy went lower than in reality. The faster dynamic of the model compared to the measure is also seen at times 685 min and 920 min, when there are major steps up of ε while the user is stopped.

The global modeling error is 3.1%, a quite low value showing here again that the model is able to successfully capture the influence of the LPPM configuration ε (control signal) and the user's mobility (disturbance) on the privacy, for this comprehensive synthetic scenario.

7.5.2.3 Evaluation on a real user mobility trace

We now investigate the ability of the model to capture privacy of a user based on real records of a mobility trace.

Figure 7.12 presents the privacy measured and modeled (middle plot) with regards to the user's speed (top plot) and a constant LPPM parametrization (bottom plot). The sampling time in this trace is not constant (in average around 10 s), for instance there are no samples between 5 and 30 min. This explains why there is no measured privacy and why the modeling do not converge in the very first minutes. In the remaining of the trace, the model capture with a good accuracy the variations of the privacy level.

The global modeling error is 0.3%, which shows quite similar performance on synthetic traces and on real data.

Figure 7.13 is the result of an experiment with the same mobility trace, but this time the control signal ε varies according to the scenario presented in Section 7.5.1.1. Except for the initial minutes for which the control predictions have not converged yet, the modeling is accurate both in steady state and in dynamics. Even for large variations of the control signal (around 120 min), the model prediction fits the measured privacy with high accuracy. It could be that the higher frequency variations in the real mobility trace compensates for the model inaccuracies.

The global modeling error is 0.6%, illustrating the ability of our controller to be used for real traces as well as for synthetic ones, and for various conditions of control signal variations.

However, the model has been developed for a control perspective. Then, even though the modeling accuracy is satisfying to reflect the user's privacy, we should now verify its main objective: the controller performance.

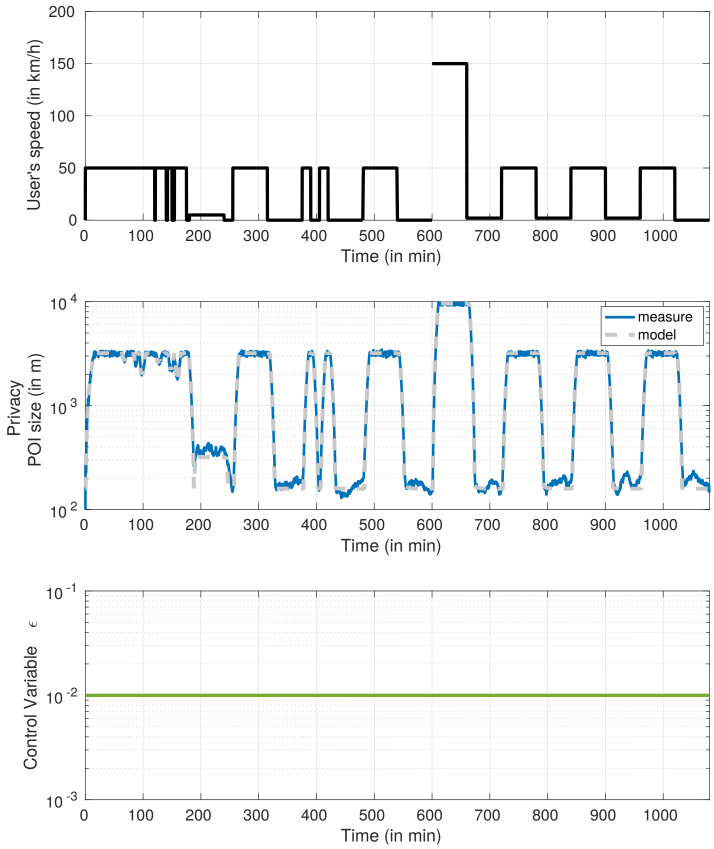


Figure 7.10 – Privacy dynamic model validation on substantial synthetic data, constant control signal.

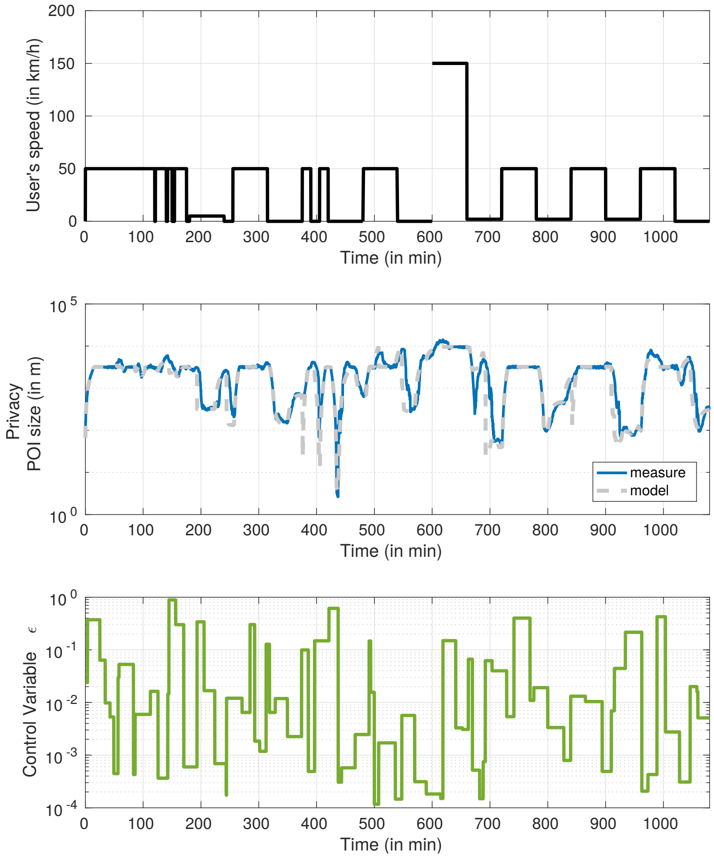


Figure 7.11 – Privacy dynamic model validation on substantial synthetic data and control signal.

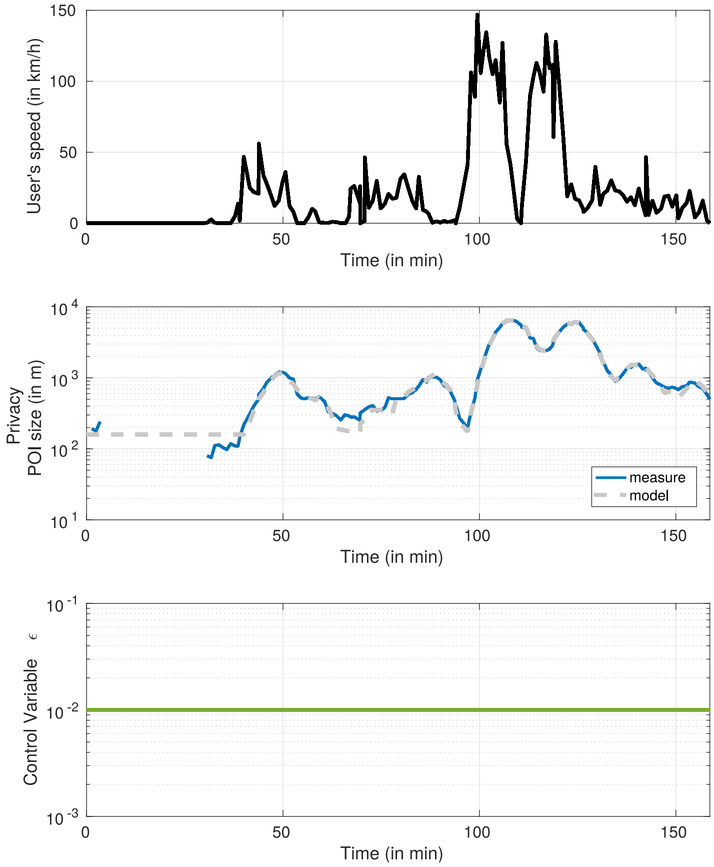


Figure 7.12 – Privacy dynamic model validation on a real mobility trace, with constant control signal.

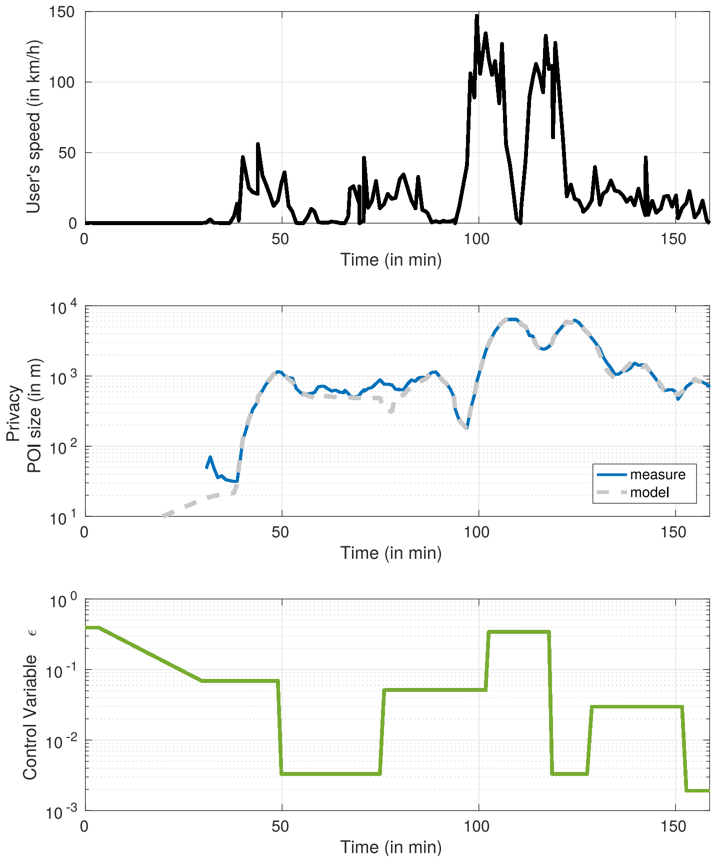


Figure 7.13 – Privacy dynamic model validation on a real mobility trace with a realistic control signal.

7.5.3 Control

The controller aims at keeping privacy at the user specified level, no matter her movements, while ensuring a high utility. The ability of the controller to achieve those objectives is validated on simple steps scenarios, evaluated on a synthetic comprehensive scenario and its practical applicability on real data is tested.

7.5.3.1 Validation on simple scenarios

The first validation experiment consists in a reference tracking challenge. The disturbance is fixed, i.e. the used is stopped, the controller is launched and the experiment do not start before it converged. Then we ask the controller to set the privacy level an order of magnitude higher (from 10^2 m to 10^3 m) and back to its initial value. Results are reported in Figure 7.14 with, from top to bottom, (i) the user's speed over time, (ii) the privacy reference, the privacy level without using LPPM nor control and the controlled privacy, (iii) the control signal and (iv) the utility metric. The privacy curve without control

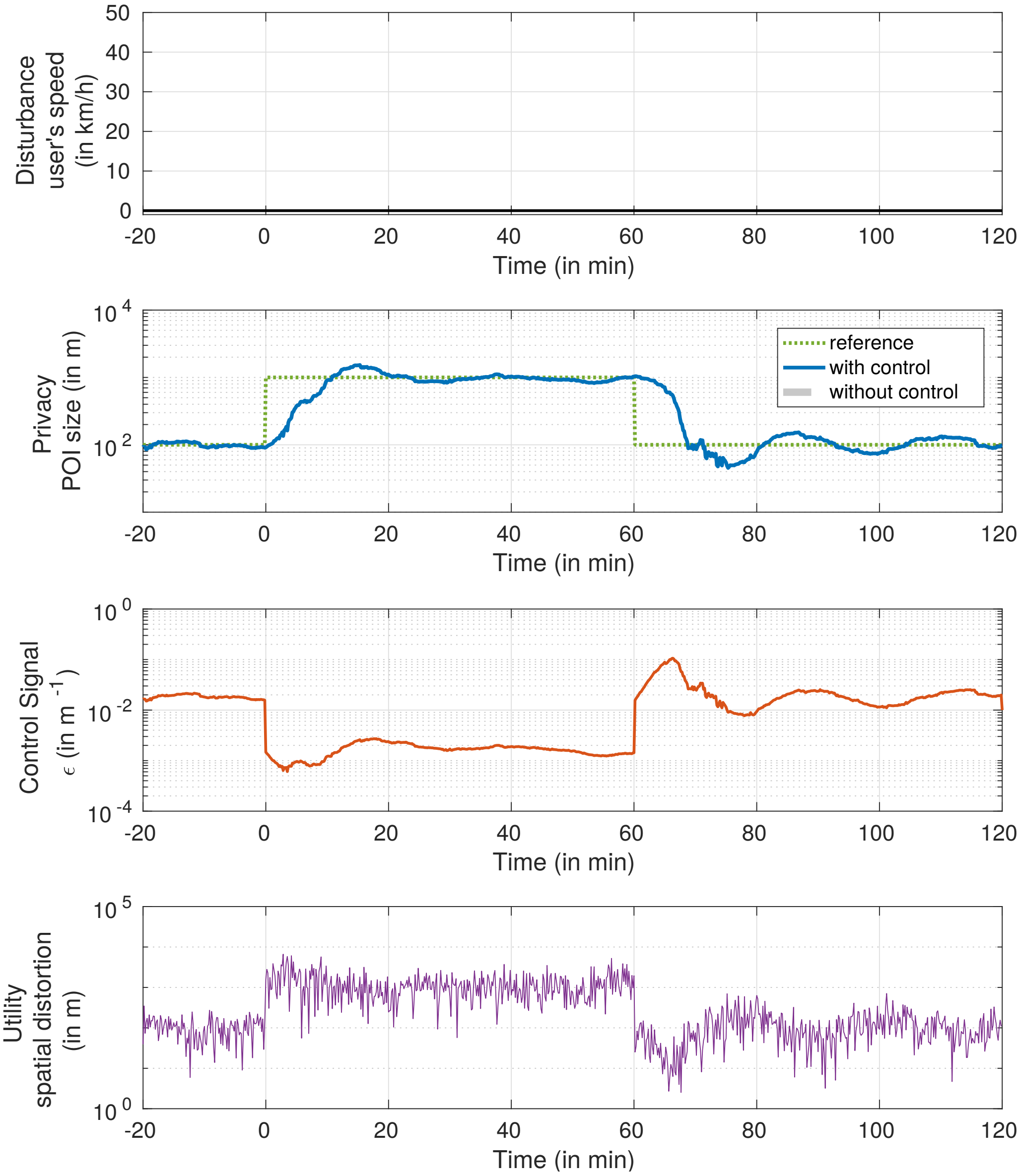


Figure 7.14 – Dynamical privacy control of a stopped user with step privacy specifications.

do not appear on this log-scale graph because, as the user is stopped, it is constantly null.

The controlled privacy follows the reference without steady state error. The time spent below 95% of the step amplitude is respectively of 17 min and 13 min. It takes longer than what was initially aimed. Overshoot is present for both steps. The maximal negative amplification (at time 75 min) is of -40%. The privacy error indicator described in eq. (7.16) is of 1.36%. The median utility is of 106 m and its 99^{th} percentile is 3.6 km, which are reasonable values. We clearly see here the trade-off between a fast privacy tracking, which would have required higher amplitudes for the control variable, and a maximal utility, which would have required to set the control variable instantly to its converged value.

Figure 7.15 show the controller behavior in a step disturbance rejection scenario, with a constant reference. Only the step down (the user reduces its speed) make the privacy level drop below its specification, but eventually converges again to its reference level without steady state error. In the step up case, the privacy level converges higher that the reference, as the user movement naturally protects her at a high level. The maximal amplitude reduction is of 48% at 15 min. The time spent

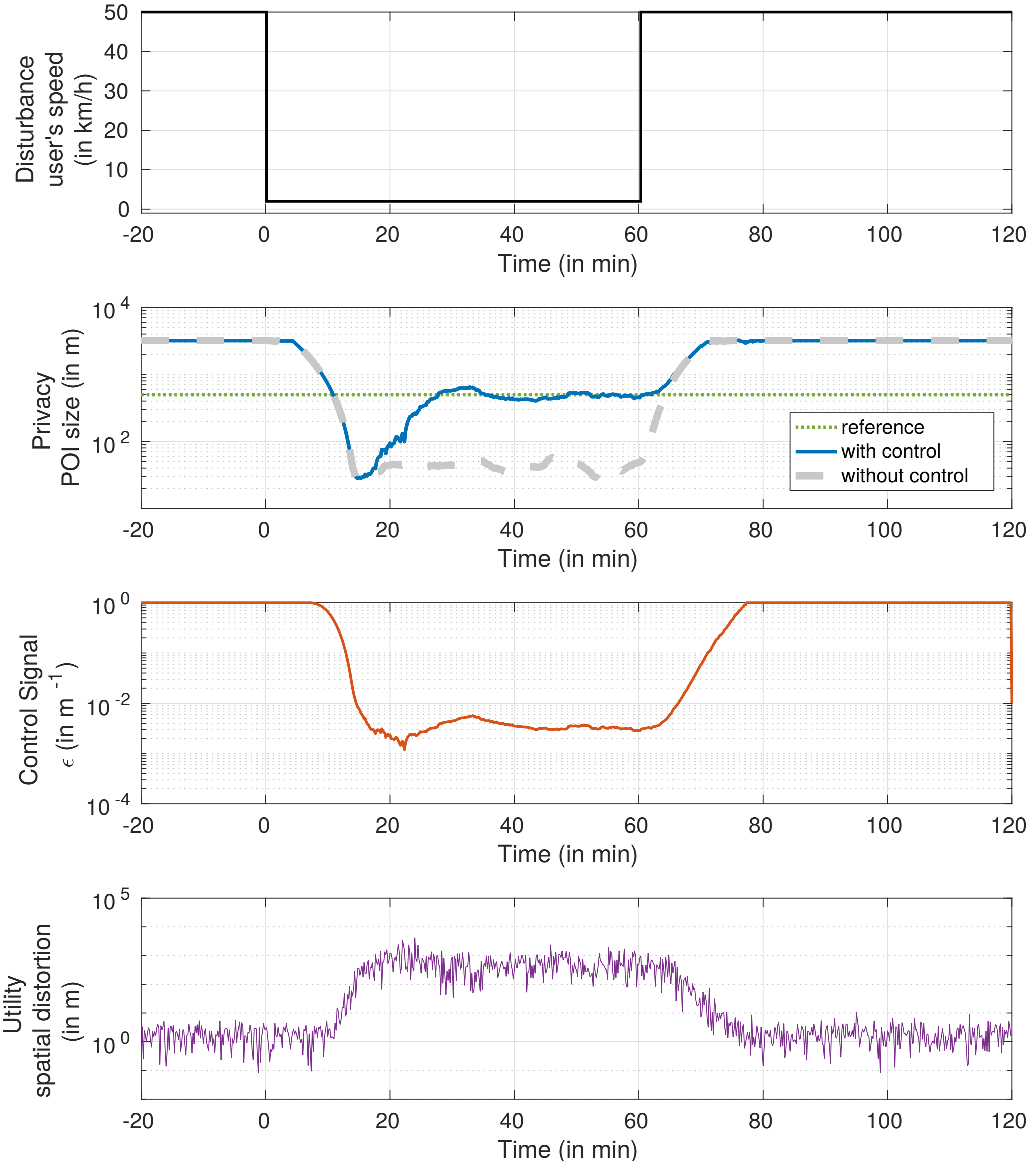


Figure 7.15 – Dynamical privacy control with constant privacy specifications, with user's speed steps.

below the reference is of 17 min. The error indicator is of 2.9%: the controller performs better for reference tracking. Median utility is of 4 m, with a 99^{th} percentile at 1.8 km. Utility is well preserved as there is no overshoot.

To conclude, the simple PI controller is stable and works perfectly in steady state. The transient phases could be optimized, particularly regarding "down"-shoots and response times. The reference tracking test works the best but the disturbance rejection is still good, which is the main the objective of the controller.

7.5.3.2 Evaluation on substantial synthetic data

The controller is then evaluated on the complete scenario, first with a constant privacy reference, see results in Figure 7.16. Similar to previous experiments, the controller keeps the privacy at its required level without steady state error but with negative overshoot when speed decreases drastically. The pri-

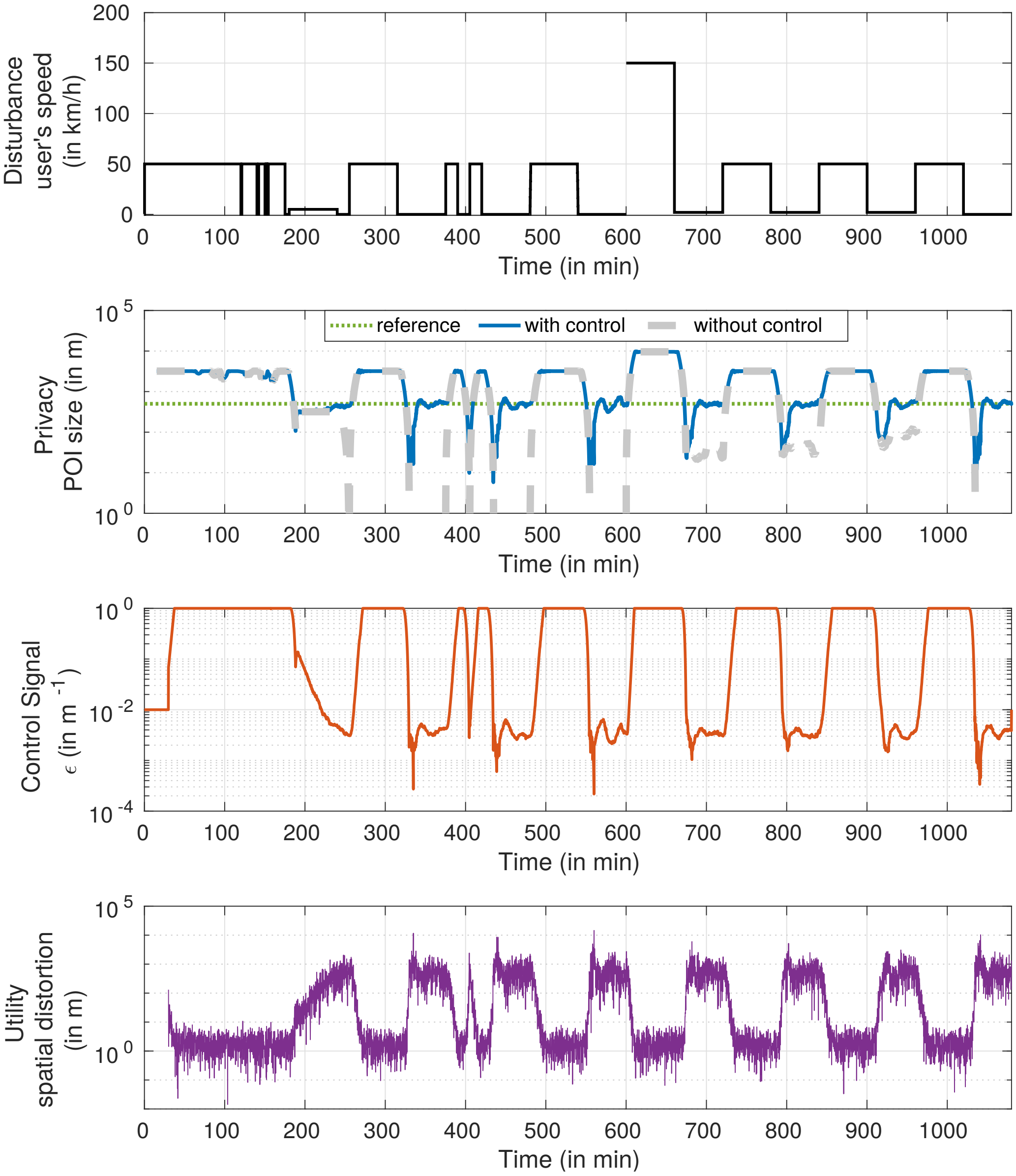


Figure 7.16 – Dynamical privacy control with constant privacy specifications on substantial synthetic data.

vacy performance indicator - mean normalized squared logarithmic error - is of 2.7%, which is close to the previous regulation experiment. Median utility is 4 m, and its 99^{th} percentile 2.0 km, which are still reasonable errors for instance for recommendation systems.

When the specification also varies, see Figure 7.17, the controller performs similarly. Results are even slightly better, as the negative overshoots seem to be reduced. However, this could be explained by the fact that the reference level is below the privacy level inherent in the trace during more time than in the previous experiment. Quantitatively, the performance indicator is of 1.5%. The median utility is around 2 m, while its 99^{th} percentile is of 1 km, significantly better than the previous experiment. This also could be explained by the relaxed specification scenario.

The controller is able to keep privacy at its specified level even in front of diverse mobility patterns. Transient phases are however still noticeable.

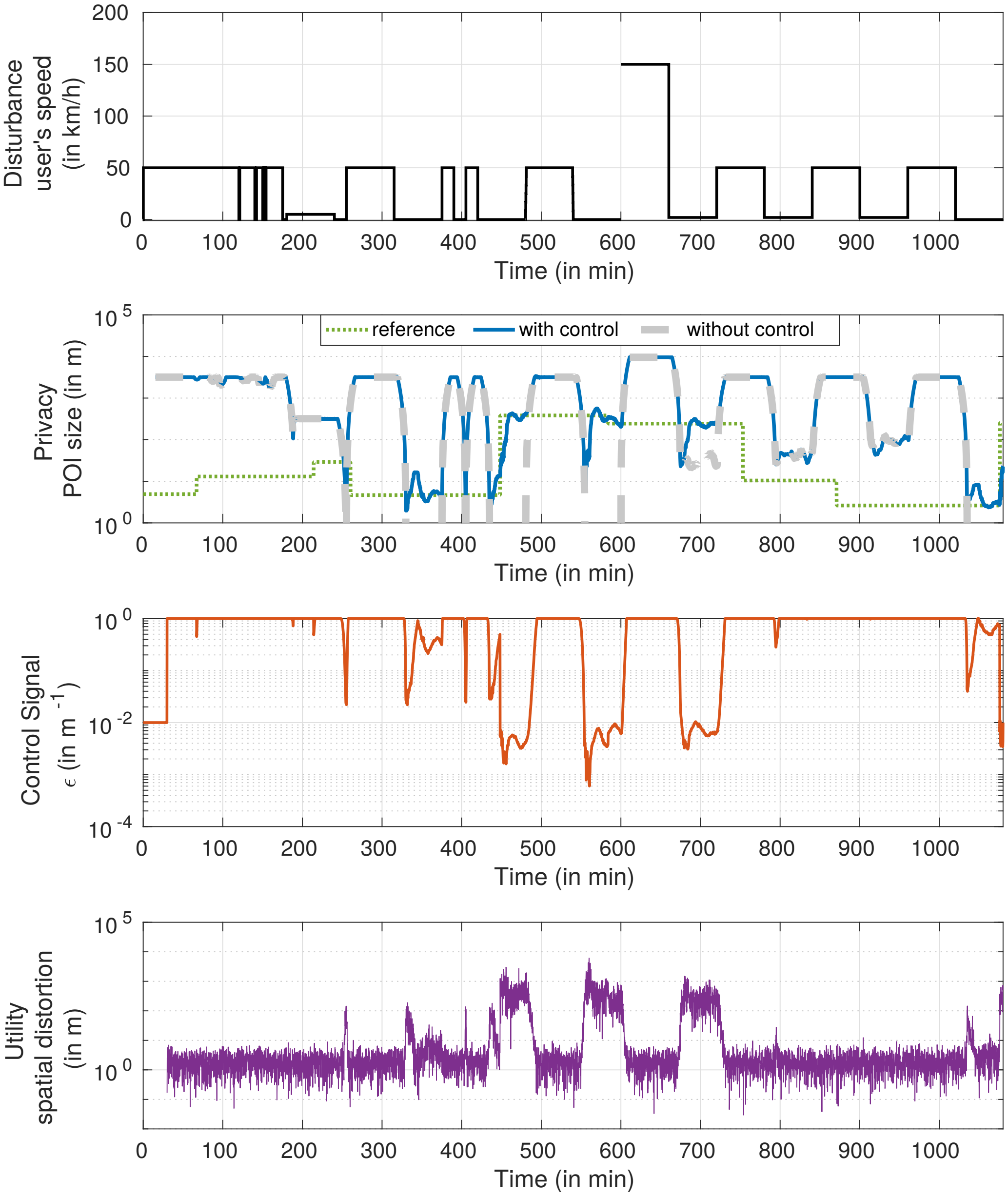


Figure 7.17 – Dynamical privacy control with substantial synthetic data and privacy specifications.

7.5.3.3 Evaluation on a real user mobility trace

The ability of the controller to cope with real data is now evaluated, using the trace of a Cabspotting user. Unlike in Section 7.5.2.3, we use the trace on a longer period, up to 800 min. This extension is useful for illustration as during the beginning of the trace, the user does not significantly stop, and thus is naturally well protected, even without control.

The first evaluation is set with a constant reference, see Figure 7.18. The privacy level is kept at it desired level, with oscillations that reflect the user's speed. When the user is fully stopped - from 580 min to 680 min - the controller manages to converge to a non oscillating steady state value. The performance indicator is 1.1%, which in comparison with the tests on synthetic traces is even better. Utility is well preserved both in general and in extreme cases, as its median value is of 11 m and it 99^{th} percentile 1.2 km.

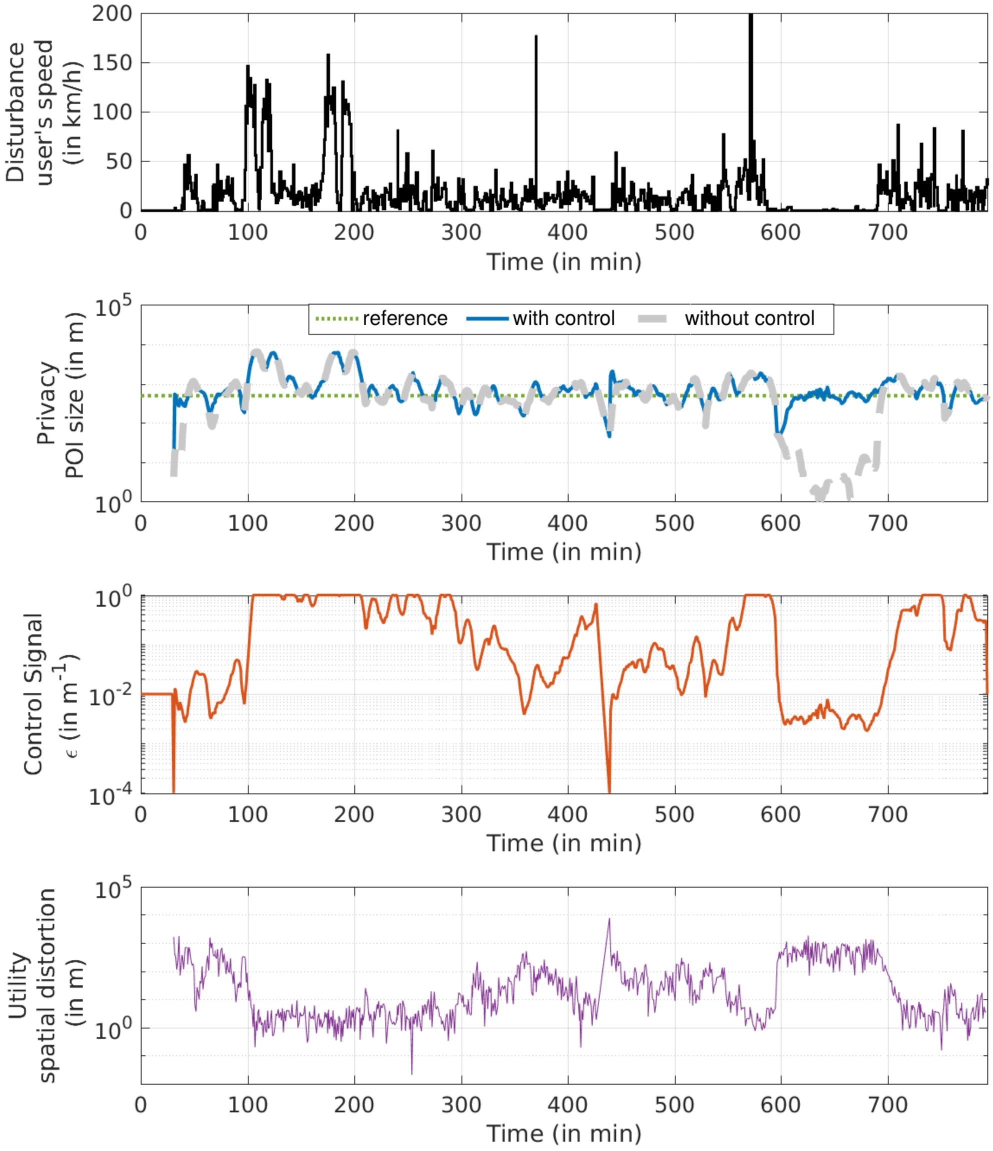


Figure 7.18 – Dynamical privacy control on a real mobility trace, with constant privacy specifications.

The same mobility trace is then considered with a varying reference scenario, see Figure 7.19. The controller performs significantly better than the non monitored scenario. Negative amplifications are very limited, and few oscillation are notable in the steady state values. The performance indicator is of 0.5%, which is significantly higher than the previous cases. Utility is however way worse, due to the high privacy requirements. The median is kept around 136 m, but the 99^{th} percentile is high: 6.6 km.

To conclude, the controller performs manage to keep track of the desired privacy level no matter the mobility scenario. However its performance in terms of reaction time and disturbance amplification are limited and could be improved, for instance with the use of a feedforward controller than would anticipate on the impact of the user mobility on privacy. Regarding utility, the controller maximizes it when the privacy is naturally reached, however an optimal controller would enable to take this specification more formally into account.

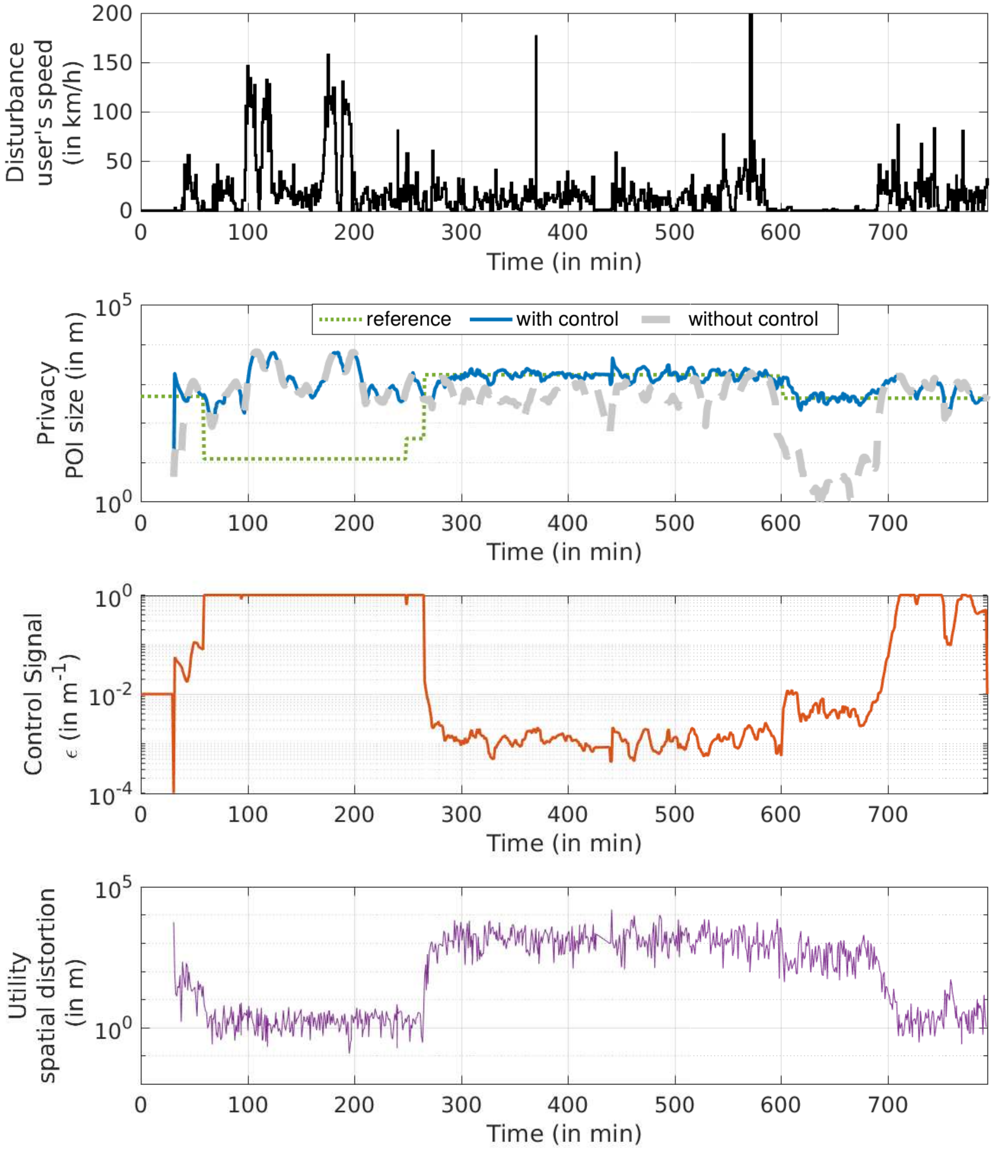


Figure 7.19 – Dynamical privacy control, real mobility trace with a privacy specifications scenario.

7.6 Conclusion

This chapter focused on the use of Location-Based Services through time, i.e. the user's location is sent periodically. This scenario is particularly sensible to privacy attacks if a malicious agent has access to the real-time position of the user. The focus has been made on the user's points of interest, a low level indicator of habits and identity. Location Privacy Protection Mechanisms (LPPMs) have been developed in order to address those threats. However, the use of LPPMs comes with an unavoidable reduction of the service utility, as informations are degraded to ensure privacy. Three issues remain open with state of the art LPPMs: (i) the usability by a non expert, especially regarding LPPMs' configuration, (ii) the possibility to change one's privacy requirements through time, and (iii) the robustness to users' mobility specificities.

In this thesis a control-based approach is proposed that enables a user to control their privacy when using such protection mechanisms while keeping an eye on utility, regardless of the user mobility behavior. Contributions are on the novel problem formulation and particularly a definition of real-time Points of Interest oriented privacy metric, on the modeling of the system and on a first PI control strategy. Evaluation was carried out both in simulation and using real data collected on the field. Results consist in a proof of concept that highlights the relevance of the control-based formulation and the efficiency of the controller to regulate privacy in a utility aware way.

Limitations of this work are on the transient efficiency of the controller, that we would like to act faster and with less overshoot. Moreover, the evaluation illustrates the possibility of using dynULP is multiple scenario, but do not consist in a proof of usability on all users and in all situations. The future of this work should thus be oriented in two directions. First its extensive evaluation using data collected from many real users with all possible mobility patterns. This large scale work requires a deeper work regarding the performance indicators of the control, to be able to significantly aggregate all the experiments results. Second, more advanced control techniques could be used to tackle the PI limitations. One can for instance add a feed-forward action on top a the feedback one in order to anticipate for the impact of the user' movement on the privacy level, and this react faster. A optimal controller could also be use in order to better manage the trade-off between utility and privacy.

As a conclusion, dynULP is a promising initial work that opens the door for deep research on the dynamical control of privacy and utility in the context of user-centric location privacy.

Chapter 8

Conclusions on Location Privacy

This ending chapter of Part II highlights this thesis contributions with regards to the Location Privacy domain, its limitations and its perspectives.

The use of mobile devices has enabled the development of Location-Based Services, and the collection and computation of location databases. The benefits of such resources are vast and cover all the spectrum of the computing world, from the smartphone user to the data scientist. However, location data are highly sensitive information for the user from which they have been recorded. It is possible, for instance, to discover someone's home or work place using mobility data, and more broadly any *point of interest* of the mobile user. During the last decade, awareness has be raised on such privacy considerations, both coming from the people and the institutions, that enforce more and more data-privacy regulations. While stopping the use of mobile devices is indeed a privacy-protecting action, computer scientists have developed more moderated solutions that aim at ensuring to the users the services while being privacy-compliant. Those so called Location Privacy Protection Mechanisms can be used either at the individual level, on a mobile device, or in the case where the data have already been collected and need to be protected, at a database level. In both cases, the question of the LPPMs usability raises. As pointed out earlier, there exist an inherent trade-off between privacy and utility, that has made most LPPM tunable. The smartphone users do want to be protected but do not necessary want to know *how* or have to configure the tool whenever they require higher utility (in case of a sudden strong service need) or higher privacy (when the service in use do not require precise location). Indeed the specificities of the user movement also make her more threatened at some moments than at others, a proper protection mechanism should be able to take this into account. From the data analyst point of view, the literature on LPPMs is so broad that the adequate choice for each individual of the database is a real challenge. And once again, the question of the configuration of the LPPM itself raises.

The two contributions of this part tackle those challenges. dynULP has been designed as a high abstraction level tool for people carrying mobile devices. The users just have to define a desired privacy level, that they can change through time, and dynULP realizes the data obfuscation before sending the location to the LBS. This solution relies on a control theory formulation of the location privacy challenge, that enables the user to be automatically protected through time, with a controller monitoring at each instant the her privacy level. *PULP* is a tool for the data scientist. It processes databases and recommends a LPPM and its configuration for all the database's users individually. *PULP* framework is objective-driven, which enables the database owner to have guarantees on the privacy and utility levels achieved in the obfuscated database.

The two systems are complementary, as they cover all the aspects of the location privacy issue. *PULP* can be either used on databases that have already been recorded and for which dynULP cannot be applied, or in combination, jointly dealing with all the sources of malicious usages of location data, see Figure 4.2 on page 32.

Some limitations still persist. First, the notion of privacy used in this work is related to Points Of Interest, which is a notion that requires two aspects that have to be fixed: the size limit of the POI and its minimal duration time. Even if this parametrization enables to distinct a stop at a traffic light from a night spent at home, they can still be hard to set precisely according to each one concept of its privacy limit. The second limitation comes from the restriction of the privacy concept to the notion of POIs. If only one definition has to be taken, it is indeed a relevant choice as the POI extraction is most of the time a preprocessing phase to more complex privacy attacks. However, this one notion do not cover all the complexity of the privacy challenge. Seemingly, the utility metric that was used (spatial distortion) is also a low level one, and is thus easily generalizable. However, specific services require specific informations regarding the mobility trace, that may not be preserved by focusing only on this utility notion.

This being said, the perspectives of the works are the following. First, awareness on the privacy threats have to be raised, more specificly on the time aspect of the issue: users have no control on data that have already been broadcasted, and the accumulation of data about an individual makes her even more vulnerable. On a more technical aspect, more metrics could be used to precise the notions of privacy and utility. Predictability of the next locations or probability of re-identification are two common notions of privacy which could benefit to *PULP* and dynULP. On the utility side, one could investigate the use of service-specific metrics. For instance, in the crowd-sourcing application, the preservation of the inference of a map of attributes (for example a map of pollution levels) in presence of obfuscated data would be an interesting challenge for database analysts. On a lower level, the preservation of the user's speed would be of great interest, for instance for navigation applications. Eventually, we aim at implementing dynULP as a smartphone application, as its low computing algorithm would enable it to be embedded. *PULP* framework could also be spread in companies, for instance to ensure their GDPR compliance [79].

As a conclusion, *PULP* and dynULP works are two solutions that tackle the location privacy applicability challenge from two complementary approaches, useful both for everyday smartphones users and data scientists.

Part III

Performance and Reliability of Hadoop Cloud Services

The second area of contribution of this thesis is on the performance monitoring of BigData cloud services. Tremendous amounts of data are generated everyday, and their processing is extremely valuable for companies and for the society. MapReduce and its open source implementation Hadoop have emerged as state of the art frameworks to realize BigData processing. However, data analytics and hardware management are two distinct computing area, requiring different expertise. This context has foster the emergence of the cloud paradigm, enabling the on-demand use of cluster resources for any type of applications or services. Indeed, many issues raise from this new configuration such as services performance monitoring, availability, cost-efficiency, guarantees, robustness, etc. This part investigates the use of advanced control theory tools to tackle those challenges.

Chapter 9 provides the required background about cloud services and the MapReduce framework and provide real-world examples. The key challenges regarding BigData cloud services are then motivated. An overview of the state of the art solutions realizing MapReduce or cloud performance monitoring is given. A specific focus is made on the works based on control theory providing the necessary problem formulation and modeling basics, required to understand the following two contribution chapters. Eventually, the state of the art limitations are highlighted.

Chapter 10 presents the first contribution: the adaptive control of MapReduce. The presented solution consists in an automated tool for MapReduce performance monitoring (in terms of jobs response time) able to work in an environment perturbed by both the clients workload and the inherent variability of the platform and network. First, this problem is motivated by illustrating the limitations of the state of the art. Then, an adaptive feedback and feedforward controller is developed which, by re-learning on-line the MapReduce model, adapts the control strategy. The controller stability verification is provided, and its performance are evaluated both in simulation and on a real cloud platforms running BigData MapReduce jobs.

The second contribution, presented in Chapter 11, focuses on the monitoring of multiple contradictory objectives: jobs performance, service availability and cost-efficiency. An optimal model predictive controller is developed, and an event-based mechanism is added for cost-awareness. However, the event-based control related works lack of a proper solution for the cost formulation specific to the cloud resources usage and pricing. Thus, a new event triggering method is developed, that enables to better deal with the cloud costs. Stability and validation of the new controller is presented. Evaluation of its performance in simulation using a real-life BigData processing workload is carried out.

Eventually, Chapter 12 puts those contributions in perspective to the BigData cloud services needs and state of the art; limitations are drawn and ideas for future works are proposed.

Chapter 9

BigData Cloud Services: Background and Related Works

The widespread use of connected devices and sensors as well as the growing use of computing tools, both for the industrial and personal usage, generates massive quantities of data. Those vast datasets are highly diverse in forms and nature: DNA transcripts, connection logs, images databases, etc. The denomination BigData gathers such datasets that have the two following properties: large scale and unstructured data (in the classic informatics definition of the word, i.e. non relational database) [54]. Handling this new form of data is a real challenge both from the hardware and the software point of view. Those two aspects have tend to decouple in the past decade, leading to the surge of the so called Cloud Services. BigData processing has triggered the development of new computing paradigms, such as the well known MapReduce framework. Those two key notions are presented in detail in the next sections. Afterwards, the problem that will interest us - the performance monitoring of BigData Cloud services - is motivated. Related works are presented, and a special attention is drawn on the control theory based approaches, thus introducing the problem formulation and preliminary models and controls.

9.1 Cloud Services

Cloud paradigm is the shared use of computing resources as an online service. The resources can be either the hardware (called IaaS, Infrastructure as a Service); the platform, including for instance the OS (PaaS); a service, such as an email client (SaaS); or even web or mobile apps (called BaaS as Backend cloud storage is provided). The notion of Cloud Services raised in the end of the 2000s, with the release or announcement of first-class clouds, such as Amazon, Google or Microsoft ones in 2008. Since then, it has became one of the biggest computing trends, with for instance Microsoft spending 90% of its R&D budget on Cloud Strategy in 2011 [57]. The biggest cloud providers are Amazon Web Services [14] and Microsoft Azure [132], see the market share diagram of 2017 published by the Cloud Security Alliance [9] in Figure 9.1.

Regarding the reasons of adoption or not of the cloud paradigm, the same report surveyed that the first reason is the scalability, that allows to adapt the resources according to the workload. The other reasons are the costs reduction and the avoidance of the investment in capital, as the cloud providers apply a pay-as-you-go pricing policy. Eventually, public clouds are trusted as more secure, which consist in another driver of adoption of the cloud paradigm.

Regarding performance and guarantees, cloud providers often come with a Service Level Agreements (SLAs), a dedicated terminology that gathers all the commitments of the provider regarding its service. When those SLAs are more specific, one can also talk about Service Level Objectives, such as a mean workload execution time for example.

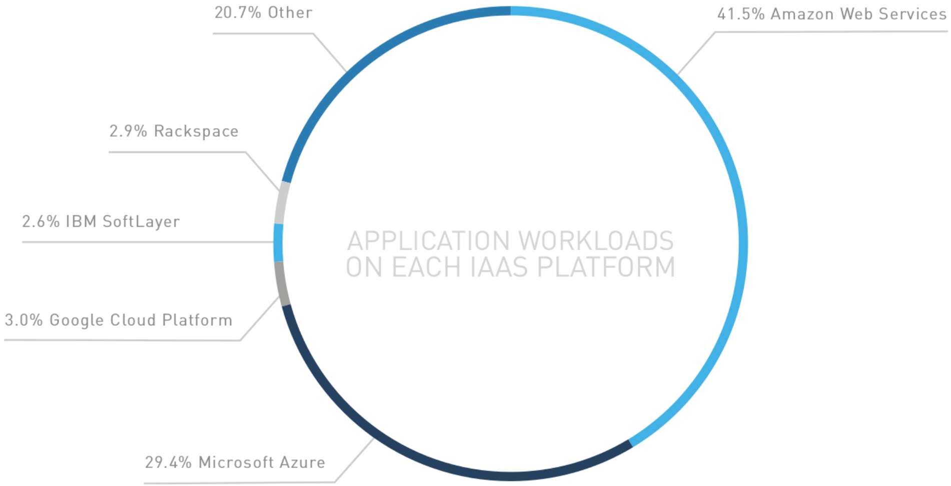


Figure 9.1 – Cloud IaaS platform adoption in 2017, percentage of applications deployed. [9]

9.2 A BigData Processing Framework: Hadoop/MapReduce

As BigData processing framework, we chose to use MapReduce under its open source implementation Hadoop. This choice has been motivated by the fact that MapReduce is one of the most popular parallel processing paradigms for Big Data systems currently in use [54]. It has been developed by Google in 2004 as a general parallel computing algorithm running on distributed platforms (such as clouds) that would automatically handle job and data management (data partitioning, consistency and replication, task distribution, scheduling, load balancing and fault tolerance [60]). Nowadays, numerous IT leaders such as Yahoo, Facebook or LinkendIn use versions of MapReduce. MapReduce is a general paradigm that aims at being able to treat extremely diverse requests, such as data mining or recommendation algorithms.

In order to better understand this new paradigm, a light analysis of how MapReduce works is presented here. MapReduce is a programming paradigm developed for parallel, distributed computations over very large amounts of data. The initial implementation of MapReduce is based on a master-slaves architecture, where the master node is in charge of task scheduling, monitoring and resource management and the slave nodes take care of starting and monitoring local mapper and reducer processes. For a user to run a MapReduce job, at least three elements need to be supplied to the framework: the input data to be treated, a Map function, and a Reduce function. To better understand how those functions that gave to the framework its name treat the data, we use the well know example of word count.

Data processing in done in five steps, see Figure 9.2 for an illustration.

(i) **Split**. The framework analyses the input data and convert it into <key,value> input pairs, that are sent to the computing nodes. In our word count example, the keys are for instance the text lines, and the values here do not really matter.

(ii) **Map**. In this step, an operation is done on the pairs, that are transformed into intermediate pairs. This operation is called mapping function. In this example, the mapper separates each word of the input key and associates the number 1.

(iii) **Collect and Sort**. Data are collected and sorted by key. Hadoop takes care that the pairs corresponding to a same key are sent to the same reducer (see step (iv)), even if several different

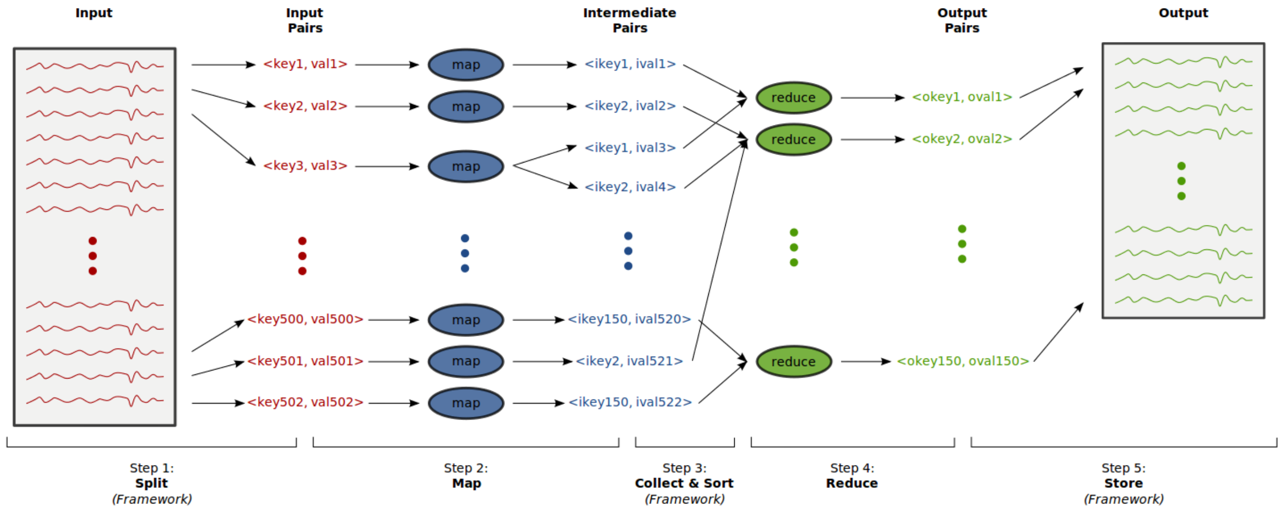


Figure 9.2 – MapReduce step-wise functional scheme. [60]

keys can be sent to a same reducer. For instance, all the pairs containing the same word are sent to the same reducer, but a reducer can treat several words.

(iv) **Reduce**. The reducer executes the second function: transforming the intermediate pairs in the output ones. For word count, the reduce function is to sum up the number of iteration of a word and return pairs with unique key and the value corresponding to its occurrence count in the entire text.

(v) **Store**. Output pairs are then wrote back in a file.

The main criticism against MapReduce is regarding the restriction of problem that it can address using its current formulation. For instance, treatments on image databases are not a suitable MapReduce problem. Advances in performance and implementation efficiency have been done since the first release of MapReduce, see Hadoop 2 (YARN) and Hadoop 3 for instance. Even if this work is developed and validated on Hadoop 1, the high level problem formulation enables to apply its principles on versions 2 and 3.

To conclude, MapReduce/Hadoop is a computing framework that enable the processing of unstructured data where the classic SQL-based approach have failed or shown too slow results. Its wide adoption in the industrial world makes it a attractive use-case for this thesis.

9.3 Problem Statement

This section highlights the challenges that rises from the use of the MapReduce framework on public clouds, see for instance Amazon EMR [15] or Azure HDInsight [133].

Public clouds are open to everyone: multiple clients can send requests at the same time, thus generating a concurrency for the access to specific resources such as network, which influences the service performance. A single client also launches multiple MapReduce jobs, that concurrently access the cluster nodes. If multiple concurrent jobs are running, the amount of resources allocated for each job is reduced and thus the job service time is low. However, the distribution of those jobs is not necessary known in advance. For instance, for an e-commerce company running MapReduce to perform sale recommendation based on client preferences, the workload is directly linked to the website frequenting which varies in time. Moreover, the jobs are highly heterogeneous between the different clients, and can also be so between the pool of a client's jobs. The workload of the system is independent of the cloud provider and can be really brisk [83], it can thus be considered as a disturbance impacting the system performance.

For a client using MapReduce, it is essential to have guarantees in term of service performance, dependability and costs. In the example of the recommendation task, the company needs to have the list of recommended products in a few tenth of seconds, otherwise their webpage will take too much time to be loaded. Poor jobs performance lead to end-user performance degradation and eventually lost of loyalty, and thus need to be avoided [137]. From the cloud provider point of view, this service time (the time it takes for a user request to be treated) is thus a performance metric that should be included in its Service Level Agreements. Aside from the performance perspective, availability of the services must be ensured, both for the MapReduce client that wants to ensure that most of its tasks are executed (and thus most of its end-users benefit from a service), and for the cloud provider, that needs all its clients to be served. A last objective is the cost efficiency. From the provider point of view, running a MapReduce service has a cost in terms of energy consumption but also hardware usage, wear and maintenance, which results in financial costs. Regarding the MapReduce clients, the pay-as-you-go pricing strategy of clouds also translates into resource minimization objectives. Obviously, all those objectives of performance, availability and cost efficiency are contradictory: a trade-off that satisfy both the cloud provider and the clients have to be achieved and ensured.

However, the guarantees formulated by the cloud providers in their SLAs are mainly focused on availability and dependability of their resources, i.e. their hardware, rather on what is of interest for their clients: the application-level availability and performance. Regarding the MapReduce framework itself, it has been designed to maximize throughput, resource utilization and jobs performance, while the clients are more interested in the workload level performance. Even if those objectives are related, they are not directly equivalent for instance in case of job concurrency. Here again, the objectives must be shifted from low level ones to service ones.

We now investigate the reasons that make this difference between resources/jobs quality and service performance. First, a cloud is a highly dynamic environment, in which the hardware is constantly changing, both at short term (connection/shutdown of new nodes, resources concurrency, network bottlenecks [113], etc.) and longer term (addition of new types of resources for instance). All those changes through time are hard to measure and sometimes come from unknown sources, due to the complexity of nowadays cloud infrastructures. As cloud providers desire to maximize the utilization of their clusters, they have mechanisms for the dynamic reallocation of unused resources in the cluster, which further add to the variability of system performance. Hence, even with the same workload and the same resource amount, an application performance may vary depending on how noisy neighboring applications are. Moreover, cloud services are highly complex paradigms that run on top of multiple software stacks and on heterogeneous nodes [19], making their behavior regarding resource provisioning highly non linear. For a software engineer, these unknown variation are particularly difficult to deal with. On top of the environment and workload dynamics, the MapReduce itself is evolving through time. At a low scale, the realization of Map and Reduce functions are not all synchronous, leading to slight variations of the large scale system behavior. The framework is also not safe from software failures [159]. Moreover, as stated before, many new versions and updates of the MapReduce Hadoop framework are released, changing every time a bit MapReduce performance.

A simple action that can be done to modify the services run time is to play with the number of resources of the cloud allocated to the jobs. For the MapReduce use case, adding resources, commonly known as *nodes*, to the cluster will increase the number of Map and Reduce functions processing the input data leading to a reduction of the service time. And conversely if the number of resources goes down. The cluster size consists then in a control knob that can influence both the service performance and indeed the costs. Another impacting factor is the admission control, i.e. the monitoring of the percentage of accepted jobs/clients. This variable impacts service availability first, but also performance (the less clients are accepted, the more resources are dedicated for the selected ones) and costs (less clients benefit from the service). Cluster size and admission control can thus be used as control knobs to monitor service performance, availability and cost.

To sum-up, MapReduce cloud services performance, availability and cost-efficiency needs to be ensured despite the dynamical workloads and the environment unpredictabilities. Those objectives could be tackled with proper resource provisioning and admission control.

9.4 Related Works

Many works have been done on the resource management, the monitoring of cloud platforms performance and on supervision of MapReduce framework. We focus in this section on giving an overview of the various techniques found in the literature, while next section details the control theory based approaches.

9.4.1 Cloud Performance Monitoring

Most of the works that aim at ensuring high performance in a cloud environment focus on the resource provisioning challenge. As their objective is similar, we categorize them by the technique they used.

The vast majority of the state of the art uses observation based rules. These are also the works that are the most implemented in large scale industrial solutions. Amazon Auto Scaling [13] is one of such examples where a load metric, such as CPU utilization, is monitored and used to scale up or down the cluster. However, there are still parameters to tune, for instance regarding thresholds, that requires the user to be well aware of its service behavior. More advanced rules-based techniques that use for instance fuzzy logic systems have also been developed [95, 94].

Queuing theory is also a famous technique for cloud control [7, 8]. AutoScale [76] is a solution that use queuing theory analytic models to decide capacity requirements by reacting to unpredicted changes in request rate, request size and/or server efficiency. Queuing theory has also been coupled with control theory, in the form of a integral discrete controller with anti-windup action, to realize load balancing and service time control, using a graceful service degradation when needed [143].

A different approach consists of using predictive techniques to estimate the future load of a service and provision accordingly, for instance using wavelets [141] or fluid modeling [124, 169].

Indeed, machine learning techniques can also be found, see for instance by Matsunaga et al. [129] to predict the time and resources consumed by applications, or by Gong et al. [85] that realize on-line resource demands predictions.

9.4.2 MapReduce Performance Monitoring

We focus now on the works that control MapReduce runtime performance using resource allocation and admission control.

Verma et al. presented ARIA [172], standing for automatic resource inference and allocation. It is an online mechanism that acts at the job level, and which is driven by runtime objectives for the jobs. Ferguson et al. have developed Jockey [70], a framework that realizes resource provisioning in order to ensure job performance objectives. It also aims at minimizing runtime costs and reducing the concurrency between the jobs. DREAMS [118] is a framework that deals with the uneven map tasks distribution among reduce tasks as it causes overhead of data repartitioning. This objective is achieved by adjusting task run-time resource allocation. MRCP-RM [114], standing for MapReduce

Constraint Programming based Resource Management is a resource allocator and scheduler. It considers that MapReduce jobs arrives online and with each a SLA consisting in time deadlines. It sees the system as an optimization problem and uses a constraint programming formulation. One can also mention the work of Zacheilas et al. [186] in which the trade-off between performance and budget is explored using a Pareto-search algorithm for scheduling and MapReduce configuration parameters optimization. CSAM-IISG [181] is one of the latest works on MapReduce resource allocation challenge. It novelty lies in its imperfect information Stackelberg Game formulation using hidden Markov models. It works online and enables to deal with multi-service providers and various resources types. Eventually, one can also refer to Cano's works on joint resource allocation for combines web services and MapReduce jobs [35], for instance in the case of a e-commerce website realizing business intelligence. The objectives are formulated as upper-bounds on the average response times of jobs and services. It considers a fixed-size private cloud cluster and thus plays with admission control and resource management. Their solution uses a non linear mathematical programming model and they find a greedy solution. Note that their framework can deal with heterogeneous workloads.

Most of the presented works realizes resource allocation or optimization at a low level, i.e. with objectives on the jobs or tasks, while our objective is to cover the higher level perspective: the whole jobs workload. This difference in the objective scale makes most of those approaches compatible with ours. Moreover, the state of the art solution are often intrusive, whereas our control theory approach aims at tuning high level parameters with an overall outside controller.

9.4.3 Benchmarking and Platforms

Technical and practical issues are detailed in this section, regarding first the MapReduce benchmark and workloads used, and then the cloud platform hosting it.

We selected the MapReduce Benchmark Suite (MRBS) [159] as our BigData processing framework. MRBS is a performance and dependability benchmark suite for MapReduce systems. It has the specificity of being able to emulate several types of workloads and inject different fault types into a MapReduce system. The workloads emulated by MRBS are selected to represent a large range of loads, from the compute-intensive to the data-intensive (e.g. business intelligence - BI) workload. One of the strong suits of MRBS is to emulate client interactions, which may consist of one or more MapReduce jobs run at the same time. We selected a data intensive BI workload, which consists of a decision support system for a wholesale supplier. Each client interaction emulates a typical business oriented query run over a large amount of data (10GB here).

Other popular MapReduce benchmarks are MRBench [101], MRPerf [176] or PUMA [4]. MRBench processes large volumes of relational data and executes highly complex queries. It addresses the specific issues of business oriented queries and concurrent data modifications. It has been made to help tune MapReduce parameters, such as data size and the number of (Map/Reduce) tasks. MRPerf is a design tool for MapReduce infrastructures. It is however not a properly speaking a benchmark but rather a simulator that reduced the number of parameters to simplify the configuration. PUMA is a MapReduce benchmark suite able to run a large range of applications with various characteristics, such as high/low computation and high/low shuffle volumes. Compared to its competitors, MRBS has the advantage of simulating workloads with multiple clients, with realistic client interaction. It is the reason why we opt for the MRBS benchmark.

Regarding the cloud platform, we used Grid5000 [36], a French nation-wide cluster infrastructure made up of 5000 CPUs. Grid5000 is a large-scale and versatile testbed for experiment-driven research in all areas of computer science, with a focus on parallel and distributed computing. It has the advantage of being highly reconfigurable and controllable, with extensive monitoring and measurement capacities. Compared to its commercial competitor such as Amazon Web Services [14] or Microsoft

Azure [132], it has the advantage of being free and open access for researcher. It is also a mastered platform which enables reproducible research, while the industrial versions are constantly evolving, making them more suitable for evaluation purposes than for the research work itself.

The combination of the MRBS benchmark and Grid5000 platform have already been used in the BigData cloud control context. Berekméri [25] developed a tool to launch and monitor MapReduce jobs on Grid5000, combined with a Matlab interface that enables to design and implement monitoring algorithms, i.e. controllers. This experimental setup is used for the evaluation of the control strategies presented in this thesis.

9.5 Background on Control Theory applied to MapReduce

The control problem formulation for BigData Cloud Control has been first introduced by Berekemeri and al. in 2014 [24] and further extended in the following years [26, 25]. This section is a detailed review of those works, on which this thesis contributions are based. In a first time, the signals used as performance indicators and control knobs are defined. Second, dynamical models of their impact on each other are introduced. Those models lead to the development of controllers, which are presented in two categories: the feedback and feedforward based runtime monitoring, and the multi-dimensions optimal control of runtime and availability.

9.5.1 Input and output signals

The output metrics of interest for our control problem are:

Service Time. Noted y_{ST}, it is the average run time of jobs that finished in the previous time window. The lower the service time is, the faster the clients requests are completed. Depending on the MapReduce tasks, it can go from few seconds to several minutes or even hours. The size of the size window is 15 min, as fixed in [25] for the BI applications.

Availability. It is the ratio of accepted MapReduce jobs over all received jobs, during the last time window, refereed to with the notation y_{AV}. Availability is thus between 0 and 1, this later value meaning a 100% availability of the service, the best value achievable.

The input signals - control knobs that can be varied or external disturbances the service undergo - impacting those performance metrics are:

Cluster Size. One simple action that can be done to control the on-line service time is to modify the number of resources of the cloud allocated to the jobs. For the MapReduce use case, adding resources, commonly known as nodes, to the cluster will increase the number of Map and Reduce functions processing the input data leading to a reduction of the service time. If the number of resources (nodes) diminishes, the results are converse. The cluster size is then one of our control signal u_N.

Workload. In the case of public clouds, multiple clients send requests at the same time thus generating a varying input workload which influences the service performance. If multiple concurrent jobs are running, the amount of resources allocated for each job is reduced and thus the job service time increases. As the workload of the system is independent of the cloud provider we will consider it as a disturbance d_C.

Admission Control. Client requests can be rejected, when the available resources are considered too few to satisfy all the incoming workload for instance. Admission control is thus another input variable for our control, it is the number of accepted clients u_{AC}. Note that there is no more accepted clients that the demand: $u_{AC} \leq d_C$.

Those input variables are linearized around a working point of 10 clients and 20 nodes [24]. Hence, the signals only express variations.

9.5.2 Modeling

The modeling of the service time and availability are presented independently and afterwards gathered into a MIMO formulation. We proceed like this as some of the controllers will only take into account service time monitoring, and thus only SISO modeling.

9.5.2.1 Service Time Model

A dynamic model that predicts MapReduce cluster performance (i.e. the average service time of the last finished jobs) with respect to the number of nodes of the cluster and the number of clients sending requests was proposed and experimentally validated by Berekméri et al. [24]. Its schematic representation is given in Figure 9.3.

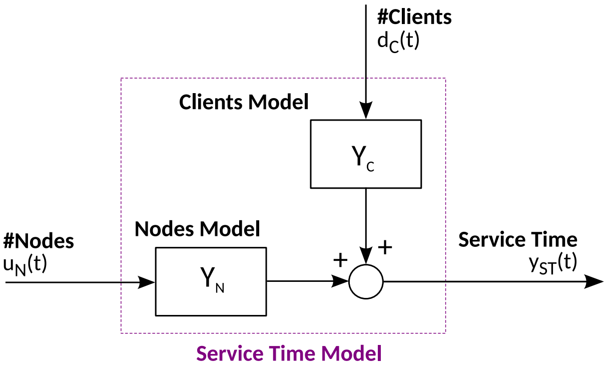


Figure 9.3 – MapReduce Service Time model.

The system is considered linear in its operating region. The models were first identified as continuous transfer functions and then discretized using the sampling period $T_s = 30\ sec$. The resulting equation is[2]:

$$y_{ST}(t) = Y_C(q^{-1})u_{AC}(t) + Y_N(q^{-1})u_N(t). \tag{9.1}$$

$Y_C(q^{-1})$ is the direct path transfer function, the link between the number of clients and the service time performance metric, and $Y_N(q^{-1})$ represents the compensatory path that enables to modify the cluster size to control the service time. Both models were identified as first-order transfer function with delays, as described in eq. (9.2):

$$Y_C(q^{-1}) = \frac{b_C q^{-1}}{1 + a_C q^{-1}} q^{-r_C}, \quad Y_N(q^{-1}) = \frac{b_N q^{-1}}{1 + a_N q^{-1}} q^{-r_N}, \tag{9.2}$$

with $b_C = 1.0716$, $a_C = -0.7915$, $r_C = 8$, $b_N = -0.17951$, $a_N = -0.919$ and $r_N = 5$.

Note that when there is no admission control, $u_{AC} = d_C$ in eq. (9.1), as illustrated in Fig. 9.3.

9.5.2.2 Availability Model

Availability is impacted by the number of clients rejected by the MapReduce framework. It is thus a combination of the workload, i.e. number of demands d_C, and the upper acceptance threshold u_{AC}.

$$y_{AV}(t) = Y_{AC}(q^{-1})[d_C(t) - u_{AC}(t)] = \frac{b_{AC} q^{-1}}{1 + a_{AC} q^{-1}}[d_C(t) - u_{AC}(t)]. \tag{9.3}$$

Parameters of the availability model are $b_{AC} = 0.1548$, $a_{AC} = -0.946$ [25].

[2] The complex variable z^{-1} will be used to characterize the system's behavior in the frequency domain and the delay operator q^{-1} will be used for the time domain analysis.

9.5.2.3 MIMO Model

The multi signal representation of MapReduce modeling combining the previous models is represented in Figure 9.4. A MIMO linear discrete model can be expressed in the following formulation:

$$\begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k, \end{cases} \tag{9.4}$$

with y_k the output vector at the k^{th} sampling time: $y = (y_{ST}, y_{AV})^T$; u_k the vector of inputs (both controllable and uncontrollable) $u = (u_N, d_C, u_{AC})^T$ and x_k the states vector. Starting from the SISO formulations of eqs. (9.1) and (9.3), there exist automated tools to derive the non unique MIMO formulation of eq. (9.4). Using Matlab *tf2ss* function, the number of states is 4. Those states have no concrete physical explanation, we thus do not detail them here.

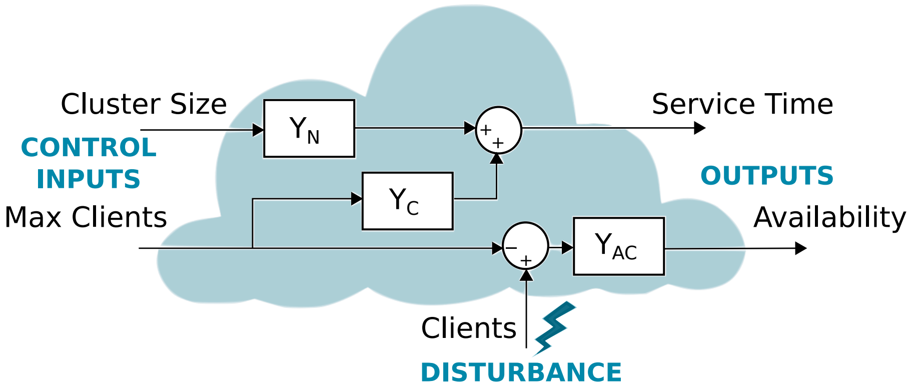


Figure 9.4 – MapReduce Service Time and Availability model.

Some work has also be done on the adaptation of those models online, both for the service time modeling [39] and on the MIMO model [40]. Those works use a Recursive Least Square Estimator to update the models parameters at each execution time, depending on an estimated gain and a covariance matrix. The adaptation of those models are used to cope with the non-linearities of the system while keeping a linear model. However, those models have shown limited results when coupled with a control strategy, as the estimation optimization in those works is toward the model accuracy, not the control performance, contrary to the work presented in Chapter 10.

9.5.3 Control

Several control laws have been developed: a feedback and feedforward controller for the service time monitoring [26], and a Model Predictive Controller for the joint service time and availability control [25, **?**].

The service time controller time-domain equation is given in eq. (9.5):

$$u_N(t) = PI(q^{-1})[y_{ST,ref}(t) - y_{ST}(t)] + FF(q^{-1})d_C(t), \tag{9.5}$$

with $y_{ST,ref}(t)$ the reference value for the service time. The PI equation is classically $PI(s) = \frac{K_I}{s} + K_P$, with s the Laplace variable and the values $K_I = 0.25584$ and $K_P = 0.0012372$. The feedforward control is in fact a static one, i.e. only a gain $FF(s) = 5.96958$ [26].

A event-based version of this feedback and feedforward control has also been developed [26], that triggers the control signal update based on the service time error rather that in a regular time-based fashion.

The multi objectives control law, called *MR-Ctrl*, is a model predictive controller that uses the following cost function:

$$J = \min_{U_k} (Y_k - Y_{ref})^T Q (Y_k - Y_{ref}) + U_k^T R U_k, \tag{9.6}$$

where $U_k = (u_k, u_{k+1}, \dots, u_{k+N})^T$, $Y_k = (y_k, y_{k+1}, \dots, y_{k+N})^T$ and $Y_{ref} = (y_{ref}, \dots, y_{ref})$ are the selected/predicted values of the signals on the optimization horizon of N samples. y_{ref} is the vector of reference values for the system outputs, fixed here: $y_{ref} = (y_{ST,ref}, y_{AV,ref})^T$. This control law is submitted to the following constraints:

$$\forall k, 0 < u_{AC}(k) \leq d_C(k) \text{ and } u_N(k) \leq u_N^{max}. \tag{9.7}$$

Integral action and compensation for actuation delay are also added to cope with modeling errors and control oscillatory behavior. States are estimated using the Luenberger observer [25].

9.5.4 Limitations

The two main limitations of the previously presented control of MapReduce are regarding robustness and cost-efficiency, as illustrated in due time in Section 10.1 and Section 11.1.

An analysis of the cloud system behavior detailed in Section 9.3 intuitively shows that the systems is non linear and time variant, thus the previous linear model is not accurate for a wide range of input or disturbance values, so neither is the control law. Therefore, Chapter 10 develops an adaptive technique to cope with system and environment uncertainties.

Moreover, the *MR-Ctrl* controller do not take into account the specificities of the cloud-based system regarding costs. For instance, resources in a cloud are rented and priced based on the time they where used, but with a minimal threshold that usually correspond the time needed to start or initialized them. To tackle this cost formulation, a new event-based optimal control law is presented in Chapter 11.

Chapter 10

Adaptive Control for Robust Cloud Services

This chapter presents a control theoretical approach aiming at controlling cloud services performance in terms of average run time, using the on-demand assigning of a varying number of shared hardware resources to the applications, called elastic resource provisioning. The selected use-case cloud application is the MapReduce framework. Given the system non-linearities and uncertainties highlighted in Chapter 9 and illustrated in Section 10.1, an adaptive feedforward controller is developed and combined with a feedback action (see Section 10.2 for the control development). The controller adaptation is based on the on-line estimation of the models parameters, derived from the observation of the system output. A stability analysis of the adaptive controller is provided. Simulations and experimentations on a real MapReduce benchmark running on a nation wide cloud infrastructure illustrate the performance of the system under various workload conditions (Section 10.3). Moreover, the use of adaptation significantly improves control efficiency and robustness with respect to variations in both the static and dynamic behaviors of the plant. Conclusions and perspectives of this work end this chapter (Section 10.4).

10.1 Introduction

10.1.1 Context and Approach

Current autonomic resource provisioning approaches that are deployed in public clouds don't work well for real time applications. The difficulty of the task partially comes from the fact that the optimal configuration can vary due to workloads fluctuations over time [20]. Indeed, some applications such as business intelligence processing do not have a constant number of tasks to treat.

The cloud service behavior itself varies over time due to the dynamic of its environment (hardware, network, etc. [103]). Cloud providers use mechanisms to reallocate the available resources to other applications (other clients or internal processing), in order to maximize the usage of the cluster's resources. On top of disturbing the physically close resources and applications (due to network bottleneck for instance), it also creates a variability for the applications running in those reused resources. Hence, even with the same workload and the same resource amount, an application performance may vary depending on how noisy and fluctuating neighboring applications are. Moreover, cloud services are highly complex paradigms that run on top of multiple software stacks, making their behavior regarding to resource provisioning highly non linear.

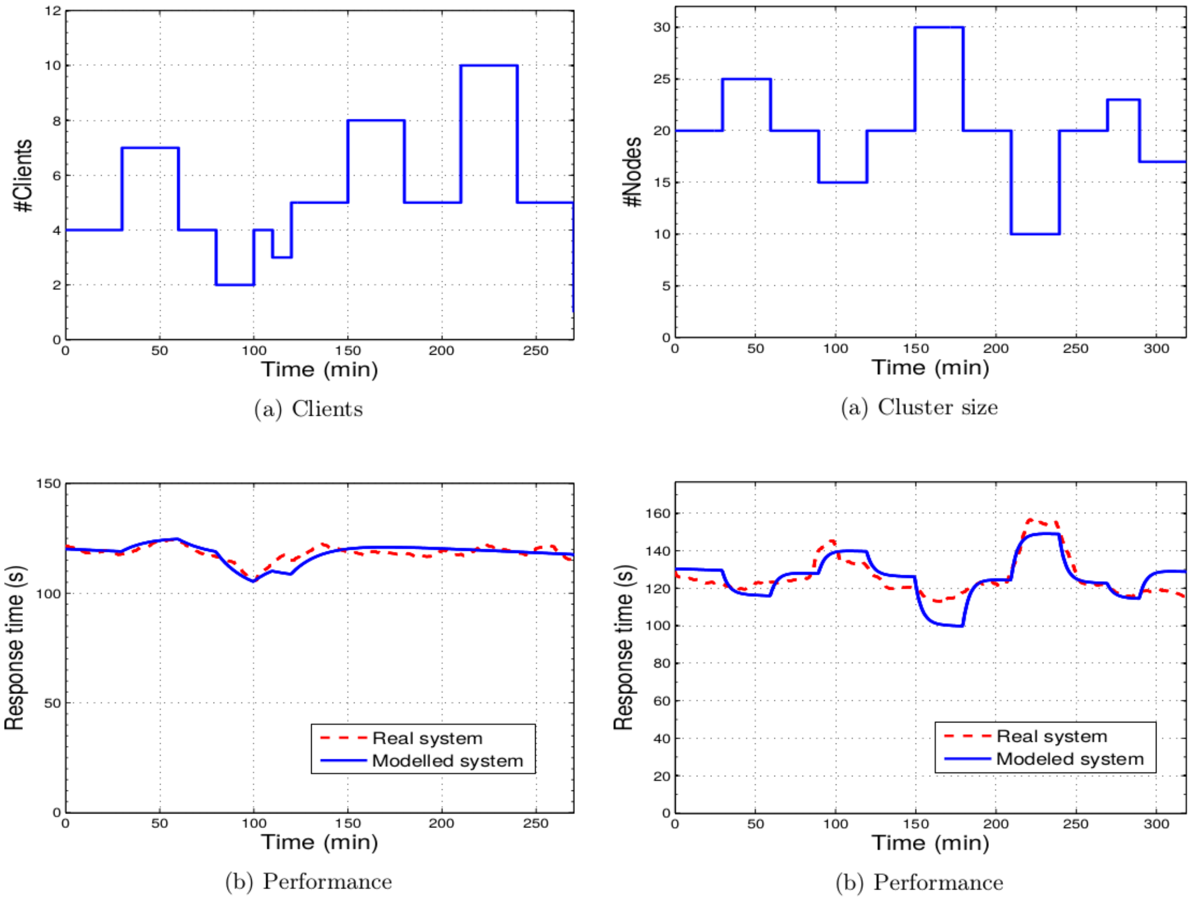
Therefore, there is a need for fully automatized and robust cluster scaling algorithms able to deal with those workload disturbances and system variability. The challenge looking into this issue of resource provisioning from a robustness point of view, for instance through control adaptation able to take into account system configuration and environment variations, is new in the state of the art.

In this chapter, we aim at providing a solution able to be efficient no matter the workload and framework state. The approach developed in this paper is control theory-based. The controller is composed of a reactive part (a feedback PI controller) that ensures steady state system convergence; and a predictive feedforward controller to ensure disturbance rejection. Our control is made robust with the addition of an adaptation algorithm that estimate the feedforward gain through time based on the system output observation.

10.1.2 Illustrated Motivation

This section aims at illustrating two points: first that the state of the art dynamical modeling of MapReduce fails to capture the system behavior, especially when the system goes beyond its linearity range; and second that the current controllers have limitations when tested in practice.

Figure 10.1 (taken from Berekméri's thesis [25]) is the comparison of the measured MapReduce service time y_{ST} compared to the prediction computed with eq. (9.1). The left plots (A) illustrate the impact of workload variation on the service time with fixed cluster size. Note that the maximum number of accepted clients is fixed to 5, the workload has an impact on the performance only in the first half of the plot. The fitting of the model predictions with the measures in this case is good. The right plots (B), the size of the cluster is periodically varied, with always the same period of 30 min. The model fits the measured with low accuracy, both in static value (from 150 min to 180 min for instance) and in dynamics (see at 90 min). Major differences are seen when the cluster size is far from its nominal value of 20 nodes (± 10 nodes).



(a) Clients

(a) Cluster size

(b) Performance

(b) Performance

A. workload variation ($u_N = 20$, $u_{AC} = 5$) B. cluster size variation ($u_{AC} = 5$, $d_C = 10$)

Figure 10.1 – MapReduce State of the Art Modeling. [25]

State of the art control performance are evaluated in Figure 10.2. More details about implementation are available in Section 10.3.1, we anticipate here to give more insights of motivation for the reader. The controller consists in a static feedforward with PI feedback action [25]. Disturbance scenario here varies reasonably around its linearization point of 10 clients (we set $d_C = u_{AC}$). The controler ability to follow its reference is not always ensured. The controller starts by diverging (min 10 to 20) but then the initial client step up is well compensated. However, the following decrease of the workload makes the controller diverge. This behavior is only mastered after a new change in the workload helps bringing the system in a stable situation. However, the following decrease in client number makes again the controller barely stable, i.e. the service time highly oscillates. The poor performance can be explained by a workload scenario that is too far from the linearization hypothesis. Indeed, the controller was not developed to cope with ±50% of its nominal workload.

The non-trustworthy node model parameters as well as the poor performance of the classical controller, with more sophisticated disturbance scenarios than a step change, motivate the need for a robust adaptive control.

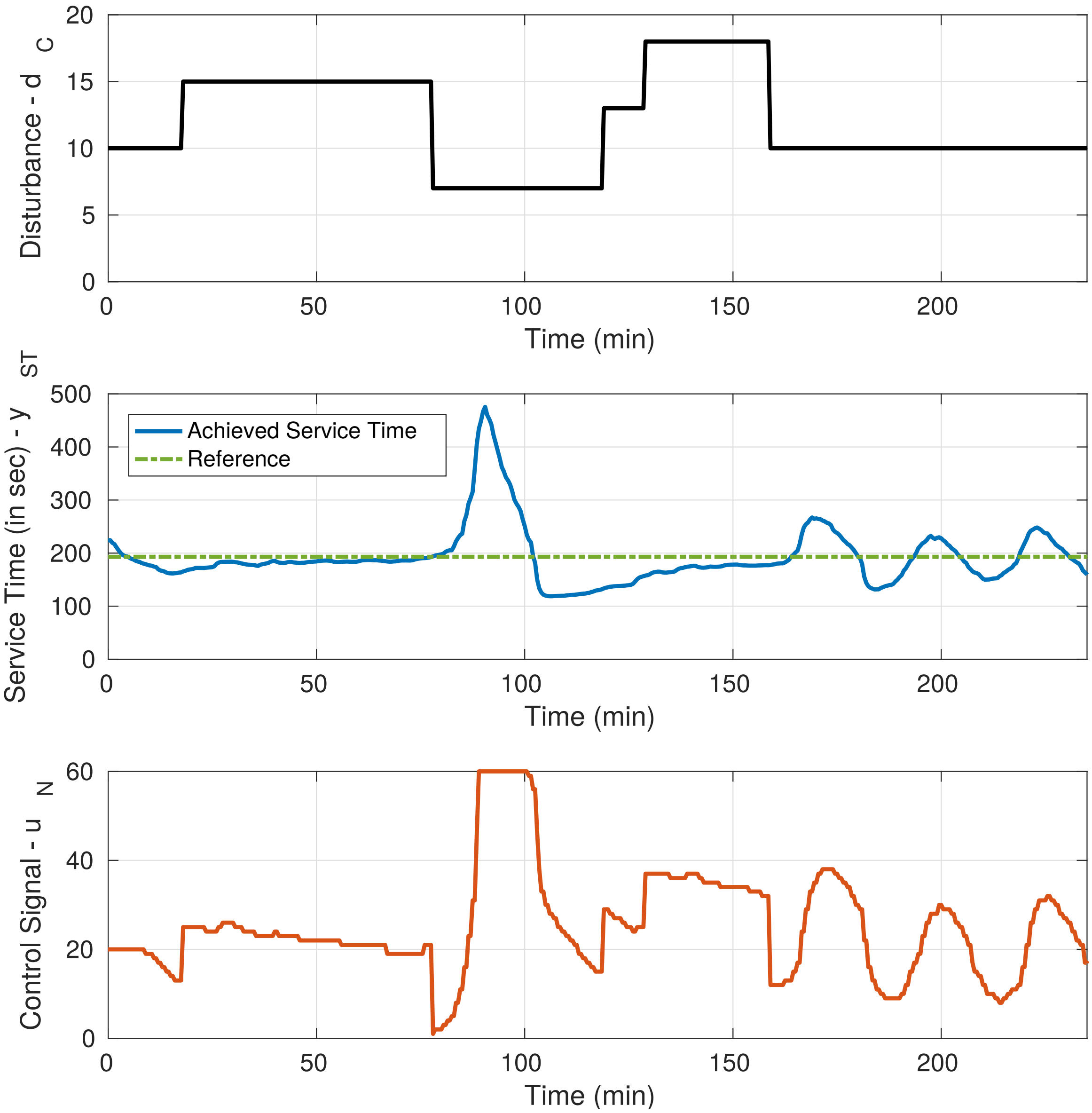


Figure 10.2 – MapReduce State of the Art Feedback and Feedforward Control experimental performance for a comprehensive client disturbance scenario.

10.2 Control Strategy

The control presented in this chapter consists of a PI controller, for asymptotic zero reference tracking error, and a feedforward loop, for dynamic disturbance compensation. A real time adaptation of the feedforward compensator is added to the state-of-the-art controller, in order to take into account the unknown and time-varying system's parameters. The parameter adaptation algorithm is based on the measures of the system outputs in reaction to client disturbances [111, 109]. The control scheme is given in Figure 10.3, with the previously seen MapReduce performance model in the light blue dotted box and the adaptation algorithm in purple.

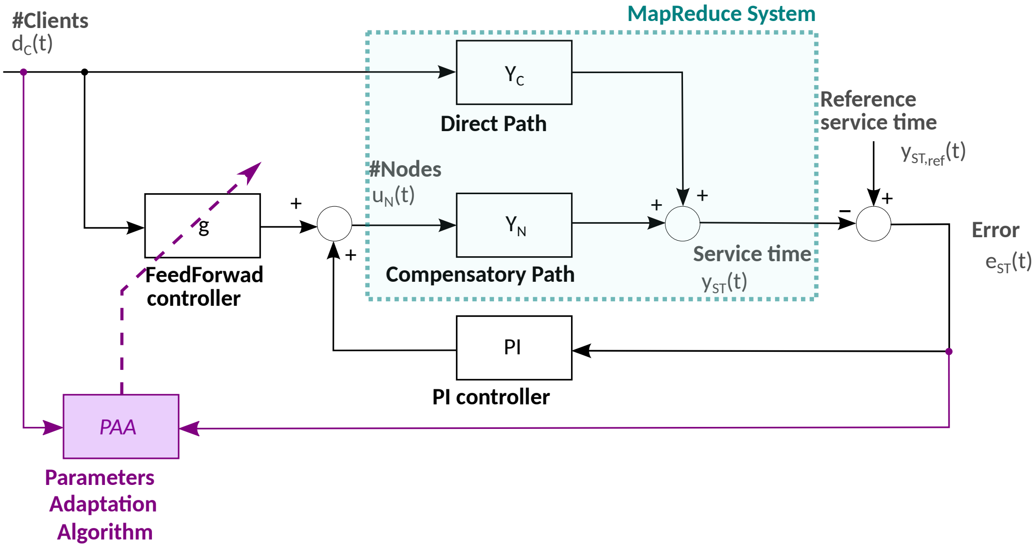


Figure 10.3 – MapReduce adaptive control schema.

10.2.1 Preliminary Formulation

Under the classical feedback-feedforward control loop (in black on Figure 10.3, without the PAA adaptation path in purple) and using the eq. (9.2), the optimal value of the 0-order feedforward controller is given by:

$$u^*_{N,ff}(t) = -\frac{b_C(1+a_N)}{b_N(1+a_C)}.d_C(t) \triangleq g^* d_C(t). \tag{10.1}$$

For the initial development of the adaptation algorithm, we will make the following hypotheses:

(H1) the effect of the PI controller can be neglected in front of the feedforward one,

(H2) equal dynamics ($a_C = a_N$),

(H3) equal and known delays ($r_C = r_N$).

Then the algorithm will be extended for the cases where hypothesis H1, H2 and H3 do not hold.

Let us define the adaptive feedforward control algorithm as:

$$u_{N,ff}(t) = \hat{g}(t) d_C(t) \tag{10.2}$$

where $\hat{g}$ is given by

$$\begin{aligned} \hat{g}(t+1) &= \hat{g}(t) + \alpha x(t+1) e_{ST}(t+1) \\ &= \hat{g}(t) + \frac{\alpha x(t+1)}{1+\alpha x^2(t+1)} e^0_{ST}(t+1) \quad \text{with } \alpha > 0. \end{aligned} \tag{10.3}$$

The filtered disturbance x is defined by:

$$x(t) \triangleq \operatorname{sgn}(b_N)F(q^{-1})d_C(t) = \frac{\operatorname{sgn}(b_N)q^{-(r_N+1)}}{1+\hat{a}_N q^{-1}}d_C(t), \tag{10.4}$$

where the *a priori* adaptation error $e^0_{ST}(t+1)$ is the value of the regulation error at time $t+1$ and is based on the estimation of $\hat{g}(t)$: $e^0_{ST}(t+1) = e^0_{ST}(t+1/\hat{g}(t))$. The *a priori* error $e^0_{ST}(t+1)$ is considered as a performance metric for our system. One can also define the *a posteriori* error $e_{ST}(t+1) = e^0_{ST}(t+1/\hat{g}(t+1))$ which, based on eq. (10.3), gives:

$$e_{ST}(t+1) = \frac{1}{1+\alpha x^2(t+1)}e^0_{ST}(t+1). \tag{10.5}$$

Theorem 1 *For the scheme represented in Figure 10.3, under the hypothesis H1, H2 and H3, using the adaptive feedforward control given by eq.* (10.2) *to* (10.4)*, one has:*

$$lim_{t\to\infty}e_{ST}(t+1) = lim_{t\to\infty}e^0_{ST}(t+1) = 0 \tag{10.6}$$

for any initial conditions provided that:

$$H'(z^{-1}) = \frac{1+\hat{a}_N z^{-1}}{1+a_N z^{-1}} \tag{10.7}$$

is a strictly positive real transfer function.

Note that this later condition is always satisfied provided that $|a_N| < 1$ (stability condition for the model) and $\hat{a}_N \leq 0$.

Proof: Under the hypothesis H1, H2 and H3, the expression of the error e_{ST} for a fixed value $\hat{g}$ is (see Figure 10.3):

$$e_{ST}(t+1) = e^0_{ST}(t+1) = \frac{b_N q^{-(r_N+1)}}{1+a_N q^{-1}}(g^* - \hat{g})d_C(t+1) \tag{10.8}$$

Filtering $d_C(t)$ by the filter F given in eq. (10.4) allows to rewrite eq. (10.8) as:

$$e_{ST}(t+1) = |b_N|\frac{1+\hat{a}_N q^{-1}}{1+a_N q^{-1}}(g^* - \hat{g})x(t) \tag{10.9}$$

When replacing $\hat{g}$ with $\hat{g}(t+1)$ and neglecting the additional vanishing term resulting from the non commutativity of time varying operators, one has:

$$e_{ST}(t+1) = |b_N|\frac{1+\hat{a}_N q^{-1}}{1+a_N q^{-1}}(g^* - \hat{g}(t+1))x(t) \tag{10.10}$$

Eq. (10.10) has the standard form of an *a posteriori adaptation error equation* and one can use straight-forwardly the results of [111], Chapter 3, Theorem (3.2). □

Note that the proof have shown the convergence of the adaptation error (i.e. the performance variable). However, the convergence of $\hat{g} \to g^*$ is related to the richness of the disturbance signal d_C.

10.2.2 Going beyond hypotheses

We know take one by one the assumptions taken during the initial formulation to overcome them; and thus have an adaptation algorithm that works in the general case.

10.2.2.1 When the effect of the PI controller cannot be neglected (H1)

The continuous time PI controller has the transfer function:

$$PI(s) = \frac{K_I}{s} + K_P \tag{10.11}$$

Using the difference approximation $\frac{1-q^{-1}}{T_S}$ (with T_S being the sampling period) for differential operator one gets:

$$PI(q^{-1}) = \frac{K_I T_S + K_P - K_P q^{-1}}{1-q^{-1}} = \frac{r_0 + r_1 q^{-1}}{1-q^{-1}} = \frac{B_K(q^{-1})}{A_K(q^{-1})} \tag{10.12}$$

Using the results of [109], Chapter 15, Section 15.17, eq. (15.86) for our particular scheme (with the values $A_K = 1-q^{-1}$, $B_K = r_0 + r_1 q^{-1}$, $B_N = b_N\, q^{-(r_N+1)}$, $A_N = 1 + a_N\, q^{-1}$, $S = 1$, $A_M = 1$, $B_M = 0$), one gets:

$$e_{ST}(t+1) = \frac{q^{-(r_N+1)}|b_N|(1-q^{-1})[g^* - \hat{g}(t+1)]d_C(t)}{(1+a_N q^{-1})(1-q^{-1}) + b_N(r_0 + r_1 q^{-1})q^{-(r_N+1)}} \tag{10.13}$$

Using the parameter adaptation algorithm given in Section 10.2.1, the stability condition becomes that the following transfer function

$$H''(z^{-1}) = \frac{(1-z^{-1})(1+\hat{a}_N z^{-1})}{(1+a_N z^{-1})(1-z^{-1}) + b_N(r_0 + r_1 z^{-1})z^{-(r_N+1)}} \tag{10.14}$$

should be a strictly positive real transfer function.

If this condition is not satisfied, taking into account eq. (10.13), one can use instead the filter F given in eq. (10.4) to filter F_{PI}:

$$F_{PI}(q^{-1}) = \frac{sgn(b_N)q^{-(r_N+1)}\hat{b}_N(1-q^{-1})}{(1+\hat{a}_N q^{-1})(1-q^{-1}) + \hat{b}_N(r_0 + r_1 q^{-1})q^{-(r_N+1)}} \tag{10.15}$$

10.2.2.2 When clients and nodes impact on service time do not have the same dynamics: $a_C \neq a_N$ (H2)

Denoting the unmeasurable output of the disturbance propagation path $Y_C(q^{-1})$ by $z(t)$, eq. (10.13) becomes in the presence of the PI controller:

$$e_{ST}(t+1) = H''(q^{-1})\left([g - \hat{g}(t+1)]d_C(t) + [a_N - a_C]z(t)\right) \tag{10.16}$$

Since the adaptation does not affect directly the second term in the left hand side of eq. (10.16), we have to analyze its influence. As $z(t)$ is the output of an asymptotically stable plant whose input is bounded, it will also be bounded. Therefore the signal $\delta(t) = [a_N - a_C]z(t)$ will be bounded. Furthermore, due to the PI controller, the effect of this disturbance term will go to 0 asymptotically in steady state for a constant value of $d_C(t)$, since in eq. (10.13) the numerator of the transfer operator contains $(1-q^{-1})$. This also holds if there is no PI since, for a constant disturbance, the adaptation algorithm will interpret this term as an additional constant disturbance which will be canceled.

10.2.2.3 Transfer delays are different but known: $r_C \neq r_N$ (H3)

Hypothesis H3 on delays can be relaxed to $r_N \leq r_G$ but both know. A delay corresponding of the difference $r_C - r_N$ is added to the feedforward-based control signal part $u_{N,ff}$.

10.3 Adaptive Controller Evaluation

This section presents the evaluation of the parameter adaptation algorithm. Both simulations using Simulink and previously identified models and experimentations of a real MapReduce system running on a distributed platform are provided. The simulations enable to investigate stability of the proposed approach, and to master the variation that we enforce on the plant. Experimentations being long and heavy processes, they are only used to compare with the state of the art. Prior to those results, the simulation and experimentation set-ups are presented, as well as the methodology and scenarios considered for evaluation.

10.3.1 Evaluation Setup

The simulation of the adaptive controller is implemented on Matlab Simulink. We will consider that H2 does not hold, that is to say that the two paths of our model do not have the same dynamics. H3 will always hold (only in simulation) while a PI is added (H1 removed). The need of the PI is however investigated in Section 10.3.2.4.

The experiments were conducted on Grid’5000 [36], on a single cluster of 60 nodes. Each node from the cluster used for the test has a quad-core Intel CPU of 2.53GHz, an internal RAM memory of 15GB, 298GB disk space and infinite band network. All the nodes in the cluster were on the same switch to minimize network skews. The interaction between the cloud and the controller implemented in Matlab (interfaces and communication protocols) have been reused from Berekméri’s work [25]. The workload used for the experiments are Business Intelligence tasks, taken from MRBS benchmark suite [158]. The MapReduce version is Hadoop (its most used open source implementation).

The adaptive feedforward is simulated with the learning parameter $\alpha = 10^{-4}$ and the initial condition $\hat{g}(0) = 0$ i.e. no *a priori* knowledge is considered. The PI is chosen with a really slow dynamics in order to prevent over reaction, just as in [26].

The disturbance scenario, i.e. incoming workload, used both for simulations and experimentations is given in Figure 10.4. After taking a constant value for stabilization of the system, the number of clients sending requests to the cloud service varies with inconstant frequency and amplitude around the nominal number of 10 clients. This scenario aims at soliciting the plant in a more comprehensive way than a step signal, often used in the state of the art.

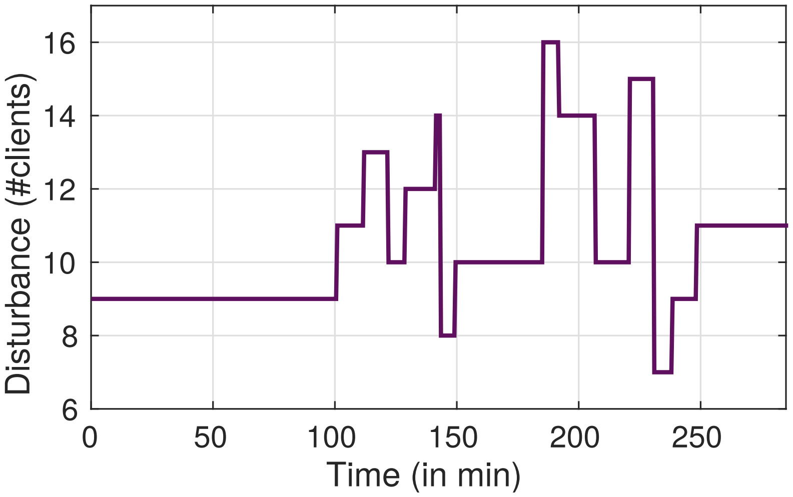


Figure 10.4 – Client disturbance Scenario

10.3.2 Simulation Results

10.3.2.1 Stability

The stability of the adaptive control system without and with the PI controller is ensured with the filter F given in eq. (10.4) since the transfer functions given in eq. (10.7) and (10.14) are strictly positive real as shown in the bode diagram of Figure 10.5. Indeed the phases always stay between -90^o and $+90^o$. When the node transfer function is mismodeled with a time constant two times slower or faster (i.e. changing $\hat{a}_N$), the stability is still ensured as illustrated with the non continuous lines. Moreover, as the plant gain value do not affect the phase (as long as the sign stays the same which is ensured in our use-case), the stability check still holds for all changes in the plant model.

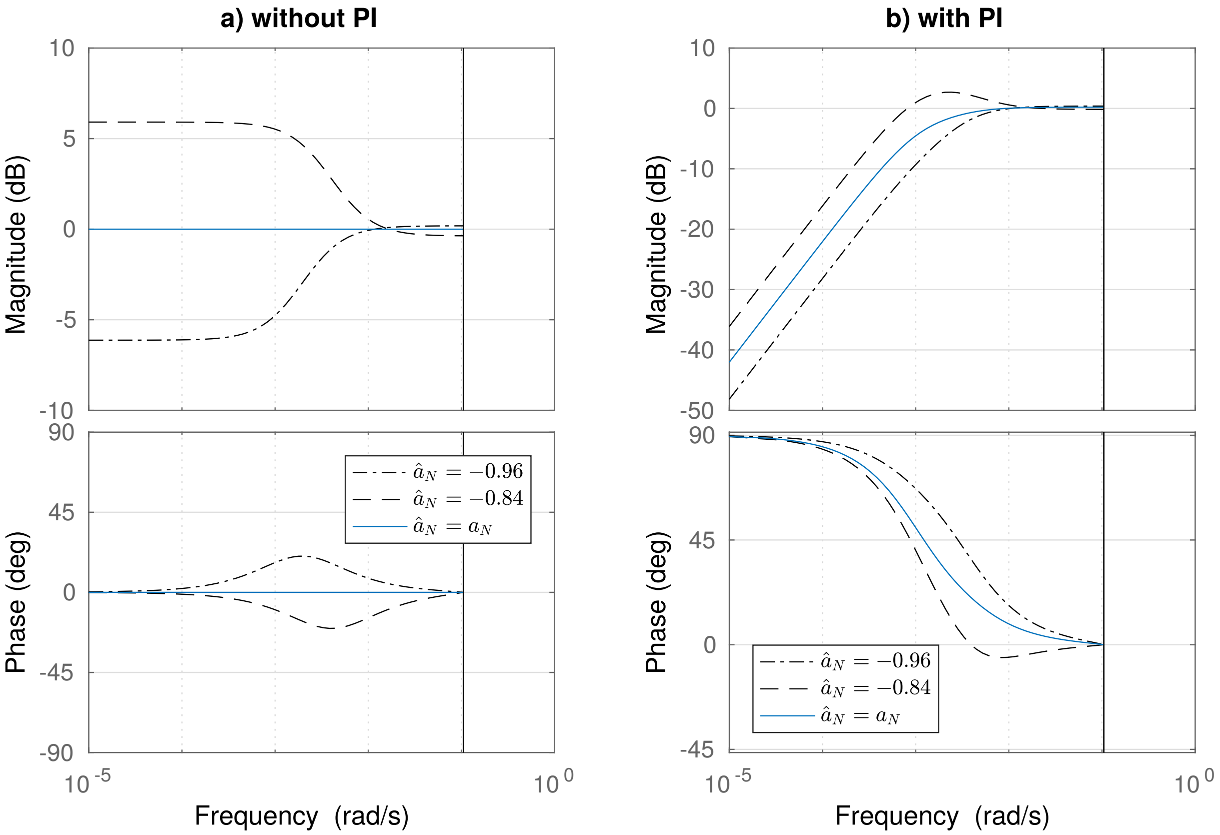


Figure 10.5 – Bode diagram of the transfer functions given in eq. (10.7) and (10.14) a) without PI b) with PI for $\hat{a}_N = a_N = -0.92$, $\hat{a}_N = -0.84$ and $\hat{a}_N = -0.96$

10.3.2.2 Robustness to gain variation

In a first time, we test our adaptation control scheme in comparison with the non adaptive state of the art, in a scenario where the node model gain b_N was overestimated by a factor two during the identification process. Only the node model is varied, as the client model is fairly accurate (see Figure 10.1 (A) and latter in Figure 10.9). Results are presented in Figure 10.6, with the performance indicator on the top plot ($e_{ST} = y_{ST} - y_{ST,ref}$) and the control signal (cluster size) in the bottom plot. The reference $y_{ST,ref}$ is set to the stable value of the model at the set point of $u_N = 20$ nodes and $d_C = u_{AC} = 10$ clients. The introduction of adaptation in the feedback and feedforward scenario enables the significant reduction of transient magnitudes and convergence times. The error variance is reduced from 5.17 s to 0.78 s during the disturbance rejection phase (on the time horizon from 100 to 300 min). Moreover, the control signal presents a more attenuated profile, which is highly beneficial for our application for which renting extra node for a short period of time induces significant costs.

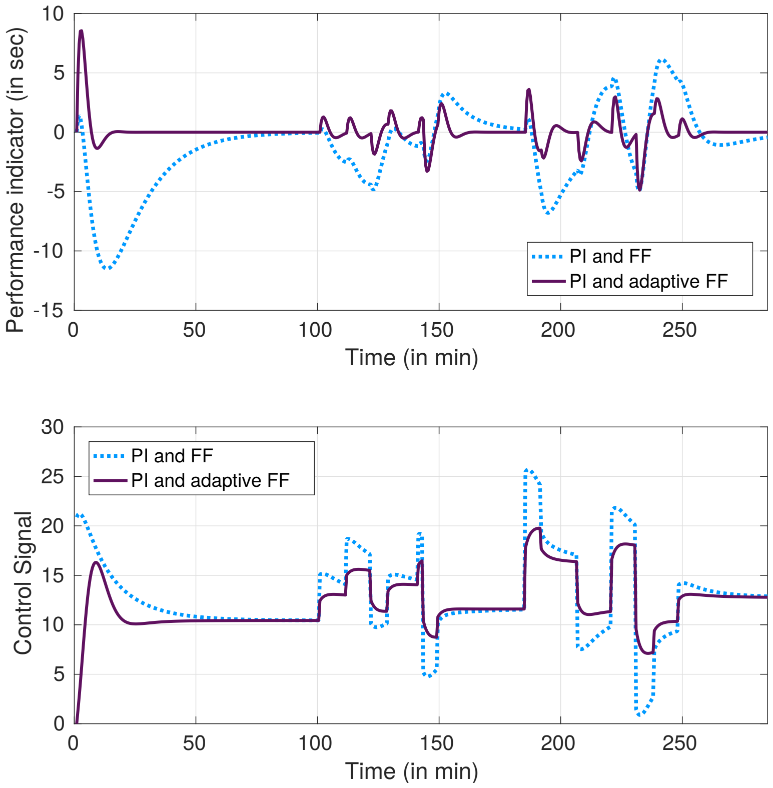


Figure 10.6 – Controllers performance comparison with overestimated model gain.

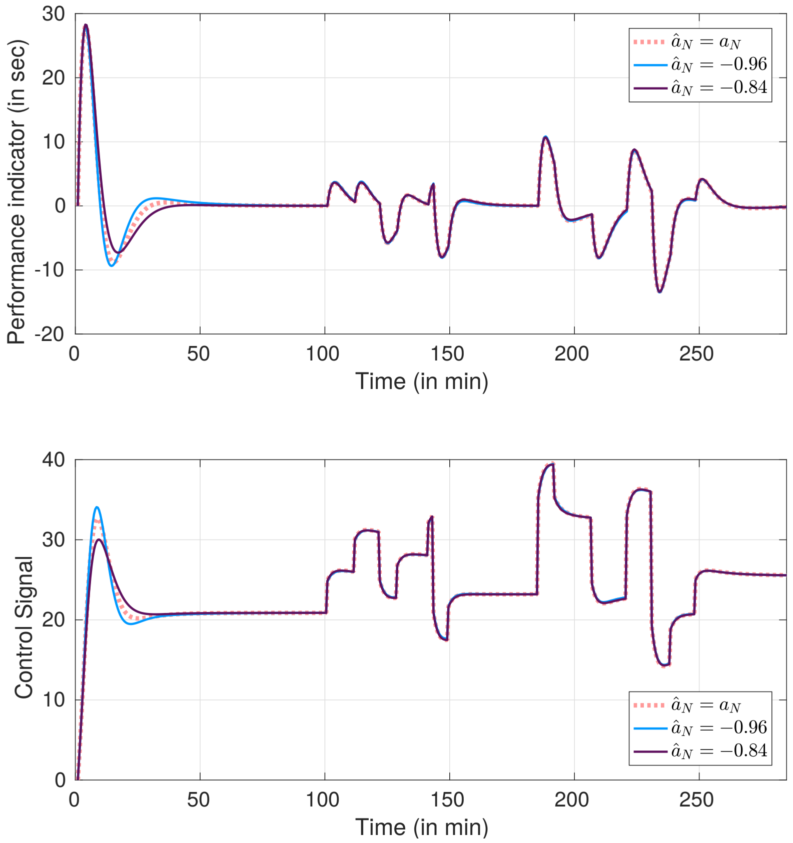


Figure 10.7 – Controller performance with characteristic time variation.

10.3.2.3 Robustness to dynamics variation

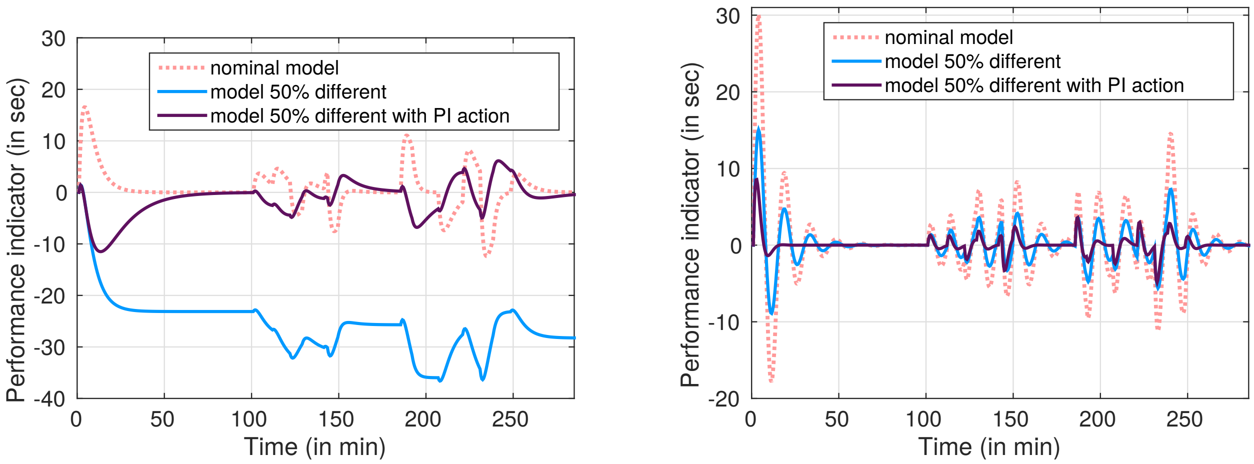
Figure 10.7 illustrates the robustness of the adaptation algorithm (combined with a PI) with respect to the value of $\hat{a}_N$ in the filter F. This represents an identification error regarding the dynamic of the plant. The node model is considered with its usual gain and the filter coefficient is taken with its nominal value ($\hat{a}_N = -0.92$), with higher value ($\hat{a}_N = -0.84$) and a lower value ($\hat{a}_N = -0.96$). These values reflect a modification of the estimated time constant by a factor 2.

After the control initialization, neither the magnitude of the transients nor the convergence time change when the filter pole varies. This allows to conclude that the performance of the adaptive controller is robust with respect to changes in the node model dynamics (a_N) or with respect to uncertainties in its estimation.

10.3.2.4 On the value of the feedback action in the controller

The behavior of the static (non adaptive) feedforward controller on the nominal model, on the model with a 50% different gain and eventually coupled with a PI controller are compared in Figure 10.8(a). When simulating with the nominal model (dotted line), the performance tends to 0 in steady state even if a transitory phase appears when a change in the disturbance occurs due to the difference of dynamics in direct and compensatory paths (H2 not holding), and to the static feedforward compensation. The modification of the plant gain changes the behavior of the controlled system and adds a steady state error (light blue line). Indeed, the feedforward gain calculated with eq. (10.1) is not correct anymore. In order to deal with this steady state error, the PI is introduced, that drives back the performance indicator to 0 in steady state (dark violet continuous line).

The evaluation of the PI with the adaptive feedforward is shown in Figure 10.8(b). Once again, three scenarios are considered: nominal model, overestimated gain during identification, both without and with PI. When simulating with the nominal model, the adaptive feedforward control drives the performance indicator to 0 but with a longer convergence time than in the static feedforward simulation and with more oscillations. However, the adaptive control is able to maintain the steady state to 0 even for the case with modified value of b_N. The addition of a PI controller considerably reduces the convergence time and the magnitude of the transients.



(a) Static Feedforward

(b) Adaptive Feedforward

Figure 10.8 – Controller performance for various plant gain scenarios, with and without PI

10.3.3 Experimental Results

The experimentation evaluation compares the presented adaptive feedforward and feedback controller (in Figure 10.10) to a system in open loop, i.e no control, (in Figure 10.9) and to the state of the art non adaptive controller (see Figure 10.2).

In the open-loop experiment of Figure 10.9, the cluster size is fixed at its nominal value, i.e. 20 nodes, see bottom plot, and the workload varies with several steps up and down around its set point value of $d_C = u_{AC} = 10$ clients, as is shown in the top plot. The reference value for the service time is chosen at the level at which it converges after the first minutes. This reference choice protocol is the same for all the other experiments and enables to cope with the unmastered differences between the various experiments (external load on the cluster, network usage, etc.).

We can recognize the first order dynamical behavior of the service time in response to the workload changes. As no adaptation of the cluster size is done, the reference is exceeded multiple times, up to +56% overshoot. When the workload is low, the service time goes down to 30% lower than

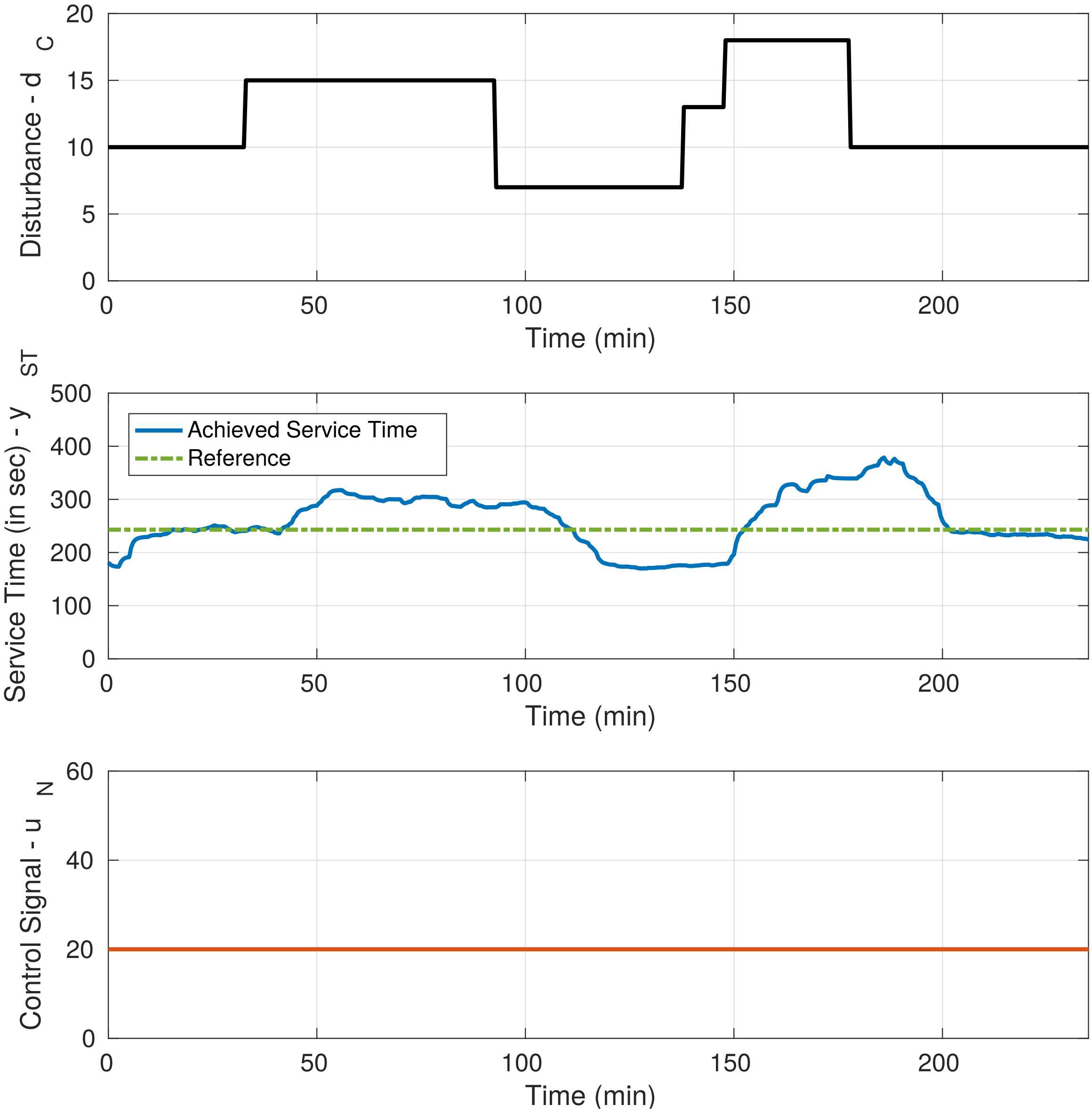


Figure 10.9 – Experimental performance in Open Loop (no control).

the desired value, which means that the resources are overused. The relative mean squared positive error (computed only when the reference is exceeded) is of 17.4 s. This value will be compared to the results of the various control strategies.

We now compare those results with the ones of the state of the art controller, see Figure 10.2. A summary in given in Table 10.1. The controller shows very high overshoot (+146%) and unstable oscillations in the end of the experiment. The relative mean squared positive error in this case is twice bigger: 39.9 s. Moreover, quite some negative overshoot can be observed too (up to -38%); which make this controller even worst than having no control, at least for this workload scenario.

Results of the addition of adaptation are plotted in Figure 10.10. Qualitatively, given the flat profile of the service time, the controller performs better. The overshoot is at its maximum +32% and at minimum -17%. Those values are not significantly far from the ones of the other control laws, partially due to the fact that the reference value for the service time here is lower that for the others. However, the relative mean squared positive error of the time when the reference is overshooted is of only 1.5 s, which is on order of magnitude smaller that the state of the art controller and the open

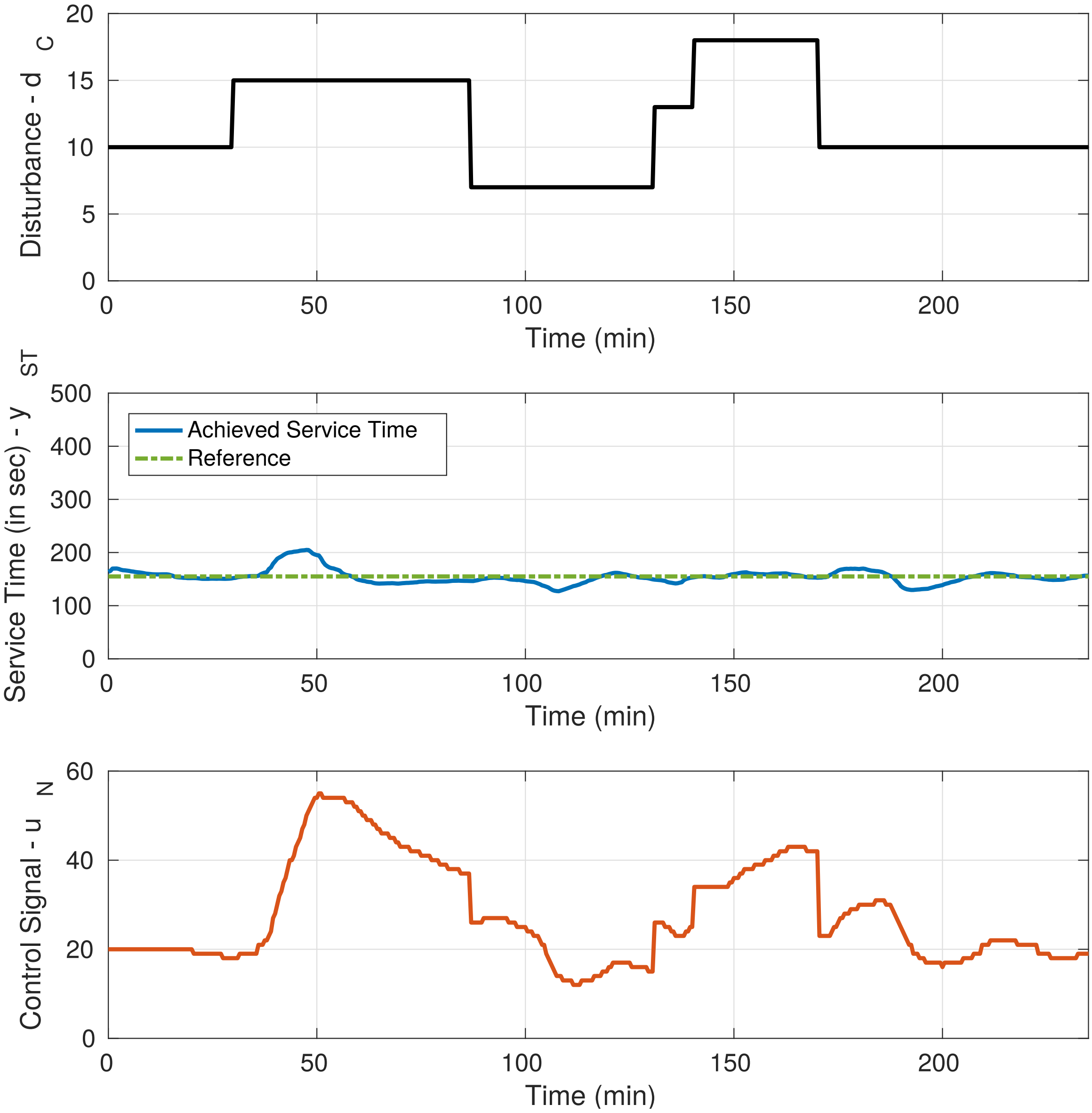


Figure 10.10 – Experimental performance with the adaptive feedforward and feedback controller.

loop. Regarding the control signal, less oscillations can be seen compared to the static feedforward controller. The cluster size is more wisely updated, which is more cost efficient.

In a second time, we evaluate our proposed controller on a comprehensive disturbance scenario, similar to the one used for the simulation validation (see Figure 10.4). We aim at validating our controller for workloads presenting large amplitude and frequency changes. Results are reported in Figure 10.11. The controller manages to keep the service time at its required level. Two overshoots can be seen when the workload is significantly high, but their value is limited to +21.5%. The relative mean squared positive error over the whole experiment is only of 3 s, slightly higher that with the previous scenario but still an acceptable value compared to the value of the reference (around 160 s). Eventually, the resources are sparsely used as the cluster size is kept below 20 nodes most of the time. However, some really small oscillations (of one or two nodes) are present in the steady phases. This behavior does not bring much to the performance while they are detrimental both from the economic point of view and the perspective of the overload that is created by constantly switching on and off some machines.

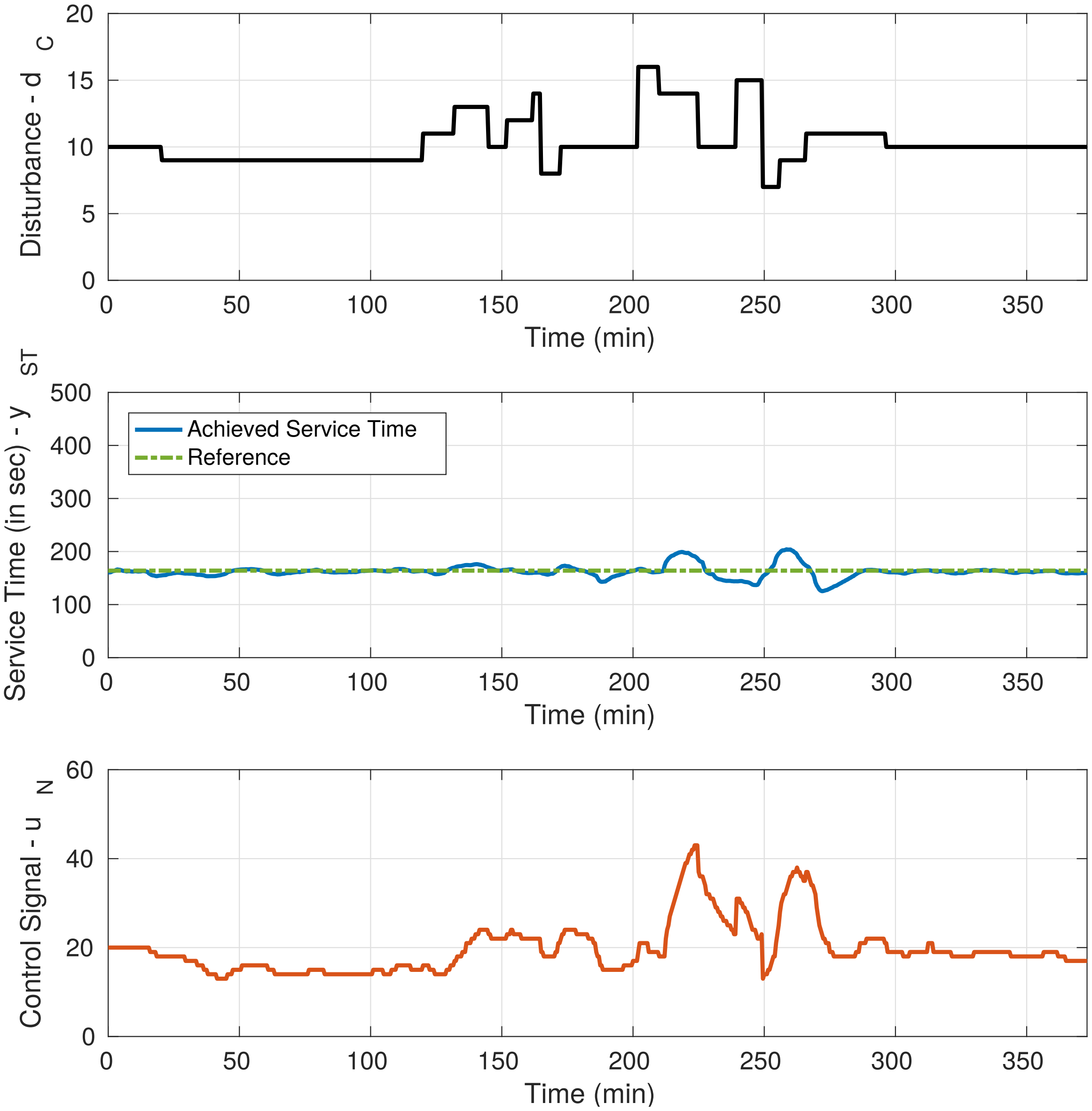


Figure 10.11 – Experimental performance of the adaptive controller for a comprehensive workload.

10.4 Conclusion

This chapter presented an adaptive control strategy that ensures efficient monitoring for the performance of Cloud Services in term of average service time of all jobs, by dynamically controlling its resource cluster size. The control algorithm is based on a PI and feedforward with an adaptive estimation of the feedforward gain. Evaluation of the approach is provided both in simulations and with experimentations. Simulations are based on a state of the art model of MapReduce, running comprehensive workloads. Experimentations use MapReduce to realize benchmark workloads on a nation-wide cluster. Results show the adaptive control stability and its ability to reduce by at least a factor of 10 the error on the service time tracking compared to a PI and optimal feedforward. Moreover, the presented adaptive control has proved its robustness as it successfully manages to control the cloud service even when the physical system is significantly different from the considered model.

Future works will consist in extending the proposed adaptation algorithm to improve its convergence behavior, as well as including more system specific constraints, such as control signal minimization. Other service metrics such as availability, dependability or costs which are crucial for service providers and users should also be considered.

Controller	**maximum overshoot**	**maximum negative overshoot**	**relative mean squared positive error**
Open Loop	+56%	-30%	17.4 s
State of the Art	+146%	-38%	39.9 s
Adaptive Approach	**+32%**	**-17%**	**1.5 s**

Table 10.1 – Experimental controllers performance comparison on the simple workload scenario. Best results are in bold.

Chapter 11

Cost-aware Control of Cloud Services

BigData cloud services performance are evaluated by the jobs runtime, as presented so far, but also by metrics such as the service availability and its cost efficiency. This chapter develops a control strategy that enables to take those multiple objectives into account. The formulation is a Model Predictive Control (MPC) with the addition of a event-based triggering mechanism (instead of a classical time-based control signal generation). The triggering mechanisms of the control theory state of the art did not correspond to the objectives of cost-efficiency specific to our cloud application. This made us develop a new cost-function based triggering mechanism. The theoretical contribution of this chapter however goes beyond the MapReduce use case, as it can be applied for all non linear discrete time systems. Stability proof of the controller is provided, as well as simulation validation of the control performance. We also evaluate the approach on the cloud use-case using simulations of a real world workload, showing 8% cost saving (based on Amazon EC2 pricing) compared to the classic error-based event triggered control.

After an introduction on motivation and related works (Section 11.1), the some background and formalism is provided on the optimal control formulation, see Section 11.2. The new control strategy is presented in Section 11.3 while Section 11.4 depicts its evaluation.

11.1 Introduction

When running requests on an online service, three major objectives can be formulated: maximization of the performance in terms of run time, maximization of the service availability and reduction of the costs. Those are the common criterias that make the clients choose one platforms rather than another. While the previous chapter was focusing only on the first objective, we are now interested in a complete framework able to satisfy performance, availability and cost requirements. Indeed, they are contradictory objectives to achieve. For instance, if very few requests are accepted and many resources are dedicated to the remaining tasks, the performance will be excellent but the other objectives won't be satisfyingly reached. To be able to monitor the cloud service, two control knobs are commonly available: resource cluster scaling and admission control. Indeed, too large clusters have high monetary and energetic costs, while too small ones bring poor performance, often resulting in financial and commercial penalties [67]. Those considerations lead to the use of a multi-input and multi-output control problem formulation.

The cost efficiency objective can be directly linked to the cluster size, as more machines means more power and network consumption. Any change in the number of resources implies a novel reconfiguration of the system which takes considerable time and costs money due to system unavailability and/or poor performance [119]. Moreover, quick changes in the control value (thus a high rate re-configuration of the cluster) can overload the communication channels and lead to an accumulation of delays, which in return increases the time needed for processing the requests [123]. Large delays along with overloaded communication channels, besides being a security threat, can increase the probability of hardware/software faults and lead to performance deterioration and skyrocket the financial costs [67]. From the point of view of the low-scale user, the example of Amazon EC2 [16] public cloud where one can rent resources for minimum ten minutes is a relevant example where scaling up and down the cluster too often is detrimental.

Cloud services also face multiple disturbances of unusual natures. The workload, the amount of requests that are received by a computing system, is an input signal that, usually, we cannot control. For the considered application, workload variations can be categorized as:

- Small predictable variations. This disturbance can be statistically modeled using a long tailed distribution, for example using a log-normal distribution [59]. The probability mass function of the log-normal distribution is $ln\mathcal{N}(\mu,\sigma^2)$, with μ its mean and σ its standard deviation. Figure 11.1 is an example of a big data workload over a day period. We can see for instance from 5 to 7 h or 20 to 23 h that the distribution has a stable mean and low variance.

- Large instantaneous variations. In cloud services, large and unpredictable events also occur quite often. An extreme example is the peak of connections to the *Michael Jackson* Wikipedia page just after his death (more than 1000 times more connections then usual and around 1 million views in 1 hour) [135]. This kind of sudden events changes the mean number of requests and affect the variance, as can been seen at around 1, 9 or 12 h in Figure 11.1.

As we would like to minimize the total utilization cost, our objective in this chapter is twofold: (i) as for computer systems a change in the number of resources automatically implies system reconfiguration which increases the expenses, we want to reduce the amount of resources we use as well as the number of cluster reconfigurations, that is to say the number of changes in the input signals but also their absolute values; (ii) in the case of computing systems the environment changes very rapidly so the system needs to react very fast to large disturbances but also be robust enough and not to react to the small natural variations of it.

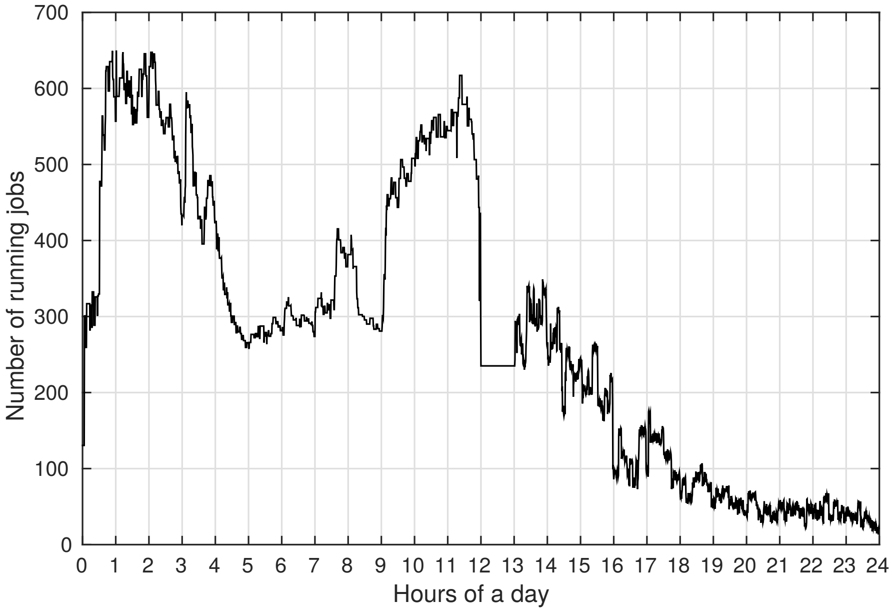


Figure 11.1 – On-line shopping website - MapReduce workload on a 2,000-node cluster [155]

Related work regarding the control of cloud services has been presented in Chapter 9. We will however mention here the work of Berekméri et al. on the control of MapReduce service time with cost efficiency objectives [26]. The used controller was with feedback and feedforward action, both coupled with an threshold-based triggering strategy to reduce the cluster size updates. However none of the state of the art solutions are explicitly designed to reduce the cluster reconfiguration rate in a multi objectives context.

In this chapter, we present event-based control techniques to reduce the number of cluster reconfigurations. For the control community, the use of event mechanisms let hope for a reduction in the use of resources [22, 64] without degrading performance [120] and with stability and robustness guarantees [127]. However, all the numerous event-based control strategies in the literature are focused on stability and performance guarantees. Those researches are motivated by the communication or computation reduction in networked controlled systems which are different objectives from reducing the number of reconfigurations while guaranteeing an acceptable level of performance.

Event based control consist of switching the control from an traditional time-based periodic computation to an event-based one. Today triggering strategies are based on level-crossing by the measurement error (for instance when using PID controllers [21, 64]) or more generally by some Lyapunov function (see for instance [171]) or on the vanishing of an event-function related to a Control Lyapunov Function (see for instance [127]). Other event triggering mechanisms are linear functions of the measurement error as in [78], [61]. In all cases, the decision of updating the control law or not usually does not use a prediction of what will happen in the future. Those formulations are *reactive*, in the sense that control is triggered only when the impact of disturbances has driven the system to cross a critical threshold. *Predictive* methods that consider explicitly the future evolution of the current state are very encouraging but until now, few methods of this type have been developed. Eqtami et al.'s work [66] is an example of this type where the prediction is based on thresholds crossing around the desired states.

Deciding on the triggering based on threshold crossing might not be the best solution in all cases. For example, an acceptable state at present time which leads to an undesirable behavior in the future can be handled by the predictive controller straight away, thus improving performance. Conversely, an undesired behavior in the present time but which leads to a stable situation in the future may not be taken care of, to reduce events number and reconfigurations. Furthermore, when dealing with multi-input and multi-output (MIMO) systems, the measurement errors alone are not sufficient to decide if the system is in a critical state and needs a control actuation. Indeed, the multiples signals impact one another and a same value of the error for an output does not necessary reflect a critical situation for the system according to other signals values. For example, a CPU usage of 90% reflect a normal behavior if memory usage is high while it is alarming if memory usage is low.

In this chapter, we suggest an triggering mechanism based upon the optimal cost function for Model Predictive Controllers (MPC) that addresses the issues presented above: the calculation of the cost function which by definition deals with the different weights of the performance signals and takes into account, the future behavior of the system on the prediction horizon using the model. Furthermore, constraints on the command shape (for instance the command should be constant on a time window) combined with an event mechanism ensure that changes in the command signal are reduced while other specifications still hold.

11.2 Preliminaries

Let us first define the notation used and formalize the optimization problem. We consider a general nonlinear discrete time system described by the equations

$$\begin{cases} x_{k+1} = f(x_k, u_k) \\ y_k = g(x_k) \end{cases} \tag{11.1}$$

where $x \in \mathbb{R}^{n_x}$ is the vector of describing the states, $y \in \mathbb{R}^{n_y}$ is the measured outputs and $u \in \mathbb{R}^{n_u}$ the control variables. We assume that $f(0,0) = 0$ and $g(0) = 0$. Only stabilization at the origin is considered here, as most problem formulations can be transformed to such stabilization points. The subscript k of a variable stands for its value at time kT_S, T_S being the sampling period. We define a time horizon of $N \in \mathbb{N}$ instants, $N > 0$. The N notation in this Chapter is not referring to the cluster size variable as in Chapters 9 and 10. In this section, we assume the system has no delay neither constraint on the control or state values. Extending the proposed results to delayed or constrained systems can however be done as classically in model predictive control. The MapReduce application typically has delays, saturation constraints on the control and positivity constraints on the states values. The applicability of this technique will be validated on such complex system, but theory is provided only for the system under the hypothesis of no delay nor constraint.

Let $U \in \mathbb{R}^{n_u \times N}$ be a control profile over the horizon N defined by:

$$U := \begin{pmatrix} u_0 & u_1 & \cdots & u_{N-1} \end{pmatrix} \tag{11.2}$$

For any initial condition x_i, and any control profile $U \in \mathbb{R}^{n_u \times N}$, we define the corresponding state trajectory $X \in \mathbb{R}^{n_x \times (N+1)}$ by:

$$X(x_i, U) := \begin{pmatrix} x_0 & x_1 & \cdots & x_N \end{pmatrix} \tag{11.3}$$

$$\text{with} \quad \begin{cases} x_0 := x_i \\ x_{k+1} = f(x_k, u_k) \quad \forall k \in \{0, \ldots, N-1\}, \end{cases} \tag{11.4}$$

and we associate to this state trajectory, the output trajectory $Y \in \mathbb{R}^{n_y \times (N+1)}$ defined by $Y(X) := \begin{pmatrix} y_0 & y_1 & \ldots & y_N \end{pmatrix}$ with $\forall k \in \{0, \ldots, N\}$, $y_k = g(x_k)$. Note that in the above definitions, the trajectories are defined on the time horizon only. Let $J : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u \times N} \to \mathbb{R}^+$ be some cost function depending on an initial condition x_i and a control profile U. A very common definition of the cost function for a stabilization objective is the quadratic cost of the form:

$$J(x_i, U) = \sum_{k=1}^{N} X|_{k+1}^T(x_i, U)\mathbf{Q_x}X|_{k+1}(x_i, U) + U|_k^T \mathbf{R} U|_k, \tag{11.5}$$

where $X|_k(x_i, U)$ (resp. $U|_k$) denotes the k^{th} column of $X(x_i, U)$ (resp. U), ^{T} is the transpose operator. $\mathbf{Q_x}$ and $\mathbf{R}$ are positive, definite matrices of appropriate sizes. Note the shift in the indexes: $U|_k = u_{k-1}$ and $X|_k = x_{k-1}$. For a reference tracking objective, the cost function could be:

$$J(x_i, U) = \sum_{k=1}^{N} \tilde{Y}|_{k+1}^T \mathbf{Q_Y} \tilde{Y}|_{k+1} + U|_k^T \mathbf{R} U|_k, \tag{11.6}$$

with $\mathbf{Q_Y}$ and $\mathbf{R}$ positive, definite matrices of appropriate sizes, $\tilde{Y}|_k := Y|_k(X(x_i, U)) - Y^{ref}|_k$ where $Y^{ref} \in \mathbb{R}^{n_y \times (N+1)}$ is the reference trajectory on the time horizon N. Finally, let us define the set $\mathscr{U}$ of controls as:

$$\mathscr{U}(x_i) := \left\{ U \in \mathbb{R}^{n_u \times N} \text{ s.t. } X|_{N+1}(x_i, U) = 0 \right\}. \tag{11.7}$$

$\mathscr{U}$ is the set of control profiles that satisfy a terminal constraint at the end of the horizon on the state variable. The final constraint is very common for stability purposes [130].

Assumption 1 *We assume that for all $x \in \mathbb{R}^{n_x}$, $\mathscr{U}(x)$ is not an empty set.*

Under Assumption 1, the model predictive optimal control problem is formulated as follows:

$$\begin{array}{rcl} \hat{U}:\mathbb{R}^{n_x} & \to & \mathbb{R}^{n_u \times N} \\ x & \mapsto & Argmin_{U\in\mathscr{U}(x)}\, J(x,U), \end{array} \tag{11.8}$$

and the control law for system (11.1) expressed in a state feedback form is classically:

$$u(x_k) := \hat{U}|_1(x_k). \tag{11.9}$$

Stability of the control law (11.8) has been studied in the literature [6]. If one defines the following operator $\Pi : \mathbb{R}^{n_u\times N} \to \mathbb{R}^{n_u\times N}$ that associates to U given by (11.2) the control profile $\Pi(U)$ defined by

$$\Pi(U) := \begin{pmatrix} u_1 & u_2 & \dots & u_{N-1} & 0_{n_u\times 1}, \end{pmatrix} \tag{11.10}$$

the stability relies on the fact that at time $(k+1)T_S$, the *translation* of one time step of the solution control profile $\hat{U}(x_k)$ at time kT_S, namely $\Pi(\hat{U}(x_k))$, belongs to the set of admissible control $\mathscr{U}(x_{k+1})$. This sort of invariance property guarantees that the cost associated to the control at time $(k+1)T_S$ is necessarily strictly smaller than the one at time kT_S and therefore the stability is guaranteed as soon as the optimal cost function $\hat{J}(x) := J(x,\hat{U}(x))$ is a Lyapunov function for system (11.1).

In our particular case, changing the control value too often may be financially expensive and need to be avoided. For this, we propose to remove some degrees of freedom by adding linear constraints on the control variable. Therefore, we extend the set $\mathscr{U}$ as follows:

$$\mathscr{V}(x_i) := \left\{ U \in \mathbb{R}^{n_u\times N} \text{ s.t. } X|_{N+1}(x_i,U) = 0 \text{ and } AU = B \right\}, \tag{11.11}$$

with A and B matrices of appropriate sizes. We assume that it is not empty:

Assumption 2 *We assume that for all $x \in \mathbb{R}^{n_x}$, $\mathscr{V}(x)$ is not an empty set.*

In order to keep the invariance property on which the stability relies, we define for the first iteration

$$\begin{array}{rrcl} \hat{\Lambda}_0 : & \mathbb{R}^{n_x} & \to & \mathbb{R}^{n_u\times N} \\ & x & \mapsto & Argmin_{V\in\mathscr{V}(x)}\, J(x,V), \end{array} \tag{11.12}$$

and for the following ones

$$\begin{array}{rrcl} \hat{\Lambda} : & \mathbb{R}^{n_x}\times\mathbb{R}^{n_u\times N} & \to & \mathbb{R}^{n_u\times N} \\ & (x,U) & \mapsto & Argmin_{V\in\mathscr{V}(x)\cup\{\Pi(U)\}}\, J(x,V). \end{array} \tag{11.13}$$

The control profile for system (11.1) is then dynamically defined for any x_k at time kT by:

$$\hat{V}_k \;:=\; \hat{\Lambda}(x_k,\hat{V}_{k-1}), \tag{11.14}$$

where $\hat{V}_{k-1}$ is the control profile at time $(k-1)T$. To initialize the process, we take at $k=0$, $\hat{V}_0 := \hat{\Lambda}_0(x_0)$. The control law for system (1) expressed in a state feedback form is then:

$$u_k := \hat{V}_k\big|_1. \tag{11.15}$$

The corresponding *optimal* cost is then $\hat{J}_k := J(x_k,\hat{V}_k)$.

Theorem 2 *Under Assumption 2, and if J is a Lyapunov function, the control law defined by (11.15) asymptotically stabilizes system (11.1).*

The proof is trivial since by construction the cost function is strictly decreasing. In particular, if one takes a cost function of the form (11.5), one can prove that:

$$\hat{J}_{k+1} - \hat{J}_k \le J(x_{k+1},\Pi(\hat{V}_k)) - \hat{J}_k = -x_{k+1}^T \mathbf{Q_x} x_{k+1} - u_k^T \mathbf{R} u_k. \tag{11.16}$$

11.3 Control Strategy

This section presents a triggering algorithm for MPC control laws (11.9) and (11.15) defined in the previous section. This strategy will decide when the control profile needs to be recalculated. For sake of simplicity, we assume in the following that the cost function J is of the form of eq. (11.5).

11.3.1 The Lyapunov based triggering strategy

To any state x_k of the system at time kT_S and any control profile W_{k-1} at time $(k-1)T_S$, we associate two corresponding costs, namely $\hat{J}_k$ defined as previously by $\hat{J}_k := J(x_k, \hat{\Lambda}_0(x_k))$ and $\tilde{J}_k := J(x_k, \Pi(W_{k-1}))$. The first cost $\hat{J}_k$ corresponds to the cost if one updates the control with its *optimal* value whereas, $\tilde{J}_k$ is the cost obtained keeping the previous control profile. Note that by construction, one has the following inequality: $\tilde{J}_k \geq \hat{J}_k$. We can now define the event function $e : \mathbb{R}^{n_x} \times \mathbb{R}^+ \to \{0,1\}$ by:

$$e_k = \begin{cases} 1 & \text{if } \tilde{J}_k - \hat{J}_k \geq \varepsilon_J \hat{J}_k \text{ or if } k = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{11.17}$$

The proposed MPC event-triggered control profile W_k at time kT_S is then:

$$W_k = \begin{cases} \hat{\Lambda}_0(x_k) & \text{if } e_k = 1 \\ \Pi(W_{k-1}) & \text{otherwise.} \end{cases} \tag{11.18}$$

The control law to apply is as previously the first element of the control profile, that is:

$$w_k = W_k|_1. \tag{11.19}$$

Note that if $e = 0$ the control profile of the last time period is applied and then $w_k = W_k|_1 = W_{k-1}|_2$. Hence, contrary to usual event-based control, the input control value is not kept constant in the proposed scheme. The control profile is updated when the cost can be reduced by a factor of $1 + \varepsilon_J$ with respect to the cost when keeping the same control profile.

Choice of ε_J The threshold ε_J from eq.(11.17) should be chosen carefully. If it is too small, the controller will unnecessarily react to noise, model uncertainties or observer error (if one), while if ε_J is too large we may not react fast enough to disturbances. The tuning of ε_J can be done in a training phase using data from an open-loop experiment with disturbances.

Whatever the value of ε_J might be, the point is that the threshold is relative to the current value of expected cost function. This means that we can detect disturbances that do not have a comparable absolute cost.

11.3.2 Stability of the proposed scheme

The proposed scheme asymptotically stabilizes the system at the origin since it ensures the strict decrease of the cost function until the origin is reached:

Theorem 3 *Under Assumption 2, and if J is a Lyapunov function, the control law defined by (11.19) asymptotically stabilizes system (11.1).*

Proof: Assume at time kT_S, the control profile is W_k with a cost $J(x_k,W_k)$ defined by (11.5). At time $(k+1)T_S$, the state value is given by $x_{k+1} = f(x_k,W_k|_1)$. Assume first that $e_{k+1} = 0$, then $W_{k+1} = \Pi(W_k)$ and

$$\begin{aligned} J(x_{k+1},W_{k+1}) &= J(x_{k+1},\Pi(W_k)) && (11.20)\\ &= J(x_k,W_k) - x_{k+1}^T \mathbf{Q_x} x_{k+1} - W_k|_1^T \mathbf{R} W_k|_1. && (11.21) \end{aligned}$$

J is therefore strictly decreasing in this case. Assume now that $e_{k+1} = 1$, the control profile needs therefore to be updated. In that case, $W_{k+1} = \hat{\Lambda}_0(x_{k+1})$ and therefore one has:

$$\begin{aligned} (1+\varepsilon_J)J(x_{k+1},W_{k+1}) &\leq J(x_{k+1},\Pi(W_k)) && (11.22)\\ &\leq J(x_k,W_k) - x_{k+1}^T \mathbf{Q_x} x_{k+1} - W_k|_1^T \mathbf{R} W_k|_1. && (11.23) \end{aligned}$$

In that case again, J is strictly decreasing as long as $x_{k+1} \neq 0$ which ends the proof of stability of the scheme. □

11.4 Evaluation

This section presents the evaluation of the cost-based event triggered MPC both as a new controller in itself and for its application to MapReduce control. First, the simulation conditions and evaluation protocol is given.

11.4.1 Experimental conditions and methodology

The controlled system is simulated using Matlab Simulink, using the models presented in Chapter 9, Section 9.5. The MPC controller has the following constraints:

- the cost function is based on the output signals: availability is a more critical output than the service time, both should be at their reference values at the end of the horizon $N = 100T_S$, with $T_S = 30\,\text{s}$;
- the cluster size should be minimized;
- for the constrained scenario, the size of the cluster can only change at regular time intervals. This value is tuned to be the indivisible minimum renting time of resources in a cloud [16], 10 min, that is to say only 5 times during the horizon N;
- the maximum number of accepted clients is free to vary as long as it does not overshoot the number of clients: $0 \leq u_{MC} \leq d_C$ and we consider a limited resources configuration where the maximum size if the cluster is 60 nodes.

The validation initial scenario is the following: the system is launched with 20 nodes, a maximum number of accepted clients at 8 and a constant disturbance load of 9 clients. Once the stabilization is reached (at $t = 0$ min) we start the control with the strong constraint that the outputs of the system should remain at theses steady state values. Neither the availability reference nor the service time one are varied. Indeed, we consider that those thresholds are fixed by the cloud provider in its SLAs and thus that it is not possible to update them online. The controller performance are then only evaluated in a disturbance rejection scenario. Initially we consider a disturbance changing by steps and we make sure that it varies slowly enough to let the system stabilize between changes. This simple simulation aims at enabling a deep analysis of the controller performance, specially regarding the choice of the triggering function threshold ε_J. Eventually, the record of a 1-day workload from a real cluster (see Figure 11.1) is used for validation of the controller in a genuine situation (Section 11.4.4).

In a first time, the performance of our new event triggered function is compared to the state of the art time triggered controller and error-based event triggered control. Then, the addition of the constraints on the cluster size control signal will be studied. Eventually, we simulate a 1-day usage of our controller based on a real scenario and compute the costs saving induced by our approach.

11.4.2 Cost based event triggering mechanism validation

First we compare our method referred to as *cost based* event solution to an *error based* one and a *time based* controller. All the solutions are calculated without specific constraints on the input signals, as described by eq. (11.9). For the error based triggering mechanism the event function is either when the control law over the whole horizon has been applied or when one of the outputs crosses a certain threshold. As for the choice of the thresholds, we fixed $\varepsilon_J = 0.5$ based on training data to ensure no false alarms (see more in Figure 11.4), then we chose the error-method thresholds so that the event-based and the cost based solutions ensure the same performance.

Results are shown in Figure 11.2. For comparable performance in term of reference tracking and inputs variations, we observe that the cost based controller generates only 6 events over a 150-minute period, which represents 98.5% less events than the time triggered solution (300 events in total), and 71.5% less than the error based controller (21 events). Moreover, due to the dynamic of the system, once the error thresholds are crossed it takes some time to stabilize the outputs signal inside the threshold-defined bounds which leads to several events generated by the error based method after each disturbance occurs. However, the cost based method takes into account this dynamic through states predictions thus removing these redundant events. Hence, with significantly less computing efforts our proposed event mechanism can guarantee comparable performance.

11.4.3 Input-constrained control

Second, we simulate the behavior of the cost-based constrained controller forced to have constant values of the input signal u_N on large time intervals (see eq. (11.15)), that we compare to its unconstrained version (eq. (11.9)). Results are given in Figure 11.3: with comparable performance in reference tracking for both outputs and the exact same number of events, the constrained controller imposes 94.5% less changes in the cluster size.

Furthermore, nonlinear delays caused by communication channel congestion happening when cluster reconfigurations are too frequents were not taken into account in our modeling. Consequently we expect that in real on-line experiments with the real MapReduce system, the constrained cost-based event method will also significantly improve the performance in terms of availability and service time.

Threshold choice

When looking at the event function (Figure 11.4), we see that the condition for triggering an event gives no false alarm and reacts each time workload conditions change. Moreover, every event is caused by a clearly marked threshold crossing, it is an indicator of the robustness of our method regarding the value of ε_J.

Furthermore, we see in Figure 11.4 the importance of the threshold expressed as a percentage. Just after an event, the cost J is way higher than just before as it takes time and resources for the system to reach specifications again, however this high value of J does not necessary reflect that another disturbance happened.

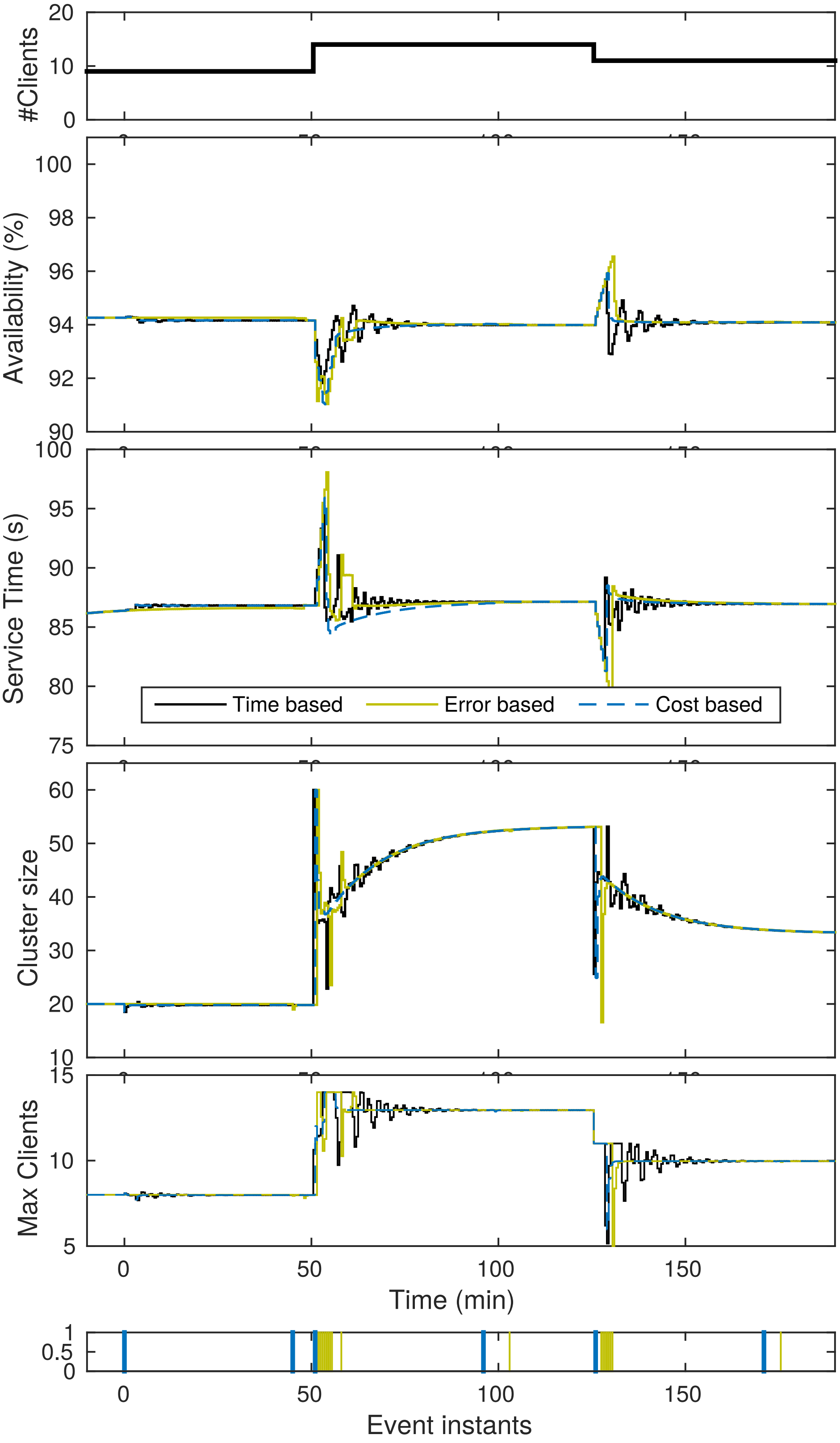


Figure 11.2 – Comparison of time-based, error-based and cost-based controller performance without constraints on the number of allowed reconfigurations

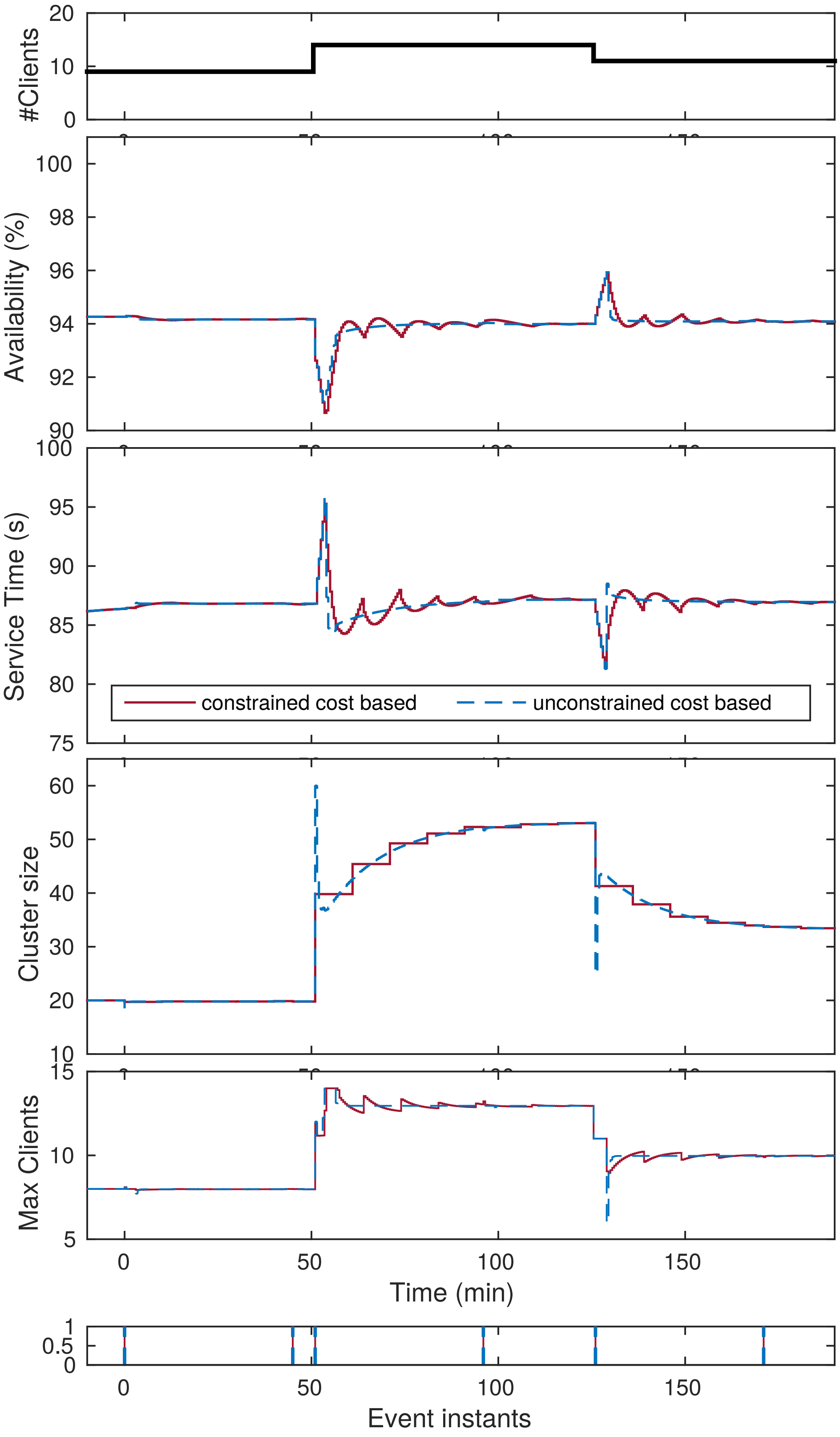


Figure 11.3 – Comparison of cost-based controller performance with and without constraints on the control changing time

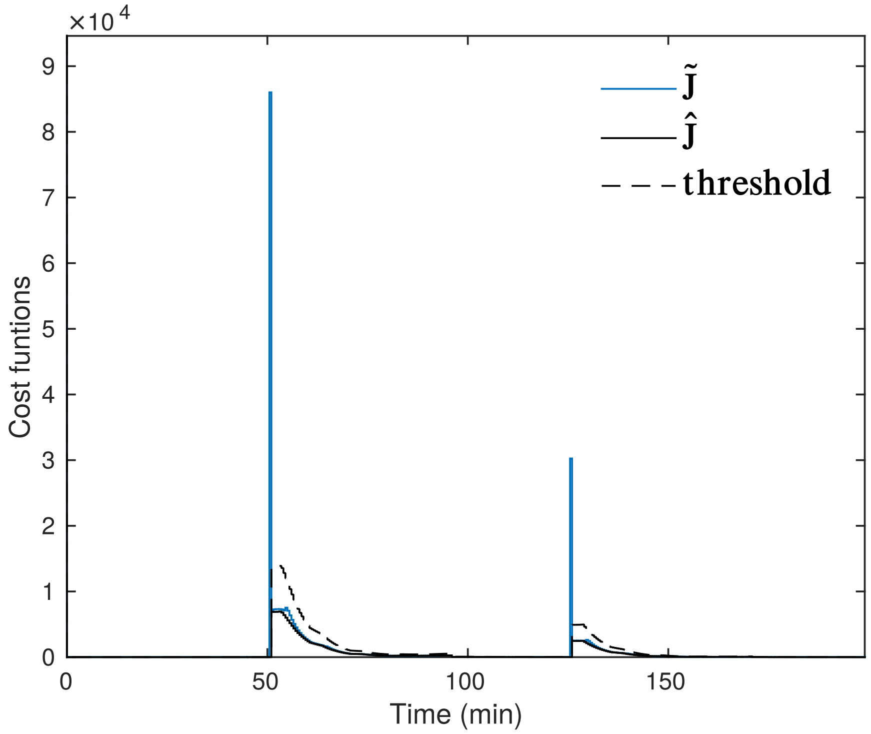


Figure 11.4 – Cost-based event function

11.4.4 Evaluation using a real MapReduce workload

We validate our proposed solution using a real 1-day MapReduce workload trace that was recorded on a 2,000-node cluster of Taobao [155] (see Figure 11.1), an e-commerce website. We scaled the values to fit our 60-nodes cluster but we did not modify the shape of the workload. In Figure 11.5 we compare the unconstrained cost-based controller to the constrained one. Results are similar to what we got with a simple 2-step workload scenario (see Figure 11.3) except that we reach our saturation levels on the cluster size on the upper bound as well as the lower bound. This leads to better performance for both controllers from 16 h to the end of the day because the minimum resources that we have available are enough to provide barely full availability and really fast service time. In this scenario we observe many peaks in the cluster size signal for the unconstrained controllers, those are useless brief cluster over-sizing that we have already noticed with the simpler workload.

Using Amazon EC2 pricing, the constrained cost based solution enables to realize major saving as detailed in Table 11.1.All the controllers performance on a 1-day workload were not plotted for readability but they are included in the prices comparison of Table 11.1. We took Amazon pricing as a reference for cost calculations as it is the first cloud provider, with around 40% of the market share while the three other big providers (see Chapter 9, Section 9.5). From Table 11.1, it is important to note that costs were calculated using the scaled workload, with maximum cluster size of 60 nodes. When considering the original Taobao workload, fees should be around thirty times higher, however saving percentages should not change.

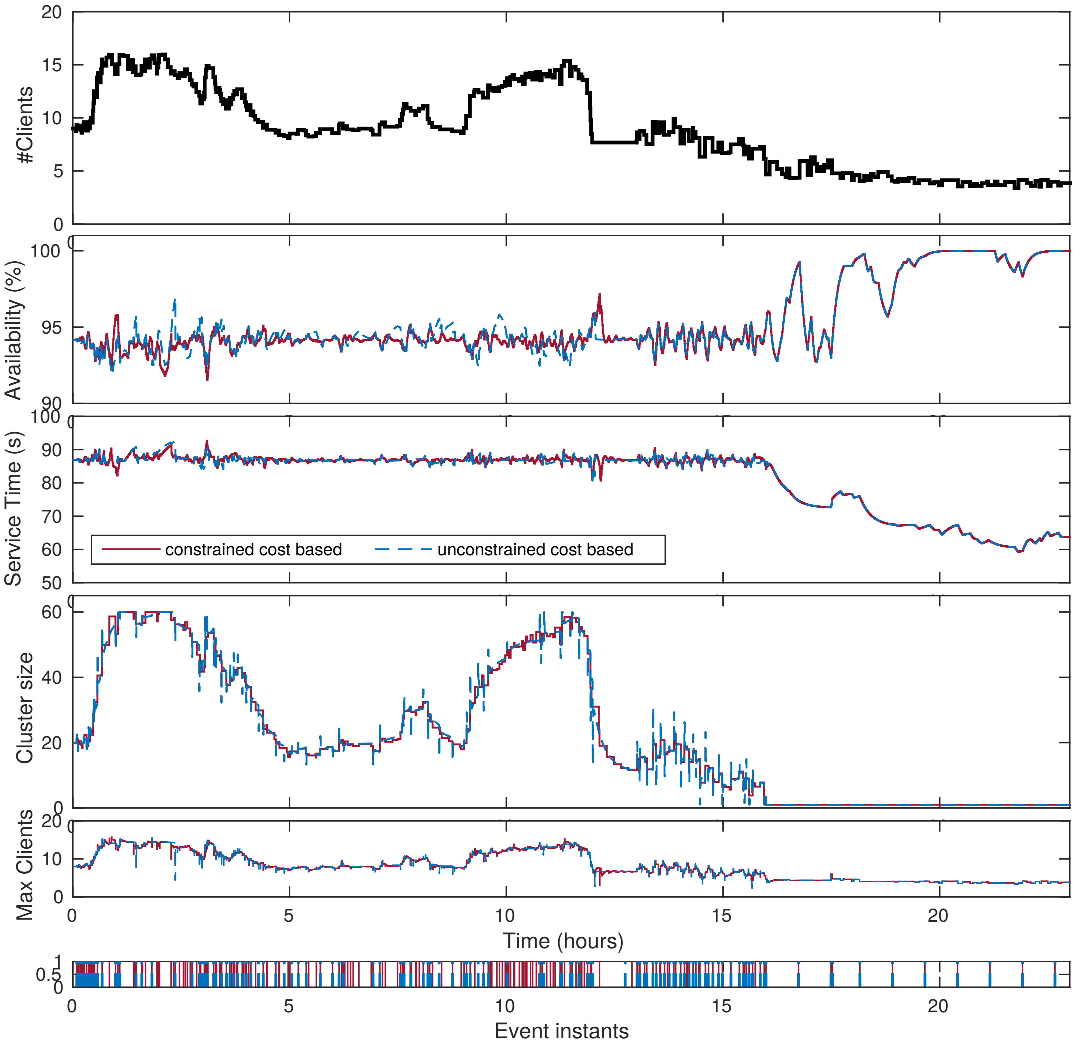


Figure 11.5 – Evaluation of cost-based controllers on a real 1-day MapReduce workload scaled to our cluster size.

Method	**fees**	**Extra costs compare to constrained cost based**	
No control	$1500	$941	**62.7%**
Time based (unconstrained)	$591	$32	**5.4%**
Error based (unconstrained)	$606	$47	**7.8%**
Cost based (unconstrained)	$590	$31	**5.3%**
Constrained cost based	$559	-	-

Table 11.1 – Cost comparison of different solutions based on a 1-day real workload

11.5 Conclusion

In this chapter we presented a new event mechanism for model predictive controllers which has two aims: (i) to reduce the number of events without degrading performance and (ii) to reduce the changes in the control inputs thus reacting to large disturbances and being robust to smaller ones.

The latter constraint is particularly relevant in the context of big-data cloud services where a high frequency of cluster reconfiguration has major financial and energetic costs and leads to performance degradation. The event mechanism developed here is a function of the cost function defined in the optimal problem, and not based on measurement error as has often be the case until now. This enables taking into account predictions of the system trajectory over a time horizon. In order to handle the reduction of input changes we add constraints to the MPC formulation, allowing changes only at regular time intervals.

The controller also has the advantage of being able to follow objectives expressed with various metrics: service time performance, service availability and costs. The multi-signal cloud control consists in a valuable improvement for the fulfillment of cloud services SLAs.

Evaluation is carried out in simulation on a model which was previously validated on a real MapReduce system. The simulations show that the constrained cost-based event triggered MPC significantly reduce the number of events as well as changes in the control law. We generate 86% less events in comparison to an error based method and we obtain at least 8% cost savings when simulating with a real 1-day workload (based on Amazon EC2 pricing).

The next steps of this work would be to validate the constrained cost-based event triggered MPC approach through experiments using the latest MapReduce release on a public Cloud such as Amazon EC2. We expect to have more promising results due to the reduction of cluster reconfigurations which are even more costly on the real system due to congestion of communication channels leading to performance degradation. However, the variability of the system highlighted in Chapter 10 might lead us to add a robustness oriented aspect to our controller.

Chapter 12

Conclusions on MapReduce Control

This ending chapter of Part III highlights this thesis contributions with regards to the bigdata cloud services monitoring domain, its limitations and its perspectives.

The growing development of computing devices as well as their capacities soar has generated massive amounts of so called bigdata. They are a windfall for companies that are able to extract from them many valuable informations. The advances in data processing has been fostered by the rise of the cloud computing paradigm, in which resources are made available and maintained as a service for clients that just want to run their processing without dealing with the hardware or low level software stacks. MapReduce/Hadoop is one of the most famous paradigm for bigdata processing able to work in such cloud infrastructure. As for any service, the guarantee of its performance is a key challenge both for the cloud provider and the client running MapReduce jobs. The execution time of the clients' requests as well as the service availability are two of the prevalent indicators of the service quality. In order to guarantee them, the cloud provider can realize elastic resource provisioning and admission control.

The works presented in this thesis use control theory to approach this performance monitoring challenge. Starting from the state of the art, we presented two main contributions. First, the problem is addressed from the robustness perspective. Given the limits of the state of the art to capture the non linear behavior of the MapReduce framework and to ensure performance in case of highly dynamic workload, an adaptive controller has been developed. It consists in a PI feedback control with gain feedforward action, which parameter is adapted on line based on the system observed behavior. This approach has been validated both in simulation and by using a MapReduce benchmark on a real cluster. However, the monitoring of jobs execution time only do not enable to apprehend the complexity of the agreements between a cloud provider and his clients. The second contribution of this part realizes the joint optimal control of jobs service time and service availability. An emphasis is done on the cost efficiency, which leads to the development of a new event-based triggering mechanism for the Model Predictive Controller used. The triggering mechanism is based on the Lyapunov cost function of the system, which enables to minimize the reconfigurations of the cluster, as it is the main source of expenses in the cloud paradigm. Results are promising but however need to be evaluated on a real cloud.

The robustness and cost-efficiency challenges are indeed complementary and a combination of both approaches of adaptation and event-based control could stand as a promising solution. However, adaptation for a MIMO model and triggering mechanisms able to deal with time varying systems are more complex problems that would require extended research work. The benefits of such solutions would go beyond the bigdata cloud control problem.

Nevertheless, those works present some limitations. The first one is the related to the metrics choice. An ideal control satisfies the objectives for the values and behavior of the signal and costs functions used for the problem formulation. Thus, aside from the technical achievements, the definition of the problem is of first importance. For instance, the performance metric that was picked in this work is the average execution time of the jobs that finished in the last time slot. However, many other choices could have been made: computing the 99^{th} percentile instead of the average, taking into account the duration of the running jobs instead of the finished ones, etc. A deeper study of the usuals SLAs of cloud providers could be of great use to refine those metrics. Moreover, the addition of new dimensions to the problem should be considered, to cope with new demands of the clients, such as dependability to hardware and software failure or security guarantees. The definition of a wider MIMO problem could enable to take into account all those aspects.

In a second time, let us recall some of the hypothesis that were taken for this work: the cluster has homogeneous resources, the jobs of the workload are of similar nature and the MapReduce version is fixed. In order to increase the robustness of the presented approaches, the cases when those assumptions do not hold should be tested. The homogeneity of current cloud is often a reality at small scale. However, the datacenters are nowadays highly heterogeneous, with application specific hardware such as GPUs. The adaptation algorithm presented in Chapter 10 could however enable to deal to some extent with those differences, whereas it still needs to be tested. Eventually, even if the experimentations were done on a real cluster deployment, Grid5000 is a research environment with eased deployment facilities. The application of the presented control technique on a commercial cloud hosting MapReduce framework, such as Amazon EMR [15], is still a technical challenge.

Part IV

Conclusions and Perspectives

This thesis presented the use of control theory for autonomous computing systems. Being in the middle of two distinct research domains, a comprehensive introduction was drawn. The two fields were presented and put in perspective, both with an accessible introduction and then in a more technical way. The related work was then overviewed, first by reviewing the various tools for software adaptation and then by focusing on detailing the solutions using control theory. The approach taken in this work was to address this issue through two computing system applications, enabling to address various problem specifications and to develop varied modeling and control techniques; but also to investigate diverse and complementary aspects of the computing world.

First, we focused on the location privacy challenge. Mobile devices' users send queries to services and include their location data to have a geo-localized answer, such as weather predictions or nearest points of interests suggestion. Then, all those location data may be collected in a database, which processing is of great interest (infer road usage frequency and the transportation mode used, derive popularity of places, etc.). Indeed, all those useful services comes at the price of personal information disclosure and is thus an major privacy breach. Our approach consisted in providing tools for the mobile device user and the data processing instance to protect data privacy without sacrificing their utility. More specifically, starting from the state of the art Location Privacy Protection Mechanisms (LPPMs), we combined them and configured them in a dynamic way that fulfill both the users' privacy and utility requirements. From a control theory perspective, this work provided the first formulation of the location privacy challenge in terms of definition of the plant, the control input, the disturbance signal as well as the performance outputs. Identification was performed to derive a model of the system and a first controller was presented to serve as a proof of concept.

Second, we considered the use-case of a bigdata processing framework running remotely on a cloud. Monitoring of performance in terms of response time of the jobs and availability of the service were the main objectives, which can be achieved by tuning the resource cluster size and by realizing admission control. Controllers were designed in order to ensure the monitoring robustness to the system changes and workload variations, and to minimize the service costs. Advanced control techniques for achieving multi-objectives have been developed and applied: a robust adaptive feedback controller and a event-based triggering mechanism for optimal control.

Conclusions regarding the contribution of this thesis, its limitations and perspective for the different application domains have been drawn in the respective parts. The works of this thesis has brought significant improvement regarding the state of the art, whether by the novelty of the problem formulation or by the resulting performance of the controlled system.

On top of aiming to provide novel solutions for the specific use-cases, this work also enable to draw lessons on benefits and difficulties of using control theory to adapt software systems.

Computing applications are ruled by laws that are far from the physics ones, as usually considered in the control theory framework. This leads to particularities in the systems' behaviors and constraints. Taking the location privacy challenge, the metrics considered were varying over several orders of magnitudes, with equal semantic importance of the performance on each of them. For the cloud control problem, the system's costs were computed not only based on resource usage but also taking into account a minimal utilization time for each instance, corresponding to the minimal renting time on a cloud platform. Those two examples illustrate that the control theory has to be extended to cope with those new behaviors, for instance by adding a logarithmic normalization of the metrics, or by defining a new cost function of the optimal controller. Software systems adaptation enables to extend the theory of control.

The benefits of the use of control for the various computing applications are even more valuable. First, it provides a well defined methodology that is able to address many monitoring challenges. Depending on how clear the problem is formulated, with regards to performance metrics and control knobs, the development of modeling and control can be straightforward. From this, the monitored closed-loop system can be further studied to derive mathematical guarantees of its stability or performance. This is highly relevant, for instance when a certification of the algorithms is needed, e.g. in the aeronautics industry.

Moreover, some limitations can be drawn. For instance, one can go back to the working hypothesis of control theory: a dynamical system is considered with at least one observable output and a controllable input. The observability and controllability can often be dealt with for software applications by the design of new sensor or actuator algorithms, but at the cost of modifying the initial system, which is not always feasible or desirable. The dynamic hypothesis is the main limitation of control theory for such applications, while being also its main advantage. It enables to deal with problem formulations that have barely been addressed in the computing world (e.g. location privacy control) while it is of no use for the static monitoring, even though such static challenge are often still unsolved problems (e.g. configuration of privacy protection mechanisms for static mobility databases).

Stepping back from the technical perspective, working at the intersection of different scientific field induces some difficulties which, as basic as they appear, significantly slow down the research process. For instance, some scientific terms are not defined and used in the same way. One can mention the expressions "response time" which is a measure of the controller convergence speed in control theory while it is the execution time of a request in the cloud computing world; or "utility" which can be either the objective function of a user to be maximized or reflect the quality of the service provided to the user. Moreover, the expanse of the contributions over distinct publication communities harden the literature review and the works diffusion.

We now take a look at the general perspectives of the association of control theory and software adaptation. Following the trends in the computing world, a promising direction of research is the inclusion of new performance metrics such as security level of applications or privacy protection guarantees. Those concerns are among the first ones for industrials and individuals whereas they are barely dealt with. The focus should particularly be done on the dynamic consideration of those metrics, acknowledging that they can evolve according to varying requirements or modifications in the external environment. Indeed, with such sensitive aspects, the results of a security or privacy monitoring should come with guarantees both on the performance and on its robustness regarding changes in its internal behavior and on the ones of its environment.

On top of considering new performance indicators, the focus should be done on the control of new kind of softwares. The recent soar of machine learning, both in the development of new algorithms and on the spreading of its applications, make it a prevalent class of systems that is interesting to address from the control theory perspective. Machine learning applications show impressive results regarding their accuracy performance, while little works have been dedicated to the study of their robustness and most of all on the guarantees of their results. Those two limitations are areas for which the control theory has decades of expertise that can benefit both domains. Moreover, learning algorithms can be considered as modeling tools, while most of the time the modeling prediction is meant to derive a monitoring strategy. The association of learning with the decision and actuation tools of control could enable to consider the challenges in their entirety, in a way to optimize the decision process and not necessary the modeling accuracy. This direction of research has been investigated alongside and following this thesis work, the main results are presented as appendix of this manuscript, see Part VI.

The combination of control theory and software systems has shown significant benefits for the research community, and still presents promising perspectives in which the science of the trade-offs, robustness and guarantees, will be combined with the state of the art solutions of the peak problems of the industrial and social worlds.

Part V

Résumé en français

Cette thèse présente l'utilisation de la théorie du contrôle pour l'automatisation des systèmes informatiques.

Les systèmes informatiques, que ce soit pour un usage personnel ou dans le monde industriel, sont de plus en plus nombreux et complexes. Les smartphones fournissent des services basés sur l'utilisateur tels que des recommandations personnalisées. Les entreprises ont accès à une quantité croissante d'informations sur leurs clients et/ou leur environnement, et les utilisent pour développer des analyses et des services basés sur l'intelligence économique. Ces deux exemples simples montrent que les outils informatiques sont utilisés, directement ou indirectement, à différents degrés de conscience, par la plupart des gens dans leur vie quotidienne.

La dernière décennie a vu le point culminant de la différenciation entre le logiciel et le matériel. Cette thèse se concentre uniquement sur l'aspect service et logiciel. Dans ce contexte, l'on retrouve souvent les mêmes objectifs généraux: performance, disponibilité, fiabilité, bas coûts, sécurité, respect de la vie privée. Ces objectifs à atteindre sont désirés avec des propriétés telles que la robustesse ou des garanties.

Une solution à tous ces défis est l'adaptation logicielle. Cette discipline vise à modifier les paramètres/mode/ressources d'un logiciel en réaction aux changements de son propre comportement ou de son environnement. De nombreux outils différents peuvent être utilisés pour réaliser l'adaptation dans la pratique, tels que la théorie des files d'attente, l'apprentissage automatique ou la théorie du contrôle. Dans ce travail, nous avons choisi de nous concentrer sur cette dernière technique. La théorie du contrôle est un domaine de l'ingénierie qui vise à surveiller les systèmes dynamiques, grâce à l'utilisation de boucles de rétroaction. Il a commencé à être bien établi depuis le début des années 1900, pour ses applications aux systèmes industriels, notamment dans l'aéronautique. En raison de son histoire pour les systèmes régis par les lois de la physique, ce n'est que depuis la fin des années 2000 que son application aux systèmes informatiques a été étudiée. Cependant, sa forte culture mathématique et ses garanties formelles sur les résultats font de la théorie du contrôle une solution prédominante pour l'adaptation logicielle. Ceci est bien illustré par l'augmentation significative du nombre d'articles sur ce sujet.

Dans ce contexte d'association de la théorie du contrôle et des systèmes informatiques, cette thèse adopte l'approche d'explorer en profondeur deux cas d'utilisation. Dans ce travail, ces applications ont été abordées avec une grande variété de techniques complémentaires. Ainsi, divers aspects de la combinaison des deux domaines ont été explorés. La première est axée sur la protection de la vie privée liée à la mobilité des utilisateurs d'appareils mobiles, tout en assurant la convivialité des services géolocalisés. La seconde traite des services BigData Cloud et de la manière d'assurer leur performance et leur fiabilité dans leur environnement extrêmement variable. Les deux applications qui sont développées dans ce manuscrit sont complémentaires à deux égards. Premièrement, ils proviennent de deux domaines du vaste monde informatique qui reflètent la complexité et la diversité des défis à relever, qu'il s'agisse de fournir un outil accessible et utile pour une grande population ou de mettre au point des outils à la fine pointe de la technologie pour les spécialistes des données. Deuxièmement, la plus grande partie du vaste domaine de la théorie du contrôle est illustrée, dès le tout début, là où le problème doit être formulé et où des outils simples peuvent être appliqués au système non linéaire le plus complexe variant dans le temps pour lequel des techniques récemment développées sont nécessaires.

Un troisième cas d'utilisation a été étudié en parallèle et à la suite de ce travail de thèse. Les systèmes informatiques considérés sont des algorithmes d'apprentissage automatique, tels que les k-means, les arbres de décision ou les célèbres réseaux des neurones. Leur contrôle a été étudié sous deux angles complémentaires. Premièrement, le défi de la robustesse du processus d'apprentissage vis-à-vis du bruit dans la base de données a été soulevé. Deuxièmement, les travaux se sont concentrées sur la paramétrisation des algorithmes, avec l'introduction d'une boucle de rétroaction.

Protection de vie privée des données de mobilité

La première application traite d'un côté du monde de l'informatique, à savoir l'utilisateur de smartphone. Plus précisément, nous étudions la protection de la vie privée des personnes lors du partage de leur localisation par le biais d'applications. Une attention particulière est portée à la qualité du service géolocalisé reçu. La littérature sur l'utilisation de la théorie du contrôle pour les questions de protection de la vie privée est presque inexistante, ce qui fait de ce cas d'utilisation un véritable terrain de jeu où tout doit être construit, de la définition des objectifs de l'utilisateur à la mise en place de la protection de la vie privée et à son évaluation. Les défis de la cybersécurité et de la protection de vie privée sont d'une importance primordiale mais cependant manquent de solutions aussi bien pratiques que théoriques. De plus, les propriétés spécifiques à cette nouvelle formulation d'un problème de contrôle ont beaucoup à apporter à la communauté des chercheurs en automatique.

La démocratisation des appareils mobiles a favorisé le développement d'applications utilisant les données de localisation des propriétaires pour fournir ou améliorer un service. Ces applications sont appelées services basés sur la localisation (LBS) et sont par exemple les applications de navigation, les systèmes de recommandation ou les applications de suivi de la condition physique.

Un grand nombre de données de mobilité sont générées, collectées, et actuellement utilisées par les entreprises et les chercheurs. En effet, le traitement de ces données peut révéler des informations précieuses qui peuvent être utilisées pour un large éventail d'applications, par exemple, la gestion du trafic routier, la planification urbaine, etc. Ces services et les données collectées correspondantes profitent donc à toutes les parties, mais au prix de la publication de données personnelles. Les fournisseurs de services, ou tout attaquant tiers, profitent de ces données pour obtenir toujours plus d'informations sur les utilisateurs. Les attaques peuvent se produire à chaque étape de traitement. Les menaces les plus courantes pour les utilisateurs sont les attaques de réidentification où l'identité d'un utilisateur anonyme est devinée à partir de données précédemment enregistrées, la prédiction de la mobilité qui anticipe les prochains déplacements des utilisateurs en fonction de leurs habitudes, l'extraction des centres d'intérêt de l'utilisateur (domicile, lieu de travail, lieu de culte, etc.) et la déduction des relations sociales (partenaires, collègues de travail, etc.). Afin d'offrir des moyens de protéger la vie privée des utilisateurs, des mécanismes de protection de la vie privée liée à la localisation (LPPM) ont été élaborés. Cette terminologie regroupe tous les algorithmes qui modifient les données de localisation afin d'améliorer la confidentialité des utilisateurs.

Des exemples simples de LPPM sont l'ajout de bruit aux données de l'utilisateur, la réduction de la précision des données ou la fusion des informations de mobilité des utilisateurs proches. Il existe une grande diversité parmi les LPPM : certains agissent au niveau de l'utilisateur, d'autres nécessitent des tiers de confiance ; certains sont des mécanismes en ligne, d'autres ne peuvent être appliqués que sur un ensemble de données ; etc.

Habituellement, les LPPMs sont des algorithmes paramétrés, par exemple la quantité de bruit ajoutée aux données peut varier. L'ajustement de ces paramètres permet de nuancer leur action, c'est-à-dire de protéger plus ou moins la vie privée des données. Cette propriété est très précieuse étant donné que la protection de la vie privée se fait souvent au prix d'une réduction de l'utilité du service. Par conséquent, un LPPM configurable permet d'ajuster le compromis entre la confidentialité et l'utilité. En effet, les données de mobilité sont directement utilisées par le LBS pour fournir un service à l'utilisateur, et lorsqu'il adopte le point de vue du LBS, ces données sont traitées conjointement pour récupérer certaines informations de haut niveau (utilisation de la route, moyens de transport, etc.). Il n'est pas facile de traiter à la fois de la protection de la vie privée et de l'utilité, étant donné le compromis naturel qui existe entre les deux. Comme les mécanismes d'amélioration de la protection de la vie privée consistent à déformer les données originales pour cacher l'information, leur utilité est par définition réduite.

De plus, chaque mobilité d'utilisateur est très dynamique, avec des vitesses et des fréquences de déplacement variables. Ainsi, l'application d'un mécanisme de protection peut donner lieu à différents niveaux de confidentialité et d'utilité selon les propriétés de la mobilité de l'utilisateur.

Toutefois, le réglage fin de ces paramètres peut nécessiter des connaissances et une expérience approfondies. L'objectif est de garantir des niveaux donnés de confidentialité et d'utilité tant du point de vue de l'utilisateur que de l'analyse des données de manière automatisée. De plus, compte tenu de la variabilité des déplacements des utilisateurs et de leur environnement, les garanties de confidentialité et d'utilité doivent être solides. Deux contributions correspondant à deux scenario d'utilisation différents ont été développés.

Scénario statique: traitement d'une base de donnée de mobilité.

Les données de mobilité collectées par les entreprises ou les chercheurs doivent protéger la vie privée des utilisateurs tout en permettant l'extraction d'informations agrégées utiles. Pour cela, des mécanismes de protection de la vie privée (LPPM) sont utilisés. Cependant, le LPPM approprié et sa configuration à utiliser peuvent varier pour chaque utilisateur et suivant son objectif. En outre, des garanties concernant les niveaux de protection et l'utilité de l'ensemble de données obtenu sont nécessaires.

Le système PULP (pour Privacy and Utility through LPPMs Parametrization) permet à l'utilisateur de recommander le meilleur LPPM à utiliser et comment le configurer. Il fonctionne en trois étapes : l'analyse de l'impact du LPPM sur la vie privée des individus et sur l'utilité du service, puis la modélisation non linéaire du comportement du LPPM, et enfin la recommandation d'un LPPM correctement réglé pour chaque individu de la base de donnée. La recommandation est fondée sur des objectifs, tels que la quantification du compromis entre protection des données et utilité. Quatre bases de données de mobilité regroupant plus de 770 utilisateurs sur plusieurs années a été utilisée pour l'évaluation de PULP.

Scénario dynamique: diffusion dynamique de localisation.

Le système dynULP (pour dynamic and Useful Location Privacy) prend le point de vue d'une personne utilisant un appareil mobile pour profiter du service géolocalisé. Les données envoyées doivent être protégées tandis que le service reçu doit être utile. Dans ce scénario dynamique et centré sur l'utilisateur, la nécessité d'une configuration correcte du LPPM se pose également. En outre, étant donné que la mobilité de l'utilisateur évolue dans le temps, le mécanisme de protection devrait également évoluer pour assurer un niveau constant de protection de la vie privée.

La théorie du contrôle est appliquée à ce cas d'utilisation. Le respect de la vie privée est considéré comme un signal mesurable à contrôler, tandis que la configuration du LPPM permet de l'influencer. Des contributions ont été apportées à la formulation du problème, c'est-à-dire à la recherche de moyens appropriés pour mesurer la protection de la vie privée et l'utilité des données afin de permettre leur contrôle. Ensuite, des outils de modélisation et des lois de contrôle ont été appliquées pour assurer le suivi de référence (en terme de niveau de vie privée) et la robustesse vis à vis de la mobilité de l'utilisateur.

Les deux systèmes sont complémentaires, car ils couvrent tous les aspects de la question de la protection de la vie privée. PULP peut être utilisé soit sur des bases de données qui ont déjà été enregistrées et pour lesquelles dynULP ne peut pas être appliqué, soit en combinaison, en traitant conjointement toutes les sources d'utilisations malveillantes des données de localisation (à la diffusion et au cours du traitement).

Des améliorations de ces deux contributions pourraient être faites, notamment sur la complétion des notions de vie privée (par exemple en ajoutant la notion de prédictibilité de la trajectoire) et d'utilité. L'implémentation de dynULP en tant qu'application pour smartphone est prévue, car son faible algorithme de calcul lui permet d'être embarqué. Le système PULP pourrait également être étendu aux entreprises, par exemple pour assurer leur conformité au GDPR.

En conclusion, les travaux PULP et dynULP sont deux solutions qui relèvent le défi de l'applicabilité de la protection de la vie privée à partir de deux approches complémentaires, utiles à la fois pour les utilisateurs quotidiens de smartphones et les experts en traitement des données.

Services BigData sur le Cloud

Le deuxième cas d'utilisation est la garantie de performance et de fiabilité des services BigData sur le cloud, par le biais de l'allocation de ressource et du contrôle des admissions. Cette application traite de l'autre face de l'écosystème informatique, à savoir les entreprises et les experts en traitement de données. Ce domaine de recherche florissant a déjà produit de nombreux travaux et solutions, les efforts se concentrent maintenant sur des résultats toujours plus avancés et fiables. Dans ce contexte: les systèmes très complexes peuvent bénéficier de toute l'expérience et de la maturité de la théorie du contrôle. D'autre part, un nouveau champ d'application signifie de nouveaux objectifs, de nouveaux défis et de nouvelles formulations qui feront croître la théorie du contrôle en elle-même.

Au cours des dernières décennies, le monde a été confronté à une forte augmentation du nombre de données produites. Le cas précédemment présenté de la collecte de données sur la mobilité des personnes est un parfait exemple de la façon dont les activités quotidiennes génèrent d'énormes quantités de données : requêtes de pages Web, transactions financières, etc. Ces BigData posent de nouveaux défis quant à leur stockage et leur analyse.

De nouveaux paradigmes de programmation sont apparus pour faire face aux spécificités de Big Data, tels que MapReduce, Hadoop ou Spark. Le cloud computing, qui promet des capacités de stockage et de traitement presque illimitées, devient une solution de plus en plus attrayante. L'utilisation du cloud permet de partager avec d'autres les ressources physiques, car la plupart des applications ne nécessitent pas leur utilisation à plein temps.

L'un des plus grands attraits du cloud computing est l'affectation à la demande de ressources matérielles partagées à des applications logicielles, appelée allocation élastique des ressources. Cela permet de fournir plus de ressources aux applications qui en ont besoin quand elles en ont besoin, et donc d'essayer d'optimiser l'utilisation de ces ressources. Par conséquent, il y a un besoin croissant de solutions pour traiter efficacement la dynamique des ressources des clusters.

Les approches actuelles d'allocation de ressources qui sont déployées dans les clouds publics ne fonctionnent toujours pas de manière optimale pour les applications en temps réel. Dans la plupart des clouds, si une application doit répondre à des critères de temps d'exécution, un algorithme réactif adapte le nombre de ressources à la demande. Cependant, il faut un haut niveau d'expertise pour décider de l'ampleur de l'intervention, car elle dépend de nombreux facteurs qui ne sont pas nécessairement directement accessibles. La difficulté de la tâche est accrue par le fait que la configuration optimale peut varier à cause de l'utilisation partagée des ressources matérielles ou des fluctuations des charges de travail au fil du temps, et de nombreux autres facteurs. Par conséquent, il est nécessaire de disposer d'algorithmes de mise à l'échelle des clusters entièrement automatisés et robustes capables de traiter ces perturbations de l'environnement.

Les fournisseurs de cloud doivent garantir la performance et la disponibilité des services afin de fidéliser les utilisateurs et d'éviter des impacts financiers importants. Garantir la performance des services cloud est un défi extrêmement complexe. Même avec la même charge de travail et la même quantité de ressources, les performances d'une application peuvent varier en fonction du niveau de bruit des applications voisines. De plus, les services cloud sont des paradigmes très complexes qui s'exécutent sur plusieurs couches de logiciels, ce qui rend leur comportement concernant l'approvisionnement en ressources non linéaire.

Les objectifs dans ce contexte sont d'assurer la performance et la disponibilité souhaitées des services cloud tout en étant robustes aux changements de l'application et de son environnement; tout en étant sensible aux coûts financiers et énergétiques.

Contrôle adaptatif pour la garantie des performances de service cloud robustes aux changements du système et de son environnement.

Le coût des services cloud ne cesse de diminuer, ce qui fait que la performance des services est en train de devenir un facteur clé de différenciation entre les fournisseurs. Des solutions qui visent à garantir des objectifs de niveau de service en terme de performance en contrôlant la taille des clusters sont déjà utilisées par les fournisseurs de cloud. Cependant, la plupart de ces solutions de contrôle sont basées sur des règles réactives statiques, elles sont donc inefficaces pour gérer la dynamique de service très variable des environnements cloud.

Dans cette thèse, une nouvelle approche de contrôle adaptatif réalisant l'allocation des ressources est présentée qui est robuste à ces phénomènes. Il se compose d'un PI et d'un régulateur à rétroaction dont les paramètres sont adaptés en ligne. L'utilisation de l'adaptation permet d'améliorer l'efficacité et la robustesse du contrôle par rapport aux variations de la dynamique du système. Cette approche a été validée à la fois en simulation et en utilisant un benchmark MapReduce sur un cluster réel.

Contrôle optimal des performances et de la disponibilité des services cloud en tenant compte des coûts.

Les multiples reconfigurations des clusters est un problème coûteux dans les services cloud traitant des BigData. Les solutions de contrôle actuelles parviennent à faire évoluer le cluster en fonction de la charge de travail, mais elles ne cherchent pas à minimiser le nombre de reconfigurations du système.

Cette thèse présente une nouvelle approche de contrôle optimale déclenchée par des événement (event-based control). Le mécanisme de déclenchement s'appuie sur un contrôleur prédictif basé sur un modèle (Model Predictive Control) et est défini sur la valeur de la fonction du coût, afin de réduire le nombre de changements de contrôle mais aussi d'assurer un comportement très réactif aux changements des entrées exogènes. Les résultats sont prometteurs mais doivent être évalués sur un vrai cloud.

Néanmoins, ces contributions présentent certaines limites, liées par exemple au choix des signaux objectifs de performance ou à l'hypothèse d'homogénéité du cluster. Afin d'accroître la robustesse des approches présentées, il convient de tester les cas où ces hypothèses ne tiennent.

Finalement, même si les expérimentations ont été faites sur un déploiement réel de cluster, Grid5000 est un environnement de recherche avec des facilités de déploiement. L'application de la technique de contrôle présentée sur un framework commercial d'hébergement en nuage MapReduce, tel que Amazon EMR, est encore un défi technique.

En plus de viser à fournir des solutions novatrices pour les cas d'utilisation spécifiques, ce travail permet également de faire un pas en arrière et de tirer des leçons sur les avantages et les difficultés d'utiliser la théorie du contrôle pour adapter les systèmes logiciels.

Les applications informatiques sont régies par des lois qui sont loin des lois de la physique, comme on le considère habituellement dans le cadre de la théorie du contrôle. Cela conduit à des particularités dans les comportements et les contraintes des systèmes. La théorie du contrôle doit être étendue pour faire face à ces nouveaux comportements.

Les avantages de l'utilisation du contrôle pour les différentes applications informatiques sont encore plus précieux. Premièrement, elle fournit une méthodologie bien définie qui permet de relever de nombreux défis en matière de surveillance. Selon la clarté de la formulation du problème, en ce qui concerne les mesures de performance et les boutons de commande, le développement de la modélisation et de la commande peut être simple. A partir de là, le système en boucle fermée surveillé peut être étudié plus avant pour en déduire des garanties mathématiques de stabilité ou de performance. Ceci est très pertinent, par exemple lorsqu'une certification des algorithmes est nécessaire, par exemple dans l'industrie aéronautique.

De plus, certaines limites peuvent être observées. L'hypothèse dynamique est la principale limite de la théorie du contrôle pour de telles applications, tout en étant aussi son principal avantage. Il permet de traiter des formulations de problèmes qui ont à peine été abordées dans le monde informatique alors qu'il n'est d'aucune utilité pour la surveillance statique, même si ce défi statique reste souvent des problèmes non résolus.

En prenant du recul par rapport à la perspective technique, le travail à l'intersection de différents domaines scientifiques induit certaines difficultés qui, ralentissent considérablement le processus de recherche. Par exemple, certains termes scientifiques ne sont pas définis et utilisés de la même manière. De plus, l'étendue des contributions sur des communautés de publication distinctes durcit l'analyse documentaire et la diffusion des contributions.

Examinons maintenant les perspectives générales de l'association de la théorie du contrôle et de l'adaptation logicielle. Suivant les tendances du monde informatique, l'inclusion de nouvelles mesures de performance telles que le niveau de sécurité des applications ou les garanties de protection de la vie privée constitue un axe de recherche prometteur. Ces préoccupations sont parmi les premières pour les industriels et les particuliers alors qu'elles sont à peine prises en compte.

En plus de considérer de nouveaux indicateurs de performance, l'accent devrait être mis sur le contrôle de nouveaux types de logiciels. L'essor récent de l'apprentissage automatique, tant dans le développement de nouveaux algorithmes que dans la diffusion de ses applications, en fait une classe dominante de systèmes qu'il est intéressant d'aborder sous l'angle de la théorie du contrôle. Les applications d'apprentissage automatique donnent des résultats impressionnants en termes de précision, alors que peu de travaux ont été consacrés à l'étude de leur robustesse au bruit et surtout à la garantie de leurs résultats. Ces deux limites sont des domaines pour lesquels la théorie du contrôle possède des décennies d'expertise qui peut profiter aux deux domaines. De plus, les algorithmes d'apprentissage sont considérés comme des outils de modélisation, alors que la plupart du temps, la prédiction de modélisation est destinée à dériver une stratégie d'action. L'association de l'apprentissage avec les outils de contrôle permet de considérer les défis dans leur globalité, de manière à optimiser le processus de décision et non pas nécessairement la précision de la modélisation. Cette direction de recherche a été étudiée en parallèle et à la suite de ce travail de thèse, les principaux résultats sont présentés en annexe de ce manuscrit, voir Partie VI.

La combinaison de la théorie de la régulation et des systèmes logiciels s'est révélée bénéfique pour la communauté des chercheurs, et présente encore des perspectives prometteuses dans lesquelles la science des compromis, de la robustesse et des garanties sera combinée avec les solutions les plus avancées des problèmes les plus importants du monde industriel et social.

Bibliography

[1] Osman Abul, Francesco Bonchi, and Mirco Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 376–385. IEEE, 2008.

[2] Osman Abul, Francesco Bonchi, and Mirco Nanni. Anonymization of moving objects databases by clustering and perturbation. *Information Systems*, 35(8):884 – 910, 2010.

[3] Berker Agir, Thanasis G. Papaioannou, Rammohan Narendula, Karl Aberer, and Jean-Pierre Hubaux. User-side adaptive protection of location privacy in participatory sensing. *GeoInformatica*, 18(1):165–191, 2014.

[4] Faraz AHMAD, Seyong LEE, Mithuna THOTTETHODI, and T Vijaykumar. Puma: Purdue mapreduce benchmarks suite. Technical report, 2012.

[5] Dhiya Al-Jumeily, Abir Hussain, and Paul Fergus. Using adaptive neural networks to provide self-healing autonomic software. *International Journal of Space-Based and Situated Computing*, 5(3):129–140, 2015.

[6] M. Alamir and G. Bornard. On the stability of receding horizon control of nonlinear discrete-time systems. *Systems & Control Letters*, 23(4):291–296, 1994.

[7] A. Ali-Eldin, J. Tordsson, and E. Elmroth. An adaptive hybrid elasticity controller for cloud infrastructures. In *IEEE Network Operations and Management Symposium (NOMS)*, pages 204–212. IEEE, 2012.

[8] Ahmed Ali-Eldin, Maria Kihl, Johan Tordsson, and Erik Elmroth. Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control. In *Proceedings of the 3rd workshop on Scientific Cloud Computing*, ScienceCloud '12, pages 31–40. ACM, ACM, 2012.

[9] Cloud Security Alliance. Custom Applications and IaaS Trends 2017. `https://cloudsecurityalliance.org/artifacts/custom-applications-and-iaas-trends-2017/`, 2019-01-04.

[10] Jussara Almeida, Virgilio Almeida, Danilo Ardagna, Chiara Francalanci, and Marco Trubian. Resource management in the autonomic service-oriented architecture. In *Autonomic Computing, 2006. ICAC'06. IEEE International Conference on*, pages 84–92. IEEE, 2006.

[11] Aymen Abdullah Alsaffar, Hung Phuoc Pham, Choong-Seon Hong, Eui-Nam Huh, and Mohammad Aazam. An architecture of iot service delegation and resource allocation based on collaboration between fog and cloud computing. *Mobile Information Systems*, 2016, 2016.

[12] Amazon. Amazon web services AWS Auto Scaling. https://aws.amazon.com/autoscaling/, 2018-11-07.

[13] Amazon. EC2 Auto Scaling. https://aws.amazon.com/ec2/autoscaling/, 2018-11-07.

[14] Amazon. Amazon Web Services (AWS) - Cloud Computing Services. https://aws.amazon.com/, 2019-01-04.

[15] Amazon. Amazon Web Services - elastic MapReduce EMR. https://aws.amazon.com/emr/, 2019-01-05.

[16] Amazon. Amazon web services EC2. https://aws.amazon.com/ec2/, 2019-01-28.

[17] Xin An, Eric Rutten, Jean-Philippe Diguet, and Abdoulaye Gamatié. Model-based design of correct controllers for dynamically reconfigurable architectures. *ACM Transactions on Embedded Computing Systems (TECS)*, 15(3):51, 2016.

[18] Miguel E. Andrés, Nicolás E. Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: Differential Privacy for Location-based Systems. In *CCS*, pages 901–914. ACM, 2013.

[19] Julio C.S. Anjos, Ivan Carrera, Wagner Kolberg, Andre Luis Tibola, Luciana B. Arantes, and Claudio R. Geyer. Mra++: Scheduling and data placement on mapreduce for heterogeneous environments. *Future Generation Computer Systems*, 42:22 – 35, 2015.

[20] Ismail Ari, Bo Hong, Ethan L Miller, Scott A Brandt, and Darrell DE Long. Managing flash crowds on the internet. In *11th IEEE/ACM Int. Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems*, pages 246–249, 2003.

[21] K. E. Årzén. A simple event-based PID controller. In *Preprints of the 14th World Congress of IFAC*, 1999.

[22] K. J. Aström. Event based control. In Alessandro Astolfi and Lorenzo Marconi, editors, *Analysis and Design of Nonlinear Control Systems*, pages 127–147. Springer Berlin Heidelberg, 2008.

[23] K. J. Åström and T. Hägglund. *PID controllers: theory, design, and tuning, 2nd Edition*, volume 2. The Instrumentation, Systems, and Automation Society, 1995.

[24] M. Berekmeri, D. Serrano, S. Bouchenak, N. Marchand, and B. Robu. A control approach for performance of big data systems. In *Proceeedings of the 19th World Congress of IFAC*, 2014.

[25] Mihaly Berekmeri. *Modeling and control of cloud services : application to MapReduce performance and dependability*. Theses, Université Grenoble Alpes, November 2015.

[26] Mihaly Berekmeri, Damián Serrano, Sara Bouchenak, Nicolas Marchand, and Bogdan Robu. Feedback Autonomic Provisioning for Guaranteeing Performance in MapReduce Systems. *IEEE Transactions on Cloud Computing*, 6(4):1004–1016, 2016.

[27] Alastair R Beresford and Frank Stajano. Location privacy in pervasive computing. *IEEE Pervasive computing*, (1):46–55, 2003.

[28] Alastair R Beresford and Frank Stajano. Mix zones: User privacy in location-aware services. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 127–131. IEEE, 2004.

[29] Igor Bilogrevic, Kévin Huguenin, Murtuza Jadliwala, Florent Lopez, Jean-Pierre Hubaux, Philip Ginzboorg, and Valtteri Niemi. Inferring Social Ties in Academic Networks Using Short-Range Wireless Communications. *Wpes*, pages 179–188, 2013.

[30] Vincent Bindschaedler and Reza Shokri. Synthesizing plausible privacy-preserving location traces. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 546–563. IEEE, 2016.

[31] Antoine Boutet, Sonia Ben Mokhtar, Louafi Bouzouina, Patrick Bonnel, Olivier Brette, Lionel Brunie, Mathieu Cunche, Stephane D'alu, Vincent Primault, Patrice Raveneau, Herve Rivano, and Razvan Stanica. PRIVA'MOV: Analysing Human Mobility Through Multi-Sensor Datasets. In *NetMob*, 2017.

[32] I. Boutsis and V. Kalogeraki. Privacy preservation for participatory sensing data. In *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 103–113, March 2013.

[33] Ioannis Boutsis and Vana Kalogeraki. *Location Privacy-Preserving Applications and Services*, pages 373–398. Springer International Publishing, 2018.

[34] Kai-Yuan Cai, Bo Gu, Hai Hu, and Yong-Chao Li. Adaptive software testing with fixed-memory feedback. *Journal of Systems and Software*, 80(8):1328–1348, 2007.

[35] Lorela Cano, Giuliana Carello, and Danilo Ardagna. A framework for joint resource allocation of mapreduce and web service applications in a shared cloud cluster. *Journal of Parallel and Distributed Computing*, 2018.

[36] F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jegou, P. Primet, E. Jeannot, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, B. Quetier, and O. Richard. Grid'5000: A large scale and highly reconfigurable grid experimental testbed. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pages 99–106, 2005.

[37] Christos G Cassandras and Stephane Lafortune. *Introduction to discrete event systems*. Springer Science & Business Media, 2009.

[38] Sophie Cerf, Sonia Ben Mokhtar, Sara Bouchenak, Nicolas Marchand, and Bogdan Robu. Dynamic Modeling of Location Privacy Protection Mechanisms. In Silvia Bonomi and Etienne Rivière, editors, *18th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS 2018), Held as Part of the 13th International Federated Conference on Distributed Computing Techniques (DisCoTec 2018)*, volume LNCS-10853 of *Lecture Notes in Computer Science*, pages 26–39, Madrid, Spain, June 2018. Springer International Publishing.

[39] Sophie Cerf, Mihaly Berekmeri, Nicolas Marchand, Sara Bouchenak, and Bogdan Robu. Adaptive Modelling and Control in Distributed Systems. In *Phd Forum 34th International Symposium on Reliable Distributed Systems (SRDS) 2015*, Montreal, Canada, September 2015. McGill University.

[40] Sophie Cerf, Mihaly Berekmeri, Bogdan Robu, Nicolas Marchand, and Sara Bouchenak. Adaptive Optimal Control of MapReduce Performance, Availability and Costs. In *11th International Workshop on Feedback Computing (Feedback Computing 2016)*, Wurzburg, Germany, July 2016.

[41] Sophie Cerf, Mihaly Berekmeri, Bogdan Robu, Nicolas Marchand, and Sara Bouchenak. Cost Function based Event Triggered Model Predictive Controllers - Application to Big Data Cloud Services. In *55th IEEE Conference on Decision and Control (CDC 2016)*, Proceedings of the 55th IEEE International Conference on Decision and Control, Las Vegas, NV, United States, December 2016.

[42] Sophie Cerf, Mihaly Berekmeri, Bogdan Robu, Nicolas Marchand, and Sara Bouchenak. Towards Control of MapReduce Performance and Availability. In Matthieu Roy, Javier Alonso Lopez, and Antonio Casimiro, editors, *46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2016)*, DSN2016-FAST-ABSTRACT, Toulouse, France, June 2016.

[43] Sophie Cerf, Mihaly Berekmeri, Bogdan Robu, Nicolas Marchand, Sara Bouchenak, and Ioan Doré Landau. Adaptive Feedforward and Feedback Control for Cloud Services. In *20th IFAC World Congress (IFAC WC 2017)*, volume 50 of *20th IFAC World Congress*, pages 5504 – 5509, Toulouse, France, July 2017.

[44] Sophie Cerf, Robert Birke, and Lydia Y. Chen. Duo Learning for Classifications with Noisy Labels. In *Continual Learning Workshop, Neural Information Processing Systems (NIPS)*, Montréal, Canada, December 2018.

[45] Sophie Cerf, Sara Bouchenak, Bogdan Robu, Nicolas Marchand, Vincent Primault, Sonia Ben Mokhtar, Antoine Boutet, and Lydia Y. Chen. Automatic Privacy and Utility Preservation of Mobility Data: A Nonlinear Model-Based Approach. *IEEE Transactions on Dependable and Secure Computing*, 2019. (to appear).

[46] Sophie Cerf, Vincent Primault, Antoine Boutet, Sonia Ben Mokhtar, Sara Bouchenak, Nicolas Marchand, and Bogdan Robu. Données de mobilité : protection de la vie privée vs. utilité des données. In *Conférence francophone d'informatique en parallélisme, architecture et système (ComPAS)*, Proceedings of the Conférence francophone d'informatique en parallélisme, architecture et système (ComPAS), Sophia Antipolis, France, June 2017.

[47] Sophie Cerf, Vincent Primault, Antoine Boutet, Sonia Ben Mokhtar, Robert Birke, Sara Bouchenak, Lydia Y Chen, Nicolas Marchand, and Bogdan Robu. Pulp: Achieving privacy and utility trade-off in user mobility data. In *Reliable Distributed Systems (SRDS), 2017 IEEE 36th Symposium on*, pages 164–173. IEEE, 2017.

[48] Sophie Cerf, Bogdan Robu, Nicolas Marchand, Antoine Boutet, Vincent Primault, Sonia Ben Mokhtar, and Sara Bouchenak. Toward an Easy Configuration of Location Privacy Protection Mechanisms. In *ACM/IFIP/USENIX Middleware conference*, Trente, Italy, December 2016. Communication orale sur poster.

[49] Sophie Cerf, Bogdan Robu, Nicolas Marchand, Sonia Ben Mokhtar, and Sara Bouchenak. A control-theoretic approach for location privacy in mobile applications. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1488–1493. IEEE, 2018.

[50] Abhishek Chandra, Weibo Gong, and Prashant Shenoy. Dynamic resource allocation for shared data centers using online measurements. In *International Workshop on Quality of Service*, pages 381–398. Springer, 2003.

[51] Abhishek Chandra, Prashant Pradhan, Renu Tewari, Sambit Sahu, and Prashant Shenoy. An observation-based approach towards self-managing web servers. *Computer Communications*, 29(8):1174–1188, 2006.

[52] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Marco Stronati. A predictive differentially-private mechanism for mobility traces. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 21–41. Springer, 2014.

[53] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Marco Stronati. Constructing elastic distinguishability metrics for location privacy. In *PETS*, volume 2015, pages 156–170, 2015.

[54] Min Chen, Shiwen Mao, and Yunhao Liu. Big data: A survey. *Mobile networks and applications*, 19(2):171–209, 2014.

[55] C Cheng, Haiqin Yang, Michael Lyu, and Irwin King. Where you like to go next: Successive point-of-interest recommendation. *IJCAI International Joint Conference on Artificial Intelligence*, pages 2605–2611, 01 2013.

[56] Chi-Yin Chow, Mohamed F Mokbel, and Xuan Liu. A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In *Proceedings of the 14th ACM international symposium on Advances in geographic information systems*, pages 171–178. ACM, 2006.

[57] CloudTimes. Microsoft says to spend 90% of r&d on cloud strategy. `https://web.archive.org/web/20131018050315/http://cloudtimes.org/2011/04/12/microsoft-says-to-spend-90-of-rd-on-cloud-strategy/`, 2011.

[58] Rogério De Lemos, Holger Giese, Hausi A Müller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Gabriel Tamura, Norha M Villegas, Thomas Vogel, et al. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*, pages 1–32. Springer, 2013.

[59] T. De Ruiter. *A workload model for MapReduce*. PhD thesis, TU Delft, Delft University of Technology, 2012.

[60] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[61] MCF Donkers and WPMH Heemels. Output-based event-triggered control with guaranteed-gain and improved and decentralized event-triggering. *Automatic Control, IEEE Transactions on*, 57(6):1362–1376, 2012.

[62] John C Doyle, Bruce A Francis, and Allen R Tannenbaum. *Feedback control theory*. Courier Corporation, 2013.

[63] Matt Duckham and Lars Kulik. A formal model of obfuscation and negotiation for location privacy. In *International conference on pervasive computing*, pages 152–170. Springer, 2005.

[64] S. Durand and N. Marchand. Further results on event-based pid controller. In *Proceedings of the European Control Conference (ECC)*, 2009.

[65] Cynthia Dwork. Differential Privacy. In *Automata, Languages and Programming*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2006.

[66] A. Eqtami, D. V. Dimarogonas, and K. J. Kyriakopoulos. Event-triggered control for discrete-time systems. In *Proceedings of the IEEE American Control Conference*, 2010.

[67] Evolven. Downtime, outages and failures - understanding their true costs. http://www.evolven.com/, 18 September 2013.

[68] Shaheen S Fatima, Michael Wooldridge, and Nicholas R Jennings. A comparative study of game theoretic and evolutionary models of bargaining for software agents. *Artificial Intelligence Review*, 23(2):187–205, 2005.

[69] Kassem Fawaz and Kang G. Shin. Location privacy protection for smartphone users. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 239–250, New York, NY, USA, 2014. ACM.

[70] Andrew D. Ferguson, Peter Bodik, Srikanth Kandula, Eric Boutin, and Rodrigo Fonseca. Jockey: guaranteed job latency in data parallel clusters. In Pascal Felber, Frank Bellosa, and Herbert Bos, editors, *EuroSys*, pages 99–112. ACM, 2012.

[71] Antonio Filieri, Martina Maggio, Konstantinos Angelopoulos, Nicolas D'Ippolito, Ilias Gerostathopoulos, Andreas Hempel, Henry Hoffmann, Pooyan Jamshidi, Evangelia Kalyvianaki, Cristian Klein, Filip Krikava, Sasa Misailovic, Vittorio Papadopoulos, Alessandro, Suprio Ray, Molzam Sharifloo, Amir, Stepan Shevtsov, Mateusz Ujma, and Thomas Vogel. Software engineering meets control theory. In *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 71–82, Firenze, Italy, May 2015. IEEE Press.

[72] Lorenzo Franceschi-Bicchierai. Redditor cracks anonymous data trove to pinpoint muslim cab drivers. `http://mashable.com/2015/01/28/redditor-muslim-cab-drivers/`, January 2015.

[73] Sebastien Gambs, Marc-Olivier Killijian, and Miguel Nunez del Prado Cortez. De-anonymization Attack on Geolocated Data. *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 789–797, 2013.

[74] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. Show Me How You Move and I Will Tell You Who You Are. *Transactions on Data Privacy*, 4(2):103–126, August 2011.

[75] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. Next place prediction using mobility markov chains. In *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*, page 3. ACM, 2012.

[76] Anshul Gandhi, Mor Harchol-Balter, Ram Raghunathan, and Michael A Kozuch. Autoscale: Dynamic, robust capacity management for multi-tier data centers. *ACM Transactions on Computer Systems (TOCS)*, 30(4):14, 2012.

[77] Jim Gao and Ratnesh Jamidar. Machine learning applications for data center optimization. *Google White Paper*, 2014.

[78] Eloy Garcia and Panos J Antsaklis. Model-based event-triggered control for systems with quantization and time-varying network delays. *Automatic Control, IEEE Transactions on*, 58(2):422–434, 2013.

[79] GDPR. reform of eu data protection rules, May 2018.

[80] Bugra Gedik and Ling Liu. A customizable k-anonymity model for protecting location privacy. Technical report, Georgia Institute of Technology, 2004.

[81] Bugra Gedik and Ling Liu. Location privacy in mobile systems: A personalized anonymization model. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 620–629. IEEE, 2005.

[82] Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian-Lee Tan. Private queries in location based services: anonymizers are not necessary. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 121–132. ACM, 2008.

[83] Bogdan Ghit, Nezih Yigitbasi, Alexandru Iosup, and Dick Epema. Balanced resource allocations across multiple dynamic mapreduce clusters. *SIGMETRICS Perform. Eval. Rev.*, 42(1):329–341, June 2014.

[84] Alain Girault and Éric Rutten. Automating the addition of fault tolerance with discrete controller synthesis. *Formal Methods in System Design*, 35(2):190, 2009.

[85] Z. Gong, X. Gu, and J. Wilkes. Press: Predictive elastic resource scaling for cloud systems. In *International Conference on Network and Service Management (CNSM)*, pages 9–16. IEEE, 2010.

[86] Graham C Goodwin, Stefan F Graebe, and Mario E Salgado. Control system design. *Upper Saddle River*, 13, 2001.

[87] Daniel Grosu and Anthony T Chronopoulos. Noncooperative load balancing in distributed systems. *Journal of parallel and distributed computing*, 65(9):1022–1034, 2005.

[88] The Guardian. Fitness tracking app strava gives away location of secret us army bases. `https://www.theguardian.com/world/2018/jan/28/fitness-tracking-app-gives-away-location-of-secret-us-army-bases`, January 2018.

[89] Jordi Guitart, Jordi Torres, and Eduard Ayguadé. A survey on performance management for internet applications. *Concurrency and Computation: Practice and Experience*, 22(1):68–106, 2010.

[90] Jian Guo, Fangming Liu, Dan Zeng, John CS Lui, and Hai Jin. A cooperative game based allocation for sharing data center networks. In *INFOCOM, 2013 Proceedings IEEE*, pages 2139–2147. IEEE, 2013.

[91] Joseph L Hellerstein, Yixin Diao, Sujay Parekh, and Dawn M Tilbury. *Feedback control of computing systems*. John Wiley & Sons, Inc., New Jersey, 2004.

[92] Baik Hoh and Marco Gruteser. Protecting location privacy through path confusion. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 194–205. IEEE, 2005.

[93] Connor Imes, David HK Kim, Martina Maggio, and Henry Hoffmann. Poet: a portable approach to minimizing energy under soft real-time constraints. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2015 IEEE*, pages 75–86. IEEE, 2015.

[94] Pooyan Jamshidi, Aakash Ahmad, and Claus Pahl. Autonomic resource provisioning for cloud-based software. In *Proceedings of the 9th international symposium on software engineering for adaptive and self-managing systems*, pages 95–104. ACM, 2014.

[95] Pooyan Jamshidi, Amir Sharifloo, Claus Pahl, Andreas Metzger, and Giovani Estrada. Self-learning cloud controllers: Fuzzy q-learning for knowledge evolution. *arXiv preprint arXiv:1507.00567*, 2015.

[96] Yonghua Ji, Vijay S Mookerjee, and Suresh P Sethi. Optimal software development: A control theoretic approach. *Information Systems Research*, 16(3):292–306, 2005.

[97] Kaifeng Jiang, Dongxu Shao, Stéphane Bressan, Thomas Kister, and Kian-Lee Tan. Publishing trajectories with differential privacy guarantees. In *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*, page 12. ACM, 2013.

[98] Jeffrey O Kephart and David M Chess. The vision of autonomic computing. *Computer*, (1):41–50, 2003.

[99] Eric C. Kerrigan. Feedback and time are essential for the optimal control of computing systems. *IFAC-PapersOnLine*, 48(23):380 – 387, 2015. 5th IFAC Conference on Nonlinear Model Predictive Control NMPC 2015.

[100] Hidetoshi Kido, Yutaka Yanagisawa, and Tetsuji Satoh. Protection of location privacy using dummies for location-based services. In *Data Engineering Workshops. 21st International Conference on*, pages 1248–1248. IEEE, 2005.

[101] Kiyoung Kim, Kyungho Jeon, Hyuck Han, Shin-gyu Kim, Hyungsoo Jung, and Heon Y Yeom. Mrbench: A benchmark for mapreduce framework. In *Parallel and Distributed Systems, 2008. ICPADS'08. 14th IEEE International Conference on*, pages 11–18. IEEE, 2008.

[102] N. Kiukkonen, Blom J., O. Dousse, Daniel Gatica-Perez, and Laurila J. Towards rich mobile phone datasets: Lausanne data collection campaign. In *ICPS*, 2010.

[103] Younggyun Koh, Rob Knauerhase, Paul Brett, Mic Bowman, Zhihua Wen, and Calton Pu. An analysis of performance interference effects in virtual environments. In *IEEE Int. Symposium on Performance Analysis of Systems & Software*, pages 200–209, 2007.

[104] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. Spotfi: Decimeter level localization using wifi. *SIGCOMM Comput. Commun. Rev.*, 45(4):269–282, August 2015.

[105] Fragkiskos Koufogiannis and George J Pappas. Location-dependent privacy. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 7586–7591. IEEE, 2016.

[106] John Krumm. Inference attacks on location tracks. In *International Conference on Pervasive Computing*, pages 127–143. Springer, 2007.

[107] John Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6):391–399, 2009.

[108] Philippe Lalanda, Ada Diaconescu, and Mccann A. Julie. *Autonomic Computing - Principles, Design and Implementation*. Undergraduate Topics in Computer Science. Springer, May 2013.

[109] Ioan D Landau, Tudor-Bogdan Airimițoaie, Abraham Castellanos-Silva, and Aurelian Constantinescu. *Adaptive and Robust Active Vibration Control: Methodology and Tests*. Springer, 2016.

[110] Ioan D Landau, Jaime Saavedra, Sophie Cerf, Bogdan Robu, Nicolas March, and Sara Bouchenak. Can adaptive feedforward control improve operation of cloud services? In *2018 26th Mediterranean Conference on Control and Automation (MED)*, pages 1–9. IEEE, 2018.

[111] Ioan Doré Landau, Rogelio Lozano, Mohammed M'Saad, and Alireza Karimi. *Adaptive control: algorithms, analysis and applications*. Springer Science & Business Media, 2011.

[112] Juha K. Laurila, Daniel Gatica-Perez, Imad Aad, Jan Blom, Olivier Bornet, Trinh Minh Tri Do, Olivier Dousse, Julien Eberle, and Markus Miettinen. From big smartphone data to worldwide research: The mobile data challenge. *Pervasive Mob. Comput.*, 9(6):752–771, December 2013.

[113] Zhao Li, Yao Shen, Bin Yao, and Minyi Guo. Ofscheduler: A dynamic network optimizer for mapreduce in heterogeneous cluster. *International Journal of Parallel Programming*, 43(3):472–488, 2015.

[114] N. Lim, S. Majumdar, and P. Ashwood-Smith. Mrcp-rm: A technique for resource allocation and scheduling of mapreduce jobs with deadlines. *IEEE Transactions on Parallel and Distributed Systems*, 28(5):1375–1389, May 2017.

[115] Xinxin Liu, Han Zhao, Miao Pan, Hao Yue, Xiaolin Li, and Yuguang Fang. Traffic-aware multiple mix zone placement for protecting location privacy. In *INFOCOM, 2012 Proceedings IEEE*, pages 972–980. IEEE, 2012.

[116] Xue Liu, Jin Heo, Lui Sha, and Xiaoyun Zhu. Adaptive control of multi-tiered web applications using queueing predictor. In *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, pages 106–114. IEEE, 2006.

[117] Zhen Liu, Mark S Squillante, and Joel L Wolf. On maximizing service-level-agreement profits. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 213–223. ACM, 2001.

[118] Zhihong Liu, Qi Zhang, Mohamed Faten Zhani, Raouf Boutaba, Yaping Liu, and Zhenghu Gong. Dreams: Dynamic resource allocation for mapreduce with data skew. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 18–26. IEEE, 2015.

[119] T. Lorido-Botran, J. Miguel-Alonso, and J. Lozano. A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of Grid Computing*, 12(4):559–592, 2014.

[120] J. Lunze and D. Lehmann. A state-feedback approach to event-based control. *Automatica*, 46:211–215, 2010.

[121] Frank D Macías-Escrivá, Rodolfo Haber, Raul Del Toro, and Vicente Hernandez. Self-adaptive systems: A survey of current approaches, research challenges and applications. *Expert Systems with Applications*, 40(18):7267–7279, 2013.

[122] Martina Maggio, Enrico Bini, Georgios Chasparis, and Karl-Erik Årzén. A game-theoretic resource manager for rt applications. In *Real-Time Systems (ECRTS), 2013 25th Euromicro Conference on*, pages 57–66. IEEE, 2013.

[123] N. Maheshwari, R. Nanduri, and V. Varma. Dynamic energy efficient data placement and cluster reconfiguration algorithm for mapreduce framework. *Future Generation Computer Systems*, 28(1):119–127, 2012.

[124] L. Malrait, S. Bouchenak, and N. Marchand. Fluid modeling and control for server system performance and availability. In *Proceedings of the 39th annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2009. inria-00367666.

[125] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pages 50–56. ACM, 2016.

[126] Mohamed Maouche, Sonia Ben Mokhtar, and Sara Bouchenak. Ap-attack: A novel user reidentification attack on mobility datasets. In *MobiQuitous*. ACM, 2017.

[127] Nicolas Marchand, Sylvain Durand, and Jose Fermi Guerrero Castellanos. A general formula for event-based stabilization of nonlinear systems. *IEEE Transactions on Automatic Control*, 58(5):1332–1337, 2013.

[128] S. Maroulis, I. Boutsis, and V. Kalogeraki. Context-aware point of interest recommendation using tensor factorization. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 963–968, Dec 2016.

[129] A. Matsunaga and J. Fortes. On the use of machine learning to predict the time and resources consumed by applications. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 495–504, 2010.

[130] D. Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *IEEE Trans. on Automatic Control*, 35(7):814–824, 1990.

[131] Kristopher Micinski, Philip Phelps, and Jeffrey S Foster. An empirical study of location truncation on android. *Weather*, 2:21, 2013.

[132] Microsoft. Azure Cloud Computing Platform & Services. *https://azure.microsoft.com/en-us/*, 2019-01-04.

[133] Microsoft. Azure HDInsight - Hadoop, Spark, & Kafka Service | Microsoft Azure. *https://azure.microsoft.com/en-us/services/hdinsight/*, 2019-01-05.

[134] Darakhshan J Mir, Sibren Isaacman, Ramón Cáceres, Margaret Martonosi, and Rebecca N Wright. Dp-where: Differentially private modeling of human mobility. In *Big Data, 2013 IEEE International Conference on*, pages 580–588. IEEE, 2013.

[135] D. Mituzas. Wikipedia embarrassment. http://dom.as/2009/06/26/embarrassment/, 29 June 2009.

[136] Mohamed F Mokbel, Chi-Yin Chow, and Walid G Aref. The new casper: Query processing for location services without compromising privacy. In *Proceedings of the 32nd international conference on Very large data bases*, pages 763–774. VLDB Endowment, 2006.

[137] Fiona Fui-Hoon Nah. A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & Information Technology*, 23(3):153–163, 2004.

[138] John Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

[139] Navigation National Coordination Office for Space-Based Positioning and Timing. Gps accuracy, 2017.

[140] Hoa Ngo and Jong Kim. Location privacy via differential private perturbation of cloaking area. In *Computer Security Foundations Symposium (CSF), 2015 IEEE 28th*, pages 63–74. IEEE, 2015.

[141] H. Nguyen, Z. Shen, X. Gu, S. Subbiah, and J. Wilkes. Agile: Elastic distributed resource scaling for infrastructure-as-a-service. In *Proceedings of the 10th International Conference on Autonomic Computing*, pages 69–82. USENIX, 2013.

[142] Nyansa. Voyance Live. `https://www.nyansa.com/voyancelive/`, May 2016.

[143] T. Nylander, M. Thelander Andrén, K. Årzén, and M. Maggio. Cloud application predictability through integrated load-balancing and service time control. In *2018 IEEE International Conference on Autonomic Computing (ICAC)*, pages 51–60, Sep. 2018.

[144] Peyman Oreizy, Nenad Medvidovic, and Richard N Taylor. Runtime software adaptation: framework, approaches, and styles. In *Companion of the 30th international conference on Software engineering*, pages 899–910. ACM, 2008.

[145] Overleaf. `https://twitter.com/overleaf/status/1063595461540495360`, Nov 2018.

[146] Xiao Pan, Jianliang Xu, and Xiaofeng Meng. Protecting location privacy against location-dependent attacks in mobile services. *IEEE Transactions on Knowledge and Data Engineering*, 24(8):1506–1519, 2012.

[147] Tharindu Patikirikorala, Alan Colman, Jun Han, and Liuping Wang. A systematic survey on the design of self-adaptive software systems using control engineering approaches. In *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012 ICSE Workshop on*, pages 33–42. IEEE, June 2012.

[148] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. CRAWDAD dataset epfl/mobility (v. 2009-02-24). Downloaded from http://crawdad.org/epfl/mobility/20090224, February 2009.

[149] Vincent Primault. *Practically Preserving and Evaluating Location Privacy*. PhD thesis, Université de Lyon; INSA Lyon, 2018.

[150] Vincent Primault, Sonia Ben Mokhtar, Cédric Lauradoux, and Lionel Brunie. Time distortion anonymization for the publication of mobility data with high utility. In *TrustCom*, pages 539–546, 2015.

[151] Vincent Primault, Antoine Boutet, Sonia Ben Mokhtar, and Lionel Brunie. Adaptive location privacy with alp. In *Reliable Distributed Systems (SRDS), 2016 IEEE 35th Symposium on*, pages 269–278. IEEE, 2016.

[152] Vincent Primault, Antoine Boutet, Sonia Ben Mokhtar, and Lionel Brunie. The long road to computational location privacy: A survey. *IEEE Communications Surveys & Tutorials*, 2018.

[153] Peter JG Ramadge and W Murray Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[154] Supranamaya Ranjan, Jerome Rolia, Ho Fu, and Edward Knightly. Qos-driven server migration for internet data centers. In *Quality of Service, 2002. Tenth IEEE International Workshop on*, pages 3–12. IEEE, 2002.

[155] Z. Ren, X. Xu, J. Wan, W. Shi, and M. Zhou. Workload characterization on a production Hadoop cluster: A case study on Taobao. In *IEEE International Symposium on Workload Characterization*, pages 3–13, 4-6 Nov 2012.

[156] Eric Rutten, Nicolas Marchand, and Daniel Simon. Feedback control as mape-k loop in autonomic computing. In *Software Engineering for Self-Adaptive Systems III. Assurances*, pages 349–373. Springer, 2017.

[157] S2, a spherical geometry library. Available online at `https://github.com/google/s2-geometry-library-java`.

[158] A. Sangroya, D. Serrano, and S. Bouchenak. Benchmarking dependability of mapreduce systems. In *Reliable Distributed Systems (SRDS), 2012 IEEE 31st Symposium on*, pages 21–30, Oct 2012.

[159] Amit Sangroya, Damián Serrano, and Sara Bouchenak. Benchmarking Dependability of MapReduce Systems. In *IEEE 31st Symposium on Reliable Distributed Systems (SRDS)*, pages 21 – 30, Irvine, CA, 8-11 Oct. 2012.

[160] Gautham Nayak Seetanadi, Luis Oliveira, Luis Almeida, Karl-Erik Arzén, and Martina Maggio. Game-theoretic network bandwidth distribution for self-adaptive cameras. *SIGBED Rev.*, 15(3):31–36, August 2018.

[161] Pravin Shankar, Vinod Ganapathy, and Liviu Iftode. Privately querying location-based services with sybilquery. In *Proceedings of the 11th international conference on Ubiquitous computing*, pages 31–40. ACM, 2009.

[162] Stepan Shevtsov, Mihaly Berekmeri, Danny Weyns, and Martina Maggio. Control-theoretical software adaptation: A systematic literature review. *IEEE Transactions on Software Engineering*, 44(8):784–810, 2018.

[163] Lingyang Song, Dusit Niyato, Zhu Han, and Ekram Hossain. Game-theoretic resource allocation methods for device-to-device communication. *IEEE Wireless Communications*, 21(3):136–144, 2014.

[164] Vítor E Silva Souza, Alexei Lapouchnian, and John Mylopoulos. (requirement) evolution requirements for adaptive systems. In *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 155–164. IEEE Press, 2012.

[165] Leon Stenneth, S Yu Phillip, and Ouri Wolfson. Mobile systems location privacy:“mobipriv” a robust k anonymous system. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2010 IEEE 6th International Conference on*, pages 54–63. IEEE, 2010.

[166] Riky Subrata, Albert Y Zomaya, and Bjorn Landfeldt. A cooperative game framework for qos guided job allocation schemes in grids. *IEEE Transactions on Computers*, 57(10):1413–1422, 2008.

[167] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[168] Adja Ndeye Sylla, Maxime Louvel, Eric Rutten, and Gwenaël Delaval. Modular and hierarchical discrete control for applications and middleware deployment in iot and smart buildings. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1472–1479. IEEE, 2018.

[169] M. Thammawichai and E. C. Kerrigan. Energy-Efficient Real-Time Scheduling for Two-Type Heterogeneous Multiprocessors. *arXiv e-prints*, July 2016.

[170] George Theodorakopoulos, Reza Shokri, Carmela Troncoso, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. Prolonging the hide-and-seek game: Optimal trajectory privacy for location-based services. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 73–82. ACM, 2014.

[171] Manel Velasco, Pau Martí, and Enrico Bini. On lyapunov sampling for event-driven controllers. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 6238–6243. IEEE, 2009.

[172] Abhishek Verma, Ludmila Cherkasova, and Roy H. Campbell. Aria: automatic resource inference and allocation for mapreduce environments. In *Proceedings of the 8th ACM international conference on Autonomic computing*, ICAC '11, pages 235–244, New York, NY, USA, 2011. ACM.

[173] Manish Verma, GR Gangadharan, Nanjangud C Narendra, Ravi Vadlamani, Vidyadhar Inamdar, Lakshmi Ramachandran, Rodrigo N Calheiros, and Rajkumar Buyya. Dynamic resource demand prediction and allocation in multi-tenant service clouds. *Concurrency and Computation: Practice and Experience*, 28(17):4429–4442, 2016.

[174] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior (commemorative edition)*. Princeton university press, 2007.

[175] D Vrancic. *Design of anti-windup and bumpless transfer protection.* PhD thesis, University of Ljubljana, J. Stefan Institute, 1996.

[176] Guanying Wang, Ali R Butt, Prashant Pandey, and Karan Gupta. Using realistic simulation for performance analysis of mapreduce setups. In *Proceedings of the 1st ACM workshop on Large-Scale system and application performance*, LSAP '09, pages 19–26, New York, NY, USA, 2009. ACM, ACM.

[177] Jun-Bo Wang, Junyuan Wang, Yongpeng Wu, Jin-Yuan Wang, Huiling Zhu, Min Lin, and Jiangzhou Wang. A machine learning framework for resource allocation assisted by cloud computing. *IEEE Network*, 32(2):144–151, 2018.

[178] Yin Wang, Hyoun Kyu Cho, Hongwei Liao, Ahmed Nazeem, Terence P Kelly, Stéphane Lafortune, Scott Mahlke, and Spyros A Reveliotis. Supervisory control of software execution for failure avoidance: Experience from the gadara project. *IFAC Proceedings Volumes*, 43(12):259–266, 2010.

[179] Yu Wang, Dingbang Xu, Xiao He, Chao Zhang, Fan Li, and Bin Xu. L2p2: Location-aware location privacy protection for location-based services. In *INFOCOM, 2012 Proceedings IEEE*, pages 1996–2004. IEEE, 2012.

[180] Guiyi Wei, Athanasios V Vasilakos, Yao Zheng, and Naixue Xiong. A game-theoretic method of fair resource allocation for cloud computing services. *The journal of supercomputing*, 54(2):252–269, 2010.

[181] Wei Wei, Xunli Fan, Houbing Song, Xiumei Fan, and Jiachen Yang. Imperfect information dynamic stackelberg game based resource allocation using hidden markov for cloud computing. *IEEE Transactions on Services Computing*, 11(1):78–89, 2018.

[182] Yonghui Xiao and Li Xiong. Protecting locations with differential privacy under temporal correlations. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1298–1309. ACM, 2015.

[183] Toby Xu and Ying Cai. Feeling-based location privacy protection for location-based services. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 348–357. ACM, 2009.

[184] Gökhan Yavaş, Dimitrios Katsaros, Özgür Ulusoy, and Yannis Manolopoulos. A data mining approach for location prediction in mobile environments. *Data & Knowledge Engineering*, 54(2):121–146, 2005.

[185] Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik-Lun Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 325–334, New York, NY, USA, 2011. ACM.

[186] Nikos Zacheilas and Vana Kalogeraki. A pareto-based scheduler for exploring cost-performance trade-offs for mapreduce workloads. *EURASIP Journal on Embedded Systems*, 2017(1):29, Jul 2017.

[187] Zilong Zhao, Sophie Cerf, Robert Birke, Bogdan Robu, Sara Bouchenak, Sonia Ben Mokhtar, and Lydia Y. Chen. Robust Anomaly Detection on Unreliable Data. In *49th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Portland, Oregon, USA, June 2019.

[188] Zilong Zhao, Sophie Cerf, Bogdan Robu, and Nicolas Marchand. Feedback Control for Online Training of Neural Networks. In *3rd IEEE Conference On Control Technology And Applications (CCTA)*, Hong Kong, China, August 2019.

[189] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. Understanding mobility based on gps data. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 312–321. ACM, 2008.

[190] Yu Zheng, Xing Xie, and Wei-Ying Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33(2):32–39, 2010.

[191] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining interesting locations and travel sequences from gps trajectories. In *WWW*, pages 791–800, 2009.

Part VI

Appendix: Machine learning algorithms as systems to control

Appendix A

Duo Learning for Classifications with Noisy Labels

The following paper results from the work conducted from July to November 2018 during an internship at IBM Research Center, Zürich, Switzerland. It has been published in the second Continual Learning Workshop at the Thirty-second Conference on Neural Information Processing Systems (NIPS 2018) [44].

Duo Learning for Classifications with Noisy Labels

Sophie Cerf
Univ. Grenoble Alpes
France
sophie.cerf@gipsa-lab.fr

Robert Birke
ABB Research
Switzerland
robert.birke@abb.ch

Lydia Y. Chen
TU Delft
Netherlands
y.chen-10@tudelft.nl

Abstract

While the vast amount of user-generated data powers up machine learning based applications in our daily life, the variation of data quality may undermine the effectiveness of learning. In this paper, we demonstrate how classification algorithms can possibly be deteriorated by incorporating the data influx with noisy labels in the context of continuous learning. We design duo-learning framework that adaptively learn the dynamics of underlying data quality in junction with an on-line classification model. Our objective is to maximize the accuracy improvement rate from the initial data by actively selecting minimum amounts of ”clean” data over time. The problem generalizes across multiple application scenarios of continuous learning with noisy label data that are intrinsic or adversarial in the label generating process. Our preliminary results show that effective data selection can achieve a good accuracy improvement using a fraction amount of data instances, and avoid the pitfall of divergence from continuously learning from noisy label data.

1 Introduction

The vast amount of user-generated data, files, and images greatly enhances the applicability of machine learning technologies in our daily life. Indeed, datasets openly available in the wild come in different qualities, due to unsophisticated and unreliable processes of data generation and label annotation, or malicious adversaries. For example, similar images from Google search could show different classification labels [12]. When continuously learning from such data that is subject to label noises, the arising question is its impact on the classification algorithm over time or alternatively its associated prices/values. Let us take the crowd sourcing example: how to determine the price for annotating images to be paid to users that can produce more accurate class labels than others.

The specific question considered in the paper is the following. How to build a classification model that intelligently selects the data and continuously improves the accuracy from a minimum amount of data instances that are subject to fluctuating label noises? The challenge lies in capturing the dynamics of data quality which is not directly observable but only via classification outcomes that in turn is coupled with the noise level of data labels. When encountering noisy data labels, the classification errors are thus attributed to the classification models as well as intrinsic wrong inputs. Moreover, the noise levels can vary significantly across datasets produced by different individuals or robots. Using data with noisy labels may degrade the classification results and slow down the learning speed in the long run, or worse, make it diverge.

In this paper, we develop an on-line duo learning framework that continuously learns both data label model and classifier from arriving datasets that are subject to label noises. Our objective is to learn the classifier efficiently from a minimum amount of non-noisy data instances. For every batch of new data, we solicit data instances with non-noisy labels based on the label model that predicts if the sample unit is noisy. Our preliminary results on the random forests classifier on five datasets show that continuously cleansing data can result into a better learning efficiency, i.e., the improvement of

Preprint. Work in progress.

classifier accuracy per additional dataset selected, compared to classifiers without continuous data cleansing.

1.1 Related work

The prior art has extensively addressed the noisy label issues with respect to different classification algorithms [5] for static datasets in an off-line setting, i.e., the classification models are learned from a single batch of data. Noisy labels are shown to degrade the effectiveness of typical classification models, e.g., k-NN [13] and SVM [1] but also deep neural network [11]. The challenges involved include missing information of label quality, and inter dependency between classification errors and label errors that can be intrinsic or malicious.

There are two types of approaches to address classification problems with noisy labels: (i) cleansing noisy labels and learning only from the predicted clean data instances, (2) designing noise-aware classification algorithms. The first builds one or multiple filter models, e.g., SVM [9] to cleanse the data, and only the data instances that have been correctly classified by one or more filters are used to train the classification models. In contrast, the second type of approach tries to capture the noise model via frequentist [6] or Bayesian methods [10] and incorporate it in the underlying classification algorithms, e.g., adjusting the loss function. [3] applies large dual weights on different training sample units when learning a SVM classifier. In [7], the metric of local intrinsic dimensionality is used to control the training of deep neural networks.

While there exist good solutions to tackle the noisy label issues, little focus has been given to the on-line setting with data instances having fluctuating noise levels. Our study presents the initial results on how to build a classifier by selectively and continuously learn from high quality data that leads to a strong classifier.

2 Methodology

We specifically consider the following problem statement. A small set of initial data instances that have clean labels is given, and another set of testing data is provided. Each data instance has d features and belongs to class k, where $k \in \mathbf{K} = \{\mathbf{1}, \ldots, K\}$. A clean label refers to samples whose given label is properly annotated, without any alteration. The data instances are continuously collected over time, subject to a certain constraint. Specifically, we assume only up to N instances can be considered for training classifier. Furthermore, data instances are assumed to arrive at the learning system in batches over time X_i and the quality of their labels $\tilde{y}_i$ may fluctuate, e.g., percentage of noisy labels changes from batches to batches. The question here is how to continuously train a classifier from live noisy data so that the accuracy improvement from the initial set can be improved given the constraint of N data instances.

To such an end, we develop a duo learning framework as shown in Fig. 1 that continuously trains the label model, $\mathfrak{L} : \mathbb{R}^d \to q \in \mathbf{Q} = \{\mathbf{0}, \mathbf{1}\}$, and the classification model, $\mathfrak{C} : \mathbb{R}^d \to k \in \mathbf{K}$. The label model tries to learn the binary classifier for label quality, i.e., $q = 0$ denotes the clean label and $q = 1$ otherwise. Upon the arrival of a new batch of data instances at epoch i, we use $\mathfrak{L}_{i-1}$ to predict the quality for each data instance j. Here we use the subscript of i to denote the label model that is trained with data instance received up to time i. If $\hat{q}_{j,i} = 1$, meaning a noisy label, we discard such a data instance and only incorporate data instances with $\hat{q}_{j,i} = 0$ into the existing training set to retrain the classifier, $\mathfrak{C}_i$. Essentially, $\mathfrak{C}_i$ is trained on all the estimated-clean data received after the arrival of i^{th} batch. In turn, we retrain the label model for the next batch $\mathfrak{L}_i$ by incorporating those clean data, indicated by the black feedback arrow.

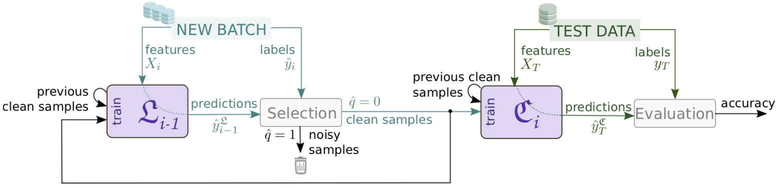


Figure 1: Duo learning framework. The flowchart is iterated at every batch.

The objectives of label model is two fold here. First of all, the label model can select the most representative instances to learn a strong classifier. The second-order objective here is to solicit data instances with clean labels, avoiding the pitfall that the classifier overfits the noise. To achieve both ends, we advocate to use supervised-learning algorithms for continuously training label model from accumulated "predicted" clean samples, which are highly correlated to a stronger classifier. Upon the arrival of data instances at batch i, we apply the previously learned label model $\mathfrak{L}_{i-1}$ to first predict their class labels $\hat{y}_i^{\mathfrak{L}}$. If predicted labels are different from their given labels, we then label such instances noisy. We then preclude them from the training set and only use the remaining subset of instances for retraining the classifier $\mathfrak{C}_i$. Eventually, the classification accuracy is measured by comparing test labels y_T to the predictions $\hat{y}_T^{\mathfrak{C}}$ based on test data X_T.

The proposed duo learning framework is generic and compatible with different learning algorithms for label and classifier models. In the evaluation section, we consider the following learning algorithms: knn, nearest centroid, SVM, Gaussian process and random forests. Considering the practical applicability on live data that might face timing constraint, we recommend to use lower order model for learning label quality than the classifier.

3 Evaluation

3.1 Experimental Setup

To evaluate the proposed duo learning, we use the following datasets: *dna*, *letter*, *pendigits*, *usps* and *mushrooms* from [2] and [4]. They vary in number of classes, features and number of samples. Their details are summarized in Table 1.

Table 1: Summary of the datasets main properties

Dataset	dna	letter	pendigits	usps	mushrooms
#-classes k	3	26	10	10	2
#-features d	180	16	16	256	24

The continual learning scenario, common for all datasets, is the following. Initially, a set of 150 clean samples is available and the testing set is clean. Then, data batches arrive continuously in sets of 50 sample units whose class labels may be altered and contains noises. We assume that there is an upper limit of $N = 1000$ samples (including the initial set) to be considered for training classifier. This number is chosen so as to ensure that the classification accuracy can converge in absence of noise for all datasets.

We add symmetric noises, i.e. following the *noise completely at random* model [5]. Time-varying dynamic noise is considered, the mean noise ratio (percentage of noisy labels in a batch) is drawn at every batch from a Gaussian distribution, with 20% of standard deviation. The mean rate is set depending on the number of classes in the dataset. We specifically consider the challenging scenarios where noisy labels affect the classification model. When the set has many classes (e.g.,*letter*, *pendigits* and *usps*), the learning is inherently more robust to noise, so the mean noise rate is set to 80%. For the datasets with fewer classes (*dna* and *mushrooms*), mean noise rate is set to 40%.

We use random forest for the classification model for multiple reasons. Random forest is applied on a variety of real applications, and is also known to be robust against label noises [5]. For fair comparison its parametrization is fixed to 100 trees, max-*depth* of 5 and *max-features* of 10. It is indeed not the optimal configuration for all datasets but it provides reasonable results on clean sets.

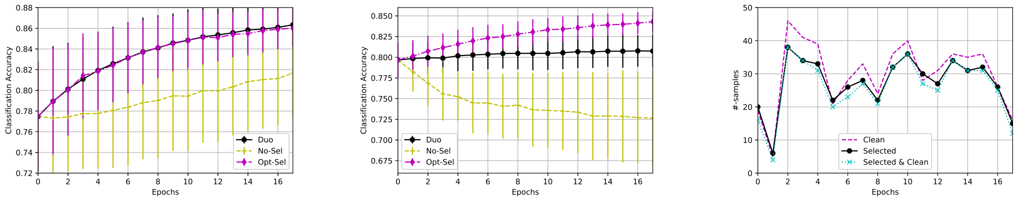
The label model is nearest centroid. We tested other learning algorithms, e.g. SVM or Gaussian process, but the simpler neighbor-based ones, e.g., nearest neighbors or nearest centroid, have shown better performance. The simple models remove not only noisy labels but also outliers. It hence increases the similarity between selected samples and thus classification robustness.

The framework is developed in Python using *scikit-learn* [8]. The main evaluation metric is the accuracy score, averaged over 100 runs. The proposed duo learning is compared against two other data selection schemes: (i) *No-Sel*, where all data instances of arriving batches are used for training classifier, (ii) *Opt-Sel*, which emulates an omniscient agent who can perfectly distinguish between clean and noisy labels.

3.2 Results

Fig. 2 (a) and (b) show the accuracy of the classification model on the *dna* and *usps* test sets for successive batches. The proposed learning framework (Duo) is compared against no selection (No-Sel) and optimal selection (Opt-Sel). These plots highlight four main results: (i) there is an advantage of continual learning compared to using only the initial set, (ii) however, learning from noisy labels over time leads to a serious accuracy degradation for the classifier, (iii) the duo learning improves the classification accuracy compared to taking all labels, (iv) the data selection of the duo learning is not far from being the optimal one.

Fig. 2 (c) presents the variations of noise over time (number of clean samples) for one run on *dna*. Overlaid is the number of data selected by the duo learning and the overlap between selection and actually clean. Results highlight the sharpness of data selection and its parsimony.



(a) Accuracy for *dna* (b) Accuracy for *usps* (c) Data selection for *dna*

Figure 2: Duo learning framework performances through time.

A full evaluation on all datasets is reported in Table 2. In addition to the average accuracy after all epochs, we also present the percentage of accuracy improvement comparing the duo to the initial set and no selection in columns 5 and 6 in Table 2. We attribute columns 5 and 6 to the impact of continual learning and intelligent selection respectively. As for columns 7 and 8, we present the percentage of accuracy difference between the duo and Opt-Sel (the smaller the better) and the fraction of samples used by duo.

Table 2: Evaluation of the duo learning framework for all datasets. Negative results are in **bold**.

Dataset	Initial	Final accuracy			Improvement due to		Distance to	Samples
	accuracy	No-Sel	Duo	Opt-Sel	Continual	Selection	optimal	used
dna	77.71	82.25	86.06	86.02	10.7%	4.6%	0%	55.6%
letter	43.76	41.04	42.78	52.51	**-2.2**%	4.2%	18.5%	10.5%
pendigits	81.64	77.05	80.07	84.87	**-1.9**%	3.9%	5.6%	19.6%
usps	79.81	73.21	81.03	84.46	1.5%	10.6%	4%	20.30%
mushrooms	97.33	94.22	80.74	98.94	**-17**%	**-14.3** %	18.3%	55.7%

Most results are positive, with varying amplitude depending on the datasets. In most cases, the proposed duo improves accuracy compared to blindly taking all samples, from 4 to 10% with the exception of *mushrooms* dataset. The impact of continual learning is more variable. It is highly valuable for *dna*, small but positive impact for *usps*, but detrimental for *pendigits*, *letter* and *mushrooms*. The poor performance of some sets is not linked with either their number of classes nor features. The optimal selection performs indeed better than the proposed duo, however for *dna* the quality estimation is close to optimal. Eventually, the total number of samples used ranges from half of the dataset to a tenth, which implies great potential savings in the scenario of data purchasing. The distance to optimal metric and *mushrooms* case show that there is still room for improvement.

4 Conclusion

In this paper we develop a duo learning framework that can continuously cleanse the live data instances with fluctuating noise levels and effectively learn a strong classifier from a small fraction of data. Our initial results show that even a simple label model, such as nearest centroid, can effectively select data instances and improve the classification accuracy and learning efficiency for random forests classifier. Our future work will explore a wider set of classification algorithms, including deep neural networks for image classification.

References

[1] Wenjuan An and Mangui Liang. Fuzzy support vector machine based on within-class scatter for classification problems with outliers or noises. *Neurocomput.*, 110:101–110, June 2013. ISSN 0925-2312.

[2] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011. ISSN 2157-6904. doi: 10.1145/1961189.1961199. URL `http://doi.acm.org/10.1145/1961189.1961199`.

[3] Ofer Dekel and Ohad Shamir. Good learners for evil teachers. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, pages 233–240, 2009.

[4] Ayhan Demiriz (demira@rpi. edu, Kristin P Bennett, John Shawe, and Taylor jst@cs. rhul. ac. uk. Linear programming boosting via column generation. (bennek@rpi.edu).

[5] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Trans. Neural Netw. Learning Syst.*, 25(5):845–869, 2014.

[6] Yunlei Li, Lodewyk F. A. Wessels, Dick de Ridder, and Marcel J. T. Reinders. Classification in the presence of class noise using a probabilistic kernel fisher method. *Pattern Recognition*, 40(12):3349–3357, 2007.

[7] Xingjun Ma, Yisen Wang, Michael E. Houle, Shuo Zhou, Sarah M. Erfani, Shu-Tao Xia, Sudanthi N. R. Wijewickrema, and James Bailey. Dimensionality-driven learning with noisy labels. In *ICML 2018*, pages 3361–3370.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] Nicola Segata, Enrico Blanzieri, Sarah Jane Delany, and Padraig Cunningham. Noise reduction for instance-based learning with a local maximal margin approach. *J. Intell. Inf. Syst.*, 35(2): 301–331, 2010.

[10] James D. Stamey and Richard Gerlach. Bayesian model selection for logistic regression with classified outcomes. *Statist. Model*, 7(3):255–273, 2007.

[11] Arash Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *NIPS*, pages 5601–5610, 2017.

[12] Yisen Wang, Weiyang Liu, Xingjun Ma, James Bailey, Hongyuan Zha, Le Song, and Shu-Tao Xia. Iterative learning with open-set noisy labels. In *CVPR 2018*.

[13] D. Randall Wilson and Tony R. Martinez. Reduction techniques for instance-basedlearning algorithms. *Mach. Learn.*, 38(3):257–286, March 2000.

Appendix B

Robust Anomaly Detection on Unreliable Data

The following paper has been accepted for publication in the 49th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2019) as a practical experience report [187].

Robust Anomaly Detection on Unreliable Data
– *Practical Experience Report* –

(Double-Blind Reviewing)

***Abstract*—Classification algorithms have been widely adopted to detect anomalies for various systems, e.g., IoT and cloud, under the common assumption that the data source is clean, i.e., features and labels are correctly set. However, data collected from the field can be unreliable due to careless annotations or malicious data transformation for incorrect anomaly detection. In this paper, we present a two-layer learning framework for robust anomaly detection (RAD) in the presence of unreliable anomaly labels.**

The first layer of quality model filters the suspicious data, where the second layer of classification model detects the anomaly types. We specifically focus on two use cases, (i) detecting 10 classes of IoT attacks and (ii) predicting 4 classes of task failures of big data jobs. Our evaluation results show that RAD can robustly improve the accuracy of anomaly detection, to reach up to 98% for IoT device attacks (i.e., +11%) and up to 83% for cloud task failures (i.e., +20%), under a significant percentage of altered anomaly labels.

***Index Terms*—Unreliable Data; Anomaly Detection; Failures; Attacks; Machine Learning**

I. Introduction

Anomaly detection is one of the core operations for enforcing dependability and performance in modern distributed systems. Anomalies can take various forms including erroneous data produced by a corrupted IoT device or the failure of a job executed in a datacenter.

Dealing with this issue has often been done in recent art by relying on machine learning-based classification algorithms over system logs [8], [10]. These systems often rely on a learning dataset from which the classifier learns to distinguish between data corresponding to a correct execution of the system from data corresponding to an abnormal execution of the latter (i.e., anomaly detection).

In this context, a rising concern when applying classification algorithms is the accessibility to a reliable ground truth for anomalies. Typically, anomaly data is manually annotated by human experts and hence the generation of anomaly labels is subject to quality variation, so-called noisy labels. For instance, annotating service failure types for data centers is done by operators.

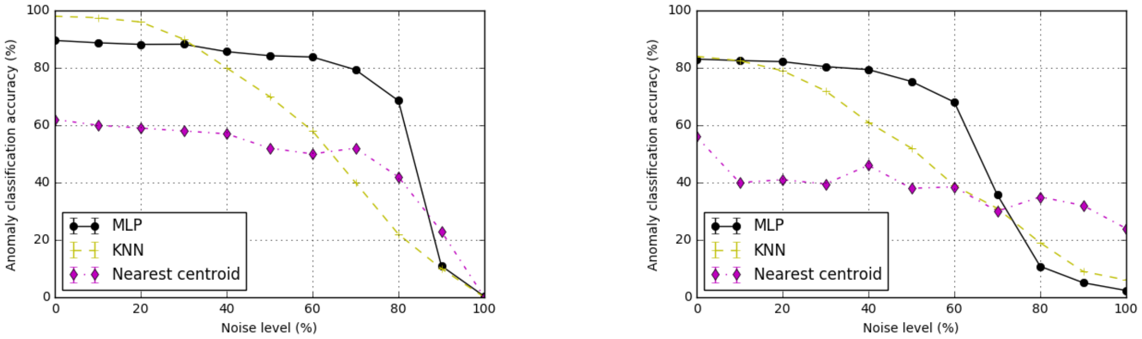
However, standard machine learning algorithms typically assume clean labels and overlook the risk of noisy labels. Moreover, recent studies point out the increasing dirty data attacks that can maliciously alter the anomaly labels to mislead the machine learning models [7], [11], [13]. As a result, anomaly detection algorithms need to capture not only anomalies that are entangled with system dynamics but also the unreliable nature of anomaly labels.

Indeed, a strong anomaly classification model can be learned by incorporating a larger amount of datasets; however learning from data with noisy labels can significantly degrade the classification accuracy, even for deep neural networks, at a non-negligible computation source [29]. Such a concern leads us to ask the following question: how to build an anomaly detection framework that can robustly differentiate the true and noisy anomalies and efficiently learn the anomaly classification models from a succinct amount of clean data. The immediate challenge of capturing the dynamics of data quality lies at the fact that label qualities are not directly observable but only via anomaly classification outcomes that in turn is coupled with the noise level of data labels.

In this paper, we develop a Robust Anomaly Detector (RAD), a generic framework that continuously learns the anomaly classification model from streams of event logs that are subject to label noises. To such an end, RAD is composed of two layers of learning models, i.e., data label model and anomaly classifier. The label model aims at differentiating the label quality, i.e., noisy v.s. true labels, for each batch of new data and only "clean" data points are fed in the anomaly classifier. The anomaly classifier predicts the event outcomes that can be in multiple classes of (non)anomalies, depending on the specific anomaly use case. The specific choices of label models and anomaly classifier include standard machine learning models, e.g., random forest, Adaboost, and discriminant analysis, and deep neural networks.

To demonstrate the effectiveness of RAD, we consider two use cases, i.e., detecting 10 classes of IoT attacks [18], and predicting four types of task failures for big data processing cluster [24], from open datasets. Our preliminary results show that RAD can effectively and continuously cleanse the data, i.e., selecting data streams with clean labels, and result better anomaly detection accuracy per additional data stream included, compared to classifiers without continuous data cleansing. Specifically, RAD achieves up to 98% and 83% accuracy for detecting IoT device attacks and predicting cluster task failures respectively.

The remainder of the paper is organized as follows. Section II describes the motivating case studies that we consider. Sections III and IV respectively present the proposed RAD framework and the results of its experimental evaluation. Section V describes the related work, and finally, Section VI draws our conclusions and the lessons learned.



(a) Use case of IoT thermostat device attacks

(b) Use case of Cluster task failures

Fig. 1: Impact of noisy data on anomaly classification

II. Motivating case studies

To qualitatively demonstrate the impact of noisy data on anomaly detection, we use two case studies.

- Detecting **IoT device attacks** from inspecting network traffic data collected from commercial IoT devices [18]. This dataset contains nine types of IoT devices which are subject to ten types of attacks. Specifically, we focus on the Ecobee thermostat device that may be infected by Mirai malware and BASHLITE malware. Here we focus on the scenario of detecting and differentiating between ten attacks. It is important to detect those attacks with high accuracy against all load conditions and data quality.
- Predicting **task execution failures** for big data jobs running at Google cluster [24], [26]. This trace contains a month-long jobs execution record from Google clusters. Each job contains multiple tasks, which can be terminated into four different states: *finish*, *fail*, *evict*, or *kill*. The last three states are considered as anomaly states. To minimise the computational resource waste due to anomaly states, it is imperative to predict the final execution state of task upon their arrivals.

The details about data definition, and statistics, e.g., no. of feature and no. of data points, can be found in Section IV-A. To detect anomalies in each case, related studies have applied machine learning classification algorithms, e.g., k-nearest neighbor (KNN), nearest centroid and multilayer perceptron (MLP) (a.k.a feed-forward deep neural networks), under the scenario where different levels of label noise are present. Here, we evaluate how the detection accuracy changes relative to different levels of noises. We focus on offline scenarios where classification models are learned from 14000 records and evaluated on a clean testing dataset of 6000 records. We specifically apply KNN, nearest centroid and MLP on IoT device attacks and cluster task failures respectively, and summarize the accuracy results in Figure 1a and Figure 1b.

One can see that noisy labels clearly deteriorate the detection results for both IoT attacks and task failures, across all three classification algorithms. For standard classifiers, like KNN and nearest centroid, the detection accuracy decays faster than MLP that is more robust to the noisy labels. Such an observation holds for both uses cases. In IoT attacks, MLP can even achieve a similar accuracy as the case of no label noises, when there is 50 percent of label classes are altered.

.

III. Design Principles of RAD Framework

A. System Model

We consider a dataset that consists of several data instances. Each data instance has f features. Each data instance belongs to a class k, where $k \in \mathcal{K} = \{1, \ldots, K\}$. A data instance is either labeled with a class k or is not labeled. Furthermore, a labeled data instance is either correctly labeled (i.e., clean data instance), or incorrectly labeled (i.e., noisy data instance). In the latter case, the data is annotated with a wrong label due to human error, or malicious error injection, etc. Obviously, a non-labeled data instance is clean. The quality of a dataset $\mathcal{D}$ is measured as the percentage of data instances with noisy labels, denoted here as $\tilde{Y}$.

Data instances structured in batches are assumed to arrive continuously in the learning system over time. $\mathcal{D}_i$ denotes the set of data instances arriving in batch at time t_i, and its set of labels $\tilde{Y}_i$, for which the quality may vary from one data instance to another inside the data batch. We assume that $\mathcal{D}_0$, a small set of initial data instances with both clean and noisy labels is given. Other collected data instances are also noisy, and the quality of datasets from batches to batches may fluctuate. Data instances belonging to $\mathcal{D}_i$ are denoted $d_{1,i}, d_{2,i}, \ldots d_{N,i}$. Up to N new data instances are considered for training the data classifier and building the learned data model. We consider that the batches arrive with constant number of data instances, $\forall \mathcal{D}_i, |\mathcal{D}_i| = N$, but not necessary at regular time intervals.

Then, a classification request consists of sets of non-labeled data instance $\mathcal{P}_i$ for which the classifier predicts the class k to which each data instance belongs. At each batch, the classification output is thus an array of the predicted classes for each data instance in the batch $\tilde{Y}_i^{\mathcal{P}}$.

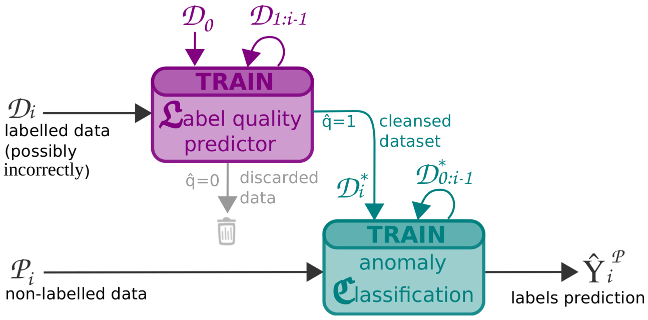


Fig. 2: RAD learning framework. The flowchart is iterated at every batch, with the new data coming in (left arrows). Each bloc is a machine learning algorithm, data used to train them are represented by colored arrows coming from the top. The data selected based on the label quality model's predictions provides the training data for the main task, the anomaly classification. Global output (right arrow) is the predicted labels of each new batch $\mathcal{P}_i$.

B. Objectives and Overview of RAD Framework

We propose the RAD learning framework. Its objective is threefold:

(i) Accurately learning a data model from noisy data.
(ii) Continuously updating the learned model with new incoming data.
(iii) Proposing a general approach that applies on different machine learning algorithms, and with different application use cases.

Figure 2 describes the overall architecture of RAD. Each of its components is detailed in the following.

C. Determining Data Noise

The first component of RAD aims at determining if a labeled data instance is correctly or incorrectly labeled. RAD learns the data label quality model $\mathfrak{L} : \mathbb{R}^f \rightarrow q \in \mathcal{Q} = \{0, 1\}$. The objective of the label quality model is to select the most representative data instances to learn a strong data classifier model. It solicits data instances with clean labels, avoiding the pitfall that the classifier overfits the noise. RAD uses supervised-learning algorithms to continuously train the label quality model from accumulated predicted clean data instances, which are highly correlated to a stronger classifier.

The label quality model tries to learn the binary classifier for label quality, i.e., $q = 1$ denotes a correct label and $q = 0$ denotes an incorrect label. $\mathfrak{L}_i$ is the label quality model that is trained with data instances received up to time t_i, that is $\mathcal{D}_0, \mathcal{D}_1 \ldots \mathcal{D}_i$. Upon the arrival of a new batch of data instances $\mathcal{D}_i$ at time t_i, we use the previously learned label quality model $\mathfrak{L}_{i-1}$ to predict the quality for each data instance $d_{j,i} \in \mathcal{D}_i$, this quality is denoted $\hat{q}_{j,i}$. If $\hat{q}_{j,i} = 0$, meaning an incorrect label, we discard such a data instance and only consider data instances with $\hat{q}_{j,i} = 1$. We thus build $\mathcal{D}^*{}_i$, the subset of $\mathcal{D}_i$ with all correct data instances from $\mathcal{D}_i$. And we incorporate $\mathcal{D}^*{}_i$ into the existing training set for the data classifier. In turn, we incorporate the clean data $\mathcal{D}^*{}_i$ to retrain the label model for the next batch, that is $\mathfrak{L}_i$.

D. Generic Approach to Handle Dynamic Data

Another component of RAD is the dynamic data classifier, $\mathfrak{C} : \mathbb{R}^f \rightarrow k \in \mathcal{K}$. Essentially, $\mathfrak{C}_i$ is trained on all the estimated clean data received until batch $\mathcal{D}_i$, that is $\mathcal{D}^*{}_0 \ldots \mathcal{D}^*{}_i$. Indeed, the received data $\mathcal{D}_1 \ldots \mathcal{D}_i$ were cleaned using the data label quality model $\mathfrak{L}_0 \ldots \mathfrak{L}_{i-1}$ to produce clean data $\mathcal{D}^*{}_1 \ldots \mathcal{D}^*{}_i$. Thus, the RAD learning framework uses the data label quality model which is updated from batch to batch, and enriches its overall learned data classification model accordingly.

Furthermore, RAD follows a generic approach since the proposed classification framework can be used with different machine learning algorithms, such as SVM, KNN, random forest, nearest centroid classifier, etc. And RAD has different applications where noisy data are collected and must be cleaned before learning data model, such as failure detection and attack diagnosis that we illustrate in Section IV.

IV. Experimental Evaluation

A. Use Cases and Datasets

In order to demonstrate the general applicability of the proposed RAD framework for anomaly detection, we consider the following two use cases: (i) Cluster task failures, and (ii) IoT botnet attacks. In our experiments, we use real data collected in cluster and IoT platforms.

The task traces comprise data instances each corresponding to a task with 27 features capturing information related to static and dynamic system state, e.g. the task start/end times, the task resource utilisations, the hosting machine, etc. Each class is labeled based on its scheduling state. A detailed description of the features and labels can be found in [24]. In particular, we are interested in the four possible termination classes: *finish*, *fail*, *evict*, or *kill*. We filter out other classes. The resulting class distribution is dominated by successful tasks (*finish*) 77.8%, followed by *kill* 22.0%, *fail* 0.2%, and *evict* <0.1%. Similar to [26], we aim to predict the task outcome to reduce the resource waste and improve the overall scheduling and system performance, e.g., in case of lack of resources and need to kill a task help choosing the task with the least probability to succeed. We apply RAD to continually train a noise-resistant model for better accuracy.

The IoT dataset comprises data instances describing 23 network packet-level statistics recursively computed over five different time scales totalling to 115 features. This traffic statistics are collected during normal operation, labeled as benign, or under one of ten different malicious attacks stemming from devices infected by either the *BASHLITE* or *Mirai* malware. Malicious traffic covers mainly scanning for vulnerable devices and various flooding attacks. The dataset provides traces collected at different IoT devices. More details are provided in [18]. We aim to apply RAD to build a noise-resistant model to categorize the attacks for post fact analysis, e.g., for threat assessment.

The main dataset characteristics are summarized in Table I.

TABLE I: Dataset description

Use case	Cluster task failures	IoT device attacks
#data instances	57,000	33,000
#classes K	4	11
#features f	27	115
data batch size	600	300

B. Experimental Setup

RAD is developed in Python using scikit-learn [21]. The main performance evaluation metric is accuracy. Experiments are carried out 5 times, results are aggregated by computing mean and standard deviation.

Noise. We inject noise into the two datasets by exchanging the true label of data instances with a random one. The label noise is symmetric, i.e., following the noise completely at random model [9] where a label is picked with equal probability from all classes except the true one. The noise level represents the percentage of data instances with noisy labels. We emulate time-varying noise by drawing for each new data batch the noise level from a Gaussian distribution with 20% standard deviation and various mean levels. We assume that all data is affected by label noise, except the testing data.

Continual learning. We start with an initial data batch of 3000 data instances for the Cluster task failures dataset, 6000 for the IoT devices one. Then, data instances arrive continuously in batches of respectively 600 and 300 data instances. Both the initial and subsequent data batches are affected by noise. To kick-start the label and classification models in RAD we assume to know which initial data instances are affected by noise (no assumptions for the subsequent data batches). We show the evolution of the model accuracy over data batch arrivals until the performance of RAD converges.

Label model. We use a multilayer perceptron to assess the quality of each label. For IoT dataset, the neural network consists of two layers with 28 neurons each. For Cluster task dataset, the network consists of two layers with 110 neurons each. The precision and robustness of the label model are critical to filter out the noisy/malicious labels and provide a clean training set to the classification model. We considered different models, but neural networks provided the best results in terms of accuracy and stability over time. Adaboost gave excellent accuracy when training from the initial data with ground truth, but resulted too sensitive to the unknown noise of subsequent data batches. Random forest is another choice and known to be robust against label noise [9], however its accuracy was below the neural network one.

Classification model. We use KNN to assign the correct class label to each data instance filtered by the label model. We set k to 5. Higher values of k can increase the resilience of the algorithm to residual noise, but also induce extra computational cost. The current choice stems from good results in preliminary experiments.

Baselines. The proposed RAD is compared against two baseline data selection schemes: (i) No-Sel, where all data instances of arriving batches are used for training the classification model; and, (ii) Opt-Sel which emulates an omniscient agent who can perfectly distinguish between clean and noisy labels. The two baselines are representative of the worst and best possible data selection strategies and we expect RAD to fall in between.

C. Handling Dynamic Data

We start by illustrating how RAD enables to increase the anomaly detection accuracy over time, despite the presence of noise. Figures 3 and 4 show the evolution of the mean and variance of the classification accuracy achieved by RAD on the thermostat and task failure datasets, respectively. Each figure moreover presents results under two levels of label noise: 30% and 40%. We compare RAD against no selection (No-Sel) and optimal selection (Opt-Sel). One can notice that learning from all data instances without cleansing (i.e., No-Sel curves) gives consistently lower accuracy in all cases. For the attacks classification on the thermostat, the accuracy even oscillates and diverges. The performance when using RAD is better. First because the accuracy does not diverge, second because it always consistently increase until it saturates. For the IoT attack dataset, the end accuracy is around 98%, for the cluster tasks around 83%. While for the first dataset the accuracy of RAD follows closely the accuracy of Opt-Sel, for the second dataset RAD follows Opt-Sel at first but then saturates after 40 data batch arrivals. RAD is efficient for various classification applications, however not optimal for all of them. Note that RAD gives also more stable results as shown by shorter errorbars which in magnitude are in line with the ones obtained by an ideal data cleansing. For No-Sel the bars are significantly larger. We discuss the results for other IoT devices in Section IV-F.

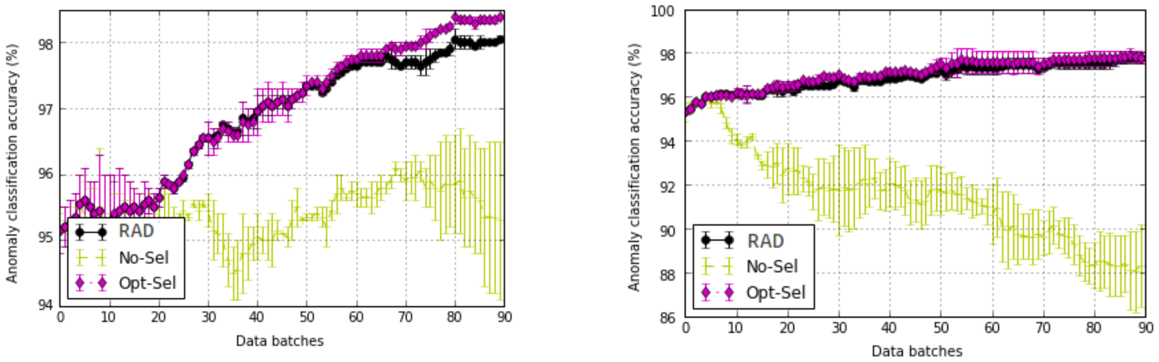
Figure 5 presents the variations of noise over time for one run on IoT thermostat dataset where mean noise rate sets to 30%. Overlaid is the number of data selected by RAD and the overlap between selection and actually clean. Results highlight the sharpness of data selection and its parsimony.

In summary: (i) continual learning is advantageous compared to using only the initial dataset; however, (ii) continual learning exposes us to possible classification accuracy degradation stemming from noisy labels if proper data selection is lacking, (iii) RAD improves the classification accuracy compared to taking all labels, (iv) the data selection of RAD is good, and close to being optimal in some cases.

D. Evaluation of Noise Robustness of RAD

Next we investigate the impact of different noise levels on the RAD performance in terms of classification accuracy.

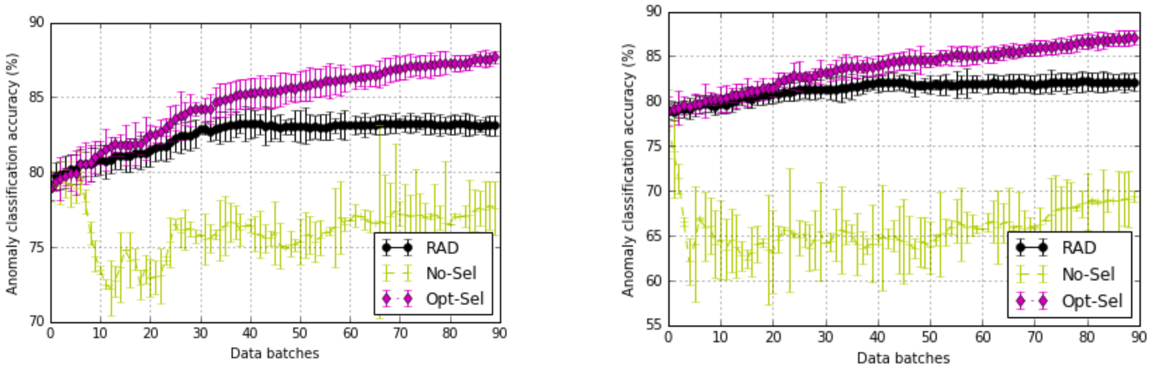
Figures 6a and 6b present the classification accuracy for various levels of noise, ranging from 0% (all data are clean)



(a) With data noise level of 30%

(b) With data noise level of 40%

Fig. 3: Evolution of learning over time – Use case of IoT thermostat device attacks



(a) With data noise level of 30%

(b) With data noise level of 40%

Fig. 4: Evolution of learning over time – Use case of Cluster task failures

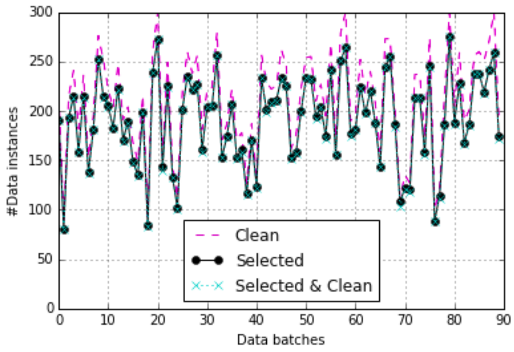


Fig. 5: Data selection – Use case of IoT thermostat device attacks with 30% noise level.

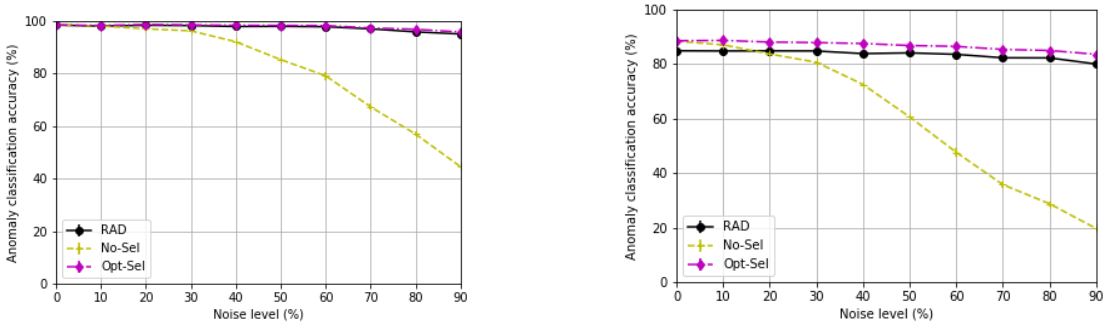
up to 90% for our two main reference datasets: IoT thermostat device attacks and Cluster task failures. Accuracy is measured once the learning has converged. Once again, the RAD performance is compared to learning from all data (No-Sel) and an omniscient data cleanser (Opt-Sel).

As illustrated in Section II, for No-Sel the noisier the data are, the worst the classification accuracy, dropping to 20% and 42% for the tasks and thermostat datasets, respectively. A decreasing trend can also be found for RAD and Opt-Sel, however the drops are significantly smaller: at most 5%. As there is by definition no noise in Opt-Sel case, the decrease in classification accuracy is only due to the reduction of the overall amount of clean data to learn from. Since the data cleansing of RAD is not perfect, the accuracy reduction is caused by both pollution from the noise and the overall clean data reduction. Nevertheless, the impact is small and any huge accuracy pitfall is avoided which results in RAD's performance being close to Opt-Sel. We can conclude that RAD can limit the impact of the amount of noise across a wide range of noise levels.

E. Impact of Initialization

Here we study the impact on RAD of the amount of ground truth data available, i.e., the size of the initial dataset $\mathcal{D}_0$.

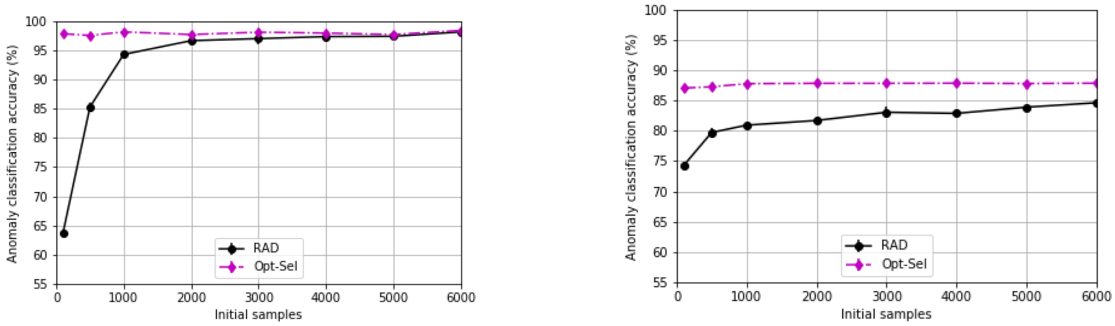
In Figure 7a, we vary the number of initial clean data instances for the IoT thermostat device dataset from 100 to 6000, and measure the classification accuracy after 90 data



(a) IoT thermostat device attacks

(b) Cluster task failures

Fig. 6: Impact of data noises on RAD accuracy



(a) IoT thermostat device attacks

(b) Cluster task failures

Fig. 7: Impact of size of initial data batch $\mathcal{D}_0$ on RAD accuracy

batch arrivals, for both RAD and Opt-Sel. We do not consider No-Sel here since this results are meant for the framework configuration, not its performance evaluation. Similarly, Figure 7b describes the results of the same experiments with the Cluster task failure dataset.

In Figure 7a for Opt-Sel, no matter the number of data instances in the initial set, the final accuracy is stable around 98%. However for RAD, the size of $\mathcal{D}_0$ does matter: the larger the better, with some saturation aftereffect. At $|\mathcal{D}_0| = 3000$ its performances is close to Opt-Sel, and at $|\mathcal{D}_0| = 6000$ they completely overlap.

This justifies our earlier choice of 6000 data instances in our initial set for the IoT device attacks classification as it enables to achieve the best accuracy. However, RAD framework could also perform well with only half of those data as initial set.

F. Analysis of All Datasets

Summary results across datasets are reported in Table II. In addition to the average accuracy after the last batch arrival, we also underline the relative accuracy improvement obtained by comparing RAD to the initial set, i.e., the impact of continual learning, and to the No-Sel strategy, i.e., the impact of intelligent data selection. Columns 5 and 6 report these results. Finally, we present the percentage of accuracy difference between RAD and Opt-Sel, i.e., how close we are to an omniscient data selection, in column 7.

All results are positive, with varying magnitude depending on the dataset. In all cases, the proposed RAD improves between 1% to 5% the accuracy compared to blindly taking all data instances. However, more important than the absolute gain is the trend. For example, for the thermostat dataset, we can observe that RAD converges over time to a stable level as well as Opt-Sel model, but No-Sel diverges. This means that as time goes by, No-Sel becomes worse and worse.

Even more than the benefits of continual learning might be important the resilience to high levels of noise. Under such levels, the classification accuracy without data cleansing diverges for all datasets. Even if it is rare to have noise levels of 90% or above, they might still happen for short periods of time in case of attacks to the auto-labelling system via flooding of malicious labels. Hence this property can be crucial for the dependability of the auto-labelling system.

TABLE II: Evaluation of the RAD learning framework for all datasets.

Dataset	Initial accuracy	Final accuracy			Improvement due to		Distance to optimal
		No-Sel	RAD	Opt-Sel	Continual learning	Selection	
Cluster task failures	78.98%	78.06%	83.03%	88.06%	4.05%	4.97%	5.03%
IoT monitor device attacks	97.60%	96.76%	99.33%	99.50%	1.73%	2.57%	0.17%
IoT doorbell device attacks	96.00%	96.50%	98.95%	99.05%	2.95%	2.45%	0.10%
IoT thermostat device attacks	95.15%	95.30%	98.05%	98.40%	2.90%	2.75%	0.35%

V. Related Work

Machine learning has been extensively used for failure detection [6], [22], [23], [25], and for attack prediction [1], [3], [4], [14], [15], [33]. Considering noisy labels in classification algorithms is also a problem that has been explored in the machine learning community as discussed in [5], [9], [19].

The problem of classification in presence of noisy labels can be organized into various categories according to, on the one hand, the type of classification algorithm subject to noise, and on the other hand, the techniques used to remove the noise.

Regarding the type of classification algorithm, the problem of noisy labels has been studied both for binary classification where noisy labels are considered as symmetric (e.g., [16]) and for classification with multiple classes where noisy labels are considered as asymmetric, e.g., [20], [27]. In the context of this paper, we consider the problem of classification with multiple classes. Furthermore, noisy labels have been considered in various types of classifiers KNN [32], SVM [2], and deep neural networks [30]. In the context of this paper, our proposed approach is agnostic to the underlying classifier type as noise removal is performed ahead of the classification.

To deal with noise, various techniques have been explored including forward loss correction. These algorithms learn about the label noise by adjusting the loss to the end of the model. However, these solutions either rely on strong assumptions or have limited accuracy as they generally do not rely on clean labels to remove the noise. More accurate solutions, which rely on clean labels during the training phase have thus been explored (e.g., [12], [17], [31]). These solutions generally train a separate network for distinguishing noisy labels from clean ones. Robustness to label noise has also been studied for GANs performing image recognition, both in the context of known and unknown noise distribution [28]. However, all these solutions have been designed and tested on static datasets and in an off-line setting. Instead in the context of this paper, we consider a dynamic model where the network has been trained using clean labels continues to learn over time.

VI. Conclusion

While machine learning classification algorithms are widely applied to detect anomalies, the commonly employed assumption of clean anomaly labels often does not hold for the data collected in the wild due to careless annotation and malicious dirty label pollution. The noisy labels can significantly degrade the accuracy of anomaly detection with an increasing amount of data and are challenging to tackle due to the lack of ground truth of label quality. In this paper, we present a framework for robust anomaly detection, RAD, which can continuously learn the system dynamics and anomaly behaviours from streams of arriving data after filtering out suspicious noisy data.

RAD is a general framework that composes of sequence of quality and classification models, where the former captures the label dynamics and the latter focus on detection anomaly. We demonstrate the effectiveness of RAD on two uses cases, i.e., detecting IoT device attacks, and predicting task failure at Google clusters. RAD can robustly improve the detection accuracy against different level of label noises, reaching up to 83% and 98% accuracy for predicting task failure and detecting IoT device attacks, respectively, whereas learning directly from all the data streams without filtering degrades the detection accuracy.

References

[1] Mayank Agarwal, Dileep Pasumarthi, Santosh Biswas, and Sukumar Nandi. Machine learning approach for detection of flooding dos attacks in 802.11 networks and attacker localization. *Int. J. Machine Learning & Cybernetics*, 7(6):1035–1051, 2016.

[2] Wenjuan An and Mangui Liang. Fuzzy support vector machine based on within-class scatter for classification problems with outliers or noises. *Neurocomput.*, 110:101–110, June 2013.

[3] Mohammed Anbar, Rosni Abdullah, Bassam Naji Al-Tamimi, and Amir Hussain. A machine learning approach to detect router advertisement flooding attacks in next-generation ipv6 networks. *Cognitive Computation*, 10(2):201–214, 2018.

[4] Sebastian Banescu, Christian S. Collberg, and Alexander Pretschner. Predicting the resilience of obfuscated code against symbolic execution attacks via machine learning. In Engin Kirda and Thomas Ristenpart, editors, *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017.*, pages 661–678. USENIX Association, 2017.

[5] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. In *Asian Conference on Machine Learning*, pages 97–112, 2011.

[6] João R. Campos, Marco Vieira, and Ernesto Costa. Exploratory study of machine learning techniques for supporting failure prediction. In *14th European Dependable Computing Conference, EDCC 2018, Iaşi, Romania, September 10-14, 2018*, pages 9–16. IEEE Computer Society, 2018.

[7] Youping Fan, Jingjiao Li, and Dai Zhang. A method for identifying critical elements of a cyber-physical system under data attack. *IEEE Access*, 6:16972–16984, 2018.

[8] Ze-Han Fang, Jian-Shuin Tzeng, Chien Chin Chen, and Tzu-Chuan Chou. A study of machine learning models in epidemic surveillance: Using the query logs of search engines. In *Pacific Asia Conference on Information Systems, PACIS 2010, Taipei, Taiwan, 9-12 July 2010*, page 137. AISeL, 2010.

[9] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Trans. Neural Netw. Learning Syst.*, 25(5):845–869, 2014.

[10] Georgios Giantamidis and Stavros Tripakis. Learning moore machines from input-output traces. In John S. Fitzgerald, Constance L. Heitmeyer, Stefania Gnesi, and Anna Philippou, editors, *FM 2016: Formal Methods - 21st International Symposium, Limassol, Cyprus, November 9-11, 2016, Proceedings*, volume 9995 of *Lecture Notes in Computer Science*, pages 291–309, 2016.

[11] Youbiao He, Gihan J. Mendis, and Jin Wei. Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism. *IEEE Trans. Smart Grid*, 8(5):2505–2516, 2017.

[12] Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In *32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.*

[13] Jeong-Won Kang, Il-Young Joo, and Dae-Hyun Choi. False data injection attacks on contingency analysis: Attack strategies and impact assessment. *IEEE Access*, 6:8841–8851, 2018.

[14] Dimitrios Karagiannis and Antonios Argyriou. Jamming attack detection in a pair of RF communicating vehicles using unsupervised machine learning. *Vehicular Communications*, 13:56–63, 2018.

[15] Rafal Kozik, Michal Choras, Massimo Ficco, and Francesco Palmieri. A scalable distributed machine learning approach for attack detection in edge computing environments. *J. Parallel Distrib. Comput.*, 119:18–26, 2018.

[16] Jan Larsen, L Nonboe, Mads Hintz-Madsen, and Lars Kai Hansen. Design of robust neural network classifiers. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 2, pages 1205–1208. IEEE, 1998.

[17] Yuncheng Li, Jianchao Yang, Yale Song, Liangliang Cao, Jiebo Luo, and Li-Jia Li. Learning from noisy labels with distillation. In *ICCV*, pages 1928–1936, 2017.

[18] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, and Yuval Elovici. N-baiotnetwork-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3):12–22, 2018.

[19] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, 2013.

[20] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.(CVPR)*, pages 2233–2241, 2017.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[22] Alessandro Pellegrini, Pierangelo di Sanzo, and Dimiter R. Avresky. A machine learning-based framework for building application failure prediction models. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop, IPDPS 2015, Hyderabad, India, May 25-29, 2015*, pages 1072–1081. IEEE Computer Society, 2015.

[23] Teerat Pitakrat, André van Hoorn, and Lars Grunske. A comparison of machine learning algorithms for proactive hard disk drive failure detection. In Philippe Kruchten and Sam Malek, editors, *Proceedings of the 4th international ACM Sigsoft symposium on Architecting critical systems, ISARCS 2013, Vancouver, BC, Canada, June 17-21, 2013*, pages 1–10. ACM, 2013.

[24] Charles Reiss, John Wilkes, and Joseph L Hellerstein. Google cluster-usage traces: format+ schema. *Google Inc., White Paper*, pages 1–14, 2011.

[25] Uwe Reuter, Ahmad Sultan, and Dirk S. Reischl. A comparative study of machine learning approaches for modeling concrete failure surfaces. *Advances in Engineering Software*, 116:67–79, 2018.

[26] Andrea Rosà, Lydia Y. Chen, and Walter Binder. Failure analysis and prediction for big-data systems. *IEEE Trans. Services Computing*, 10(6):984–998, 2017.

[27] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014.

[28] Kiran K Thekumparampil, Ashish Khetan, Zinan Lin, and Sewoong Oh. Robustness of conditional gans to noisy labels. In *Advances in Neural Information Processing Systems*, pages 10291–10302, 2018.

[29] Vadim N. Vagin and Marina V. Fomina. Problem of knowledge discovery in noisy databases. *Int. J. Machine Learning & Cybernetics*, 2(3):135–145, 2011.

[30] Arash Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *NIPS*, pages 5601–5610, 2017.

[31] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge J Belongie. Learning from noisy large-scale datasets with minimal supervision. In *CVPR*, pages 6575–6583, 2017.

[32] D. Randall Wilson and Tony R. Martinez. Reduction techniques for instance-basedlearning algorithms. *Mach. Learn.*, 38(3):257–286, March 2000.

[33] Baojun Zhou, Jie Li, Jinsong Wu, Song Guo, Yu Gu, and Zhetao Li. Machine-learning-based online distributed denial-of-service attack detection using spark streaming. In *2018 IEEE International Conference on Communications, ICC 2018, Kansas City, MO, USA, May 20-24, 2018*, pages 1–6. IEEE, 2018.

Appendix C

Feedback Control for Online Training of Neural Networks

The following paper has been accepted for publication in the third IEEE Conference On Control Technology And Applications (CCTA 2019) [188].

Feedback Control for Online Training of Neural Networks

Zilong Zhao[1], Sophie Cerf[1] , Bogdan Robu[1] and Nicolas Marchand[1]

Abstract— **Convolutional neural networks (CNNs) are commonly used for image classification tasks, raising the challenge of their application on data flows. During their training, adaptation is often performed by tuning the learning rate. Usual learning rate strategies are time-based i.e. monotonously decreasing. In this paper, we advocate switching to a performance-based adaptation, in order to improve the learning efficiency. We present E (Exponential)/PI (Proportional Integral)-Control, a conditional learning rate strategy that combines a feedback PI controller based on the CNN loss function, with an exponential control signal to smartly boost the learning and adapt the PI parameters. Stability proof is provided as well as an experimental evaluation using two state of the art image datasets (CIFAR-10 and Fashion-MNIST). Results show better performances than the related works (faster network accuracy growth reaching higher levels) and robustness of the E/PI-Control regarding its parametrization.**

I. Introduction

Convolutional neural networks (CNNs) are popular machine learning algorithms for image classification , as they are well suited for visual pattern recognition and require low preprocessing [1]. Like all neural networks, CNNs are parametrized with so-called weights that enable to tune the network prediction model to fit the task. Those weights are learned iteratively based on training data using methods such as gradient descent. In this paper, we consider a scenario where data comes dynamically in batches (not all data is initially available), previous data batches being discarded at the arrival of a new one. This type of scenarios is very common in our everyday life if we think about sequential collection of a video flow or daily crowdsourcing [2] [3].

The learning rate parameter is used to weight the impact of a new epoch on the previously learned model. Thus, in the gradient-based algorithms, there are two factors that influence the reach of the gradient global minimum: the network's weights initialization and the learning rate policy. The weights initialization is often dealt with by setting them all null or generated from a uniform distribution [4]. The learning rate controls the speed to approach the minimum. A large learning rate will accelerate the converging speed but at the risk of diverging [5]. A small learning rate will slowly approach the minimum with less tendency to skip over it, but may fall into a local minimum.

The objective is thus to set the learning rate strategy in order to learn from the data as fast as possible to reach the maximum CNN's predictions accuracy. The dynamic data collection scenario raises the challenge of a learning rate scheduling able to deal with the combination of epoch learning (take the most out of the currently available data) with batch learning (being able to include new data without forgetting the previous ones).

A learning rate strategy is defined by its initial value and its evolution law. The tuning of both is a significant challenge for the deep learning community [6]. According to Bengio [5], a learning rate of 0.01 typically works as a default value for standard multi-layer neural networks. He also recommends a classic strategy to find a more suitable value for a given architecture and dataset. Its principle is to try several values on a subset of the dataset and compare the best validation accuracy for a fixed training time; and the lowest training time to reach a given validation accuracy [7]. Learning rate evolution laws are usually of two kinds: time-based or adaptive. Time-based learning strategies [5] are the most famous ones: the learning rate follows a predefined function (polynomial, exponential, etc.) that should decrease through time to ensure stability. The adaptive techniques are based on the gradient: they are reactive techniques that set the learning rate according to the past values of the gradient [8], [9]. Given the definition of the gradient descent training law, a learning rate indexed on the gradient value is always decreasing with time. The monotonous decrease of the learning rate is a well-established use that has rarely been questioned. However, Smith [7] presents promising results with a cyclical learning rate law of triangular shape, and An et al. [10] introduced a time decreasing law with small sine oscillations. Indeed, we advocate that a brief increase in the learning rate could enable both to reach faster the global minimum and avoid being blocked in a local one.

Moreover, the issue with the state of the art learning rate laws regarding our continual learning scenario is that they do not take into account the dynamic of data coming in. They are predefined functions that do not adapt to the performances of the CNN training. Even by re-initializing the learning rate rule at each batch, it will not take into account the precision improvement through time that results from the memory of the previous batches.

In this work, we advocate using a control-based approach to adapt the learning rate in order to reach a high network accuracy in a short amount of time. The principle is to switch from time decreasing rules to a performance-based rule to be able to increase the learning rate when necessary. P and PI control strategies are initially developed. Then we present E/PI-Control, a hybrid strategy for setting the learning rate that combines both a time-based rule, with a first initial phase of an exponential growth of the learning rate, with a PI controller triggered by the network loss function. The initial E phase additionally allows the PI to be tuned on-line, thus

[1] GIPSA-lab, Univ. Grenoble Alpes, CNRS, Grenoble INP, 38000 Grenoble, France `firstname.lastname@gipsa-lab.fr`

getting rid of the need of an off-line profiling phase to adapt to a new dataset or network architecture. The E/PI-Control is evaluated on two classical state of the art datasets (CIFAR-10 [11] and Fashion-MNIST [12]), which are labelled image datasets commonly used to train computer vision algorithms [13]. Our control shows higher accuracy, faster rising time, lower final loss and more stable results than the state of the art techniques. Robustness regarding the initial value of the learning rate is also illustrated.

In the remaining of the paper, we first present the problem statement in a control theory formulation (Section II) and illustrate two state of the art learning rate strategies (Section III). The control law is presented in Section IV and its stability analysis and performance evaluation are given in Section V.

II. Background

This section presents, in control terms, the system we aim at monitoring, its disturbances, the signals that evaluate its performances and the available control knob.

A. The plant: a convolutional neural network

Convolutional Neural Networks (CNN) are state of the art learning mechanisms that give the best results comparing to other learning algorithms when modeling image datasets [14]. CNNs are inspired by the organization of animal visual cortex [15], [16]. They are a type of deep learning model made to process data that have grid patterns (neighboring features form a local structure such as in images). They are designed to automatically and adaptively learn spatial hierarchies of features, from low to high-level patterns. CNN training is done using algorithms such as the Stochastic Gradient Descent (SGD) optimizer. We use SGD as it enables to set the learning rate at the beginning of each training epoch.

We consider a CNN as being our plant, and the data used in training are seen as a disturbance, see Figure 1.

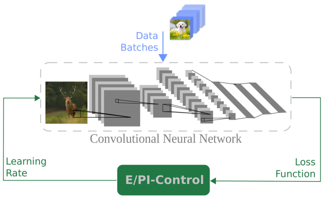


Fig. 1: CNN control schema.

B. The disturbance: data batches

We consider a training dataset that consists of several data instances, such as images. And each data instance belongs to only one class c, where $c \in \{1, \dots, C\}$, representing for instance the main object on the picture. Data instances, structured in batches, are assumed to arrive sequentially to the learning system over time. We set that each batch consists of B data instances. One iteration on the whole batch is called an epoch, E epochs are run on each batch. When the data of the new batch arrives, we discard the previous data and continue learning only on the new ones. This enables to reduce the storing space and processing time compared to keeping all the data. The control system sampling time is thus one epoch, while the disturbance time scale is the batches' one.

C. Performance Metrics: accuracy and loss

Two signals can be used to evaluate a neural network performance: validation accuracy and loss function, both varying through epochs. Validation accuracy, computed on the test set, is the percentage of instances for which the predicted class matches the ground truth. Loss function also compares the model predictions with the ground truth, but includes the notion of confidence in the prediction through the use of a distance. Cross-entropy is one of the most used multiclass classification loss function, and is thus the one we selected to be our output signal. It is the sum of cross-entropy error between targets and the predicted values done on the test set. We define $y_{i,c}$ as being the ground truth, indicating for each image i if it belongs to class c ($y_{i,c} = 1$) or not ($y_{i,c} = 0$). $\hat{y}_{i,c}$ is the CNN output, indicating the predicted probability of the image i to belong to class c. The loss function thus defined as follows:

$$loss = -\frac{1}{T}\sum_{i=1}^{T}\sum_{c=1}^{C} y_{i,c}log(\hat{y}_{i,c}) + (1-y_{i,c})log(1-\hat{y}_{i,c}) \tag{1}$$

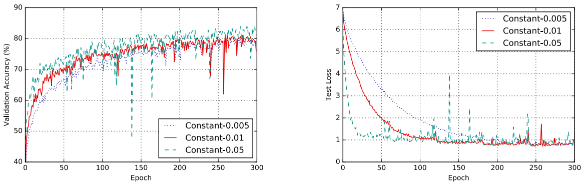
with T the size of the test set. The lower the value of loss function, the better the model is for the data, and as the entropy can't be negative, the target loss value is 0. Hence, the loss function is our error signal.

Validation accuracy is used as an *a posteriori* evaluation signal, as eventually, only the final prediction matters, whatever its confidence. Several metrics are extracted from the accuracy signal to reflect the CNN training performances. The end value of the accuracy is indeed the key factor. However, accuracy's converging speed is also important engineering concern: for complex image datasets such as ImageNet, state of the art training strategies take up to a few days. Another important metric is the stability of the accuracy curve, especially for the epochs toward the end of the learning. A low standard deviation of the accuracy provides more guarantees on the final CNN performances. Eventually, the final value of the loss function is also of interest, as it allows to evaluate the model overfitting on data when compared to accuracy value.

D. The Control Signal: the learning rate

The learning rate λ is our control signal. To illustrate that the learning rate is a control signal for our online scenario, we study the impact of different constant learning rate on the variation of the accuracy and loss functions over epochs. Figure 2 is the application of three constant learning rates ($\lambda \in \{0.005, 0.01, 0.05\}$, corresponding to Bengio's recommendation [5], larger and smaller) for training on CIFAR-10 dataset, with new data batches arriving every 60 epochs (see Section V-A for more details). The accuracy and loss signals differ according to the learning rate (see Figure 2): with $\lambda = 0.05$, the accuracy improves the fastest and the loss also quickly converges to its lower limit. However, the noise of the curve at the last epochs is also higher than with the two other scenarios because a large learning rate oscillates around the minimum. When $\lambda = 0.005$, the accuracy increase is slower but the loss function varies more smoothly, the loss value rarely rises.

Thus, the learning rate is able to influence our performance indicators: it is suitable as a control signal.



(a) Validation accuracy

(b) Loss function

Fig. 2: Impact of different constant learning rates on accuracy and loss (CIFAR-10).

III. Motivation

The experiments shown in Figure 2 illustrate the advantages and drawbacks of large and small learning rates. A natural thought is to combine their benefits through learning rate scheduling: an initial phase with a large learning rate to quickly converge to a high-level accuracy, then a smaller value to smoothly approach the minimum and avoid the bumps on validation accuracy and loss. In the state of the art, there are some commonly used learning rate strategies that vary the learning rate through time. In the following of the paper, we will introduce two learning rate laws: (i) Keras-Time-Based-decay and (ii) Exponential-Sine-Wave-decay. They will later be used in Section V to compare with our proposed methods.

Keras-Time-Based-decay is a commonly and widely used learning rate strategy in Keras [17], which is a famous python deep learning library. The learning rate is computed as follow:

$$\lambda(k) = \frac{\lambda(k-1)}{1+\delta k} \tag{2}$$

where k is the number of epochs since the arrival of the last batch, $k \in \{1, \ldots, E\}$. δ is a hyperparameter enabling to tune the steepness of the time decay. We set $\lambda(0) = 0.01$ and $\delta = 0.001$ as suggested in [5].

The second common schedule is the exponential decay, it has been successfully used in neural network training. A good implementation is exponential decay sine wave learning rate schedule [10]. The original schedule is implemented to an offline setting, so to adapt this learning rate schedule into our online setting, we need to adjust their strategy to allow the learning rate decays to around 0 at the ending epochs of each batch. We will refer to this strategy as Exponential-Sine-Wave-decay. The adapted version is calculated as follow:

$$\lambda(k) = \lambda(0)e^{\frac{-\alpha k}{E}}(\gamma sin(\frac{\beta k}{2\pi}) + e^{\frac{-\alpha k}{E}} + 0.5) \tag{3}$$

where k shares the same definition as in eq. (2). E is the training epochs per batch. α, β and γ are three hyperparameters. In order to have a same behavior as in [10] during our shorter E, we set $\alpha = 3$, $\beta = 6$ and $\gamma = 0.4$. The constant 0.5 in the equation is important, it makes sure that $\lambda(k)$ is strictly positive.

IV. Performance-based Learning Rate Laws

In all the related work strategies, the learning rate is decreasing with time, in a predefined manner. The only differences between these strategies is that for some the learning rate decreases slowly in the beginning and faster in the end and for others is the opposite. We therefore introduce our control strategy where the learning rate is automatically computed based on the loss function (see Fig 1). Nevertheless, according to the definition of the loss function, the absolute loss value in itself does not give us much information since different size of training dataset can change the absolute loss value. Therefore, we normalize the value of $loss(k)$ by $loss(k = 0)$, where $loss(k)$ represents the loss value at k^{th} epoch since the arrival of the last batch.

Subsequently, we try three different control laws for computing the learning rate λ: Proportional-Control (P-Control), Proportional Integral-Control (PI-Control) and a Mixed Exponential PI-Control (E/PI-Control).

A. P-Control

In this case the learning rate depends proportionally on the loss value as follows:

$$\lambda(k) = K_P \frac{loss(k)}{loss(0)} \tag{4}$$

In general the value of $\frac{loss(k)}{loss(0)}$ varies between 0 and 1. Indeed, as the loss function decreases thanks to the Stochastic Gradient Descent, we know that we are approaching the minimum of the loss function and therefore the learning rate should be decreased in order not to skip it. The choice of K_P is important for the speed of convergence. Based on trial and error tests, we make it equal to the same value as the empirical starting learning rate from [17]: $\lambda(0) = K_P = 0.01$.

B. PI-Control

On one side, the hypothesis behind P-Control is that the loss is always decreasing; as we are getting closer to a minimum, the learning rate should slow down to better approach it. On the other side if the loss has decreased during last epoch we are in the good direction to find the minimum so we should reward last learning epoch by increasing the learning rate. This can be seen as adding an integral action to our controller. We express our PI-Control as follows:

$$\lambda(k) = K_P \frac{loss(k)}{loss(0)} - K_I \frac{loss(k) - loss(k-1)}{loss(0)} \tag{5}$$

where $K_P = 0.01$, and the integral parameter is empirically chosen at 5 times $\lambda(0)$, as we choose $\lambda(0) = 0.01$, then $K_I = 0.05$. As $-(loss(k) - loss(k-1))$ could also be negative, the integral part will introduce oscillations to the learning rate. In order to avoid that $\lambda(k)$ becomes negative due to the integral part, the PI-Control is turned to a P-Control if $\lambda(k)$ in PI-Control gets a negative value. Indeed, the P-Control will always return a positive value for the learning rate, as the loss function is by definition positive.

C. *E/PI-Control*

This third control law tries to accelerate the convergence speed by exponentially increasing the learning rate at the beginning of learning a new batch, as the data are new so there are more informations to learn. We present a two phases algorithm to control the learning rate: (i) an initial Exponential growth followed by (ii) a PI-Control. During the exponential growth period, the learning rate is increased each time step by a factor 2 to quickly reach the minimum. This phase is stopped when the loss starts increasing, and the learning rate is afterwards ruled by the PI-Control law. The PI phase is initialized with the last value of the learning rate before loss growth. The PI parameters are set according to the behavior of the systems during the E-phase. The E/PI-Control law during one batch is summarized in Algorithm 1.

Algorithm 1 E/PI-Control

```
 1: $\lambda(0) = 0.01$
 2: $k = 1$
 3: while $loss(k) \leq loss(k-1)$ do
 4:     $\lambda(k) = 2\lambda(k-1) = 2^k\lambda(0)$
 5:     $k = k+1$
 6: end while
 7: $\lambda(k+1) = \lambda(k-1)$
 8: $K_P = \lambda(k-1)$
 9: $K_I = 5 \times \lambda(0)$
10: for $i \in \{k+2, \dots, E\}$ do
11:     $\lambda(k) = K_P \frac{loss(k)}{loss(0)} - K_I \frac{loss(k) - loss(k-1)}{loss(0)}$
12: end for
```

V. Control Laws Evaluation

In this section, the P, PI and E/PI-Control laws are evaluated in comparison with the state of the art. The stability of the E/PI-Control law is highlighted, and its robustness with regards to its initial configuration is presented. First, details on the datasets, CNNs and evaluation indicators are given.

A. *Experimental setup*

The controllers are evaluated on two datasets: CIFAR-10 and Fashion-MNIST. The CNN and scenario configurations for the two datasets is sum-up in Table I. As the images in CIFAR-10 have colors and are larger than the ones of Fashion-MNIST, a more complex CNN setting with more layers and parameters is used. Meanwhile, as there are more informations to extract from CIFAR-10, the number of epochs per batch is larger, allowing the accuracy curve to converge. All the values of hyperparameters of eq. (2) and (3) we showed in section.III are tuned for CIFAR-10, as Fashion-MNIST has shorter epochs per batch, we will change δ to 0.01 of eq. (2), and set $\alpha = 2$, $\beta = 18$ of eq. (3) for Fashion-MNIST experiment learning rate schedule.

To eliminate the influence of the CNN's weights starting point to the final accuracy, we initialize the weights of each layer of CNN by Xavier uniform initializer [4], all the results will be averaged on 3 time experiment results. All code is implemented with Keras library [17].

TABLE I: CNN configuration

Use case	**CIFAR-10**	**Fashion-MNIST**
#data instances to train	50,000	60,000
#data instances to test T	10,000	10,000
#classes C	10	10
image size	32×32	28×28
data batch size B	10000	10000
#trainng epochs per batch E	60	20
#CNN layers	28	15
#CNN parameters	1,641,858	422,538

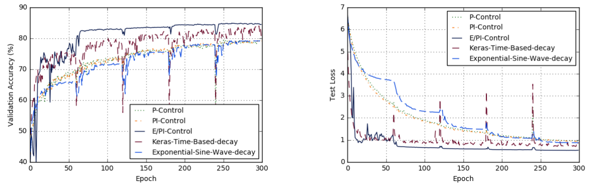
For performance evaluation, we measure several indicators on the accuracy and loss signals, the first one being their final value. To quantify the accuracy's converging speed, we will report for each experiment the epoch at which they reached 95% of their final accuracy. To compare the influence of each learning rate strategy on stability of validation accuracy, we will also compare the standard deviation of the accuracy curve on the last 10% epochs of each experiment.

B. *Stability analysis*

Stability of the presented algorithms needs to be proved to ensure that the error signal (the loss function) will not diverge, and ideally converge to 0. The stability theory behind the algorithm is based on the SGD: the direction of the gradient is always set to decrease the loss. The only case that loss will increase is because the learning rate is too large (i.e. we skipped the minimum). The stability of the P and PI-Controllers are ensured via a proper parametrization of K_P and K_I. The E/PI-Control law allows the learning rate to exponentially grow, however the learning rate is switched to a PI law as soon as the loss increases. The reset of the learning rate to the previously stable value ($7^{th} - 9^{th}$ line of Algorithm 1) enables to properly initialize the PI.

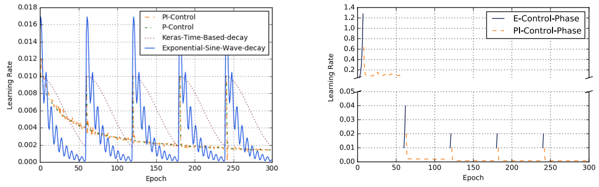
C. *P, PI and E/PI-Control Performances Validation*

The three control laws presented in Section IV are evaluated on CIFAR-10. Results are reported in Figure 3 through the accuracy 3a and loss functions 3b, and the corresponding control signals are illustrated in Figure 4, For the P and PI in 4a and for the E/PI law in 4b. There are few differences between P and PI control performances, while the E/PI-Control is significantly faster (61 epochs rising time compared to 130 for the P and PI), converges to a higher accuracy (+7%) and lower loss (-37%) and the standard deviation of the accuracy at the end of the experiment is three times lower. We see from the first epochs that the E-phase enables to properly tune the



(a) Validation accuracy (b) Loss function

Fig. 3: Performances of state of the art, P, PI and E/PI-Control (CIFAR-10).



(a) P and PI-Control (b) E/PI-Control

Fig. 4: Control signal of state of the art, P, PI and E/PI-Control (CIFAR-10).

initial value of the PI, which then significantly increases the validation accuracy. The P and PI-Control learning rate signal (Figure 4a) illustrates that a reset of the learning rate at the arrival of a new batch is not necessary beneficial if the value is not carefully chosen, as for the E/PI-Control.

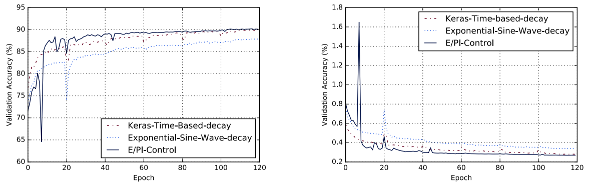
The loss function with the PI-Control declines a little bit faster than with the P-Control at beginning, and do not present a large peak around epoch 240. Those advantages made us opt for the PI-Control to combine with the initial E-phase.

D. Comparison with state of the art

Comparison of the state of the art learning rate strategies to our E/PI-Controller is provided for CIFAR-10 (Figures 3 and 4) and for Fashion-MNIST (see Figure 5 for the accuracy and loss and Figure 6 for the learning rate evolution through epochs).

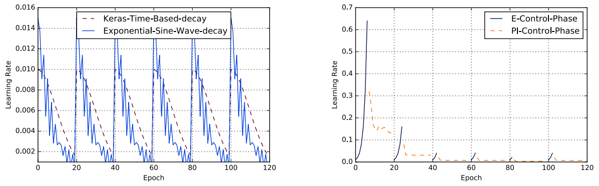
E/PI-Control provides the best results for all the indicators for CIFAR-10. It converges faster and has a smallest standard deviation of last 10% epochs among all the strategies, it reaches at a higher final validation accuracy (+3%) and a lower loss. Keras-Time-Based-decay has a closer final accuracy and loss to E/PI-Control. But the deviation of its accuracy curve is bigger than E/PI-Control, especially at the beginning of learning a new batch.

Regarding Fashion-MNIST dataset, results are similarly in favor of the E/PI-Controller, even if the differences are smaller. As this dataset is easier than CIFAR-10, all strategies reached a high validation accuracy and lower loss, the standard deviation of accuracy of last 10% epochs is also very small.



(a) Validation accuracy (b) Loss function

Fig. 5: Performances of state of the art and E/PI-Control on Fashion-MNIST.



(a) State of the art strategies (b) E/PI-Control

Fig. 6: Control signal for the state of the art and E/PI-Control on Fashion-MNIST.

E. Robustness to initial value of the learning rate

The E/PI-Control is now compared with the best strategy from the state of the art (Keras-Time-Based-decay) when the initial learning rate varies. Results are showed in Table II for CIFAR-10 and Table III for Fashion-MNIST.

Among all the experiments on CIFAR-10, there are only one case for which Keras-Time-Based-decay law has a better indicator (final validation accuracy at initial learning rate 0.05). The difference is very small, and standard deviation of the indicator itself is large. Moreover, if we check the accuracy's standard deviation during the last 10% epochs, Keras-Time-Based-decay still has a strong oscillation, which makes the model unpredictable. E/PI-Control also shows the advantage on converging time, it makes the model converge faster and rarely affected by the initial values.

Table III shows the robustness results on Fashion-MNIST. E/PI-Control still shows a fast converging speed, reaching 95% of final accuracy just using 7 to 10 epochs. The performances for the final loss and accuracy final standard deviation are similar for the two strategies. The E/PI-Control's final accuracy performances among all the experiments is more stable than with Keras-Time-Based-decay, which again, shows that E/PI-Control is more robust to the initial learning rate variations.

VI. Conclusion

When performing image classification tasks with neural networks, often comes the issue of on-line training, from sequential batches of data. Iterative training of CNNs is driven by a learning rate - how much to update the network weights with the new data - which value is usually ruled by a time decreasing function. This paper presents a control approach to the challenge of on-line training of CNNs, that

TABLE II: Robustness experiments with varying initial learning rate on CIFAR-10. Mean value (and standard deviation) over 3 runs. The best results are highlighted in **bold**.

Algorithm	**Initial learning rate**	**Final loss**	**Final validation accuracy (%)**	**Final accuracy standard deviation**	**First epoch to reach 95% accuracy**
Keras	0.001	0.849(0.023)	79.075(0.485)	0.371(0.053)	161.333(24.495)/300
E/PI-Control	0.001	**0.648(0.015)**	**82.035(0.465)**	**0.057(0.013)**	**61.333(0.471)/300**
Keras	0.002	0.745(0.026)	80.180(0.445)	0.415(0.049)	118.333(9.78)/300
E/PI-Control	0.002	**0.586(0.006)**	**83.150(0.225)**	**0.077(0.017)**	**62(0)/300**
Keras	0.05	0.727(0.006)	**85.640(0.523)**	1.630(0.085)	103.333(2.859)/300
E/PI-Control	0.05	**0.555(0.005)**	85.060(0.090)	**0.117(0.001)**	**61.333(0.471)/300**
Keras	0.1	0.829(0.180)	84.433(2.82)	1.609(0.432)	77.333(32.785)/300
E/PI-Control	0.1	**0.578(0.013)**	**85.075(0.585)**	**0.345(0.16)**	**65(0.816)/300**

TABLE III: Robustness experiments with varying initial learning rate on Fashion-MNIST. Mean value (and standard deviation) over 3 runs. The best results are highlighted in **bold**.

Algorithm	**Initial learning rate**	**Final loss**	**Final validation accuracy (%)**	**Final accuracy standard deviation**	**First epoch to reach 95% accuracy**
Keras	0.001	0.413(0)	85.055(0.035)	0.054(0)	37(0.816)/120
E/PI-Control	0.001	**0.334(0.002)**	**87.955(0.105)**	**0.023(0.008)**	**10.667(1.247)/120**
Keras	0.002	0.360(0.001)	86.850(0.005)	0.066(0.002)	25.667(1.247)/120
E/PI-Control	0.002	**0.350(0.008)**	**87.415(0.103)**	**0.057(0.011)**	**8.333(0.471)/120**
Keras	0.05	0.282(0.026)	89.785(0.920)	0.145(0.012)	16.667(6.532)/120
E/PI-Control	0.05	**0.263(0.006)**	**90.425(0.200)**	**0.094(0.013)**	**9.333(1.700)/120**
Keras	0.1	0.265(0.016)	90.400(0.674)	0.133(0.010)	9(3.265)/120
E/PI-Control	0.1	**0.249(0.003)**	**91.340(0.140)**	**0.114(0.015)**	**7(0)/120**

decides the learning rate value based on the expected learning need (i.e. the CNN loss function) instead of being time-based. E/PI-Control is a strategy that combines a phase of exponential growth of the control signal (i.e. learning rate) with a PI controller, which parameters are automatically adapted based on the E-phase.

Stability of the control strategy is provided, and evaluation highlights that E/PI-Control achieves a higher accuracy level in a shorter time than the state of the art solutions. Robustness of the approach is illustrated by its performances on two different datasets, and enforced by a sensitivity analysis regarding its initialization.

This work could be further extended by the addition of a triggering mechanism to smartly adapt the number of epochs needed at each batch processing. Moreover, we want to investigate the performances of the E/PI-Control in the scenario when new classes appear in some batches.

References

[1] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann.lecun.com/exdb/lenet*, p. 20, 2015.

[2] B. T. Morris and M. M. Trivedi, "Learning, modeling, and classification of vehicle track patterns from live video," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 3, pp. 425–437, Sep. 2008.

[3] M. Lease, "On quality control and machine learning in crowdsourcing," in *Human Computation*, vol. 11, no. 11, 2011.

[4] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15*, 2010, pp. 249–256.

[5] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 437–478.

[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[7] L. N. Smith, "Cyclical learning rates for training neural networks," in *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE, 2017, pp. 464–472.

[8] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[9] X. Wu, R. Ward, and L. Bottou, "Wngrad: Learn the learning rate in gradient descent," *arXiv preprint arXiv:1803.02865*, 2018.

[10] W. An, H. Wang, Y. Zhang, and Q. Dai, "Exponential decay sine wave learning rate for fast deep neural network training," in *2017 IEEE Visual Communications and Image Processing (VCIP)*, Dec 2017, pp. 1–4.

[11] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[12] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[14] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, Aug 2018.

[15] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *Journal of Physiology (London)*, vol. 195, pp. 215–243, 1968.

[16] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, pp. 193–202, 1980.

[17] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

Titre — Contrôle des systèmes informatiques: application aux services clouds et à la protection de vie privée.

Résumé — Un algorithme de contrôle peut gérer des systèmes complexes, même lorsqu'ils sont particulièrement sensibles aux variations de leur environnement. Cependant, l'application du contrôle aux systèmes informatiques soulève plusieurs défis, par exemple dû au fait qu'aucune loi physique ne régisse leur comportement. D'une part, le cadre mathématique fourni par la théorie du contrôle peut être utilisé pour améliorer l'automatisation, la robustesse et la fiabilité des systèmes informatiques. D'autre part, les défis spécifiques de ces cas d'étude permettent d'élargir la théorie du contrôle elle-même. L'approche adoptée dans ce travail consiste à utiliser deux systèmes informatiques d'application: la protection de vie privée liée à la mobilité et les performances des services clouds. Un troisième cas d'utilisation sur le contrôle des algorithmes d'apprentissage automatique est présenté en annexe. Ces cas d'utilisation sont complémentaires par la nature de leurs technologies, par leur échelle et par leurs utilisateurs finaux.

La popularité des appareils mobiles a favorisé la diffusion et la collecte des données de localisation, que ce soit pour que l'utilisateur bénéficie d'un service personnalisé ou pour que le prestataire de services tire des informations utiles des bases de données de mobilité, au prix de la diffusion de données personnelles parfois très sensibles. Pour remédier à cette atteinte à la vie privée, des mécanismes de protection spécifiques aux données de mobilité (LPPM) ont été élaborés. Cependant, ces outils ne sont pas facilement configurables par des novices et ne s'adaptent pas à la mobilité de l'utilisateur. Dans cette thèse, nous développons deux outils, l'un pour les bases de données déjà collectées et l'autre pour l'utilisation en ligne, qui garantissent aux utilisateurs des niveaux de protection de la vie privée et de préservation de la qualité des services en configurant les LPPMs. Nous présentons la première formulation du problème en termes de théorie du contrôle (système et contrôleur, signaux d'entrée et de sortie), et un contrôleur PI pour servir de démonstration d'applicabilité. Dans les deux cas, la conception, la mise en œuvre et la validation ont été effectuées par le biais d'expériences utilisant des données réelles.

L'essor récent des bigdata a conduit au développement de programmes capables de les analyser, tel que MapReduce. Les progrès des pratiques informatiques ont également permis d'établir le modèle du cloud (location de ressources en ligne prêtes à l'emploi) comme une solution incontournable pour tous types d'utilisateurs. Dans ce travail, nous nous intéressons aux performances des tâches MapReduce exécutées sur les clouds et développons des techniques avancées de contrôle du temps d'exécution des tâches et de la disponibilité de la plate-forme; en ajustant la taille du cluster de ressources et en réalisant un contrôle d'admission, fonctionnant quelle que soit la charge de clients. Afin de traiter les non linéarités, un contrôleur adaptatif a été conçu. Pour réduire l'utilisation du cluster et ses coûts, nous présentons une nouvelle formulation du mécanisme de déclenchement du contrôle événementiel, combiné à un contrôleur prédictif optimal. L'évaluation est effectuée sur un benchmark s'exécutant en temps réel sur un cluster, et en utilisant des charges de travail industrielles.

Les algorithmes d'apprentissage automatiques sont maintenant répandus dans le monde industriel et académique. Malgré d'excellentes performances, d'autres aspects ont été négligés jusqu'à présent, tels que l'automatisation de leur configuration ou leur robustesse. Nous réalisons donc le contrôle d'algorithmes d'apprentissage de deux manières complémentaires: garantie de la robustesse vis-à-vis du bruit de la base de données, et le paramétrage automatique des algorithmes par contre-réaction. Les résultats sont validés à l'aide de bases de données classiques et industrielles.

Mots clés — Contrôle pour les systèmes informatiques; Problème inverse; Contrôle de la vie privée pour la mobilité; Automatisation des clouds; Apprentissage automatique.

Title — Control Theory for Computing Systems. Application to big-data cloud services & location privacy protection.

Abstract — This thesis aims at investigating techniques to build and control efficient, dependable and privacy-preserving computing systems. Ad-hoc service configuration require a high level of expertise which could benefit from automation in many ways. A control algorithm can handle bigger and more complex systems, even when they are extremely sensitive to variations in their environment. However, applying control to computing systems raises several challenges, e.g. no physics governs the applications. On one hand, the mathematical framework provided by control theory can be used to improve automation and robustness of computing systems. Moreover, the control theory provides by definition mathematical guarantees that its objectives will be fulfilled. On the other hand, the specific challenges of such use cases enable to expand the control theory itself. The approach taken in this work is to explore in details two application computing systems: location privacy and cloud services. A third use-case on the use of control for machine learning algorithm is presented in appendix. Those use-cases are complementary in the nature of their technologies, scale and end-users.

The widespread of mobile devices has fostered the broadcasting and collection of users' location data. It enables users to benefit from a personalized service and service providers or any other third party to derive useful information from the mobility databases, whereas it also exposes highly sensitive personal data. To overcome this privacy breach, algorithms have been developed that modify the user's mobility data, hopefully to hide some sensitive information, called Location Privacy Protection Mechanisms (LPPMs). However, those tools are not easily configurable by non experts and are static processes that do not adapt to the user's mobility. We develop two tools, one for already collected databases and one for online usage, that, by tuning the LPPMs, guarantee to the users objective-driven levels of privacy protection and of service utility preservation. First, we present an automated tool able to choose and configure LPPMs to protect already collected databases while ensuring a trade-off between privacy protection and database processing quality. Second, we present the first formulation of the location privacy challenge in control theory terms (plant and control, disturbance and performance signals), and a feedback controller to serve as a proof of concept. In both cases, design, implementation and validation has been done through experiments using data of real users.

The surge in data generation of the last decades, the so-called bigdata, has lead to the development of frameworks able to analyze them, such as the well known MapReduce. Advances in computing practices have also settled the cloud paradigms (online ready-to-use resources to rent) as premium solution for all kind of users. In this work, we focus on performance of MapReduce jobs running on clouds and thus develop advanced monitoring techniques of the jobs execution time and the platform availability; by tuning the resource cluster size and realizing admission control, in spite of the unpredictable client workload. In order to deal with the non linearities of the MapReduce system, a robust adaptive feedback controller has been designed. To reduce the cluster utilization and costs, we present a new event-based triggering mechanism formulation combined with an optimal predictive controller. Evaluation is done on a MapReduce benchmark suite running on a large-scale cluster, and using real jobs workloads.

Learning algorithms are now prevalent in both the research and industry worlds. While they show impressive results in terms of performance, other aspects has been neglected so far, such as automation, robustness or privacy. Machine learning algorithms control is investigated in two complementary ways: robustness regarding noise in the dataset, and the parametrization of the algorithms, with the introduction of feedback action. Results are validated using classic datasets and task-specific ones.

Keywords — Control for Software Systems; Inverse Problem; Control of Location Privacy; Cloud Control; Machine Learning.