The Pennsylvania State University

The Graduate School

Department of Electrical Engineering

TRAFFIC ENGINEERING AND TIME-VARYING CONVEX OPTIMIZATION

A Dissertation in

Electrical Engineering

by

Wenjing Su

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2009

The dissertation of Wenjing Su was reviewed and approved* by the following:

Constantino M. Lagoa
Associate Professor of Electrical Engineering
Dissertation Adviser
Chair of Committee

Tom M. Cavalier
Professor of Industrial Engineering

George Kesidis
Professor of Electrical Engineering
Professor of Computer Science and Engineering

Ji-Woong Lee
Assistant Professor of Electrical Engineering

Mario Sznaier
Professor of Electrical and Computer Engineering
Northeastern University

W. Kenneth Jenkins
Professor of Electrical Engineering
Head of the Department of Electrical Engineering

*Signatures are on file in the Graduate School.

Abstract

Computer network traffic engineering aims at providing algorithms supporting Traffic Engineering (TE) for resource optimization (multi-path load balancing), Quality of Service (QoS), and Fast Failure Recovery (FFR) (dealing with link/node failures). This dissertation addresses two computer network traffic engineering problems: the multi-domain traffic engineering problem, and the overlay network traffic engineering problem. The solutions provided have their basis on nonlinear control theory. More precisely, they use concepts from sliding mode theory.

The motivation for the study of the multi-domain traffic engineering problem comes from the increasing demand for the Internet to provide rich service quality features in support of sophisticated applications at a global scale, including TE, QoS, and FFR. We believe that, to be deployable at a global scale and to be integrated into the basic Internet architecture, a successful solution for a multi-domain environment must be distributed by design and in line with the distributed, two-level routing structure, and hop-by-hop forwarding paradigm of today's Internet. In this dissertation, as a first but critical step toward finding such a solution, we develop a well-grounded theoretical underpinning for it. This family of control laws proposed for a multi-domain environment enables QoS-based TE and FFR features by performing per-hop edge-to-edge based traffic control at two levels (i.e., inter-domain and intra-domain), in alignment with the two-level routing structure and hop-by-hop forwarding paradigm of the Internet.

Besides the multi-domain traffic engineering problem, we also address the overlay traffic engineering problem. Again, the approach used in the development of the control laws is based on the sliding mode theory. An overlay network, built at network application layer, is another prominent approach to provide QoS feature in the current best-effort Internet infrastructure. Given an overlay network, the goal of overlay network traffic engineering problems is to distribute traffic among available overlay paths in order to optimize the use of time-varying network resources. Due to the presence of time-varying network resources (link capacity), as well as the feasible set and the set

of optimal solutions being time-varying, the problem is fundamentally a time-varying optimization problem and different from the problems previously addressed in the literature. And in this dissertation, we are able to find a new family of control laws, which addresses the time-varying problem. The family of control laws presented in this dissertation is shown to converge to the optimal (time-varying) traffic allocation and uses only the number of congested links in a forwarding path as feedback for the control, making it an ideal traffic control solution for the overlay network.

Since the overlay network traffic engineering problem is a time-varying optimization problem, we also extend our research to more general time-varying optimization problems. For the problem with a twice differentiable strictly convex objective function, a Continuous First Order Algorithm (CFoA) is proposed. Moreover, in order to achieve "smoother" behavior than the CFoA, a Continuous Second Order Algorithm (CSoA) is also proposed. Both the CFoA and CSoA are shown to converge to and track the time-varying optimum. For a subclass of strictly convex objective functions having derivatives with "linear" discontinuity, a Sliding Algorithm (SA) is proposed, which is shown to converge to an arbitrarily small neighborhood of the time-varying optimum. Moreover, the SA is applied to solve time-varying linearly constrained optimization problems, and sufficient conditions for asymptotical convergence of the SA are provided.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I am greatly indebted to my adviser, Dr. Constantino Lagoa, for his guidance and support throughout my doctoral program, without his help, this dissertation would not be possible. He is a very intelligent, responsible, considerate person, and a great mentor. I deeply respect him, and will miss all these years working under his guidance. I would also like to thank my committee members: Dr. Tom Cavalier, Dr. George Kesidis, Dr. Ji-Woong Lee, and Dr. Mario Sznaier, for their valuable time and guidance that helped to improve this dissertation. Special thanks also goes to Dr. Hao Che for his guidance.

I would like to thank Dr. Jialing Chen, Dr. Xiang Li, Dr. Bernardo Movsichoff, Dr. Wenjing Ma, Abdullah Ashoor, and Chao Feng, for their help and friendship.

I would like to thank my parents and parents-in-law, for their support and encouragement.

I dedicate this dissertation to my husband.

Chapter 1

Introduction

Traffic Engineering aims at providing Quality-of-Service (QoS) in the current best-effort Internet. In this dissertation, by using the sliding mode control, we address two traffic engineering problems: the multi-domain traffic engineering problem, and the overlay network traffic engineering problem. Moreover, we extend these results to more general time-varying optimization problems. We now briefly review the related work, our work about the traffic engineering problems addressed in this dissertation, and time-varying optimization problems, respectively.

1.1 Traffic Engineering Problem

The Internet has quickly evolved into a very critical communications infrastructure, supporting significant economic, educational, and social activities. Simultaneously, the delivery of Internet communication services has become very competitive, and there has been an increasing demand for the Internet to provide rich service quality features in support of sophisticated applications at a global scale. However, despite the great effort made in the past decades, today's Internet, for a large part, can only provide a single Best Effort (BE) service and there has been no large-scale deployment of better-than-BE service quality features to date.

Traffic Engineering [3] has been considered as one of the vital components of an autonomous system required to achieve both high resource utilization and high quality of service for both real time and non real-time applications. The basic idea is to split the traffic flows among multiple paths or steer the traffic away from a shortest path found by the interior gateway protocols, so that the congestion is avoided and network resource utilization is maximized.

In this dissertation, we address two traffic engineering problems: the multi-domain traffic engineering problem, and the overlay network traffic engineering problem, by using an optimization-based method. We first review the pricing and fairness issue in Traffic Engineering, and the optimization-based methods to address traffic engineering problems, then introduce the multi-domain traffic engineering problem in Section 1.1.3, and the overlay network traffic engineering problem in Section 1.1.4, respectively.

1.1.1 Pricing and Fairness in Computer Network

In this dissertation, we will use an optimization-based approach to design data rate control algorithms for the multi-domain traffic engineering problem, and the overlay traffic engineering problem. It has been widely acknowledged that economic problems, such as pricing and fairness, play an important role in such a computer network traffic engineering problem. Though these issues are beyond the scope of our research, a short introduction is helpful for understanding the problem formulation in the later chapters.

If the available resource, in this case bandwidth, far exceeds the demand, there is little role for pricing mechanisms. However, this is not the case in the current networks. When the demand exceeds the available resources, pricing network services become an important issue, due to the fact that the network behavior depends on the aggregated traffic load of the network – the result of many users' individual decisions on how to use the network, and these decisions are affected by prices users face [30]. Pricing mechanisms provide incentives and penalties which prompt users to choose their service requirement while considering the price. Hence, pricing mechanism is an effective way to alleviate congestion, and additionally, it will automatically generate appropriate amount of revenue to finance capacity expansion [23].

The concept of fairness as seen by the users' point of view, has been widely used in optimization-based Traffic Engineering, as a reference for the design of objective functions that measure the efficiency of the network resources utilization. We now introduce some notions of fairness that are commonly used in most related works. Assume a network in which there exist several source nodes, each connected to a destination node. Each source–destination pair establishes a flow that is assigned a transmission rate x_i,

for $i \in \mathcal{S} \doteq \{1, 2, ..., S\}$. Furthermore, let $\boldsymbol{x}$ be the vector containing all these rates. The objective of the rate adaptation algorithm is to determine the rate vector $\boldsymbol{x}$ that maximizes a given optimality criterion. However, this optimal allocation should be "fair" in some sense. Three standard fairness criteria that are widely used in the field of computer network are presented below [26].

A rate vector $\boldsymbol{x}$ is min-max fair if it is feasible (i.e., satisfies all the constraints of the optimization problem) and for each $i \in \mathcal{S}$, x_i cannot be increased without decreasing x_j for some $j \neq i$ with a lower rate; i.e., $x_j \leq x_i$. This concept of fairness prioritizes smaller flows; i.e., if $x_j \leq x_i$ then no increase in x_i can compensate for a decrease in x_j.

On the other hand, a rate vector $\boldsymbol{x}$ is proportionally fair if it is feasible and given any other feasible $\hat{\boldsymbol{x}}$ the aggregate of proportional changes is not positive

$$\sum_{i \in \mathcal{S}} \frac{\hat{x}_i - x_i}{x_i} \leq 0. \tag{1.1}$$

If the concept of price per unit time w_i that the user at source i is willing to pay is introduced, then the proportional fairness concept can be slightly modified. A rate vector $\boldsymbol{x}$ provides proportionally fair rates per unit charge if instead of (1.1) the following inequality is satisfied under the same assumptions of proportional fairness

$$\sum_{i \in \mathcal{S}} w_i \frac{\hat{x}_i - x_i}{x_i} \leq 0.$$

Given the considerations above, in this dissertation we will address the problem of maximizing a given utility function subject to both resource and Class-of-Service (CoS) constraints. Although not explicitly mentioned in the remainder of this dissertation, it is assumed that the utility function to be maximized is related to the economic and fairness issues discussed above.

1.1.2 Traffic Engineering and Optimization-based Distributed Method

The general approach taken to address the optimization-based, distributed traffic control problem is to formulate it as an optimization problem based on a fluid-flow model,

taking into account link bandwidth constraints. The objective function represents desired pricing policies. The aim is to find distributed traffic control laws which, by working independently of one another, will drive the network to an operation point where the given utility function of flow rates is maximized. Since different flows share the link resources which are constrained, the key challenge in the design of distributed control laws is the high degree of interaction between different flows.

A significant amount of work has been done on adaptive rate allocation algorithms. In this section, we focus exclusively on surveying some of the literature relevant to the work presented in this dissertation, i.e., optimization-based, multipath-enabled, distributed traffic control schemes.

The first approach (e.g. [6, 9]) is to incorporate link congestion costs into the overall utility function in order to convert a constrained problem into a non-constrained problem. The optimization problem is then solved using a gradient type of algorithm. Iterative algorithms have been proposed where individual sources periodically adjust/balance their flow sending rates to multiple paths based on the congestion cost information fed back from the (congested) links along each path.

The second approach (e.g., [11, 15, 16]) is to solve a relaxation of the original problem, by incorporating a price function into the overall utility function. Distributed control laws have been found, proven to be locally stable in the presence of variable feedback delays. Working independently at a source, a control law adjusts/balances its flow sending rates to multiple paths based on periodical, cumulative price feedbacks from the destination node. Each component price is collected from the intermediate links along the forwarding path.

The third approach is to solve the original problem directly (e.g., [19, 20, 24, 25, 35]). Using a duality model, an algorithm was provided [35]. Similar to the second approach, this solution requires a cumulative price to be conveyed to the source periodically. In [19, 20, 24, 25], the authors have tackled this problem using a technique based on the sliding mode theory. Both end-to-end [19, 20, 25] and hop-by-hop [24] optimal control laws have been found. These algorithms allow multipath forwarding, enable multiple CoSes, and require minimum information feedback for the control.

Motivated by the third approach reviewed in this section, which is based on the sliding mode theory, in this dissertation we use the same approach to address the multi-domain traffic engineering problem and the overlay traffic engineering problem, and propose data rate control algorithms for them. In the next sections, we give the motivation to study these two problems.

1.1.3 Multi-domain Traffic Engineering Problem

As the Internet has evolved into a global commercial infrastructure, there has been an increasing demand for it to provide rich service quality features in support of sophisticated applications at a global scale, including Quality-of-Service (QoS), Traffic Engineering (TE), and Fast Failure Recovery (FFR). However, despite great effort made over the past decades, today's Internet can only provide a single BE service and there has been no large-scale deployment of better-than-BE service quality features to date. The root cause of this status quo has much to do with the existing design approaches taken to enable better features, which are at odds with the design approach that has made the Internet a success in terms of global reachability. The central idea of the existing design approaches is to try to embed reliable end-to-end path (with protection) over an unreliable, connectionless, multi-domain Internet, such as Multi-Protocol Label Switching (MPLS). Such design approaches are complex and significantly deviate from the one that has made the current Internet highly scalable. They require adding a connection-oriented forwarding paradigm and hence, strong coordination among domains over a connectionless, loosely-coordinated-multi-domain Internet. As a result, they are expensive; cannot be implemented at large-scale; and are difficult to integrate into the basic Internet architecture.

Given the size of the Internet, a solution for providing service-quality features must be distributed by design. Moreover, it must be able to allow quick response to network congestions and link/node failures. Therefore, we now focus on reviewing literature about dynamic, distributed traffic control.

First, most previous literature on the existing online distributed traffic control approaches at the the IP layer has focused on work about a single domain. In particular,

a large number of optimization-based intra-domain TE and/or FFR mechanisms have been proposed (e.g., [6, 9, 13, 14, 15, 18, 22]). These mechanisms can only deal with rate-adaptive traffic and hence cannot provide QoS features. In another series of papers [19, 20, 24, 25], the authors have found large families of optimization-based, distributed controllers that provide QoS, TE, and FFR features simultaneously.

Second, although TE has been an extensively studied subject at the inter-domain level, most studies have been carried out in the context of BGP route selections (e.g., [4, 7, 34, 36]). Some of the solutions of this kind complement the solution proposed in this dissertation by providing it with well-engineered inter-domain multiple routes (e.g., [4, 36]). To date, there has been no TE solution which jointly optimizes inter-domain and intra-domain routing. Most multi-domain-based QoS solutions are provisioning based and generally involve policy servers or bandwidth brokers to coordinate resource provisioning within and between domains (e.g., [12, 27]). In particular, [12] provides algorithms for joint inter-domain-and-intra-domain QoS provisioning. Again, these solutions complement the solution proposed in this dissertation by providing it with resource guarantee for QoS-based flows or lossless failure recovery.

To the best of our knowledge, there has been no existing online, distributed traffic control mechanism that can jointly optimize inter-domain and intra-domain traffic allocation. We believe that to be deployable at a global scale and to be integrated into the basic Internet architecture, a successful solution must be distributed by design and in line with the distributed two-level routing structure and the hop-by-hop forwarding paradigm of today's Internet. As a first but critical step toward finding such a solution, we develop a well-grounded theoretical underpinning for it. More specifically, based on a fluid-flow model, we find a large family of optimization-based control laws. This family of control laws enables CoS-based TE and FFR features by performing per-hop, edge-to-edge based traffic control at two levels (i.e., inter-domain and intra-domain), in alignment with the two-level routing structure and hop-by-hop forwarding paradigm of the Internet. To the best of our knowledge, the family of control laws presented in this dissertation is the only optimization-based, distributed, multi-domain traffic control algorithm being found to date.

1.1.4 Overlay Network Traffic Engineering Problem

An overlay network is an alternative approach to support QoS-sensitive applications, which is formed by a subset of nodes of underlying physical nodes. The connections between the overlay nodes are provided by overlay links, each of which is usually comprised of one or more physical links [21]. Based on the overlay node types, overlay networks can be classified into two categories: end-user centric and network centric. In an end-user centric overlay network, the overlay nodes are end-hosts, which are linked together through network layer to form an overlay network. In a network centric overlay network, the overlay nodes are pre-selected network layer nodes/servers, which are connected by overlay links. Since overlay applications are usually built at the application layer, they can effectively use the underlying best-effort network layer as a lower-level infrastructure to provide high-level services to end users while retaining the protocols of network layer.

Some literature documents the emergence of several service-based overlay architectures, e.g., [1, 2, 5, 11, 21, 32], aimed at providing *service quality features* largely unattainable in today's best-effort Internet. Resilient Overlay Network (RON) [2] is an overlay architecture which allows distributed Internet applications to detect and recover from path outages and degraded performance periods quickly, providing better routing. Service Overlay Network [5] is a logical end-to-end service delivery infrastructure addressing end-to-end QoS problem. QoS-aware routing protocols for overlay networks (QRON) [21] provide QoS-satisfied overlay paths and balance the overlay traffic.

However, because the resource availability and topology of underlying network layer are largely hidden from the overlay network, these overlay architectures require a proprietary functionality support, such as static or dynamic resource isolation or active probing/discovery. As a result, the approaches taken in solving QoS issues are quite diversified and strongly dependent on the actual techniques in use. Moreover, each of these architectures solves just part of the overall service quality issues. These problems make the adoption of any one or any combination of these architectures for addressing the overall service quality issues more difficult. Hence, from a fundamental viewpoint, in

order to facilitate the support for existing and new overlay architectures, it is desirable to provide a unified mathematical framework which enables TE, QoS, and FFR.

However, due to the presence of time-varying network resources (link capacity), as well as the feasible set and the set of optimal solutions being time-varying, the existing optimization-based, distributed traffic control approaches reviewed in Section 1.1.2 can not handle the time-varying link capacity. The novelty of our work in this dissertation lies in the fact that, motivated by [19, 20], we are able to find a new family of control laws, which supports TE, QoS, and FFR. The family of control laws is shown to converge to the optimal (time-varying) traffic allocation and uses only the number of congested links in a forwarding path as feedback for the control, making it an ideal traffic control solution for the overlay. This is the case since each upstream overlay node can detect whether its downstream overlay link is congested, through a source inferred congestion detection mechanism, and then pass this information to the source. Hence, this approach allows both underlying network topology and resources to be hidden from the overlay network.

1.2 Time-varying Optimization Problem

The overlay traffic engineering problem reviewed in Section 1.1.4 is a resource allocation problem with time-varying resources. Hence, it is fundamentally a time-varying optimization problem, for which we design an optimization-based rate control algorithm. Having this as a starting point, we extend these results to more general time-varying optimization problems. We now briefly review the literature on time-varying optimization problems.

Many problems in engineering, economics, and management, such as optimal control, system identification, and optimal resource allocation, can be formulated as time-varying optimization problems. A Time-varying problem involves an objective function and/or constraints which depend on time.

The optimal solution of a time-varying optimization problem, $\boldsymbol{x}_{opt}(t)$, is a function of time t. In the literature on parametric optimization, one can find sensitivity analysis

results that provide the dynamics of the time-varying optimum of unconstrained problems [28] and constrained problems [10], under the condition that the optimum $\boldsymbol{x}_{opt}(t)$ is differentiable with respect to the parameter t. If at some time t_0, one starts at the optimum $\boldsymbol{x}_{opt}(t_0)$, then the dynamics of the optimum of $t > t_0$ can be calculated. However, sensitivity analysis is mainly of theoretical significance, since $\boldsymbol{x}_{opt}(t_0)$ cannot, usually, be calculated analytically.

In our opinion, to address this problem in a practical setting, an algorithm should have the following two properties: convergence to the time-varying optimum and the ability to track it. From this viewpoint, we now review the literature on algorithms for time-varying optimization problems. To the best of our knowledge, there are only limited results in this area. In [29], it was proven that the gradient method is able to converge to an arbitrarily small neighborhood of the optimum $\boldsymbol{x}_{opt}(t)$ for a twice differentiable strictly convex objective function. An algorithm was proposed in [38], which is able to converge, in the sense that $\lim_{t\to\infty} \partial U(\boldsymbol{x}, t)/\partial \boldsymbol{x} = 0$, if the objective function $U(\boldsymbol{x}, t)$ is twice differentiable with respect to $\boldsymbol{x}$ for all t.

In this dissertation, we focus on a class of time-varying optimization problems with objective functions having "linear" discontinuous derivatives (the computer network traffic engineering problem with time-varying link capacity [15, 31] is an example of such problems), and use the smallest norm of gradient/subgradient as a descent function to design algorithms. These algorithms only require local information of the objective function.

First, for the time-varying optimization problem with a twice differentiable strictly convex/concave objective function, a Continuous First Order Algorithm (CFoA) is proposed. And in order to achieve "smoother" dynamics than the CFoA, a Continuous Second Order Algorithm (CSoA) is proposed. Both the CFoA and CSoA are shown to converge to and track the time-varying optimum. These results serve as a step stone for the algorithm design for a class of time-varying optimization problems with strictly convex/concave objective functions having derivatives with "linear" discontinuity. Then, for strictly convex/concave objective functions having derivatives with "linear" discontinuity, a Sliding Algorithm (SA) is proposed, which is shown to converge to an arbitrarily

small neighborhood of the time-varying optimum. Moreover, the SA can be applied to solve time-varying linearly constrained strictly convex optimization problems, and sufficient conditions for asymptotic convergence are presented.

1.3 The Sequel

The remainder of this dissertation is organized as follows. Chapter 2 introduces sliding mode theory which is the basic tool used to design control algorithms throughout this dissertation. Chapter 3 and Chapter 4 present our data rate control algorithms for the multi-domain traffic engineering problem and the overlay network traffic engineering problem, respectively. Chapter 5 presents our results for more general time-varying optimization problems. Chapter 6 gives concluding remarks. And finally, the proofs of the results in this dissertation are given in the Appendices.

Chapter 2

Sliding Modes in Mathematical Programming

In this chapter, we introduce the sliding mode control and its application in mathematical programming. The sliding mode theory is the basic tool used to design our algorithms throughout this dissertation.

2.1 Sliding Modes

The concept of sliding modes arises from the need to describe a number of systems which are characterized by the fact that the right hand sides of the differential equations describing their dynamics feature discontinuity with respect to the current process state [33]. In this dissertation, we consider systems of the general form

$$\dot{x} = f(x, u, t), \tag{2.1}$$

where $x \in \mathcal{R}^n$ is the state variable, $t \in \mathcal{R}$ is time, $u \in \mathcal{R}^M$ is the control vector defined as

$$u_i(x,t) = \begin{cases} u_i^+(x,t) & \text{if } s_i(x) > 0, \\ & i = 1, 2, ..., M, \\ u_i^-(x,t) & \text{if } s_i(x) < 0, \end{cases} \tag{2.2}$$

where $s_i(x) = 0, i = 1, 2, ..., M$ define discontinuity surfaces. The signals $u_i^+(x,t)$ and $u_i^-(x,t)$ are assumed to be continuous.

When the set of discontinuity points has a nonzero measure in time, the trajectory given by any combination of continuous controls $u_i^+(x,t)$ and $u_i^-(x,t)$ differs from the system trajectories [33]. Sliding modes is an accepted term for the motion on discontinuity surfaces. Sliding mode does exist on a discontinuity surface whenever the following

conditions are satisfied

$$\lim_{s \to -0} \dot{s} > 0 \quad \text{and} \quad \lim_{s \to +0} \dot{s} < 0.$$

2.2 Equivalent Control Method

The mathematical description of sliding modes is quite challenging, because the classical theory of differential equations can not adequately describe the behavior of systems on the discontinuity surfaces. The equivalent control method is one way to solve this problem. It reduces the original problem to a form which has a solution close, in a sense, to the solution of the original problem, enabling one to use classical analysis techniques [33].

For the system defined by equations (2.1) and (2.2), assume that sliding mode exists on the discontinuity surfaces $s(x) = \left[s_1(x), s_2(x), ..., s_m(x) \right]^T = 0$, the equivalent control method provides a way of determining the motion of the system by finding a continuous control such that $\dot{s} = 0$, for any x satisfying $s(x) = 0$. More precisely, find u such that:

$$\dot{s} = Gf(x, u, t) = 0, \tag{2.3}$$

where $G = \left[\nabla s_1 \quad \nabla s_2 \quad ... \quad \nabla s_m \right]^T$. Assume that at least one solution of equation (2.3) with respect to $u \in \mathcal{R}^m$ exists, such a solution is referred to as the equivalent control u_{eq}, substitute it into equation (2.1) to get

$$\dot{x} = f(x, u_{eq}, t). \tag{2.4}$$

Obviously, such a motion, starting from x_0 satisfying $s(x_0) = 0$, will keep satisfying $s(x) = 0$, and it can be used as one of the ways of describing the motion on the intersection of discontinuity surfaces $s(x) = 0$. The above procedure is referred to as the equivalent control method.

From the geometric view, illustrated by Fig. 2.1, the equivalent control method replaces the discontinuous control with a continuous control which directs the velocity vector along the discontinuity surfaces intersection [33].

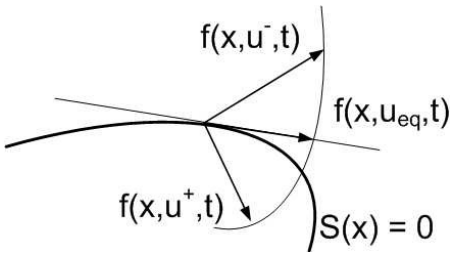


Fig. 2.1. Equivalent control method

2.3 Sliding Mode Algorithms in Mathematical Programming

An application of sliding mode theory of particular interest to the problems addressed in this dissertation is its use in nonlinear programming. In this section,we briefly introduce the sliding modes theory for solution of nonlinear optimization problem [33].

2.3.1 Convex Optimization Problem

A convex optimization problem is one of the form

$$\begin{aligned} &\text{minimize} \;\; U(x), \\ &\text{subject to} \;\; s_i(x) \leq 0, i = 1, 2, ..., m, \\ &\qquad\qquad s_i(x) = 0, i = m+1, m+2, ..., m+l, \end{aligned}$$

where the objective function $U(x) : \mathcal{R}^n \rightarrow \mathcal{R}$ is convex, the inequality constraints $s_i(x) \leq 0$, $s_i(x) : \mathcal{R}^n \rightarrow \mathcal{R}, i = 1, 2, ..., m$ are convex, and the equality constraints $s_i(x) = 0$, $s_i(x) : \mathcal{R}^n \rightarrow \mathcal{R}$, $i = m+1, m+2, ..., m+l$ are affine. For convex optimization problems,

every local minimum is a global minimum, and in addition if $U(x)$ is strict convex, this problem has a unique global minimum.

The following conditions are called Karush-Kuhn-Tucker (KKT) conditions

$$\nabla U(x) + \sum_{i=1}^{m} \lambda_i \nabla s_i(x) + \sum_{i=m}^{m+l} \nu_i \nabla s_i(x) = 0,$$

$$s_i(x) \leq 0, i = 1, 2, ..., m,$$

$$\lambda_i \geq 0, i = 1, 2, ..., m,$$

$$\lambda_i s_i(x) = 0, i = 1, 2, ..., m,$$

$$s_i(x) = 0, i = m+1, m+2, ..., m+l,$$

For convex optimization problems, the KKT condition is a sufficient condition for x to be an optimal point. Moreover, if a convex optimization problem only has linear constraints, the KKT condition is also a necessary condition. This is the case for the problems addressed in this dissertation.

2.3.2 Sliding Mode Algorithms in Mathematical Programming

For the convex optimization problem defined in Section 2.3.1, consider a piecewise continuous penalty function defined as

$$P(x) = s^T(x)u, \tag{2.5}$$

where

$$s(x) = \left[s_1(x), s_2(x), ..., s_{m+l}(x)\right]^T,$$

$$u = \left[u_1, u_2, ..., u_{m+l}\right]^T,$$

$$u_i = \begin{cases} \lambda_i \ \text{ if } \ s_i(x) > 0, \\ \qquad\qquad i = 1, 2, ..., m, \\ 0 \ \text{ if } \ s_i(x) < 0, \end{cases}$$

$$u_i = \begin{cases} \lambda_i \ \text{ if } \ s_i(x) > 0, \\ \qquad\qquad\qquad i = m+1, m+2, ..., m+l, \\ -\lambda_i \ \text{ if } \ s_i(x) < 0, \end{cases}$$

$$\lambda_i > 0, \lambda_i = constant, i = 1, 2, ..., m+l.$$

The penalty function $P(x)$ is constructed in the way that: it is zero for any feasible point, and positive for any non-feasible point.

Define the auxiliary function $F(x) = U(x) + P(x)$, where $U(x)$ is the objective function. It was proven by Zangwill [37] that: there exists a positive constant λ_0 such that, if for all $i = 1, 2, ..., m+l$,

$$\lambda_i \geq \lambda_0, \tag{2.6}$$

a minimum of $F(x)$ coincides with a solution of the original convex problem defined in Section 2.3.1 [33].

Outside $s_i(x) = 0, i = 1, 2, ..., m+l$, the gradient of $F(x)$ is well defined, hence a possible gradient procedure to find the minimum of $F(x)$ is

$$\dot{x} = -\nabla U - G^T u, \tag{2.7}$$

where $G = \begin{bmatrix} \nabla s_1 & \nabla s_2 & ... & \nabla s_{m+l} \end{bmatrix}^T$. On surfaces $s_i(x) = 0, \forall i = 1, 2, ..., m+l$, which are referred to as the discontinuity surfaces, the gradient of $F(x)$ is discontinuous, and sliding mode may occur on intersection of discontinuity surfaces.

Assume sliding mode occurs on the intersection of some discontinuity surfaces (this condition is equivalent to the condition given by equation (2.6)), and let X_F^*, defined as the set of all minimum points x_F^* of $F(x)$, be bounded, by using the equivalent control method introduced in Section 2.2 to analyze the system defined by equation (2.7), convergence of this system to minimum point of $F(x)$ can be established in the following sense [33]

$$\lim_{t \to \infty} \min_{x_F^* \in X_F^*} \| x - x_F^* \| = 0.$$

Chapter 3

Multi-domain Traffic Engineering Problem

In the last chapter, we provided a review of the concept of sliding modes and its application to convex optimization. This is the basic tool used to address the multi-domain traffic engineering problem which is studied in this chapter. More precisely, a family of control laws is developed, which supports TE, QoS, and FFR in a multi-domain environment. This family of control laws is optimization-based, distributed by design, and in line with the distributed, two-level routing structure, and hop-by-hop forwarding paradigm of today's Internet. Hence, it is deployable at a global scale.

This chapter is organized as follows. First, a high-level view of the multi-domain control structure is provided. Second, some additional notation, assumptions, and a precise problem statement are given. Next, the optimal control laws and an alternative algorithm are presented. Finally, the simulations of the proposed algorithms are given. The proofs of the main results in this chapter are provided in the Appendix.

3.1 Multi-domain Control Structure

Before presenting the precise mathematical problem formulation of the multi-domain traffic engineering problem and the mathematical details of the proposed family of control laws, in this section, we provide a high-level view of the multi-domain control structure based on the family of control laws.

As shown in Fig. 3.1, the Internet backbone is composed of multiple Internet Service Provider (ISP) domains, known as autonomous systems (AS). Internet routing is performed at two levels, intra-domain and inter-domain. In line with this distributed routing structure, the proposed solution works at two levels, intra-domain and inter-domain, based on three types of control laws - class A, class B, and class C. These

control laws perform fully distributed, per-hop control of class-of-service (CoS)-and-IP-prefix based flow aggregates at the Internet access points (Class A) and domain edge nodes (Class B at the inter-domain level and Class C at the intra-domain level) to enable FFR, QoS, and TE features simultaneously. These features are achieved through CoS-based, multi-next-hop dynamic load balancing and/or rate adaptation among edge nodes in response to congestions/failures. More specifically:

- Class A control laws running at the access/aggregation points perform one-hop control of one or multiple, CoS-aware flows sent to an edge node of an ISP domain.[1]

- At the inter-domain level, the Class B controller sitting at a domain edge node performs single-hop, dynamic load balancing and/or rate adaptation for a CoS-and-IP-prefix-based flow aggregate in response to congestion/failures.

- At the intra-domain level, the Class C controller running at an AS domain edge node performs domain edge-to-edge, dynamic multi-path load balancing and/or rate adaptation for a CoS-and-IP-prefix-based flow aggregate.[2] Since the Class C control law performs edge-to-edge control without having to involve any core nodes for the control (i.e., no need for path pinning), such edge-to-edge, per-hop control allows the flexibility to support both connection-oriented and connectionless services in the domain.

In our solution, the distributed combination of A, B, and C controllers enable scalable and adaptive per-hop control/forwarding across the Internet.

Since this solution allows next-hop edge nodes and related link resource allocation to be determined purely locally, it can work properly under the constraints imposed by the local policies. Moreover, due to its "per-hop" control among edge nodes at the intra-domain level, which can be locally engineered to cater to the specific conditions

[1]If load balancing at an access point is desirable, e.g., through multi-homing, Class A controllers can be pushed towards end-hosts and the access points are treated as edge nodes, running Class B and Class C controllers.

[2]Multi-path can be connection-oriented, e.g., built by MPLS, or connectionless, e.g., based on a set of shortest paths found by an IP routing protocol. In the case of a connectionless IP network, since a given destination network or IP prefix may be reachable through multiple egress edge nodes, the multi-path control may involve multiple egress edge nodes, or point-to-multi-point multi-path.

of the domain without sacrificing the global convergence properties, the proposed solution can address heterogeneity and edge diversity of the Internet domains. For example, customized Class B controllers can be developed for a wireless network for performance enhancement. The implementation of these controllers requires minimum software upgrades in domain edge routers with programmable interfaces as explained in [24].

Here we note that the proposed architecture enables basic QoS, TE, and FFR features in a highly distributed, scalable fashion. The end-to-end service qualities can still be affected by other factors. For instance, since a multi-path in an ISP domain may be created by various mechanisms, such as MPLS and IP routing protocols, the multi-path quality may vary. However, this topic is beyond the scope of this dissertation (which focuses mainly on higher level algorithms), and its detailed study will be the focus of future work.

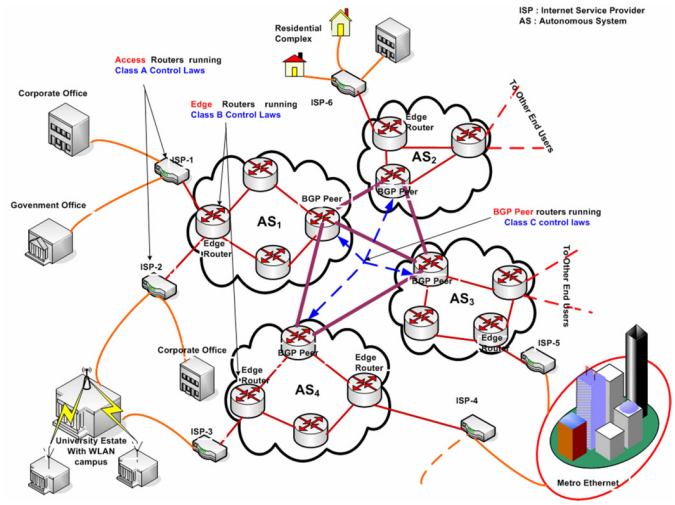


Fig. 3.1. Multi-domain control structure

3.2 Problem Statement

We now provide a precise problem statement of the optimal traffic engineering problem in a multi-domain environment addressed in this chapter. To better explain the proposed approach, we introduce some of the notations in Table 3.1, and use a

four domain internet as given in Fig. 3.2, Fig. 3.3 and Fig. 3.4 to guide the discussion. Throughout this chapter, traffic flows in a multi-domain environment are assumed to be described by a fluid flow model and the only resource considered is link bandwidth. Consider a multi-domain internet with access points attached to various internet domain edge nodes, serving as traffic source and sink nodes. For example, S and D nodes in Fig. 3.2 are access points. Note that each edge node may have multiple access points connected to it (only one is given in Fig. 3.2). Define a flow of given type x_i, $i = 1, 2, ..., \eta$, as a Class-of-Service (CoS)-based flow aggregate between a source access point and a sink access point. The objective is to find the allocation of traffic that leads to the maximization of the sum of individual flow utility functions $U_i(x_i)$

$$\sum_{i=1}^{\eta} U_i(x_i),$$

subject to the network resource constraints and CoS requirements. The functions $U_i(x_i)$, $i = 1, ..., \eta$, are assumed to be differentiable concave functions and increasing in their arguments.

For simplicity of exposition, only two CoSs are considered: Flows of type i, for $i = 1, 2, \ldots, \eta_1$, are assumed, without loss of generality, to be of the Assured Forwarding CoS category with a fixed target rate (AF); On the other hand, flows of type i, for $i = \eta_1 + 1, \eta_1 + 2, \ldots, \eta$, are assumed to be of the Best Effort CoS category (BE). Other CoS categories such as service with minimum rate guarantee, upper bounded rate, and/or service with both minimum and upper bounded rate can also be easily incorporated [31].

The proposed control laws, class A, B and C run at both access points and domain edge nodes. They perform inter-domain and intra-domain traffic controls. At the inter-domain level, the class A running at access points and class B running at domain edge nodes, perform load balancing/rate adaptation among multiple next-hop nodes. At the intra-domain level, the class C running at domain edge nodes perform load balancing/rate adaptation among multiple paths. In what follows, such an edge-to-edge multi-path is called a *virtual link*. Hence, logically, one can view this approach as a hop-by-hop

distributed traffic control scheme among edge nodes interconnected by both non-virtual and virtual links.

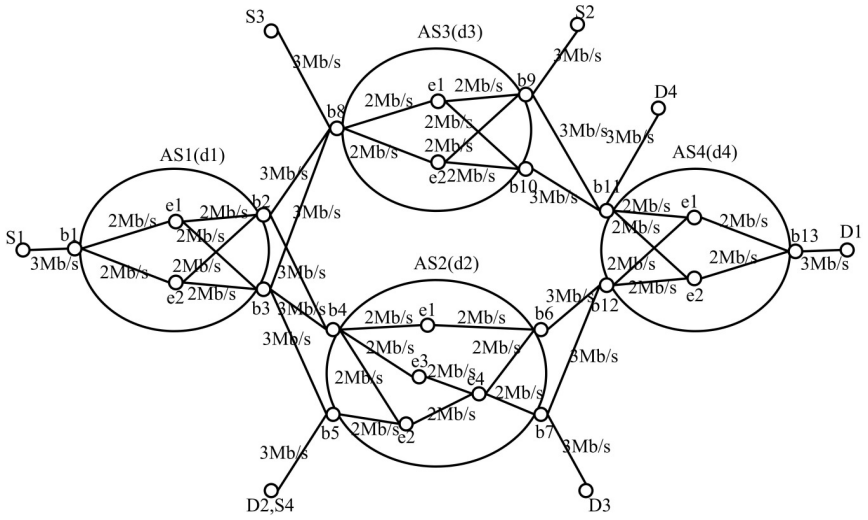


Fig. 3.2. Network topology

Each virtual link l connecting an ingress-egress node pair consists of one or more paths. For example, in Fig. 3.2, we may define three virtual uni-directional virtual links $l4$, $l5$, and $l6$, which one may see in Fig. 3.4, corresponding to the following multi-paths, respectively: the virtual link $l4$ has one path: $b5-e2-e4-b6$; the virtual link $l5$ has two paths: $b4-e3-e4-b7$, and $b4-e2-e4-b7$; and the virtual link $l6$ has three paths: $b4-e1-b6$, $b4-e3-e4-b6$, and $b4-e2-e4-b6$.

Let $\boldsymbol{x}$ be an $n \times 1$ vector consisting of (see Table 3.1)

$$
\begin{aligned}
& x_i && \text{for each } i, \\
& x^{out}_{i,b,l} && \text{for each } b,\ i \in I_b, \text{ and } l \in \mathcal{L}_b, \\
& x^{out}_{i,b,l(j)} && \text{for each } b,\ i \in I_b, \text{ virtual link } l \in \mathcal{L}_b \text{ and each } l(j), \\
& x^{in}_{i,b,l} && \text{for each } b,\ i \in I_b, \text{ and } l \in \mathcal{L}_b, \\
& x^{in}_{i,b,l(j)} && \text{for each } b,\ i \in I_b, \text{ virtual link } l \in \mathcal{L}_b \text{ and each } l(j).
\end{aligned}
$$

Table 3.1. Notation

domain d	domain d
set $\mathcal{D}$	set of all domains d
node b	node at the inter-domain level
set B	set of all nodes b
link l	link (non-virtual or virtual) at the inter-domain level
non-virtual link l	link connecting an egress-ingress node pair example: $l1$ connecting $b2$ and $b4$
virtual link l	link connecting an ingress-egress node pair, and consisting of one of more paths example: $l6$ connecting $b4$ and $b6$
set $\mathcal{L}_b$	set of all links l connected to node b
path $l(j)$	path j of virtual link l
node e	node at the intra-domain level except edge nodes
physical link g	physical link at the intra-domain level interconnecting $b-e$ and $e-e$ node pairs
set $\mathcal{G}^d$	set of all physical links g in domain d
set $\mathcal{G}_{i,l(j)}$	set of all physical links g taken by $x^{out}_{i,b,l(j)}$
type i	type i of flows
set I_b	set of all types i of flows visiting node b
set $\mathcal{L}_{b,i}$	set of all outgoing links l the flows of type i use
node $\varsigma(b,l)$	next hop from node b through link l
x_i	data rate of flows of type i
$x^{out}_{i,b,l}(x^{in}_{i,b,l})$	data rate of flows of type i arriving at (departing from) node b through link l
$x^{out}_{i,b,l(j)}(x^{in}_{i,b,l(j)})$	data rate of flows of type i arriving at (departing from) node b through path $l(j)$ of virtual link l

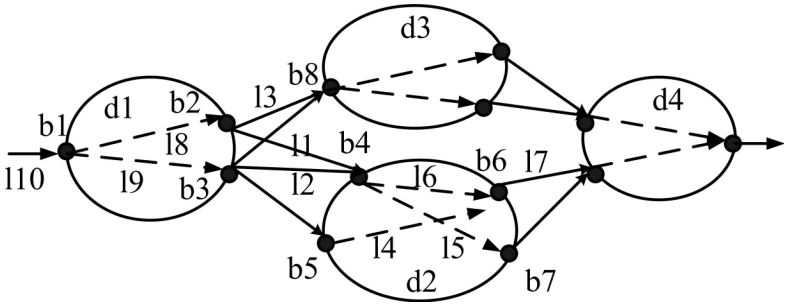


Fig. 3.3. Inter-domain

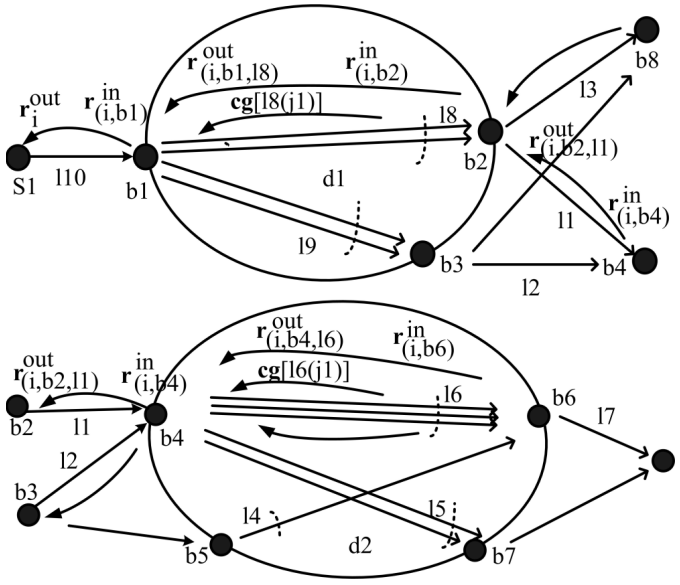


Fig. 3.4. Example of Class A,B and C

Furthermore, we assume that all the data rates are bounded; i.e., there exists an $\iota \in \boldsymbol{R}$ such that $\boldsymbol{x}$ always belongs to the set

$$\mathcal{X} \doteq \left\{ \boldsymbol{x} \in \boldsymbol{R}^n : x_i, x^{out}_{i,b,l}, x^{out}_{i,b,l(j)}, x^{in}_{i,b,l}, x^{in}_{i,b,l(j)} \leq \iota; \quad \text{for each } i, b, l, \text{ and } l(j) \right\}.$$

Moreover, assume that at the optimal traffic allocation each congested link has at least one non-binding class of service or a BE flow with a nonzero data rate, and that the components of the gradient of the utility function ∇U, are bounded.

Given the assumptions and requirements above, the problem of optimal traffic allocation can be formulated as the following optimization problem

$$\max_{\boldsymbol{x}} U(\boldsymbol{x}) = \sum_{i=1}^{\eta} U_i(x_i),$$

subject to: inter-domain level link capacity constraints: for each non-virtual link l, and each $b \in B$

$$\sum_{i,l \in \mathcal{L}_b} x^{in}_{i,b,l} + x^{out}_{i,b,l} \leq c_l,$$

intra-domain level link capacity constraints: for each domain $d \in \mathcal{D}$, and each physical link $\mathsf{g} \in \mathcal{G}^d$

$$\sum_{i,l(j):\mathsf{g} \in \mathcal{G}_{i,l(j)}} x^{out}_{i,b,l(j)} + x^{in}_{i,b,l(j)} \leq c_{\mathsf{g}},$$

flow conservation constraints: for each node $b \in B$, and each $i \in I_b$

$$\sum_{l \in \mathcal{L}_b} x^{out}_{i,b,l} - \sum_{l \in \mathcal{L}_b} x^{in}_{i,b,l} = 0,$$

where

$$x^{out}_{i,b,l} = \sum_{l(j)} x^{out}_{i,b,l(j)}, \qquad \text{if } l \text{ is virtual link,}$$

$$x^{in}_{i,b,l} = \sum_{l(j)} x^{in}_{i,b,l(j)}, \qquad \text{if } l \text{ is virtual link,}$$

flow conservation constraints for non-virtual links: for each node $b \in B$, each non-virtual link $l \in \mathcal{L}_b$, and each type $i \in I_b$

$$x^{out}_{i,b,l} - x^{in}_{i,\varsigma(b,l),l} = 0,$$

flow conservation constraints for virtual links: for each node $b \in B$, each virtual link $l \in \mathcal{L}_b$, each path $l(j)$, and each type $i \in I_b$

$$x^{out}_{i,b,l(j)} - x^{in}_{i,\varsigma(b,l),l(j)} = 0,$$

Assured Forwarding (AF) requirements

$$x_i - \Lambda_i = 0, \qquad i = 1, 2, \ldots, \eta_1,$$

and non-negativity of all the data rates, $\boldsymbol{x} \geq 0$. Note that the decision variable $\boldsymbol{x}$ contains x_i, $x^{out}_{i,b,l}$, and $x^{out}_{i,b,l(j)}$, but the objective function only depends on x_i.

3.3 Optimization-based Distributed Control Laws

This section presents a family of adaptation control laws, class A, B and C, that converge to the optimal solution of the optimization problem formulated in Section 3.2. Moreover, this family of control laws only needs binary feedback information from the network, which can be inferred from the edge nodes without any assistance from the core nodes. Our approach has its basis in nonlinear control theory. More precisely, we use results from the sliding mode theory [33].

3.3.1 Additional Notation

Let function $h_i(x_i)$ be defined as

$$h_i(x_i) = (1 - e^{-\partial U/\partial x_i}).$$

A non-virtual link l is said to be congested if the aggregated data rates using the link reaches its capacity c_l. If l is a non-virtual link, define $cg(l)$

$$cg(l) = \begin{cases} 1, & \text{if link } l \text{ is congested,} \\ 0, & \text{otherwise.} \end{cases}$$

If l is a virtual link, it contains multipaths $l(j)$. A path $l(j)$ contained in virtual link l is said to be congested if any physical link $\mathtt{g}$ of path $l(j)$ is congested. We define $cg[l(j)]$

$$cg[l(j)] = \begin{cases} 1, & \text{if path } l(j) \text{ is congested,} \\ 0, & \text{otherwise.} \end{cases}$$

Moreover, define $\overline{cg}(l)$ and $\overline{cg}[l(j)]$ as the logical negations of $cg(l)$ and $cg[l(j)]$; i.e.,

$$\overline{cg}(l) = \begin{cases} 1, & \text{if } \ cg(l) = 0, \\ 0, & \text{if } \ cg(l) = 1, \end{cases}$$

$$\overline{cg}[l(j)] = \begin{cases} 1, & \text{if } \ cg[l(j)] = 0, \\ 0, & \text{if } \ cg[l(j)] = 1. \end{cases}$$

The proposed control laws and examples of Class A, B and C are given in the following sections. Note that in this family of control laws, $w_i(t, x_i, cg(l), r_i^{out})$, $w_{i,b}(t, x_{i,b,l}^{out}, cg(l), r_{i,b}^{in}, r_{i,b,l}^{out})$, $w_{i,b}(t, x_{i,b,l(j)}^{out}, cg[l(j)], r_{i,b}^{in}, r_{i,b,l}^{out})$, r_{min}^{CoS}, r_{max}^{CoS}, r_{min}, and r_{max} are design parameters that have to be determined to achieve convergence and provide an acceptable transient behavior.

3.3.2 Class A Control Laws

Class A control laws run at access points. At each access point b_{si}, the class A control laws are

$$\dot{x}_i = w_i(t, x_i, cg(l), r_i^{out})\Big[h_i(x_i) - (1 - \overline{cg}(l) r_i r_i^{out})\Big],$$

where

$$\text{if } i \text{ is AS flow, } \; r_i = \begin{cases} r_{min}^{CoS} < 1, & \text{if } x_i > \Lambda_i, \\ r_{max}^{CoS} > 1, & \text{if } x_i < \Lambda_i, \end{cases} \tag{3.1}$$

$$\text{if } i \text{ is BE flow, } \; r_i = 1,$$

and

$$r_i^{out} = \begin{cases} r_{min} < 1, & \text{if } x_i > \displaystyle\sum_{\tilde{l} \in \mathcal{L}_{\varsigma(b_{si}, l)}} x_{i,\varsigma(b_{si},l),\tilde{l}}^{out}, \\ r_{max} > 1, & \text{if } x_i < \displaystyle\sum_{\tilde{l} \in \mathcal{L}_{\varsigma(b_{si}, l)}} x_{i,\varsigma(b_{si},l),\tilde{l}}^{out}. \end{cases} \tag{3.2}$$

Remark: The term r_i^{out} is used to enforce the flow conservation constraint at $\varsigma(b_{si}, l)$ (the downstream node of access point b_{si}), if the total data rates arriving at $\varsigma(b_{si}, l)$, which is exactly x_i, is greater (less) than the total data rates departing from $\varsigma(b_{si}, l)$, then r_i^{out} will be assigned a value strictly less than 1 (strictly greater than 1). The term r_i is used to enforce the CoS requirement $x_i - \Lambda_i = 0$. Here is an example of the Class A control laws. The data rate running through the non-virtual link $l10$ in Fig. 3.4 is controlled by the following class A control laws

$$\dot{x}_1 = w_1(t, x_1, cg(l10), r_1^{out})\Big[h_1(x_1) - (1 - \overline{cg}(l10) r_1 r_1^{out})\Big],$$

where

$$r_1 = \begin{cases} r_{min}^{CoS}, & \text{if } x_1 > \Lambda_1, \\ r_{max}^{CoS}, & \text{if } x_1 < \Lambda_1, \end{cases}$$

and

$$r_1^{out} = \begin{cases} r_{min}, & \text{if } x_1 > x_{1,b1,l8(j1)}^{out} + x_{1,b1,l8(j2)}^{out} + x_{1,b1,l9(j1)}^{out} + x_{1,b1,l9(j2)}^{out}, \\ r_{max}, & \text{if } x_1 < x_{1,b1,l8(j1)}^{out} + x_{1,b1,l8(j2)}^{out} + x_{1,b1,l9(j1)}^{out} + x_{1,b1,l9(j2)}^{out}. \end{cases}$$

3.3.3 Class B Control Laws

Class B control laws running at domain edge egress nodes perform load balancing/rate adaptation among multiple next-hop nodes. At each domain edge egress node b, for each non-virtual link $l \in \mathcal{L}_b$, and each type $i \in I_b$, the class B control laws are

$$\dot{x}_{i,b,l}^{out} = w_{i,b}(t, x_{i,b,l}^{out}, cg(l), r_{i,b}^{in}, r_{i,b,l}^{out})\Big[-1 + \overline{cg}(l) r_{i,b}^{in} r_{i,b,l}^{out}\Big],$$

where

$$r_{i,b}^{in} = \begin{cases} r_{max} > 1, & \text{if } \sum_{l \in \mathcal{L}_b} x_{i,b,l}^{in} > \sum_{l \in \mathcal{L}_b} x_{i,b,l}^{out}, \\ r_{min} < 1, & \text{if } \sum_{l \in \mathcal{L}_b} x_{i,b,l}^{in} < \sum_{l \in \mathcal{L}_b} x_{i,b,l}^{out}, \end{cases} \tag{3.3}$$

and

$$r_{i,b,l}^{out} = \begin{cases} r_{min} < 1, & \text{if } \sum_{\tilde{l} \in \mathcal{L}_{\varsigma(b,l)}} x_{i,\varsigma(b,l),\tilde{l}}^{in} > \sum_{\tilde{l} \in \mathcal{L}_{\varsigma(b,l)}} x_{i,\varsigma(b,l),\tilde{l}}^{out}, \\ r_{max} > 1, & \text{if } \sum_{\tilde{l} \in \mathcal{L}_{\varsigma(b,l)}} x_{i,\varsigma(b,l),\tilde{l}}^{in} < \sum_{\tilde{l} \in \mathcal{L}_{\varsigma(b,l)}} x_{i,\varsigma(b,l),\tilde{l}}^{out}. \end{cases} \tag{3.4}$$

Remark: The term $r_{i,b}^{in}$ is used to enforce the flow conservation constraint at node b, if the total data rates arriving at b is greater (less) than the total data rates departing

from b, then $r^{in}_{i,b}$ will be assigned a value strictly larger than 1 (strictly less than 1). Similarly, the term $r^{out}_{i,b,l}$ is used to enforce the flow conservation constraint at node $\varsigma(b,l)$ (downstream node of node b through link l). Here is an example of the Class B control laws. The data rate running through the non-virtual link $l1$ in Fig. 3.4 is controlled by the following class B control laws

$$\dot{x}^{out}_{i,b2,l1} = w_{i,b2}(t, x^{out}_{i,b2,l1}, cg(l1), r^{in}_{i,b2}, r^{out}_{i,b2,l1})\Big[-1 + \overline{cg}(l1) r^{in}_{i,b2} r^{out}_{i,b2,l1}\Big],$$

where

$$r^{in}_{i,b2} = \begin{cases} r_{max}, \text{ if } x^{in}_{i,b2,l8(j1)} + x^{in}_{i,b2,l8(j2)} > x^{out}_{i,b2,l1} + x^{out}_{i,b2,l3}, \\ r_{min}, \text{ if } x^{in}_{i,b2,l8(j1)} + x^{in}_{i,b2,l8(j2)} < x^{out}_{i,b2,l1} + x^{out}_{i,b2,l3}, \end{cases}$$

and

$$r^{out}_{i,b2,l1} = \begin{cases} r_{min}, \text{ if } x^{in}_{i,b4,l1} + x^{in}_{i,b4,l2} > x^{out}_{i,b4,l5(j1)} + x^{out}_{i,b4,l5(j2)} \\ \qquad + x^{out}_{i,b4,l6(j1)} + x^{out}_{i,b4,l6(j2)} + x^{out}_{i,b4,l6(j3)}, \\ r_{max}, \text{ if } x^{in}_{i,b4,l1} + x^{in}_{i,b4,l2} < x^{out}_{i,b4,l5(j1)} + x^{out}_{i,b4,l5(j2)} \\ \qquad + x^{out}_{i,b4,l6(j1)} + x^{out}_{i,b4,l6(j2)} + x^{out}_{i,b4,l6(j3)}. \end{cases}$$

3.3.4 Class C Control Laws

Class C control laws running at domain edge ingress nodes perform load balancing/rate adaptation among multiple paths. At each domain edge ingress node b, for each virtual link $l \in \mathcal{L}_b$, each $l(j)$, and each type $i \in I_b$, the class C control laws are

$$\dot{x}^{out}_{i,b,l(j)} = w_{i,b}(t, x^{out}_{i,b,l(j)}, cg[l(j)], r^{in}_{i,b}, r^{out}_{i,b,l})\Big[-1 + \overline{cg}[l(j)] r^{in}_{i,b} r^{out}_{i,b,l}\Big],$$

where $r^{in}_{i,b}$ and $r^{out}_{i,b,l}$ are defined by (3.3) and (3.4). Here is an example of the Class C control laws. The three data rates running through the virtual link $l6$ in Fig. 3.4 are

controlled by the following class C control laws

$$\dot{x}^{out}_{i,b4,l6(j1)} = w_{i,b4}(t, x^{out}_{i,b4,l6(j1)}, cg[l6(j1)], r^{in}_{i,b4}, r^{out}_{i,b4,l6}) \Big[-1 + \overline{cg}[l6(j1)] r^{in}_{i,b4} r^{out}_{i,b4,l6} \Big],$$

$$\dot{x}^{out}_{i,b4,l6(j2)} = w_{i,b4}(t, x^{out}_{i,b4,l6(j2)}, cg[l6(j2)], r^{in}_{i,b4}, r^{out}_{i,b4,l6}) \Big[-1 + \overline{cg}[l6(j2)] r^{in}_{i,b4} r^{out}_{i,b4,l6} \Big],$$

$$\dot{x}^{out}_{i,b4,l6(j3)} = w_{i,b4}(t, x^{out}_{i,b4,l6(j3)}, cg[l6(j3)], r^{in}_{i,b4}, r^{out}_{i,b4,l6}) \Big[-1 + \overline{cg}[l6(j3)] r^{in}_{i,b4} r^{out}_{i,b4,l6} \Big],$$

where

$$r^{in}_{i,b4} = \begin{cases} r_{min}, \text{ if } x^{in}_{i,b4,l1} + x^{in}_{i,b4,l2} > x^{out}_{i,b4,l5(j1)} + x^{out}_{i,b4,l5(j2)} \\ \qquad + x^{out}_{i,b4,l6(j1)} + x^{out}_{i,b4,l6(j2)} + x^{out}_{i,b4,l6(j3)}, \\ r_{max}, \text{ if } x^{in}_{i,b4,l1} + x^{in}_{i,b4,l2} < x^{out}_{i,b4,l5(j1)} + x^{out}_{i,b4,l5(j2)} \\ \qquad + x^{out}_{i,b4,l6(j1)} + x^{out}_{i,b4,l6(j2)} + x^{out}_{i,b4,l6(j3)}, \end{cases}$$

and

$$r^{out}_{i,b4,l6} = \begin{cases} r_{min}, \text{ if } x^{in}_{i,b6,l6(j1)} + x^{in}_{i,b6,l6(j2)} + x^{out}_{i,b4,l6(j3)} + x^{in}_{i,b6,l4(j1)} > x^{out}_{i,b6,l7}, \\ r_{max}, \text{ if } x^{in}_{i,b6,l6(j1)} + x^{in}_{i,b6,l6(j2)} + x^{out}_{i,b4,l6(j3)} + x^{in}_{i,b6,l4(j1)} < x^{out}_{i,b6,l7}. \end{cases}$$

Given the control laws above, the data rates are then forced to be greater than or equal to zero. More precisely, if any of the data rates above is zero, then the corresponding derivative is taken as the maximum between zero and the expression given above.

Note that the above control laws are indeed distributed in the sense that control laws controlling different flows are independent of each other. The coupling effect among different flows is reflected through $cg(l)$, $cg[l(j)]$, $r^{in}_{i,b}$, and $r^{out}_{i,b,l}$ only.

Note that in this family of control laws, $w_i(t, x_i, cg(l), r^{out}_i)$, $w_{i,b}(t, x^{out}_{i,b,l}, cg(l), r^{in}_{i,b}, r^{out}_{i,b,l})$, $w_{i,b}(t, x^{out}_{i,b,l(j)}, cg[l(j)], r^{in}_{i,b}, r^{out}_{i,b,l})$, r^{CoS}_{min}, r^{CoS}_{max}, r_{min}, and r_{max}

are design parameters that have to be determined to achieve convergence and provide an acceptable transient behavior.

The following theorem states that the control laws converge to optimal traffic allocation.

THEOREM 3.1. *Let $\zeta > 0$ be a given (arbitrarily small) constant. Also, let $w_i(t, x_i, cg(l), r_i^{out})$, $w_{i,b}(t, x_{i,b,l}^{out}, cg(l), r_{i,b}^{in}, r_{i,b,l}^{out})$, and $w_{i,b}(t, x_{i,b,l(j)}^{out}, cg[l(j)], r_{i,b}^{in}, r_{i,b,l}^{out})$ be scalar functions continuous in t, satisfying*

$$w_i(t, x_i, cg(l), r_i^{out}), w_{i,b}(t, x_{i,b,l}^{out}, cg(l), r_{i,b}^{in}, r_{i,b,l}^{out}), w_{i,b}(t, x_{i,b,l(j)}^{out}, cg[l(j)], r_{i,b}^{in}, r_{i,b,l}^{out}) > \zeta,$$

for all $t > 0$. Furthermore, let

$$0 < r_{min} < r_{lower} < 1 < r_{upper} < r_{max},$$
$$0 < r_{min}^{CoS} < r_{lower}^{CoS} < 1 < r_{upper}^{CoS} < r_{max}^{CoS},$$

where

$$r_{lower} = e^{-Cg\alpha^*}, \quad r_{upper} = e^{Cg\alpha^*}, \quad r_{lower}^{CoS} = e^{-\alpha^*}, \quad r_{upper}^{CoS} = e^{\alpha^*}, \quad \alpha^* = \max_i |\frac{dU_i}{dx_i}|_{x_i=0},$$

and Cg is the maximum possible number of congested links in path $l(j)$ taken by $x_{i,b,l(j)}^{out}$; i.e., the path length. Then the control law presented above converges to optimal traffic allocation.

The proof of Theorem 3.1 is given in Appendix A.2. Finally, note that the family of control laws presented in this chapter reduces to the the family of control laws found in [25] when the control laws at the access points run end-to-end. Moreover, it reduces to the family of control laws found in [24] when a virtual link degenerates to a physical link.

3.4 Percentage Adaptation

Although distributed, the family of control laws proposed in Section 3.3 requires that each edge node keeps track of and runs a separate control law for each and every flow x_i that traverses the node. This is not only un-scalable but also infeasible because a domain edge node may not be able to identify individual flows running between access points. To address this issue, the family of control laws is simplified to allow the control of flows destined to the same destination network as a whole using one control law. This control law controls the *percentage* of rates sent to different paths/next-hops and there is no need to measure and control the flow rates.

Define $p_{i,b,l}$ and $p_{i,b,l(j)}$ as the percentage of incoming traffic of type i at node b that is routed along each available outgoing non-virtual link l and each path j of virtual link l.

$$x^{out}_{i,b,l} = p_{i,b,l} \sum_{\substack{\tilde{l}\in\mathcal{L}_b \\ \tilde{l}\notin\mathcal{L}_{b,i}}} x^{in}_{i,b,\tilde{l}}, \quad b \in B, l \in \mathcal{L}_{b,i}, l \text{ is non-virtual link},$$

$$x^{out}_{i,b,l(j)} = p_{i,b,l(j)} \sum_{\substack{\tilde{l}\in\mathcal{L}_b \\ \tilde{l}\notin\mathcal{L}_{b,i}}} x^{in}_{i,b,\tilde{l}}, \quad b \in B, l \in \mathcal{L}_{b,i}, l \text{ is virtual link}.$$

The following percentage adaptation laws are proposed for each edge node:

• if l is a non-virtual link, the class B control laws are

$$\dot{p}_{i,b,l} = w_{i,b}(t, x^{out}_{i,b,l}, cg(l), r^{in}_{i,b}, r^{out}_{i}) \Big(\dot{x}^{out}_{i,b,l} \sum_{\tilde{l}\in\mathcal{L}_{b,i};\tilde{l}\neq l} p_{i,b,\tilde{l}} - p_{i,b,l} \sum_{\tilde{l}\in\mathcal{L}_{b,i};\tilde{l}\neq l} \dot{x}^{out}_{i,b,\tilde{l}} \Big),$$

• if l is a virtual link, the class C control laws are

$$\dot{p}_{i,b,l(j)} = w_{i,b}(t, x^{out}_{i,b,l(j)}, cg[l(j)], r^{in}_{i,b}, r^{out}_{i}) \Big(\dot{x}^{out}_{i,b,l(j)} (\sum_{\tilde{l}\in\mathcal{L}_{b,i}} p_{i,b,\tilde{l}} - p_{i,b,l(j)}) - p_{i,b,l(j)} (\sum_{\tilde{l}\in\mathcal{L}_{b,i}} \dot{x}^{out}_{i,b,\tilde{l}} - \dot{x}^{out}_{i,b,l(j)}) \Big),$$

where

$$x^{out}_{i,b,l} = \sum_{l(j)} x^{out}_{i,b,l(j)}, \qquad \text{if } l \text{ is virtual link,}$$

$$p_{i,b,l} = \sum_{l(j)} p_{i,b,l(j)}, \qquad \text{if } l \text{ is virtual link,}$$

$$\dot{x}^{out}_{i,b,l} = \left[-1 + \overline{cg}(l) r^{in}_{i,b} r^{out}_{i,b,l}\right], \quad l \text{ is non-virtual link,}$$

$$\dot{x}^{out}_{i,b,l(j)} = [-1 + \overline{cg}[l(j)] r^{in}_{i,b} r^{out}_{i,b,l}], \quad l \text{ is virtual link,}$$

$\overline{cg}(l)$ and $\overline{cg}[l(j)]$ are defined in Section 3.2, and $r^{in}_{i,b}$, $r^{out}_{i,b,l}$ are given by (3.3) and (3.4) respectively. These laws are derived directly from the control laws for the data rates presented in Section 3.3.

Remark: Note that the laws above do not require the measurement of the data rates if the values of $r^{in}_{i,b}$ and $r^{out}_{i,b,l}$ are available by some other means. The practical computation of $r^{in}_{i,b}$ and $r^{out}_{i,b,l}$ is discussed in Section 3.4.1.

The following theorem states that under some conditions these laws are indeed optimal.

THEOREM 3.2. *Assume the conditions in Theorem 3.1 hold. Moreover, assume that the total incoming traffic is always strictly positive; i.e., there exists* $\epsilon > 0$ *such that, for each node* $b \in B$*, each type* $i \in I_b$ *and all* $t \geq 0$

$$\sum_{\substack{\tilde{l} \in \mathcal{L}_b \\ \tilde{l} \notin \mathcal{L}_{b,i}}} x^{in}_{i,b,\tilde{l}}(t) > \epsilon. \tag{3.5}$$

Then the percentage adaptation laws converge to optimal traffic allocation.

The proof of Theorem 3.2 is given in Appendix A.3.

Remark: The percentage control laws perform CoS-based, multi-path-multi-next-hop dynamic load balancing by controlling the percentage of rates sent to different paths/next-hops. From practical point of view, if the total data rate of flow of type i

arriving at node b is less than ϵ (where ϵ is very small compared to link capacities), then one can "freeze" the adaptation of the percentage allocation and keep it fixed until the data rate arriving at the node is larger than ϵ. Since ϵ is "very small", the traffic allocation obtained by such a procedure is very close to the optimal one. Additionally, as it is mentioned in Section 3.4.1, when one uses the percentage adaptation laws, the flow conservation constraints are implicitly satisfied. The quantities $r_{i,b}^{in}$ and $r_{i,b,l}^{out}$ are, in this case, just a useful tool for conveying congestion information. Moreover, the comments in Section 3.4.2 show that the computational burden can be further reduced, and the condition that the total data rate is strictly lower bounded away from zero can be further relaxed.

3.4.1 Practical Computation of $r_{i,b}^{in}$ and $r_{i,b,l}^{out}$

Although optimal, the laws in Section 3.3 require access to data rate information for the computation of $r_{i,b}^{in}$ and $r_{i,b,l}^{out}$. Hence, an alternative (empirical) way to compute this information is presented.

Note that when implementing percentage adaptation, the aggregate incoming data rate is always larger than or equal to the aggregate outgoing rate; i.e.,

$$\sum_{l \in \mathcal{L}_b} x_{i,b,l}^{in} \geq \sum_{l \in \mathcal{L}_{b,i}} x_{i,b,l}^{out}. \tag{3.6}$$

Hence, it is not necessary for $r_{i,b}^{in}$ in (3.3) to assume the value r_{min}; i.e., in this case only $r_{i,b}^{in} = r_{max}$ or 1 is needed. Similarly, only $r_{i,b,l}^{out} = r_{min}$ or 1 is needed. Note also only when there is some type of congestion, either in the connected links or further downstream, (3.6) is a strict inequality, hence $r_{i,b}^{in} = r_{max}$. When there is available network resource, (3.6) is equality, $r_{i,b}^{in} = 1$. This prompts the following computation for $r_{i,b}^{in}$:

- if $\varsigma(b,l)$ is the destination of flow i, then there is no $r^{out}_{i,b,l}$, and

$$r^{in}_{i,b} = \begin{cases} 1, & \text{if } l \text{ is a non-virtual link, and } cg(l) = 0, \\ 1, & \text{if } l \text{ is a virtual link, and any path } l(j),\ cg[l(j)] = 0, \\ r_{max} > 1, & \text{otherwise.} \end{cases}$$

- if $\varsigma(b,l)$ is not the destination of flow i, then

$$r^{in}_{i,b} = \begin{cases} 1, & \text{if any } l \in \mathcal{L}_{b,i} \text{ is a non-virtual link, and } cg(l) = 0,\ r^{out}_{b,l} = 1, \\ 1, & \text{if any } l \in \mathcal{L}_{b,i} \text{ is a virtual link, and any path } l(j),\ cg[l(j)] = 0,\ r^{out}_{b,l} = 1, \\ r_{max} > 1, & \text{otherwise.} \end{cases}$$

The quantity $r^{out}_{i,b,l}$ is received from the downstream node $\varsigma(b,l)$, and is given by

$$r^{out}_{i,b,l} = \begin{cases} r_{min}, & \text{if } r^{in}_{i,\varsigma(b,l)} = r_{max}, \\ 1, & \text{if } r^{in}_{i,\varsigma(b,l)} = 1, \end{cases}$$

for each link $l \in \mathcal{L}_{b,i}$. With this expression, $r^{in}_{i,b}$ will only be r_{max} if all the available paths are congested and 1 otherwise.

In general, if any optimal data rate is zero, the percentages obtained by these means might exceed one or become negative, although they will always add up to one. This issue is addressed by means of a normalization procedure that is explained in Section 3.5, along with the problem of discretization of the continuous-time adaptation laws. These optimal adaptation laws, together with the empirical expression for the computation of $r^{in}_{i,b}$ and $r^{out}_{i,b,l}$, lead to a tractable control law that will approximately mimic the behavior of the optimal ones, and keep the optimality.

3.4.2 Destination-based Aggregate Flow Control

By carefully looking at the way $r^{in}_{i,b}$ and $r^{out}_{i,b,l}$ are computed in the previous section, it can be seen that the computational burden at each node can be further reduced. Indeed, consider a node b and two types of flows $i1$ and $i2$ arriving at b which match the same destination network address or IP prefix. Now, the formulas proposed in the previous section imply that

$$r^{in}_{i1,b} = r^{in}_{i2,b},$$

$$r^{out}_{i1,b,l} = r^{out}_{i2,b,l}, \quad \text{if } l \text{ is non-virtual link.}$$

Hence, if the initial conditions are the same, then

$$p_{i1,b,l}(t) = p_{i2,b,l}(t), \quad \text{if } l \text{ is non-virtual link,}$$

$$p_{i1,b,l(j)}(t) = p_{i2,b,l(j)}(t), \quad \text{if } l \text{ is virtual link,}$$

and, as a consequence, there is no need to independently adapt the percentages for these two flow types.

Therefore, the same percentages can be used for all the flows matching the same route prefix. More precisely, given a node b let the set of next hops for flows with destination D, $\mathcal{L}^D_b$, replace the set of next hops per type $\mathcal{L}_{b,i}$. Similarly, let per destination percentages $p^D_{b,l}$ take the place of per type percentages $p_{i,b,l}$. Then, node b needs only to adapt per destination percentages $p^D_{b,l}$, $l \in \mathcal{L}^D_b$; i.e., per type information *no longer* needs to be maintained. Note that per destination or per route prefix information is readily available in the routing table. Moreover, condition (3.5) can be relaxed to the condition that the total data rate with destination D is strictly lower bounded away from zero. An approach to efficiently implement such control laws in a high-speed router is presented in [24].

3.4.3 Robustness with respect to Failures

A salient feature of the proposed control laws is that, once implemented, the resulting network will be robust with respect to link/node failures. In other words, after a small modification discussed below, the control laws will automatically reroute traffic away from the nodes/links that fail. The distributed nature of the laws allows for traffic rerouting to be done by the nodes adjacent to the failure and without any change of the control law parameters.

Indeed, this can be accomplished by the following procedure: Upon detection of a link or node failure in an adjacent node or a connected link, each node performs an update of the set $\mathcal{L}_b^D$; i.e., it updates the set of available next-hops for flows with destination D. Once the update is performed, given the distributed nature of the laws, it can be seen that the control laws are optimal for the "new" network configuration and will provide the desired traffic allocation; i.e., traffic is rerouted away from the failed components and a new optimal steady state allocation will be achieved [24]. Since the rerouting is done locally, these control laws enable optimal FFR features.

Finally, we note that the proposed approach can address tussles between domains for the following reason. Since the decision on the selection of next-hop nodes at the inter-domain level is purely a local matter, it can be selected based on policies or service level agreement between domains. The proposed control framework can also be extended (the work is underway) to allow destination-based, CoS-aware policy control between domains. This extension will allow the current framework to be seamlessly integrated with a DiffServ-like per-hop QoS architecture to enable sophisticated QoS, TE, and FFR.

3.5 Numerical Examples

In this section, simulation results are presented which exemplify the behavior of the algorithms proposed.

3.5.1 Implementation Considerations

The implementation of the algorithms in a real network has to be performed in discrete time. In the simulations to follow, this is accomplished by obtaining a difference equation using the forward rule approximation as follows

$$\boldsymbol{x}^d[(k+1)t_d] = \boldsymbol{x}^d[kt_d] + t_d\dot{\boldsymbol{x}}^d[(k+1)t_d],$$

for $k = 0, 1, 2, ...$, where t_d is the integration step.

The following proposition establishes under what conditions this approach will lead to a successful realization of the discrete-time version of the proposed algorithms.

PROPOSITION 1. *Let $\boldsymbol{x}(t)$ be the trajectory obtained using the algorithms in Section 3.3 and 3.4, and let $\boldsymbol{x}^d(t)$ be the corresponding discrete-time trajectory obtained using the discretization algorithm above. Given any time interval $[t_0, t_1]$ and constant $\varepsilon > 0$, there exists a $\xi > 0$ such that if $t_d w_{i,b} < \xi$, $t_d w_i < \xi$, for all $t > 0$ and $\boldsymbol{x} \in \mathcal{X}$, then $\|\boldsymbol{x}(t) - \boldsymbol{x}^d(t)\| < \varepsilon$, for all $t \in [t_0, t_1]$.*

3.5.2 Simulation Setup

Based on the above Proposition, the discretization for the percentage adaptation laws is performed following way. Taking a non-virtual link as an example:

$$\hat{p}^d_{i,b,l}\big[(k+1)t_d\big] = \hat{p}^d_{i,b,l}[kt_d] + t_d\frac{d\hat{p}_{i,b,l}(kt_d)}{dt}.$$

Moreover, the following normalization forces all the percentages to add up to one and to lie in $[0, 1]$. Taking a virtual link as an example:

$$\hat{p}_{i,b,l(j)} = \max\big\{\min\{\hat{p}^d_{i,b,l(j)}, 1\}, 0\big\}, \quad \hat{p}_{i,b,l(j)} = \frac{\hat{p}_{i,b,l(j)}}{\sum\limits_{l\in\mathcal{L}_b}\hat{p}^d_{i,b,l}}.$$

The above discretization and the existence of congestion feedback delays result in a non-ideal behavior of the system. Due to this fact, an adaptation reduction scheme [20] is used to mitigate this phenomenon.

The network topology used in the simulation is given in Fig. 3.2, where all the links' bandwidths, as well as source and destination nodes are shown. The inter-domain delay is set as $10ms$, and intra-domain delay $5ms$. We assume that each flow has multiple paths, shown in Fig. 3.5. Flows 1 and 2 are of AF type $\Lambda_1 = \Lambda_2 = 1Mb/s$, and flows 3 and 4 are of BE type. The following utility function is assumed

$$U(\boldsymbol{x}) = \sum_i log(x_i + 0.5), \quad i = 1, 2, ..., 4.$$

As a first step, the control laws were tested in almost ideal conditions by setting network delays to 0 and sampling time to $1ms$. The design parameters are chosen as $r_{min} = 1/10$, $r_{max} = 10$, $r_{min}^{CoS} = 1/15$, $r_{max}^{CoS} = 15$, and $w_i(t) = w_{i,b}(t) = 5$. It can be seen in the left column of Fig. 3.6 that, under these conditions, the network converges to an optimal traffic allocation. Furthermore, the network resources are fully utilized, and the AS requirements are satisfied.

The right column of Fig. 3.6 shows the network behavior with the delays and sampling time $1ms$. The design parameters are the same as above. Except that for oscillation reduction, with $T = 3s$, the $w_{i,b}(t)$ is [20]:

$$w_{i,b}(t) = \nu(t - t_0), \quad t_0 \leq t \leq t_0 + T,$$

$$\nu(t) = 4(0.25 + 0.65^t).$$

Here one observes the expected oscillation caused by delays. The data rates, after a transient period, converge to a point "close" to the optimal traffic allocation.

3.5.3 Link Failure

This section shows the simulation results in the presence of link failure. The same utility function given in the last section is assumed, and at $t = 15s$, the link between nodes b_5 and e_2 fails.

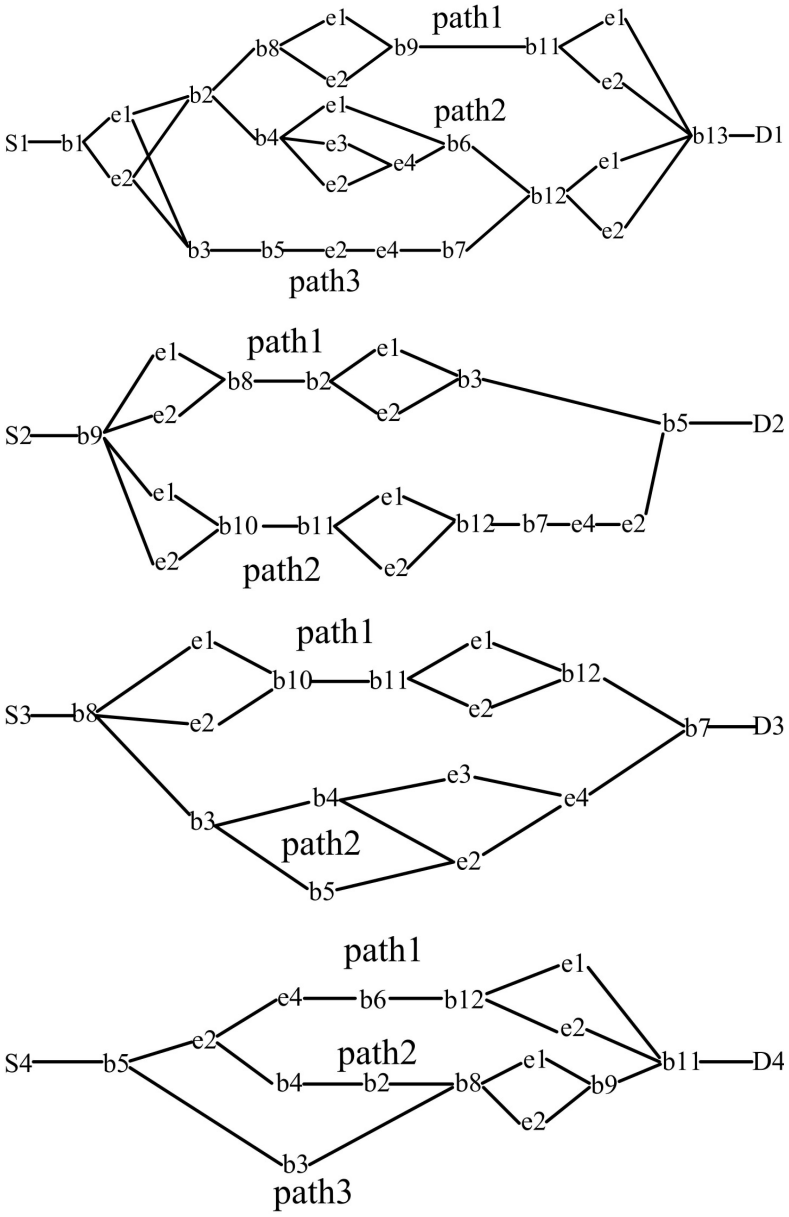
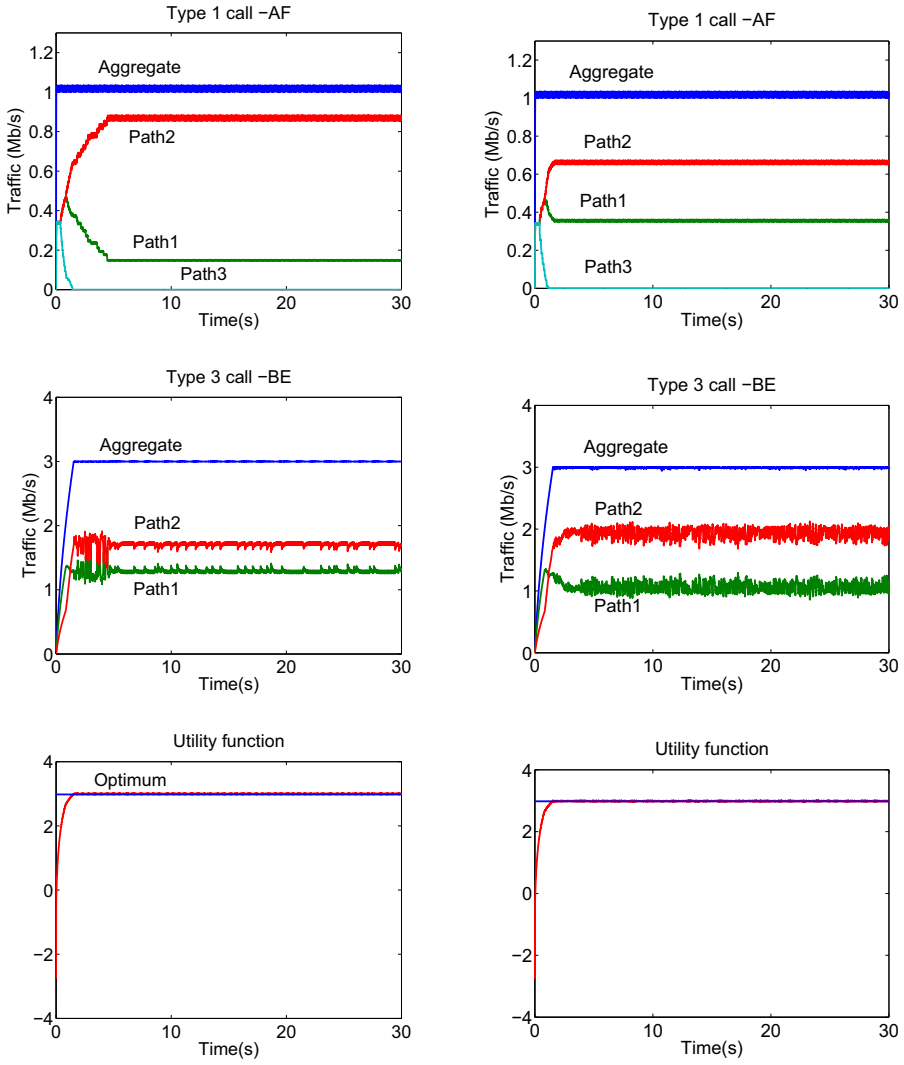


Fig. 3.5. Multi-path for each source-destination pair



without delay with delays

Fig. 3.6. Simulation

The Fig. 3.7 shows the network behavior with delays given in Fig. 3.2. The design parameters are the same as above. The simulation shows that the proposed control laws can also handle link failure.

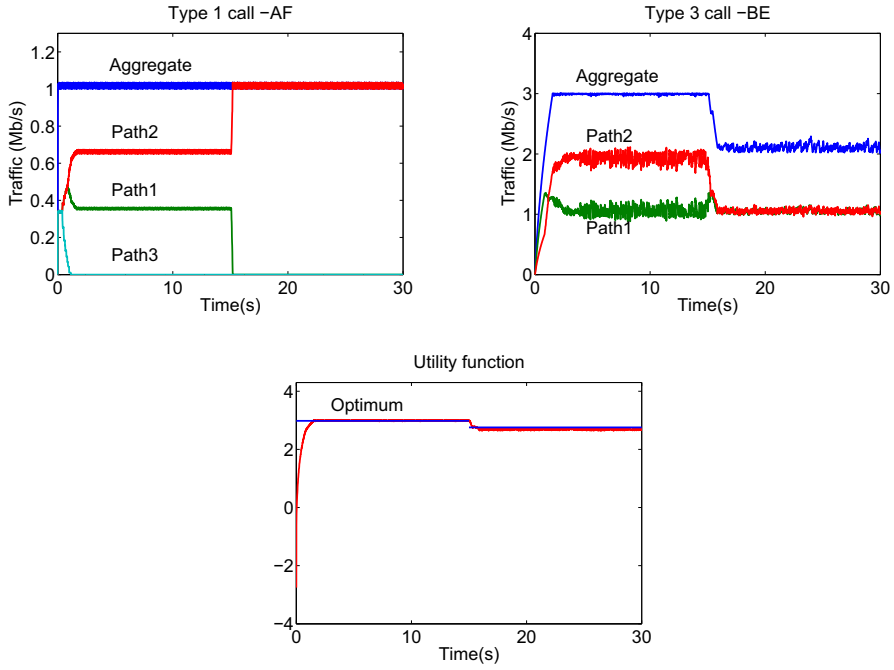


Fig. 3.7. Simulation with link failure and delay

3.6 Conclusion

In this chapter, based on the sliding mode control, a family of adaptation control laws for optimal rate adaptation and load balancing in a multi-domain internet is proposed. Moreover, the so-called percentage adaptation laws are presented which enables one to further improve the scalability of the control laws. The control laws run at the access points and domain edge nodes only, and can work properly with locally inferrable information only (e.g., through exchange of information with its next-hop node only). As a result, the proposed control approach is highly scalable. Simulation results presented demonstrate that the proposed approach can provide good QoS, TE, and FFR features in a multi-domain environment.

Chapter 4

Overlay Network Traffic Engineering Problem

In the last chapter, we proposed a family of control laws which converges to an optimal traffic allocation in a multi-domain environment. The main mathematical tool used is sliding mode theory. In this chapter, we use an approach based on the same mathematical concepts to address the traffic engineering problem in overlay networks. The family of control laws presented in this chapter is shown to converge to the optimal (time-varying) traffic allocation and uses only the number of congested links in a forwarding path as feedback for the control, making it an ideal traffic control solution for the overlay networks. This chapter is organized as follows. First, a precise problem statement and some additional assumptions are provided; next, the optimal distributed algorithm is presented; finally, some simulation results are given. A proof of the main result in this chapter is provided in the Appendix.

4.1 Notation

Throughout this chapter, traffic flows in the overlay network are assumed to be described by a fluid flow model and the only resource considered is link bandwidth.

Consider an overlay network where flows with different service requirements are present. Divide these flows into types. Types are aggregates of flows that can be treated as a unit; i.e., a type is an aggregate of flows with the same ingress and egress nodes. Moreover, service requirements are to be applied to the aggregate, not individual flows. Note that one can have flow types with just one flow. Assume that each flow type has several overlay paths available. The objective is to find an allocation of traffic that leads to the maximization of a given utility function subject to the network resource constraints and Class of Service (CoS) requirements.

More precisely, consider an overlay network whose set of overlay links is denoted by $\mathcal{L}$, with cardinality $card(\mathcal{L})$ equal to its number of links. Let $c_l(t)$ be the capacity of a link $l \in \mathcal{L}$. Due to the fact that overlay link resource uncertainty $c_l(t)$ is time-varying, define a constant $\dot{c}_{max}$ such that, for all t

$$-\dot{c}_{max} < \min_{l \in \mathcal{L}} \dot{c}_l(t) < \max_{l \in \mathcal{L}} \dot{c}_l(t) < \dot{c}_{max}.$$

Let $x_{i,j}$ be the data rate of flows of type i taking path j, n_i be the number of available paths for type i, and η be the number of types of flows. Let $\mathcal{L}_{i,j}$ be the set of links used by flows of type i taking path j. Also, let

$$\boldsymbol{x}_i \doteq \left[x_{i,1}, x_{i,2}, ..., x_{i,n_i}\right]^T \in \mathcal{R}^{n_i},$$

denote the vector containing the data rates allocated to the different paths by flows of type i, and

$$\boldsymbol{x} \doteq \left[\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, ..., \boldsymbol{x}_\eta^T\right]^T \in \mathcal{R}^n,$$

denote the vector containing all the data rates allocated to different types and respective paths, where $n = n_1 + n_2 + \cdots + n_\eta$.

A link $l \in \mathcal{L}$ is said to be congested if the aggregated data rate of the flows using the link is larger than or equal to $c_l(t)$. Given this, define $b_{i,j}(\boldsymbol{x})$ as the number of congested links along the j-th path of type i. Moreover, let $B_{i,j}$ be the number of links used by flows of type i taking path j. Finally, let $\mu(x)$ be the unit step function; i.e., $\mu(x) = 1$ if $x \geq 0$ and $\mu(x) = 0$ otherwise.

4.2 Problem Statement

In this chapter, we assume that some of the types of calls have service requirements. Flows of type i, $i = 1, ..., \eta_1$, are assumed to be Minimum Rate Guaranteed and Upper Bounded Rate Service (MRGUBS) category; i.e., there exist $\theta_i > 0$, $\Theta_i > 0$ such

that

$$\theta_i \leq \sum_{j=1,...,n_i} x_{i,j} \leq \Theta_i.$$

Flows of type i, $i = \eta_1 + 1, ..., \eta$, are assumed to be Best Effort (BE) category, type $\boldsymbol{x}_i$ of this class does not have any further service requirements. Note that one may have other classes of service, like Assured Forwarding Service (AF) (a target rate to be guaranteed), Minimum Rate Guaranteed Service (MRGS), and Upper Bounded Rate Service (UBRS). These are straightforward modification of MRGUBS CoS by appropriately choosing θ_i and Θ_i.

Assume that all data rates are bounded; i.e., there exists an $\iota \in \boldsymbol{R}$ such that $\boldsymbol{x}$ always belongs to the set

$$\mathcal{X} \doteq \left\{\boldsymbol{x} \in \boldsymbol{R}^n : x_{i,j} \leq \iota; j = 1, 2, \ldots, n_i;\ i = 1, 2, \ldots, \eta\right\}.$$

Let $U(\boldsymbol{x})$ be a given utility function representing the desired policy for assigning resources in the network

$$U(\boldsymbol{x}) = \sum_{i=1,...,\eta} U_i(\boldsymbol{x}_i),$$

where each $U_i(\cdot)$ is a differentiable strictly concave function in domain $\boldsymbol{R}^{n_i}$, and is increasing in each of its arguments in $\mathcal{X}$. Hence, the Hessian matrix H of $U(\boldsymbol{x})$ is invertible. Given this, the problem of optimizing utilization of the network resources can be formulated as

$$\max_{\boldsymbol{x}} U(\boldsymbol{x}),$$

subject to network capacity constraints

$$\sum_{i,j:l\in\mathcal{L}_{i,j}} x_{i,j} - c_l(t) \leq 0, \quad l \in \mathcal{L},$$

CoS requirement constraints

$$\theta_i \leq \sum_{j=1,...,n_i} x_{i,j} \leq \Theta_i, \quad i = 1, ..., \eta_1,$$

and nonnegativity of all data rates

$$x_{i,j} \geq 0, \quad i = 1, ..., \eta, j = 1, ..., n_i.$$

One can write this optimization problem in a matrix form

$$\max_{\boldsymbol{x}} U(\boldsymbol{x}), \quad \text{subject to constraints} \quad \boldsymbol{S}(\boldsymbol{x}, \boldsymbol{c}) \leq 0,$$

where $\boldsymbol{S}(\boldsymbol{x}, \boldsymbol{c}) = \mathcal{G}\boldsymbol{x} - \boldsymbol{c}(t)$, $\mathcal{G}$ is the gradient of $\boldsymbol{S}(\boldsymbol{x}, \boldsymbol{c})$ with respect to $\boldsymbol{x}$, and is an $M \times n$ constant matrix; i.e. there are M constraints.

Clearly this is a time-varying parametric convex optimization problem since the link capacities $c_l(t)$, for all $l \in \mathcal{L}$, are allowed to be time-varying. This problem is not a traditional optimization problem since the feasible set is time-varying. Correspondingly, the optimal solution of this problem, referred to as $\boldsymbol{x}_{opt}(t)$, is a function of t.

4.2.1 Additional Notation and Assumption

Assume that at the optimal traffic allocation each congested link has at least one non-binding class of service or a BE flow with a nonzero data rate.

Given any boundary point $\boldsymbol{x}_0$, one has an active constraint set

$$s_{\boldsymbol{x}_0}(\boldsymbol{x}) = \begin{bmatrix} s_1(\boldsymbol{x}) & ... & s_m(\boldsymbol{x}) \end{bmatrix}^T,$$

where $s_i(\boldsymbol{x}_0) = 0$. Without loss of generality, assume that the rows of the gradient $G_{\boldsymbol{x}_0}$ are linearly independent, where

$$G_{\boldsymbol{x}_0} = \frac{d\boldsymbol{s}_{\boldsymbol{x}_0}(\boldsymbol{x})}{d\boldsymbol{x}}.$$

Define a set of boundary points as

$$X_{sm} = \{\boldsymbol{x}_0 : (G_{\boldsymbol{x}_0} H^{-1}(\boldsymbol{x}_0) G^T_{\boldsymbol{x}_0})^{-1} G_{\boldsymbol{x}_0} H^{-1}(\boldsymbol{x}_0) \nabla U(\boldsymbol{x}_0) > 0\},$$

where H is the Hessian matrix. Assume that for all $\boldsymbol{x} \in X_{sm}$, each element of the vector $(G_{\boldsymbol{x}} H^{-1}(\boldsymbol{x}) G^T_{\boldsymbol{x}})^{-1} G_{\boldsymbol{x}} H^{-1}(\boldsymbol{x}) \nabla U(\boldsymbol{x})$ is lower bounded, i.e., there exists a positive constant φ,

$$\inf_{\boldsymbol{x} \in X_{sm}} ((G_{\boldsymbol{x}} H^{-1}(\boldsymbol{x}) G^T_{\boldsymbol{x}})^{-1} G_{\boldsymbol{x}} H^{-1}(\boldsymbol{x}) \nabla U(\boldsymbol{x})) > \varphi > 0. \tag{4.1}$$

Define constants $\Psi > 0$ and $\Phi > 0$, such that

$$\begin{aligned} \max_{\boldsymbol{x} \text{ feasible}} \|(G_{\boldsymbol{x}} H^{-1}(\boldsymbol{x}) G^T_{\boldsymbol{x}})^{-1}\| < \Psi, \\ \max_{\boldsymbol{x} \text{ feasible}} \|(G_{\boldsymbol{x}} H^{-1}(\boldsymbol{x}) G^T_{\boldsymbol{x}})\| < \Phi. \end{aligned} \tag{4.2}$$

Remarks: 1) Though condition (4.1) looks abstract, it has very intuitive meaning. From the optimization problem point of view, it means that the unconstrained optimal solution is outside the feasible set. In other words, if there is no link capacity constraints, the data rates will keep increasing. This condition holds true in computer networks, due to the fact that one will try to achieve a higher utility function value if the network has more resources (i.e., link capacity).

2) Condition (4.2) guarantees that for $\boldsymbol{x} \in X_{sm}$, with appropriately chosen control parameters, the state variable $\boldsymbol{x}$ is able to "catch" and track the optimal solution; i.e., the senders can increase the data rate fast enough to reach and keep following the optimal traffic allocation.

Moreover, the following conditions are assumed to be met throughout this chapter

- the function $c_l(t) > 0$, for all $l \in \mathcal{L}$ and for all t, and $c_l(t)$ is upper bounded by, e.g., the physical link capacity.
- the constant $\dot{c}_{max} > 0$ is finite.

4.3 Optimal Control Laws

We propose the following distributed family of control laws:
for MRGUBS flows

$$\dot{\boldsymbol{x}}_i = -\zeta H_i^{-1}[\frac{dU_i}{d\boldsymbol{x}_i} - \alpha \boldsymbol{b}_i(\boldsymbol{x}) + \beta_i^m \boldsymbol{r}_i^m(\boldsymbol{x}_i) - \beta_i^M \boldsymbol{r}_i^M(\boldsymbol{x}_i) + \varpi_i \boldsymbol{\mu}_i(-\boldsymbol{x}_i)],$$

for BE flows

$$\dot{\boldsymbol{x}}_i = -\zeta H_i^{-1}[\frac{dU_i}{d\boldsymbol{x}_i} - \alpha \boldsymbol{b}_i(\boldsymbol{x}) + \varpi_i \boldsymbol{\mu}_i(-\boldsymbol{x}_i)],$$

where H_i^{-1} is the i-th block of the inverse of the Hessian matrix H (which is block diagonal), $\boldsymbol{b}_i(\boldsymbol{x})$, $\boldsymbol{\mu}_i(-\boldsymbol{x}_i)$, $\boldsymbol{r}_i^m(\boldsymbol{x}_i)$ and $\boldsymbol{r}_i^M(\boldsymbol{x}_i)$ are $n_i \times 1$ vectors

$$\boldsymbol{b}_i(\boldsymbol{x}) = \begin{bmatrix} b_{i,1}(\boldsymbol{x}) & b_{i,2}(\boldsymbol{x}) & ... & b_{i,n_i}(\boldsymbol{x}) \end{bmatrix}^T,$$

$$\boldsymbol{\mu}_i(-\boldsymbol{x}_i) = \begin{bmatrix} \mu(-x_{i,1}) & \mu(-x_{i,2}) & ... & \mu(-x_{i,n_i}) \end{bmatrix}^T,$$

$$\boldsymbol{r}_i^m(\boldsymbol{x}_i) = \begin{cases} \begin{bmatrix} 0 & 0 & ... & 0 \end{bmatrix}^T, & \text{if } \sum_{j=1,...,n_i} x_{i,j} > \theta_i, \\ \begin{bmatrix} 1 & 1 & ... & 1 \end{bmatrix}^T, & \text{if } \sum_{j=1,...,n_i} x_{i,j} < \theta_i, \end{cases}$$

$$\boldsymbol{r}_i^M(\boldsymbol{x}_i) = \begin{cases} \begin{bmatrix} 1 & 1 & ... & 1 \end{bmatrix}^T, & \text{if } \sum_{j=1,...,n_i} x_{i,j} > \Theta_i, \\ \begin{bmatrix} 0 & 0 & ... & 0 \end{bmatrix}^T, & \text{if } \sum_{j=1,...,n_i} x_{i,j} < \Theta_i, \end{cases}$$

ζ, α, β_i, β_i^m, β_i^M and ϖ_i are design parameters. Note that the control laws are *distributed* in the sense that flows are controlled independently and the only interactions among flows is through $b_{i,j}$, the number of congested links in a forwarding path j for flow i. In an overlay environment, each path is composed of multiple intermediate overlay nodes interconnected through overlay links. Overlay link congestion information can be easily inferred by the overlay nodes themselves through per-hop probing. This information is

then fed back from each intermediate overlay nodes to the source nodes. Hence, this approach does not require the exposure of the underlying resources and topology to the overlay.

As mentioned before, the approach presented in this section can be extended to other classes of service, such as Assured Forwarding Service (AF), Minimum Rate Guaranteed Service (MRGS), and Upper Bounded Rate Service (UBRS).

We now show that the control laws converge to the optimal traffic allocation.

THEOREM 4.1. *If all the conditions in Section 4.2 hold, there exists a constant* $\alpha^*(\dot{c}_{max}) > 0$ *such that if for all* $\boldsymbol{x} \in \mathcal{X}$, $i = 1, 2, \ldots, \eta$, $j = 1, 2, \ldots, n_i$, *the parameters* ζ, α, β_i, β_i^m, β_i^M, ϖ_i, *and* $\dot{c}_{max}$ *satisfy*

$$\zeta > \rho = \frac{\dot{c}_{max} n^{1/2} \Psi}{\varphi};\ \alpha > \alpha^*;\ \beta_i, \beta_i^m, \beta_i^M \geq \alpha^* \max_{i,j} B_{i,j};\ \varpi_i \geq 2\alpha^* \max_{i,j} B_{i,j}; \tag{4.3}$$

the family of control laws above converges to the optimal traffic allocation; i.e.,

$$\lim_{t \to \infty} \|\boldsymbol{x}(t) - \boldsymbol{x}_{opt}(t)\| = 0.$$

Remark: If the Hessian H is diagonal (not just block diagonal), α^* can be calculated as follows

$$\alpha^* = \max_{x \in \mathcal{X}} (\nabla U) + \frac{\dot{c}_{max}}{\rho \min_{x \in \mathcal{X}} (-H^{-1})}, \tag{4.4}$$

where $\min(-H^{-1})$ is the smallest diagonal element of $-H^{-1}$. Intuitively, α should be set large enough so that when a link is congested, and the link capacity is decreasing, the control laws can "push" state variable $\boldsymbol{x}$ back to feasible set.

4.4 Numerical Examples

In this section, some simulation results are presented, which exemplify the behavior of the algorithms proposed. Note that, similarly to the simulation given in Section 3.5,

the implementation of the proposed algorithm in a real network is performed in discrete time as well.

4.4.1 Simulation Setup

The model of the network used in the simulation is based on the one in [20] and is shown in Fig. 4.1, where all the links' bandwidths and delays, as well as source and destination nodes are shown. We assume that one has multiple paths and multiple CoSs. More precisely, there exist a total of $\eta = 8$ types of flows corresponding to 8 different combinations of source/destination nodes. The paths available for each pair of ingress/egress nodes are described in Table 4.1. Flows of type 3 and 5 are assumed to be of AF type with target rates $\theta_3 = \Theta_3 = 1Mb/s$, $\theta_5 = \Theta_5 = 0.8Mb/s$, flows of other types are assumed to be BE type.

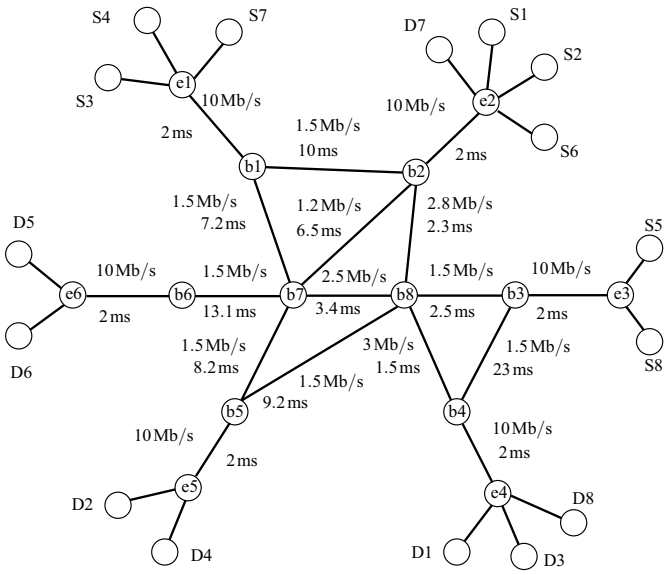


Fig. 4.1. Network topology

Table 4.1. Paths available for each type of calls.

type 1 $n_1 = 4$	$x_{1,1}$: $e_2b_2b_8b_4e_4$ $x_{1,2}$: $e_2b_2b_8b_3b_4e_4$ $x_{1,3}$: $e_2b_2b_7b_8b_3b_4e_4$ $x_{1,4}$: $e_2b_2b_7b_8b_4e_4$	type 5 $n_5 = 2$	$x_{5,1}$: $e_3b_3b_8b_7b_6e_6$ $x_{5,2}$: $e_3b_3b_4b_8b_5b_7b_6e_6$
type 2 $n_2 = 3$	$x_{2,1}$: $e_2b_2b_8b_5e_5$ $x_{2,2}$: $e_2b_2b_7b_5e_5$ $x_{2,3}$: $e_2b_2b_1b_7b_5e_5$	type 6 $n_6 = 3$	$x_{6,1}$: $e_2b_2b_1b_7b_6e_6$ $x_{6,2}$: $e_2b_2b_8b_7b_6e_6$ $x_{6,3}$: $e_2b_2b_7b_6e_6$
type 3 $n_3 = 2$	$x_{3,1}$: $e_1b_1b_7b_8b_4e_4$ $x_{3,2}$: $e_1b_1b_2b_8b_4e_4$	type 7 $n_7 = 3$	$x_{7,1}$: $e_1b_1b_2e_2$ $x_{7,2}$: $e_1b_1b_7b_2e_2$ $x_{7,3}$: $e_1b_1b_7b_8b_2e_2$
type 4 $n_4 = 4$	$x_{4,1}$: $e_1b_1b_7b_5e_5$ $x_{4,2}$: $e_1b_1b_7b_8b_5e_5$ $x_{4,3}$: $e_1b_1b_2b_7b_5e_5$ $x_{4,4}$: $e_1b_1b_2b_8b_5e_5$	type 8 $n_8 = 2$	$x_{8,1}$: $e_3b_3b_4e_4$ $x_{8,2}$: $e_3b_3b_8b_4e_4$

The capacity of the link between node b2 and node b7 is assumed to be time-varying. More precisely, we have

$$c(t) = 1.2 - \cos(0.5t).$$

Since the term involving $\varpi_{i,j}$ can lead to large oscillation, when implementing the control laws, we take $\dot{x}_{i,j} = 0$, if $x_{i,j} \leq 0$ and $\dot{x}_{i,j} < 0$. This is equivalent to setting $\varpi_i = +\infty$ in the original control laws.

The following utility function is assumed

$$U(\boldsymbol{x}) = \sum log(x_{i,j} + 3), \quad i = 1, ..., 8, j = 1, ..., n_i.$$

As a first step, the control laws were tested in almost ideal conditions by setting network delays to 0 and sampling time to $0.1ms$. The design parameters are chosen as $\alpha = 3$, $\beta_i, \beta_i^m, \beta_i^M = \beta = 3\alpha$, and $\zeta = 1$. It can be seen in the left column of Fig. 4.2 that, under these conditions, the network converges to an optimal operation point, and keeps being optimal. Furthermore, the backbone links are fully utilized, and the AF requirements are satisfied.

The right column of Fig. 4.2 shows the network behavior with the delays given in Fig. 4.1 and sampling time $0.5ms$. The design parameters are chosen as $\alpha = 0.5$, $\beta_i, \beta_i^m, \beta_i^M = \beta = 3\alpha$, and $\zeta = 1$. Here one observes the expected oscillations caused by delays. The data rates, after a transient period, converge to a point close to the optimal point and the optimal traffic allocation is closely followed.

4.4.2 Link Failure

This section shows the simulation of link failure(which is equivalent to discontinuous jumps in the capacity). The same utility function is assumed, and at $t = 8s$, the link capacity of the link between node b2 and b8 becomes 0.

The left column of Fig. 4.3 shows the network behavior without delay. The design parameters are chosen as $\alpha = 3$, $\beta_i, \beta_i^m, \beta_i^M = \beta = 3\alpha$, and $\zeta = 1$. The right column of Fig. 4.3 shows the network behavior with delays given in Fig. 4.1. The design parameters are chosen as $\alpha = 0.5$, $\beta_i, \beta_i^m, \beta_i^M = \beta = 3\alpha$, and $\zeta = 1$. The simulation shows that the proposed control laws can also handle link uncertainty/failure.

4.5 Conclusion

In this chapter, an optimization-based distributed algorithm for an overlay network is developed, under the assumption that the utility function is strictly concave. This algorithm converges to and tracks the time-varying optimal solution. The simulation results show that the proposed algorithm leads to rather high performance in terms of resource utilization, as measured by utility function.

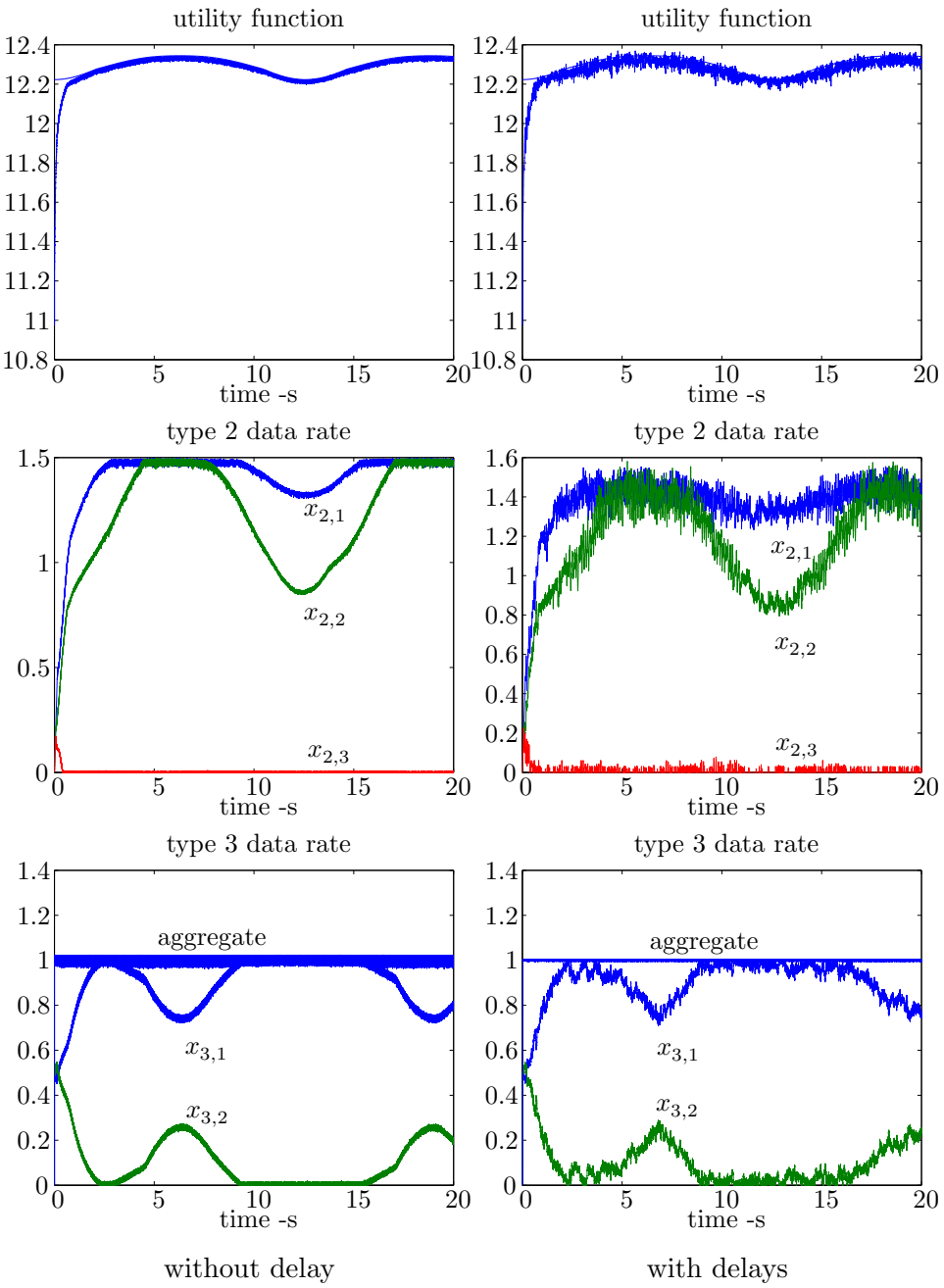


Fig. 4.2. Simulation results

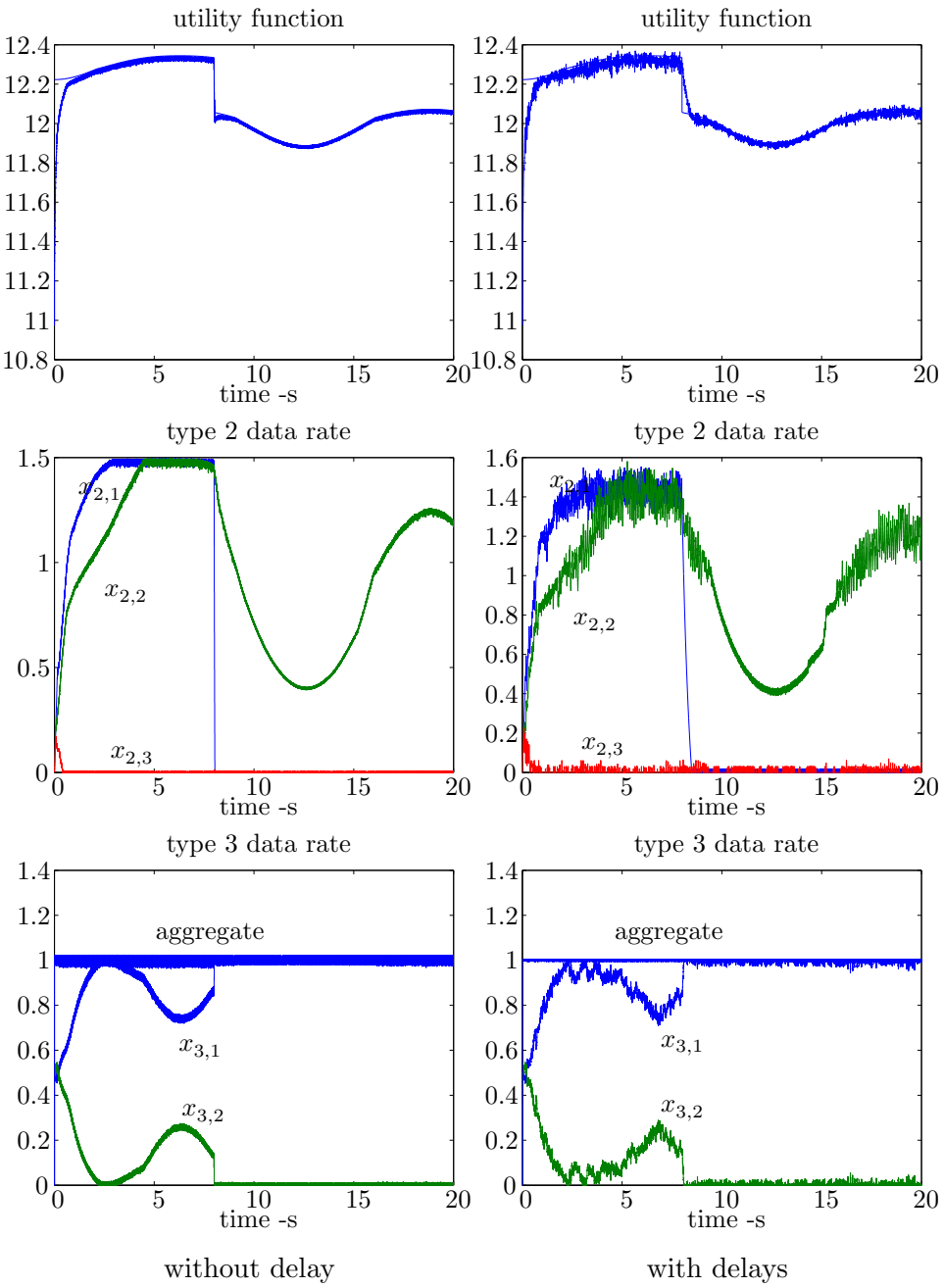


Fig. 4.3. Simulation results with link failure

Chapter 5

Time-varying Optimization Problem

The overlay traffic engineering problem presented in Chapter 4 has time-varying link capacities. Hence, it is fundamentally a time-varying optimization problem. We addressed this problem and designed data rate control algorithm for it, by using optimization-based methods. In this chapter, we extend our research to more general time-varying optimization problems. Our work focuses on a class of time-varying objective functions having derivatives with "linear" discontinuity, and proposes an algorithm that converges to an arbitrarily small neighborhood of the time-varying optimum. Moreover, such an algorithm is applied to time-varying linearly constrained strictly convex optimization problems, and sufficient conditions for asymptotic convergence are presented.

This chapter presents our results about time-varying optimization problems, and is organized as follows. In Section 5.1, we provide some notation and definitions used throughout this chapter. In Section 5.2, a Continuous First Order Algorithm (CFoA) is proposed for unconstrained optimization problems having twice differentiable strictly convex/concave objective functions. Moreover, in order to achieve "smoother" behavior than the CFoA, a Continuous Second Order Algorithm (CSoA) is also provided. Assuming knowledge of $\partial^2 U(\boldsymbol{x}, t)/\partial \boldsymbol{x} \partial t$, both of these algorithms are shown to converge to and track the time-varying optimum. The results in Section 5.2 serve as a stepping stone for the results in the following sections. Section 5.3 addresses a class of time-varying objective functions having derivatives with "linear" discontinuity, and a so-called Sliding Algorithm (SA) is proposed. The SA does not require any "global" time-varying information, and it is shown to converge to an arbitrarily small neighborhood of the time-varying optimum. Section 5.4 focuses on time-varying linearly constrained optimization problems. Sufficient conditions for asymptotic convergence of the SA are provided. Simulations are presented in Section 5.5 to exemplify the behavior of the algorithms proposed in previous sections. The proofs of the results in this chapter are given in the Appendices.

5.1 Notation and Definitions

In this section, we provide the notation and definitions which are used throughout the chapter. For a general time-varying function $U(\boldsymbol{x}, t)$, $U(\boldsymbol{x}, t)$ is said to be convex (concave) if $U(\boldsymbol{x}, t)$ is a convex (concave) function of $\boldsymbol{x}$ for fixed t. Moreover, $U(\boldsymbol{x}, t)$ is said to be (not to be) differentiable if $U(\boldsymbol{x}, t)$ is (is not) differentiable with respect to $\boldsymbol{x}$ for fixed t.

For a given $\boldsymbol{x}_0$, the set of subgradients of $U(\boldsymbol{x}, t)$ with respect to $\boldsymbol{x}$ at $\boldsymbol{x}_0$ is denoted by

$$\partial_{\boldsymbol{x}} U(\boldsymbol{x}_0, t).$$

Moreover, if $U(\boldsymbol{x}, t)$ is differentiable with respect to $\boldsymbol{x}$, the following notation is used

$$\nabla U(\boldsymbol{x}_0, t) = \left. \frac{\partial U(\boldsymbol{x}, t)}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}=\boldsymbol{x}_0}.$$

Also, if $U(\boldsymbol{x}, t)$ is twice differentiable with respect to $\boldsymbol{x}$ for fixed t, let $H(\boldsymbol{x}_0, t)$ denote the Hessian matrix of $U(\boldsymbol{x}, t)$ with respect to $\boldsymbol{x}$ at $\boldsymbol{x} = \boldsymbol{x}_0$ and time t. Moreover, taking the quantity $H(\boldsymbol{x}, t)$ as example, if the trajectory $\boldsymbol{x}(t)$ is given, one has the function

$$H[\boldsymbol{x}(t), t] = H(\boldsymbol{x}, t)|_{\boldsymbol{x}=\boldsymbol{x}(t)}.$$

Note that $H(\boldsymbol{x}, t)$ depends on $\boldsymbol{x}$ and t, while $H[\boldsymbol{x}(t), t]$ depends only on t. Moreover, we use $H(t)$ or just H to denote $H[\boldsymbol{x}(t), t]$ when clear. One should note that this convention applies to other quantities as well.

Let $v \in R^n$ be a vector, and a be a scalar. Then $v > a$ means that $v_i > a$, $i = 1, ..., n$. The matrix I denotes the identity matrix. Unless otherwise specified, the vector norm used throughout this chapter is the 2-norm, and the matrix norm is the

induced 2-norm. Also define the function $T(\zeta, \epsilon, a)$ as

$$T(\zeta, \epsilon, a) = \begin{cases} \frac{1}{\zeta} \ln \frac{a+\epsilon}{\epsilon}, & a \geq 0, \\ \\ 0, & a < 0, \end{cases} \tag{5.1}$$

where $\zeta, \epsilon > 0$ are scalars.

Finally, for a general time-varying function $U(\boldsymbol{x}, t)$, we define the following condition which we refer to as the *Basic Condition*:

CONDITION 1. *For a general time-varying function $U(\boldsymbol{x}, t)$, the Basic Condition holds, if*

- *for all $\boldsymbol{x}$ and t, $U(\boldsymbol{x}, t)$ is twice differentiable with respect to $\boldsymbol{x}$,*
- *for all $\boldsymbol{x}$ and t, there exist positive constants p and P, such that*

$$pI < -H(\boldsymbol{x}, t) < PI, \quad \textit{and} \quad pI < -H^{-1}(\boldsymbol{x}, t) < PI,$$

- *for all $\boldsymbol{x}$ and t, $\nabla U(\boldsymbol{x}, t)$ is differentiable with respect to t, and there exists a positive constant Q, such that*

$$\left\| \frac{\partial \nabla U(\boldsymbol{x}, t)}{\partial t} \right\| < Q < \infty.$$

5.2 Time-varying Optimization Problem with Continuous Derivative

This section focuses on time-varying unconstrained convex optimization problems with twice differentiable objective functions. For such a problem, a first algorithm, referred to as the Continuous First Order Algorithm (CFoA), is proposed in Section 5.2.1. Then, a few comments on how to use the CFoA in a constrained problem context are given. This specific application provides the motivation for the development of another algorithm, referred to as the Continuous Second Order Algorithm (CSoA), which has less oscillations than the CFoA, in the case when one has a "rapidly" changing gradient.

The results in this section serve as a stepping stone for results in the next section, which focus on a class of time-varying problems; i.e., the objective function is strictly concave but its derivative is discontinuous.

5.2.1 First Order Algorithm

Consider the following time-varying unconstrained convex optimization problem

$$\max_{\boldsymbol{x}} U(\boldsymbol{x}, t), \tag{5.2}$$

where function $U(\boldsymbol{x}, t)$ satisfies Condition 1. It can be shown that the time-varying optimum is unique and a continuous function of t.

Given an $n \times n$ matrix function $K(\boldsymbol{x})$, we propose the following algorithm to solve problem (5.2), which we refer to as the Continuous First Order Algorithm (CFoA)

$$\dot{\boldsymbol{x}} = -K[\boldsymbol{x}(t)]H[\boldsymbol{x}(t), t]\nabla U[\boldsymbol{x}(t), t] - H^{-1}[\boldsymbol{x}(t), t]\frac{\partial \nabla U[\boldsymbol{x}(t), t]}{\partial t}. \tag{5.3}$$

We now provide a result that establishes the convergence of the algorithm above. Its proof is given in Appendix C.1, and simulation results are given in Section 5.5.1.

THEOREM 5.1. *If there exists a constant $z > 0$, such that $K(\boldsymbol{x}) > zI$ for all $\boldsymbol{x}$, and $K(\boldsymbol{x})$ is continuous with respect to $\boldsymbol{x}$, algorithm (5.3) converges; i.e.,*

$$\lim_{t \to \infty} \|\nabla U[\boldsymbol{x}(t), t]\| = 0.$$

Remark: For example, if one takes $K(\boldsymbol{x}) = \zeta H^{-2}(\boldsymbol{x})$ with $\zeta > 0$, then the following algorithm (proposed in [38]) is a special case of CFoA (5.3)

$$\dot{\boldsymbol{x}} = -H^{-1}[\boldsymbol{x}(t), t]\left[\zeta \nabla U[\boldsymbol{x}(t), t] + \frac{\partial \nabla U[\boldsymbol{x}(t), t]}{\partial t}\right]. \tag{5.4}$$

5.2.2 A Few Remarks on Constrained Optimization Problems

The CFoA can also be used to approximate the solution of a time-varying constrained optimization problem. Consider a general time-varying constrained convex optimization problem

$$\max_{\boldsymbol{x}} U(\boldsymbol{x},t), \quad \text{subject to} \quad s_i(\boldsymbol{x},t) \leq 0, \quad i=1,...,M. \tag{5.5}$$

Assume that the objective function $U(\boldsymbol{x},t)$ satisfies Condition 1, and that the functions $s_i(\boldsymbol{x},t)$, $i=1,...,M$, are twice differentiable convex functions which satisfy

$$\left\| \frac{\partial s_i(\boldsymbol{x},t)}{\partial t} \right\| < \infty, \quad \text{and} \quad \left\| \frac{\partial \nabla s_i(\boldsymbol{x},t)}{\partial t} \right\| < \infty, \quad \forall (\boldsymbol{x},t). \tag{5.6}$$

Hence problem (5.5) is a convex optimization problem for any t, and the optimal solution $\boldsymbol{x}_{opt}(t)$ of problem (5.5) is unique and a continuous function of t.

Given problem (5.5), define the following unconstrained problem

$$\max_{\boldsymbol{x}} \tilde{U}(\boldsymbol{x},t) = U(\boldsymbol{x},t) - \sum_{i=1,...,M} u_i s_i(\boldsymbol{x},t), \tag{5.7}$$

where

$$u_i = \begin{cases} \alpha_i, & \text{if } \ s_i > 0, \\ \\ 0, & \text{if } \ s_i < 0. \end{cases}$$

Let $\tilde{\boldsymbol{x}}_{opt}(t)$ be the optimal solution of problem (5.7). For any given t_0, it is well known that problem (5.5) and problem (5.7) have the same solution for large enough α_i, $i=1,...,M$ [33]; i.e., there exists a constant $\alpha^*(t_0) > 0$, if the constants α_i of problem (5.7), $i=1,...,M$, satisfy $\alpha_i > \alpha^*(t_0) > 0$, then $\tilde{\boldsymbol{x}}_{opt}(t_0) = \boldsymbol{x}_{opt}(t_0)$. Moreover, if there exists a constant α^* which satisfies $\alpha^* = \sup_t \alpha^*(t) < \infty$, and $\alpha_i > \alpha^* > 0$, then $\tilde{\boldsymbol{x}}_{opt}(t) = \boldsymbol{x}_{opt}(t)$.

In problem (5.7), although $\tilde{U}(\boldsymbol{x},t)$ is not differentiable everywhere, due to the fact that $U(\boldsymbol{x},t)$ satisfies Condition 1 and the functions $s_i(\boldsymbol{x},t)$ satisfy condition (5.6),

one may use a function $\bar{U}(\boldsymbol{x},t)$ which satisfies Condition 1 to approximate $\tilde{U}(\boldsymbol{x},t)$. This means one may apply the CFoA to solve the following problem

$$\max_{\boldsymbol{x}} \bar{U}(\boldsymbol{x},t). \tag{5.8}$$

Denote the optimal solution of the problem above by $\bar{\boldsymbol{x}}_{opt}(t)$. If $\bar{U}(\boldsymbol{x},t)$ is very "close" to $\tilde{U}(\boldsymbol{x},t)$, then $\bar{\boldsymbol{x}}_{opt}(t)$ will be very "close" to $\tilde{\boldsymbol{x}}_{opt}(t)$. More precisely, given any $\epsilon > 0$, there exists $\xi > 0$, such that, if for all t

$$\|\tilde{U}(\boldsymbol{x},t) - \bar{U}(\boldsymbol{x},t)\|_{\infty} < \xi,$$

then

$$\|\tilde{\boldsymbol{x}}_{opt}(t) - \bar{\boldsymbol{x}}_{opt}(t)\| < \epsilon.$$

Hence, one can apply CFoA (5.3) to solve problem (5.8), and therefore, provide an approximate solution of the constrained time-varying optimization problem (5.5).

5.2.3 Second Order Algorithm

In the approximation problem (5.8) provided in the last section, the derivative may change "fast" near the boundary of the feasible set of the constrained optimization problem (5.5) (i.e., s_i is "close" to zero for some i). In such a case, a direct implementation of CFoA (5.3) might lead to high frequency oscillations. A simulation of this situation is given in Section 5.5.1. This motivates the study of the so-called Continuous Second Order Algorithm (CSoA) for problem (5.2), which is presented below

$$\begin{cases} \dot{\boldsymbol{x}} = -H^{-1}[\boldsymbol{x}(t),t]\left[K[\boldsymbol{x}(t)]\boldsymbol{y}(t) + \dfrac{\partial \nabla U[\boldsymbol{x}(t),t]}{\partial t}\right], \\ \dot{\boldsymbol{y}} = K[\boldsymbol{x}(t)]\nabla U[\boldsymbol{x}(t),t] - q[\boldsymbol{y}(t)]. \end{cases} \tag{5.9}$$

where $K(\boldsymbol{x})$ is a given $n \times n$ matrix function of $\boldsymbol{x}$. The vector $q(\boldsymbol{y}) = [...q_i(y_i)...]^T$, where $q_i(y_i)$, $i = 1, ..., n$, are continuous functions of y_i, satisfying

$$q_i(0) = 0,$$

$$y_i q_i(y_i) > 0, \quad \text{for} \quad y_i \neq 0.$$

We now provide a result that establishes the convergence of the Continuous Second Order Algorithm (5.9). Its proof is given in Appendix C.2, and simulation results are provided in Section 5.5.1.

THEOREM 5.2. *If there exists a constant $z > 0$, such that $K(\boldsymbol{x}) > zI$ for all $\boldsymbol{x}$, and $K(\boldsymbol{x})$ is continuous with respect to $\boldsymbol{x}$, then for any bounded initial condition $\boldsymbol{y}(0)$, algorithm (5.9) converges; i.e.,*

$$\lim_{t \to \infty} \|\nabla U[\boldsymbol{x}(t), t]\| = 0.$$

5.3 Time-varying Optimization Problem with Discontinuous Derivative

The discussion in Section 5.2 shows that, by using the approximation problem (5.8), CFoA (5.3) and CSoA (5.9) both converge to an arbitrarily small neighborhood of the optimal solution of the constrained problem (5.7). However, to determine the true optimal solution, we need address optimization problems with objective functions having discontinuous derivatives.

In this section, we only consider time-varying optimization problems having derivatives with "linear" discontinuity; i.e., optimization problems of the following form

$$\max_{\boldsymbol{x}} \tilde{U}(\boldsymbol{x}, t) = U(\boldsymbol{x}, t) - \sum_{i=1,...,M} u_i s_i(\boldsymbol{x}, t) = U(\boldsymbol{x}, t) - \sum f_i(s_i), \tag{5.10}$$

where

$$s_i(\boldsymbol{x}, t) = g_i^T \boldsymbol{x} - c_i(t), \quad i = 1, ..., M,$$

$$u_i = \begin{cases} \alpha_i > 0, & \text{if } s_i > 0, \\ 0, & \text{if } s_i < 0. \end{cases}$$

Without loss of generality, it is assumed that

$$\|g_i\| = 1, \quad i = 1, ..., M.$$

Moreover assume that $U(\boldsymbol{x}, t)$ satisfies Condition 1, and that functions $s_i(\boldsymbol{x}, t)$, $i = 1, ..., M$, are linear. Hence for any t, $\tilde{U}(\boldsymbol{x}, t)$ is strictly concave with respect to $\boldsymbol{x}$.

It is also assumed that there exists a positive constant $\dot{c}_{max}$, such that

$$-\dot{c}_{max} < \inf_{i,t} \dot{c}_i(t) < \sup_{i,t} \dot{c}_i(t) < \dot{c}_{max}, \tag{5.11}$$

with $\dot{c}_{max} < \infty$. Since problem (5.10) is strictly concave and $\dot{c}_{max}$ is bounded, it can be proved that the time-varying optimal solution of problem (5.10) is unique and a continuous function of t.

The gradient of $\tilde{U}(\boldsymbol{x}, t)$ is given by

$$\nabla\tilde{U}(\boldsymbol{x}, t) = \nabla U(\boldsymbol{x}, t) - \mathcal{G}^T u, \quad \text{if at } \boldsymbol{x},\ s_i(\boldsymbol{x}, t) \neq 0 \text{ for all } i,$$

where the ith column of $\mathcal{G}^T$ is g_i, and the ith entry of the vector u is u_i. The Hessian matrix $\tilde{H}(\boldsymbol{x}, t)$ of $\tilde{U}(\boldsymbol{x}, t)$ is given by

$$\tilde{H}(\boldsymbol{x}, t) = H(\boldsymbol{x}, t), \quad \text{if at } \boldsymbol{x},\ s_i(\boldsymbol{x}, t) \neq 0 \text{ for all } i.$$

Note that due to the structure of $\tilde{U}(\boldsymbol{x}, t)$, $\nabla\tilde{U}(\boldsymbol{x}, t)$ and $\tilde{H}(\boldsymbol{x}, t)$ are not defined at $(\boldsymbol{x}, t)$, if for some i, $s_i(\boldsymbol{x}, t) = 0$. But for any given t, $\nabla\tilde{U}(\boldsymbol{x}, t)$ and $\tilde{H}(\boldsymbol{x}, t)$ are defined almost everywhere.

To solve the time-varying optimization problem (5.10), we propose the following algorithm, which we refer to as the Sliding Algorithm (SA)

$$\begin{aligned}\dot{\boldsymbol{x}} &= -\tilde{H}^{-1}[\boldsymbol{x}(t),t]\left[\zeta\nabla\tilde{U}[\boldsymbol{x}(t),t]+\frac{\partial\nabla\tilde{U}[\boldsymbol{x}(t),t]}{\partial t}\right]\\ &= -H^{-1}[\boldsymbol{x}(t),t]\left[\zeta\nabla U[\boldsymbol{x}(t),t]+\frac{\partial\nabla U[\boldsymbol{x}(t),t]}{\partial t}-\zeta\mathcal{G}^{T}u\right],\end{aligned} \tag{5.12}$$

where $\zeta > 0$ is a constant.[1]

5.3.1 Descent Function

We now define the descent function used throughout this chapter, for time-varying problems with objective functions having discontinuous derivatives. Given an objective function $\tilde{U}(\boldsymbol{x},t)$, define $L(\boldsymbol{x},t)$ as

$$L(\boldsymbol{x},t)=\begin{cases}\|\nabla\tilde{U}(\boldsymbol{x},t)\|, & \text{if at } \boldsymbol{x},\ \tilde{U}(\boldsymbol{x},t) \text{ is differentiable,}\\ \min\limits_{v\in\partial_{\boldsymbol{x}}\tilde{U}(\boldsymbol{x},t)}\|v\|, & \text{if at } \boldsymbol{x},\ \tilde{U}(\boldsymbol{x},t) \text{ is not differentiable.}\end{cases} \tag{5.13}$$

Recall that $\partial_{\boldsymbol{x}}\tilde{U}(\boldsymbol{x},t)$ is the set of subgradients of $\tilde{U}(\boldsymbol{x},t)$ at $\boldsymbol{x}$.

5.3.2 Convergence of Sliding Algorithm

The following result states that with Sliding Algorithm (5.12), the descent function $L[\boldsymbol{x}(t),t]$ of problem (5.10) converges to an arbitrarily small neighborhood of zero. Its proof is given in Appendix C.3 and simulation results are provided in Section 5.5.2.

THEOREM 5.3. *The Sliding Algorithm (5.12) tracks the time-varying optimal solution of problem (5.10) in the following sense*

$$\limsup_{t\to\infty} L[\boldsymbol{x}(t),t] \le \frac{M\dot{c}_{max}}{\zeta p}+\frac{MQ}{\zeta}, \tag{5.14}$$

[1]The algorithm is not defined at $(\boldsymbol{x},t)$, if, for some i, $s_i(\boldsymbol{x},t)=0$. However, the behavior is completely defined by the behaviors at $s_i(\boldsymbol{x},t)>0$ and $s_i(\boldsymbol{x},t)<0$.

where the constants $\dot{c}_{max}$, Q, *and* p *are defined in (5.11) and in Condition 1. Moreover, for any given* $\epsilon > 0$, *and initial condition* $L[\boldsymbol{x}(t_0), t_0]$, *let*

$$\hat{T} = T(\zeta, \epsilon, L[\boldsymbol{x}(t_0), t_0] - \epsilon) + MT(\zeta, \epsilon, \frac{\dot{c}_{max}}{\zeta p} + \frac{Q}{\zeta}),$$

where the function $T(\zeta, \epsilon, a)$ *is given by (5.1). Then,*

$$L[\boldsymbol{x}(t), t] \leq \frac{M\dot{c}_{max}}{\zeta p} + \frac{MQ}{\zeta} + \epsilon, \quad \textit{for all} \;\; t \geq \hat{T}.$$

Remark 1: Recall that $T(\zeta, \epsilon, a)$ is defined by (5.1) in Section 5.1, hence, $\hat{T}$ decreases with respect to the parameters ζ, ϵ, p, and increases with respect to M, $L[\boldsymbol{x}(t_0), t_0]$, $\dot{c}_{max}$, and Q. Therefore, with a large enough ζ, $L(t)$ will converge to an arbitrarily small neighborhood of zero.

Remark 2: There are cases for which $L(t)$ will converge to zero with a finite ζ. A precise definition of such situations is only given and used in the proof (see Appendix C.3.2), due to the fact that it is hard to check it in real implementation. In the next section sufficient conditions for asymptotic convergence of the SA with a finite ζ are provided.

5.4 Asymptotic Convergence for Optimization Problem with Time-varying Linear Constraints

As seen in Theorem 5.3, as the algorithm parameter ζ goes to infinity, one has

$$\lim_{\zeta \to \infty} (\limsup_{t \to \infty} L[\boldsymbol{x}(t), t]) \leq \lim_{\zeta \to \infty} \frac{M\dot{c}_{max}}{\zeta p} + \frac{MQ}{\zeta} = 0.$$

In this section, we will focus on the following question: under which conditions, there exists a finite ζ, such that

$$\lim_{t \to \infty} L[\boldsymbol{x}(t), t] = 0.$$

Consider the following time-varying linear constrained problem

$$\max U(\boldsymbol{x}), \quad \text{subject to} \quad s_{i}(\boldsymbol{x}, t) \leq 0, \quad i = 1, ..., M. \tag{5.15}$$

Again, $U(\boldsymbol{x})$ is assumed to be twice differentiable and strictly concave with respect to $\boldsymbol{x}$. Moreover, assume that there exist positive constants p and P, such that for all $\boldsymbol{x}$

$$pI < -H(\boldsymbol{x}) < PI, \quad \text{and} \quad pI < -H^{-1}(\boldsymbol{x}) < PI. \tag{5.16}$$

All constraints $s_{i}(\boldsymbol{x}, t) = g_{i}^{T}\boldsymbol{x} - c_{i}(t) \leq 0$, $i = 1, ..., M$, are linear inequality constraints. Without loss of generality, it is also assumed that

$$\|g_{i}\| = 1, \quad i = 1, ..., M.$$

Recall that $\dot{c}_{max}$ is defined by (5.11) in Section 5.3, and as before, it is assumed that

$$\dot{c}_{max} < \infty. \tag{5.17}$$

A point $\boldsymbol{x}$ being feasible at time t means that $s_{i}(\boldsymbol{x}, t) \leq 0$ for all $i = 1, 2, ..., M$. Due to the fact that $c_{i}(t)$ depends on t, the feasible set depends on time t. We assume that there exists a bounded closed set $\mathcal{X} \subset R^{n}$, such that for any t, the feasible set is contained in $\mathcal{X}$; i.e., for any t

$$\{\boldsymbol{x} : s_{i}(\boldsymbol{x}, t) \leq 0, i = 1, ..., M\} \subset \mathcal{X}. \tag{5.18}$$

The optimal solution of problem (5.15), $\boldsymbol{x}_{opt}(t)$, is unique, bounded, and a continuous function of t. Given the optimal solution $\boldsymbol{x}_{opt}(t)$, let $G_{opt}(t)$ be the matrix such that, the columns of $G_{opt}^{T}(t)$ are the gradients of the constraints which satisfy $s_{i}[\boldsymbol{x}_{opt}(t), t] = 0$.

If $G_{opt}(t)$ has linear dependent rows, the Lagrangian multiplier is not unique. In such a case, let $G_{opt,LI}(t)$ be any matrix with linear independent rows, such that: (i) if g^{T} is a row of $G_{opt,LI}(t)$, then it is also a row of $G_{opt}(t)$; (ii) the column space

of $G^T_{opt,LI}(t)$ equals the column space of $G^T_{opt}(t)$. Then there exists a unique Lagrangian multiplier $\boldsymbol{\lambda}_{opt,LI}(t) > 0$, such that $\nabla U[\boldsymbol{x}_{opt}(t), t] = G^T_{opt,LI}(t)\boldsymbol{\lambda}_{opt,LI}(t)$.

It is also assumed that there exist positive constants $\underline{\lambda}$ and $\bar{\lambda}$, such that for any t, and any $G_{opt,LI}(t)$, the following condition holds

$$0 < \underline{\lambda} < \boldsymbol{\lambda}_{opt,LI}(t) < \bar{\lambda} < \infty. \tag{5.19}$$

For simplicity, from now on, without loss of generality, we only consider the case $G_{opt}(t)$ has linear independent rows for any t, and there is a unique Lagrangian multiplier vector $\boldsymbol{\lambda}(t)$.

The constrained problem (5.15) can be written as the following unconstrained problem

$$\max_{\boldsymbol{x}} \tilde{U}(\boldsymbol{x}, t) = U(\boldsymbol{x}) - \sum_{i=1,...,M} u_i s_i(\boldsymbol{x}, t), \tag{5.20}$$

where

$$s_i(\boldsymbol{x}, t) = g_i^T \boldsymbol{x} - c_i(t), \quad i = 1, ..., M,$$

$$u_i = \begin{cases} \alpha_i, & \text{if } s_i > 0, \\ 0, & \text{if } s_i < 0. \end{cases}$$

Note that, since $U(\boldsymbol{x})$ only depends on $\boldsymbol{x}$, the optimization problem above is a special case of problem (5.10). Assume the following condition holds

$$\alpha_i \geq \bar{\lambda}, \quad i = 1, ..., M, \tag{5.21}$$

The above unconstrained form (5.20) is equivalent to the constrained problem (5.15), in the sense that: for all t

$$\tilde{\boldsymbol{x}}_{opt}(t) = \boldsymbol{x}_{opt}(t)$$

where $\tilde{\boldsymbol{x}}_{opt}(t)$ is the optimum of the unconstrained problem (5.20), and $\boldsymbol{x}_{opt}(t)$ is the optimal solution of problem (5.15).

To solve the unconstrained problem, we apply Sliding Algorithm (5.12). Note that, for this special case, Sliding Algorithm (5.12) has the following special form

$$\dot{\boldsymbol{x}} = -H^{-1}(\boldsymbol{x})\left[\zeta \nabla U(\boldsymbol{x}) - \zeta \mathcal{G}^T u\right]. \tag{5.22}$$

The following theorem establishes the convergence of algorithm (5.22). The proof is given in Appendix C.4, and simulation results are provided in Section 5.5.3.

THEOREM 5.4. *Assume all the conditions (5.16, 5.17, 5.18, 5.19, 5.21) hold. Then there exist $\zeta^* > 0$ and $\alpha^* \geq \bar{\lambda}$ such that if $\zeta > \zeta^*$, and $\alpha_i > \alpha^*$, $i = 1, ...M$, algorithm (5.22) converges; i.e.,*

$$\lim_{t\to\infty} \boldsymbol{x}(t) = \boldsymbol{x}_{opt}(t), \quad \textit{and} \quad \lim_{t\to\infty} L[\boldsymbol{x}(t), t] = 0.$$

5.5 Numerical Examples

In this section, some simulation results are presented, which exemplify the behavior of the algorithms proposed. Note that, similarly to the simulation given in Section 3.5, the implementation of the proposed algorithm is performed in discrete time as well.

5.5.1 Continuous Derivative

We now exemplify the convergence of CFoA (5.3) and CSoA (5.9) with a first example of problem (5.2). The objective function $U(\boldsymbol{x}, t)$ used in this example is

$$U(\boldsymbol{x}, t) = -\frac{1}{2}\frac{1}{10}(t x_{(1)} - 6)^2 - \frac{1}{2}x_{(2)}^2.$$

It can be proven that this objective function satisfies Condition 1. Fig. 5.1 shows the behavior of CFoA. The parameters of the optimization algorithm CFoA are: $K(\boldsymbol{x}) = \zeta H^{-2}(\boldsymbol{x})$, sampling time $1ms$, $\boldsymbol{x}|_{t=1} = \begin{bmatrix} 2 & 0.2 \end{bmatrix}^T$, and $\zeta = 5$. Fig. 5.2 shows the behavior of CSoA. The parameters of the optimization algorithm CSoA are: $K(\boldsymbol{x}) = \zeta I$, sampling time $1ms$, $\boldsymbol{x}|_{t=1} = \begin{bmatrix} 2 & 0.2 \end{bmatrix}^T$, $\zeta = 5$, $q_i(y_i) = 10y_i$, and $\boldsymbol{y}|_{t=1} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$. The simulation results show that both of CFoA and CSoA converge to the time-varying optimum.

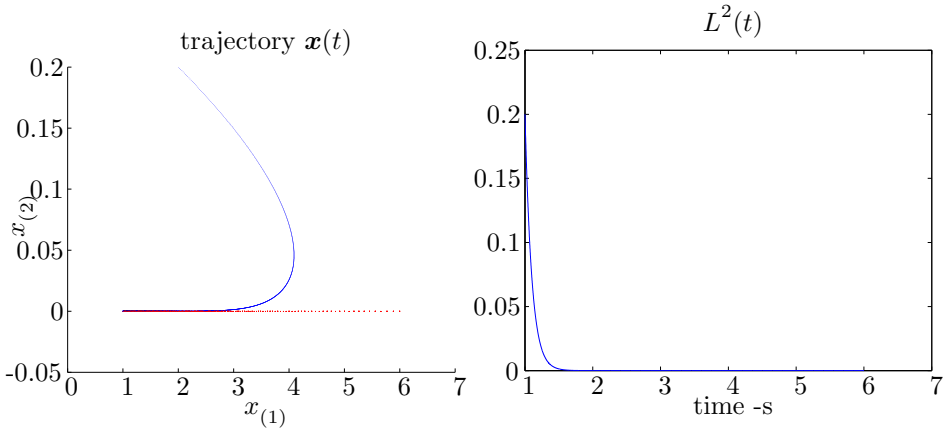


Fig. 5.1. First example of continuous derivative problem: CFoA

As mentioned in Section 5.2.3, the motivation for the study of CSoA is that if the gradient of an objective function changes fast, an implementation of CFoA may have high frequency oscillations. We now provide another example to exemplify this phenomenon and compare the behavior of CFoA and CSoA. Fig. 5.3 shows the simulation results.

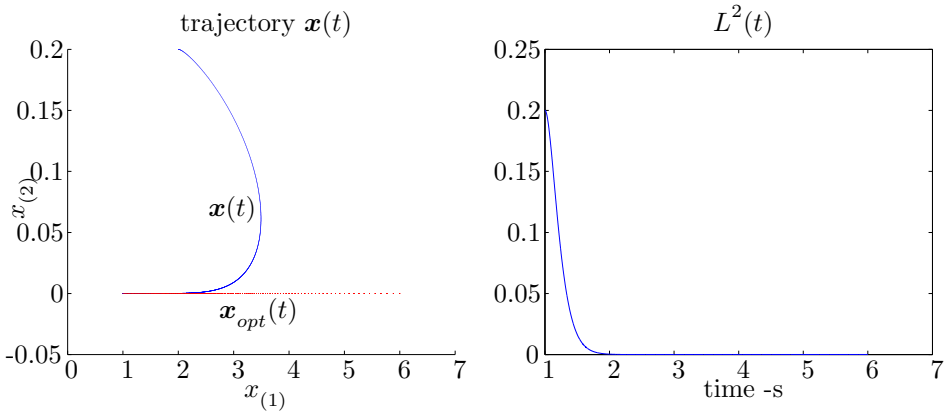


Fig. 5.2. First example of continuous derivative problem: CSoA

The objective function used in this example is

$$U(\boldsymbol{x},t) = -\frac{1}{2}\frac{1}{10}(x_{(1)}-6)^2 - \frac{1}{2}x_{(2)}^2 - f_{\delta,\alpha}(s),$$

where $s = \begin{bmatrix}1 & 1\end{bmatrix}\boldsymbol{x} - c(t) = g^T\boldsymbol{x} - c(t),$

$$c|_{t=0} = 3,$$

$$\dot{c} = \begin{cases} 0.8, & \text{if } t \in [\frac{5}{8}k, \frac{5}{8}(k+1)), \\ -0.8, & \text{if } t \in [\frac{5}{8}(k+1), \frac{5}{8}(k+2)), \end{cases} \quad k = 0, 2, 4, ...$$

$$\alpha = 2,$$

$$\delta = 0.01,$$

$$f_{\delta,\alpha}(s) = \begin{cases} 0, & \text{if } s \le -\delta, \\ \alpha s, & \text{if } s \ge \delta, \\ a_4 s^4 + a_2 s^2 + a_1 s + a_0, & \text{otherwise}, \end{cases}$$

$$a_4 = -1.2500e+005, \quad a_2 = 75, \quad a_1 = 1, \quad a_0 = 0.0037.$$

It can be proven that it satisfies Condition 1, and the behavior of the time-varying parameter $c(t)$ is shown in Fig. 5.3. Note that in this example, the gradient of the objective function changes quite fast when s is "close" to 0; i.e., $-\delta < s < \delta$. The parameters of CFoA and CSoA are chosen as: $K(\boldsymbol{x}) = \zeta H^{-2}(\boldsymbol{x})$ for the CFoA, and $K(\boldsymbol{x}) = \zeta I$ for the CSoA; The sampling time is $1ms$; $\boldsymbol{x}|_{t=0} = \begin{bmatrix} 2.5 & 0.3 \end{bmatrix}^T$; For both CFoA and CSoA, $\zeta = 8$; For CSoA, $q_i(y_i) = 70y_i$ and $\boldsymbol{y}|_{t=0} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$.

The left column of Fig. 5.3 shows the behavior of CFoA, and the right column shows the behavior of CSoA. Although both of them converge to the time-varying optimal solution, one should note that, since the gradient of the objective function changes quite fast when s is "close" to 0, CFoA shows high frequency oscillations when s is "close" to 0, in both trajectories and objective functions, while CSoA has a "smoother" behavior than CFoA.

5.5.2 Discontinuous Derivative

We now provide an example of problem (5.10), which has a discontinuous derivative, to exemplify the behavior of SA (5.12). Fig. 5.4 shows the simulation result. The

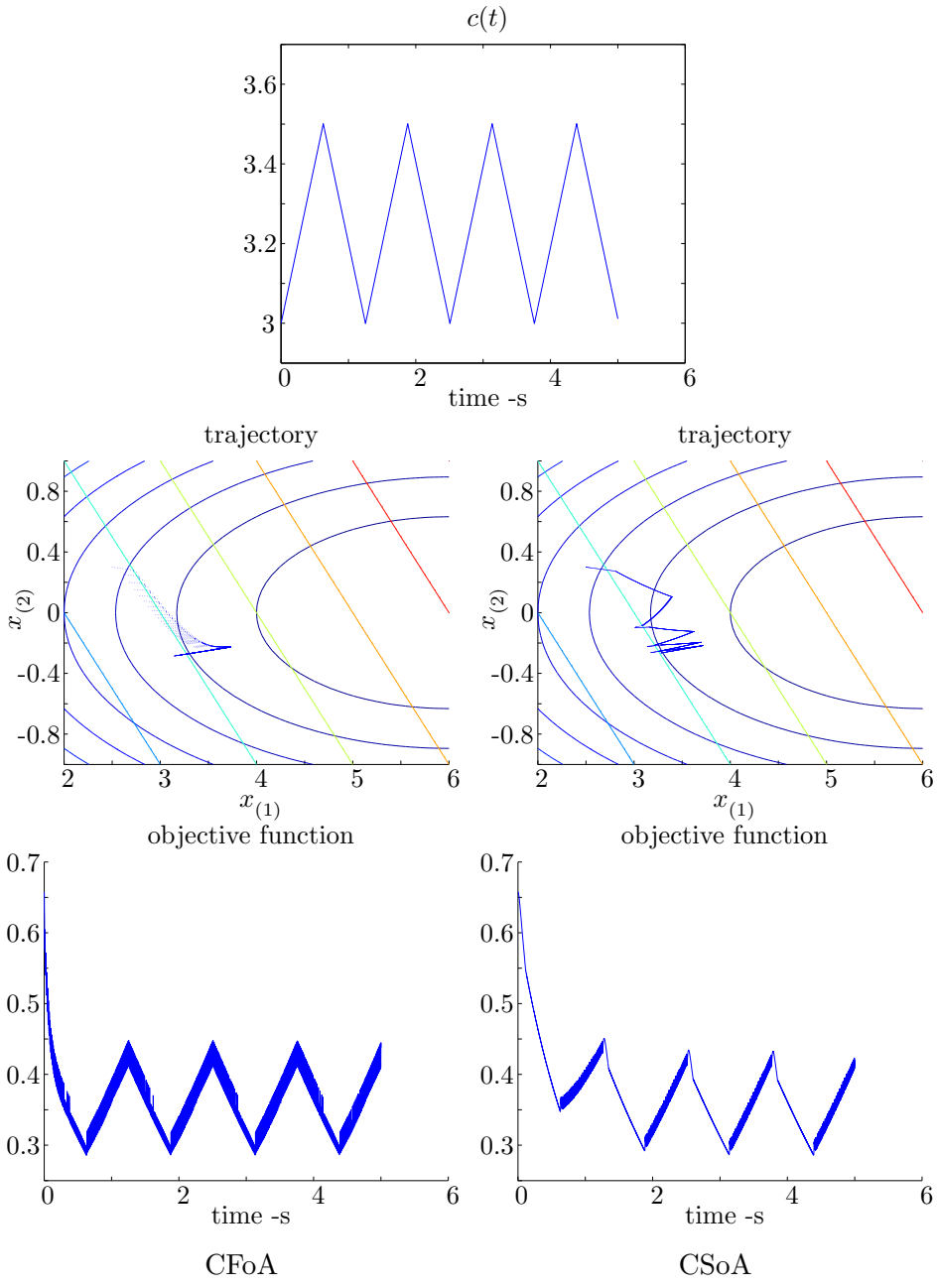


Fig. 5.3. Second example of continuous derivative problem

objective function $U(\boldsymbol{x},t)$ used in this example is

$$U(\boldsymbol{x},t) = -\frac{1}{2}\frac{1}{10}(x_{(1)} - 6)^2 - \frac{1}{2}x_{(2)}^2 - \boldsymbol{u}\boldsymbol{s}(\boldsymbol{x},t),$$

$$\text{where } \boldsymbol{s}(\boldsymbol{x},t) = \begin{bmatrix} s_1(\boldsymbol{x},t) \\ s_2(\boldsymbol{x},t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0.2 & 1 \end{bmatrix} \boldsymbol{x} - \begin{bmatrix} c_1(t) \\ c_2(t) \end{bmatrix},$$

$$u_i = \begin{cases} 1, \text{ if } s_i > 0, \\ \\ 0, \text{ if } s_i < 0, \end{cases} \quad i = 1,2,$$

$$c_1|_{t=0} = c_2|_{t=0} = -0.2,$$

$$\dot{c}_1(t) = \begin{cases} 1, \text{ if } t \in [0.3k, 0.3(k+1)), \\ \\ -1, \text{ if } t \in [0.3(k+1), 0.3(k+2)), \end{cases} \quad k = 0,2,4,...$$

$$\dot{c}_2(t) = \begin{cases} 0.75, \text{ if } t \in [\frac{4}{15}k, \frac{4}{15}(k+1)), \\ \\ -0.75, \text{ if } t \in [\frac{4}{15}(k+1), \frac{4}{15}(k+2)), \end{cases} \quad k = 0,2,4,...$$

It can be proven that this objective function satisfies the conditions of problem (5.10). The behavior of the time-varying parameter $\boldsymbol{c}(t)$ is shown in the left of Fig. 5.4. The parameters are chosen as: sampling time $1ms$, $\boldsymbol{x}|_{t=0} = \begin{bmatrix} -2 & -0.2 \end{bmatrix}^T$ and $\zeta = 4$. The right of Fig. 5.4 shows the behavior of SA (5.12). And as expected, the descent function $L(t)$ converges to a neighborhood of zero, and the Sliding Algorithm converges to a neighborhood of the time-varying optimum.

5.5.3 Time-varying Optimization with Time-varying Linear Constraints

We now provide an example of problem (5.15) to exemplify that the Sliding Algorithm converges to the time-varying optimum for such a problem. Fig. 5.5 shows the

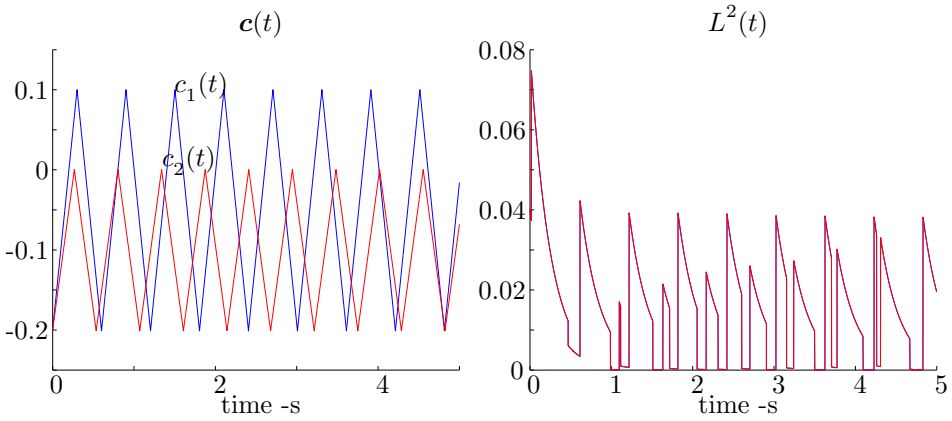


Fig. 5.4. Discontinuous derivative example

simulation results. The optimization problem used in this example is

$$\max U(\boldsymbol{x},t) = -\frac{1}{2}\frac{1}{10}(x_{(1)}-6)^2 - \frac{1}{2}x_{(2)}^2,$$

$$\text{subject to} \quad s(\boldsymbol{x},t) = \begin{bmatrix}1 & 1\end{bmatrix}\boldsymbol{x} - c(t) = g^T\boldsymbol{x} - c(t) \le 0,$$

$$\text{where} \quad c|_{t=0} = 3,$$

$$\dot{c} = \begin{cases} 0.5, & \text{if } t \in [4k, 4k+2), \\ & \qquad\qquad k = 0,1,2,... \\ -0.5, & \text{if } t \in [4k+2, 4k+4), \end{cases}$$

It can be proven that this example satisfies the conditions of problem (5.15). The behavior of the time-varying parameter $c(t)$ is shown in Fig. 5.5. Note that in this example, the Lagrangian Multiplier is strictly lower bounded away from zero. The parameters are chosen as: the sampling time $1ms$, $\boldsymbol{x}|_{t=0} = \begin{bmatrix}2.5 & 0.1\end{bmatrix}^T$, $\alpha = 2$, and $\zeta = 1$. The simulation shows that the descent function $L(t)$ converges to zero, and the Sliding Algorithm converges to the time-varying optimum.

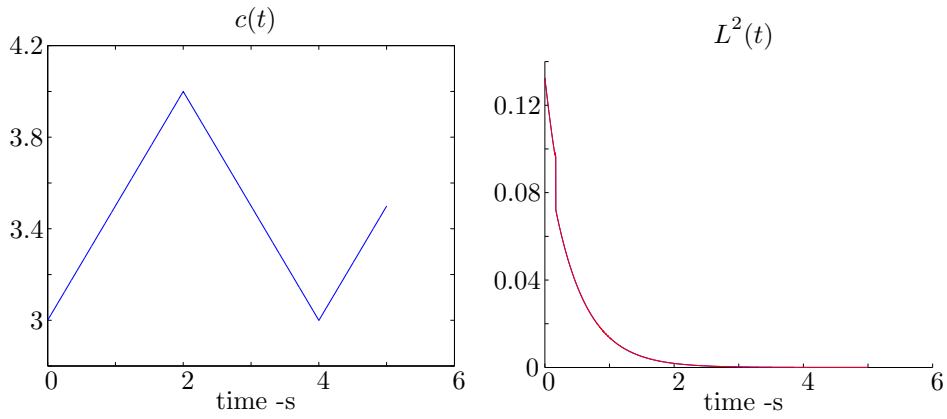


Fig. 5.5. Example of problem with time-varying linear constraints

5.6 Conclusion

This chapter focuses on a class of time-varying optimization problems, and uses the smallest norm of the gradient/subgradient as a descent function. The algorithms designed only require local information of the objective function.

First, for the time-varying optimization problems with twice differentiable strictly convex/concave objective functions (Problem (5.2)), the Continuous First Order Algorithm (CFoA) is proposed. Moreover, in order to achieve "smoother" behavior, the Continuous Second Order Algorithm (CSoA) is also proposed. Both of these algorithms are shown to converge to and track the time-varying optimum. Then, for the time-varying optimization problems with strictly convex/concave objective functions having derivatives with "linear" discontinuity (Problem (5.10)), the Sliding Algorithm (SA) is proposed, which makes the descent function converge to an arbitrarily small neighborhood of zero. Moreover, the SA is applied to time-varying linearly constrained optimization problems, sufficient conditions for asymptotic convergence of the Sliding Algorithm are provided.

Chapter 6

Concluding Remarks

By using the sliding mode control, this dissertation addressed two computer network traffic engineering problems: the multi-domain traffic engineering problem, and the overlay network traffic engineering problem. Due to the fact that the overlay traffic engineering problem is fundamentally a time-varying optimization problem, we also extended our research to more general time-varying optimization problems.

For the multi-domain traffic engineering problem presented in Chapter 3, a family of adaptation control laws for optimal rate adaptation and load balancing in a multi-domain internet was proposed. Moreover, the percentage adaptation laws were developed to further improve the scalability of the control laws. To the best of our knowledge, this family of control laws is the first of its kind that jointly optimizes the traffic allocation at the inter-domain and intra-domain levels. The control laws run at the access points and domain edge nodes only. Moreover, the control laws can work properly with locally inferrable information only (e.g., through exchange of information with its next-hop node only). As a result, the proposed control approach is highly scalable. Simulation results demonstrated that the proposed approach can provide good QoS, TE, and FFR features in a multi-domain environment.

In Chapter 4, the overlay network traffic engineering problem was presented and formulated as a time-varying convex optimization problem. A family of distributed algorithms for optimal traffic allocation under CoS constraints was developed, under the assumption that the utility function is strictly concave. It was shown that the proposed control laws converge to the time-varying optimal solution. And the simulation results showed that the proposed algorithms lead to rather high performance in terms of resource utilization, as measured by utility function.

Moreover, since the overlay traffic engineering problem addressed in Chapter 4 is fundamentally a time-varying optimization problem, we extended our research to more

general time-varying optimization problems. In Chapter 5, we focused on a class of convex objective functions having derivatives with "linear" discontinuity. First, the Continuous First Order Algorithm (CFoA) was proposed for the time-varying optimization problem with a twice differentiable strictly convex/concave objective function. In order to achieve "smoother" dynamics than the CFoA, the Continuous Second Order Algorithm (CSoA) was also proposed. Both the CFoA and CSoA were shown to converge to and track the time-varying optimum. These results serve as a step stone of the algorithm design for a class of time-varying optimization problems with strictly convex/concave objective functions having "linear" discontinuous derivatives. Then, for strictly convex/concave objective functions having derivatives with "linear" discontinuity, the Sliding Algorithm (SA) was proposed. This algorithm is shown to converge to an arbitrarily small neighborhood of the time-varying optimum. Moreover, the SA was applied to time-varying linearly constrained strictly convex optimization problems, and sufficient conditions for asymptotical convergence of the SA were provided. Simulation results were presented to exemplify the behaviors of the algorithms proposed.

Appendix A

Proof of Results in Chapter 3

This Appendix is devoted to present the proofs of Theorem 3.1 and Theorem 3.2 in Chapter 3.

A.1 Preliminary

Define the $n \times n$ matrix $W(\boldsymbol{x}, t)$

$$W(\boldsymbol{x}, t) = diag\big(w_{i}(t, x_{i}, cg(l), r_{i}^{out}), w_{i,b}(t, x_{i,b,l}^{out}, cg(l), r_{i,b}^{in}, r_{i,b,l}^{out}),$$
$$w_{i,b}(t, x_{i,b,l(j)}^{out}, cg[l(j)], r_{i,b}^{in}, r_{i,b,l}^{out})\big).$$

The optimization problem presented in Section 3.2 can be represented in the following standard form

$$\max_{\boldsymbol{x}} U(\boldsymbol{x}), \tag{A.1}$$

subject to the inequality and equality constraints

$$s_{k}(\boldsymbol{x}) = g_{k}^{T}\boldsymbol{x} - c_{k} \leq 0, \quad k = 1, 2, ..., m,$$
$$s_{k}(\boldsymbol{x}) = g_{k}^{T}\boldsymbol{x} - c_{k} = 0, \quad k = m+1, m+2, ..., M,$$

where $U(\boldsymbol{x})$ is a concave differentiable function increasing in each one of its arguments, $s_{k}(\boldsymbol{x})$ are affine functions of $\boldsymbol{x}$ for all $k = 1, 2, ..., M$. Define $\mathcal{C}$ as the feasible set. Moreover, define $\mathcal{G}$ as

$$\mathcal{G}^{T} = \begin{bmatrix} g_{1} & g_{2} & ... & g_{M} \end{bmatrix},$$

and define $\boldsymbol{u}(\boldsymbol{x}) = \begin{bmatrix} u_1(\boldsymbol{x}) & u_2(\boldsymbol{x}) & ... & u_M(\boldsymbol{x}) \end{bmatrix}^T$, an $M \times 1$ vector whose components are of the form

$$u_k(\boldsymbol{x}) = \begin{cases} \alpha_k, & \text{if } s_k(\boldsymbol{x}) > 0, \\ 0, & \text{if } s_k(\boldsymbol{x}) < 0, \end{cases} \qquad k = 1, 2, ..., m,$$

$$u_k(\boldsymbol{x}) = \begin{cases} \alpha_k, & \text{if } s_k(\boldsymbol{x}) > 0, \\ -\alpha_k, & \text{if } s_k(\boldsymbol{x}) < 0, \end{cases} \qquad k = m+1, m+2, ..., M.$$

One states and proves the following theorem.

THEOREM A.1. *[25] If a family of control laws for problem (A.1) is given as*

$$\dot{\boldsymbol{x}} = W(\boldsymbol{x}, t)\big[\nabla U(\boldsymbol{x}) - \mathcal{G}^T \boldsymbol{u}(\boldsymbol{x})\big],$$

and for $k = 1, 2, ..., M$, α_k satisfy the following conditions:

- *for u_k associated with link capacity constraints:* $\alpha_k \geq \alpha^*$
- *for u_k associated with CoS constraints:* $\alpha_k \geq \alpha^*$ (A.2)
- *for u_k associated with flow conservation constraints:* $\alpha_k \geq \alpha^* Cg$

where

$$\alpha^* = \max_i \left| \frac{dU_i}{dx_i} \right|_{x_i = 0},$$

the control laws converge to the optimal resource allocation.

A.2 Proof of Theorem 3.1

LEMMA A.1. *If the conditions in Theorem 3.1 hold, then vector $\boldsymbol{x}$ converges to the feasible set $\mathcal{C}$ in finite time.*

Proof: Let $\boldsymbol{x} \geq 0$, and assume that $\boldsymbol{x}$ is outside the feasible set $\mathcal{C}$ due to some given x_i; i.e., there exists at least one constraint involving x_i is violated. Note that

$$\dot{x}_i = w_i(t, x_i, cg(l), r_i^{out})\Big[h_i(x_i) - (1 - \overline{cg}(l) r_i r_i^{out})\Big],$$

if a link capacity constraint involving x_i is violated (i.e., $\overline{cg}(l) = 0$), such that $\boldsymbol{x}$ is outside the feasible set $\mathcal{C}$, then there exists positive constant $\epsilon > 0$, such that $\dot{x}_i \leq -\epsilon < 0$. Similarly, if a CoS requirement constraint and/or a flow conservation constraint involving x_i is violated, due to the fact that

$$\alpha^* = \max_i \left| \frac{dU_i}{dx_i} \right|_{x_i = 0},$$

one also has that there exists positive constant $\epsilon > 0$, such that $\dot{x}_i \leq -\epsilon < 0$. Note that the same reasoning can be applied to $\dot{x}_{i,b,l}^{out}$ and $\dot{x}_{i,b,l(j)}^{out}$. Therefore, since the derivative is strictly negative outside the feasible set, $\boldsymbol{x}$ reaches the feasible set in finite time.

LEMMA A.2. *The control laws given in Section 3.3 can be expressed as*

$$\dot{\boldsymbol{x}} = \hat{W}(\boldsymbol{x}, t)\big[\nabla U(\boldsymbol{x}) - \mathcal{G}^T \boldsymbol{u}(\boldsymbol{x})\big].$$

Proof: The control laws presented in Section 3.3 can be formulated as follows: For class A control laws,

$$\dot{x}_i = w_i(t, x_i, cg(l), r_i^{out})\Big[h_i(x_i) - (1 - \overline{cg}(l) r_i r_i^{out})\Big],$$

let $\mathcal{K}_i$ be the set of indices $k \in \{1, 2, ..., M\}$ such that the constraints $s_k(\boldsymbol{x})$ involve the data rate x_i. Then,

$$\dot{x}_i = w_i(t, x_i, cg(l), r_i^{out})\left[h_i(x_i) - \left(1 - \prod_{k \in \mathcal{K}_i} \tilde{u}_k\right)\right],$$

where the quantities $\tilde{u}_k$ are defined as follows For AF constraints, let

$$\tilde{u}_k = r_i.$$

For link capacity constraints, let

$$\tilde{u}_k = \overline{cg}(l).$$

For flow conservation constraints, let

$$\tilde{u}_k = r_i^{out}.$$

Now, when $\boldsymbol{x} \in \mathcal{C}$, either $\boldsymbol{x}$ is an inner point of $\mathcal{C}$, or a sliding mode occurs on some surface $s(\boldsymbol{x}) = 0$, where $\boldsymbol{x} \in \partial\mathcal{C}$ (the boundary of $\mathcal{C}$). In the latter case, using the equivalent control method [33], there exists $\tilde{u}_{k,eq}$ such that

$$\dot{x}_i = w_i(\boldsymbol{x}, t)\left[-\big(1 - h_i(x_i)\big) + \prod_{k\in\mathcal{K}_i} \tilde{u}_{k,eq}\right].$$

Moreover, since $\max_{\boldsymbol{x}\in\mathcal{C}} h_i(x_i) = \chi_1 < 1$, and for any $\boldsymbol{x} \in \mathcal{C}$, there exist constant $\chi_2, \chi_3 > 0$ such that

$$\chi_2 < \tilde{u}_{k,\text{eq}} < \chi_3, \quad \forall \boldsymbol{x} \in \mathcal{C}.$$

Hence, given that the logarithm function has a bounded derivative in the closed interval $\left[\min(1-\chi_1, \chi_2^3), \max(1, \chi_3^3)\right]$, the evolution of x_i can be represented as

$$\begin{aligned}
&\dot{x}_i \\
=&\hat{w}_i(t, x_i, cg(l), r_i^{out})\left[\log\frac{1}{1-h_i(x_i)} + \sum_{k\in\mathcal{K}_i}\log\tilde{u}_{k,eq}\right] \\
=&\hat{w}_i(t, x_i, cg(l), r_i^{out})\left[\frac{\partial U}{\partial x_i} + \sum_{k\in\mathcal{K}_i}\log\tilde{u}_{k,eq}\right],
\end{aligned}$$

and there exists $\hat{\zeta} > 0$, such that $\hat{w}_i \geq \hat{\zeta} > 0$. By applying the same reasoning to $\dot{x}^{out}_{i,b,l}$ and $\dot{x}^{out}_{i,b,l(j)}$, we have,

$$\dot{\boldsymbol{x}} = \hat{W}(\boldsymbol{x}, t)\big[\nabla U(\boldsymbol{x}) - \mathcal{G}^T \boldsymbol{u}(\boldsymbol{x})\big].$$

By comparing it with the control laws given in Theorem A.1, if the conditions in Theorem 3.1 hold, the control laws given in Section 3.3 converge to an optimal resource allocation.

A.3 Proof of Theorem 3.2

Since $p_{i,b,l}$ is defined as

$$p_{i,b,l}(t) = \frac{x^{out}_{i,b,l}(t)}{\sum\limits_{\tilde{l}\in\mathcal{L}_{b,i}} x^{out}_{i,b,\tilde{l}}(t)}, \qquad l \in \mathcal{L}_{b,i}.$$

Straightforward computation of the time derivative of $p_{i,b,l}$ yields

$$\frac{dp_{i,b,l}(t)}{dt} = \frac{\dot{x}^{out}_{i,b,l}(t) \sum\limits_{\tilde{l}\in\mathcal{L}_{b,i};\tilde{l}\neq l} p_{i,b,\tilde{l}}(t) - p_{i,b,l}(t) \sum\limits_{\tilde{l}\in\mathcal{L}_{b,i};\tilde{l}\neq l} \dot{x}^{out}_{i,b,\tilde{l}}(t)}{\sum_{l\in\mathcal{L}_{b,i}} x^{out}_{i,b,l}(t)},$$

where the derivatives $\dot{x}^{out}_{i,b,\tilde{l}}$ are computed according to the optimal control laws given in Section 3.4. Now, if the total data rate is strictly positive, the following positive multiplication factor can be introduced

$$w_{i,b}\big(t, x^{out}_{i,b,l}, cg(l), r^{in}_{i,b}, r^{out}_{i}\big) \sum_{l\in\mathcal{L}_{b,i}} x^{out}_{i,b,l}.$$

This factor has only effect to change the adaptation speed but it does not affect the steady state behavior of these laws, since it is strictly positive.

Proceeding in this way and dropping function arguments for notational convenience, the proposed percentage adaptation laws are obtained

$$\dot{p}_{i,b,l} = w_{i,b}\Big(\dot{x}^{out}_{i,b,l} \sum_{\tilde{l}\in\mathcal{L}_{b,i};\tilde{l}\neq l} p_{i,b,\tilde{l}} - p_{i,b,l} \sum_{\tilde{l}\in\mathcal{L}_{b,i};\,\tilde{l}\neq l} \dot{x}^{out}_{i,b,\tilde{l}}\Big).$$

Note that the same reasoning can be applied to $p_{i,b,l(j)}$. Therefore, these laws are equivalent to the convergent data rate adaptation laws for the case of strictly positive total data rate, and thus, they also converge to the optimal data rate of the optimization problem at hand.

Appendix B

Proof of Results in Chapter 4

This Appendix is devoted to present the proof of Theorem 4.1 in Chapter 4.

B.1 Sliding Mode Condition and Equivalent Motion

The algorithm in Section 4.3 can be written as

$$\dot{\boldsymbol{x}} = \zeta Z(\frac{\partial U}{\partial \boldsymbol{x}} - \mathcal{G}^T \boldsymbol{u}),$$

where $Z = -H^{-1}$ is block diagonal, and $\mathcal{G}$ is the gradient of the constraints. Due to the fact that one only has linear constrains, $\mathcal{G}$ is an $M \times n$ constant matrix. Moreover $\boldsymbol{u}$ is an $M \times 1$ vector where M is the number of constraints

$$u_i = \begin{cases} \alpha_i, & \text{at } s_i(\boldsymbol{x}, t) > 0, \\ 0, & \text{at } s_i(\boldsymbol{x}, t) < 0, \end{cases}$$

for $i = 1, 2, ..., M$, α_i satisfy following conditions

$$\begin{aligned} &\text{for link capacity constraints: } \alpha_i \geq \alpha^*, \\ &\text{for CoS constraints: } \alpha_i \geq \alpha^* \max_{i,j} B_{i,j}, \\ &\text{for nonnegative rate constraints: } \alpha_i \geq 2\alpha^* \max_{i,j} B_{i,j}. \end{aligned} \tag{B.1}$$

Sliding mode exists on a discontinuity surface $s(\boldsymbol{x}) = 0$ whenever the following conditions are satisfied

$$\lim_{s \to -0} \dot{s} > 0 \quad \text{and} \quad \lim_{s \to +0} \dot{s} < 0. \tag{B.2}$$

We now prove that the Sliding Mode Condition (B.2) holds for $\boldsymbol{x} \in X_{sm}$.

Since condition (4.3) holds (for simplicity, the index $\boldsymbol{x}$ is omitted), we have

$$\inf_{\boldsymbol{x} \in X_{sm}} ((GZG^T)^{-1}(GZ\nabla U - \frac{1}{\zeta}\dot{\boldsymbol{c}}) > \varphi - \frac{1}{\rho}\Psi n^{1/2}\frac{\rho\varphi}{n^{1/2}\Psi} \geq 0,$$

where $\rho = \frac{\dot{c}_{max} n^{1/2}\Psi}{\varphi}$. This implies

$$(GZG^T)^{-1}(GZ\zeta\nabla U - \dot{\boldsymbol{c}}) > 0. \tag{B.3}$$

If sliding mode occurs on active constraints $\bar{s} = 0$, the equivalent motion is [33]

$$\dot{\boldsymbol{x}}_{eq} = \zeta(Z - Z\bar{G}^T(\bar{G}Z\bar{G}^T)^{-1}\bar{G}Z)\nabla U + Z\bar{G}^T(\bar{G}Z\bar{G}^T)^{-1}\dot{\boldsymbol{c}}.$$

If $\boldsymbol{x}$ encounters a constraint s_g having gradient g at some $\boldsymbol{x} \in X_{sm}$, and g is not a linear combination of $\bar{G}^T$, by condition (B.3)

$$(\begin{bmatrix} \bar{G} \\ g^T \end{bmatrix} Z \begin{bmatrix} \bar{G}^T & g \end{bmatrix})^{-1}(\begin{bmatrix} \bar{G} \\ g^T \end{bmatrix} Z\zeta\nabla U - \begin{bmatrix} \dot{\boldsymbol{c}} \\ \dot{c}_g \end{bmatrix}) > 0,$$

which implies that

$$\begin{bmatrix} * & * \\ -C_2^{-1} g^T Z\bar{G}^T(\bar{G}Z\bar{G}^T)^{-1} & C_2^{-1} \end{bmatrix} \begin{bmatrix} \bar{G}Z\zeta\nabla U - \dot{\boldsymbol{c}} \\ g^T Z\zeta\nabla U - \dot{c}_g \end{bmatrix} > 0,$$

where $C_2 = g^T Zg - g^T Z\bar{G}^T(\bar{G}Z\bar{G}^T)^{-1}GZg > 0$. Then

$$g^T Z\zeta\nabla U - \dot{c}_g - g^T Z\bar{G}^T(\bar{G}Z\bar{G}^T)^{-1}(\bar{G}Z\zeta\nabla U - \dot{\boldsymbol{c}}) > 0.$$

If $s_g < 0$, $u_g = 0$, then

$$\dot{s}_g = g^T\dot{\boldsymbol{x}}_{eq} - \dot{c}_g = g^T\zeta Z(\nabla U - \bar{G}^T(\bar{G}Z\bar{G}^T)^{-1}(\bar{G}Z\nabla U - \frac{1}{\zeta}\dot{\boldsymbol{c}})) - \dot{c}_g > 0.$$

Define

$$zgn(s) = \begin{cases} 0, & s \leq 0, \\ 1, & s > 0, \end{cases}$$

and $zgn(\boldsymbol{s}) = \begin{bmatrix} zgn(s_1) & zgn(s_2) & ... & zgn(s_m) \end{bmatrix}^T$. The control law is

$$\dot{\boldsymbol{x}} = \zeta Z(\nabla U - \bar{G}^T \boldsymbol{u}) = \zeta Z(\nabla U - \alpha \bar{G}^T zgn(\boldsymbol{s})),$$

the equivalent motion is

$$\dot{\boldsymbol{x}}_{eq} = \zeta Z(\nabla U - \bar{G}^T \boldsymbol{u}_{eq}) = \zeta Z(\nabla U - \alpha \bar{G}^T zgn(\boldsymbol{s})_{eq}),$$

also note that $zgn(\boldsymbol{s})_{eq,i} \in [0, 1]$.

If $s_g > 0$, $u_g = \alpha$, condition (4.4) implies

$$\begin{aligned} \dot{s}_g &= g^T \dot{\boldsymbol{x}}_{eq} - \dot{c}_g \\ &= g^T \zeta Z(\nabla U - \begin{bmatrix} \bar{G}^T & g \end{bmatrix} \begin{bmatrix} \alpha zgn(\boldsymbol{s})_{eq}) \\ \alpha \end{bmatrix}) - \dot{c}_g \\ &= \zeta g^T Z(\nabla U - g\alpha) - \alpha \zeta g^T Z \bar{G}^T zgn(\boldsymbol{s})_{\bar{G},eq} - \dot{c}_g \\ &< \rho \min_{\boldsymbol{x}}(Z)(\max_{\boldsymbol{x}}(\nabla U) - \alpha) + \dot{c}_{max} \leq 0, \end{aligned}$$

and hence a sliding mode occurs where $\boldsymbol{x} \in X_{sm}$.

B.2 Convergence

In this section, we will prove that the algorithm given in Section 4.3 converges. To do that, first, based on the optimization problem in Section 4.2, two modified optimization problems are formulated, and two optimization algorithms are proposed; second, the optimization algorithm for the second modified problem is proved to converge; third, the

relationship between the two optimization algorithms are given; finally, the convergence of the algorithm given in Section 4.3 is provided. Without loss of generality, it is assumed that with $\zeta = 1$, the control law is fast enough to follow the time-varying constraints.

B.2.1 Original-problem and Two Modified Problems

The optimization problem given in Section 4.2 is referred to as the Original-problem(OP)

$$\max U(\boldsymbol{x}), \quad \text{subject to} \quad s_i(\boldsymbol{x}) \leq 0, \quad i = 1, ..., M, \tag{B.4}$$

all M constraints $s_i(\boldsymbol{x}, t) = g_i^T \boldsymbol{x} - c_i(t) \leq 0$ are linear inequality constraints.

The First-modified-problem (FMP) is defined as

$$\max \tilde{U}(\boldsymbol{x}, t), \tag{B.5}$$

where

$$\tilde{U}(\boldsymbol{x}, t) = U(\boldsymbol{x}) - \sum_{i=1,...,M} u_i s_i(\boldsymbol{x}) = U(\boldsymbol{x}) - \sum f_i(s_i),$$

$$u_i = \begin{cases} \alpha_i, & \text{if} \quad s_i > 0, \\ 0, & \text{if} \quad s_i < 0, \end{cases} \tag{B.6}$$

the α_i, $i = 1, ..., M$, satisfy condition (C.16). The objective function $\tilde{U}(\boldsymbol{x}, t)$ is continuous, strictly concave. It has a unique time-varying optimal solution $\tilde{\boldsymbol{x}}_{opt}(t)$. Moreover [20, 33]

$$\tilde{\boldsymbol{x}}_{opt}(t) = \boldsymbol{x}_{opt}(t). \tag{B.7}$$

The gradient of $\tilde{U}(\boldsymbol{x},t)$ is given by

$$\nabla\tilde{U}(\boldsymbol{x},t)=\nabla U(\boldsymbol{x})-\mathcal{G}^{T}\boldsymbol{u},\quad\text{where}\quad\forall i,s_{i}(t)\neq 0,$$

the ith column of $\mathcal{G}^{T}$ is g_{i}, the ith element of $\boldsymbol{u}$ is u_{i}. The Hessian matrix $\tilde{H}$ of $\tilde{U}(\boldsymbol{x},t)$ is given by

$$\tilde{H}(t)=H,\quad\text{where}\quad\forall i,s_{i}(t)\neq 0.$$

Note that $\nabla\tilde{U}(\boldsymbol{x},t)$ and $\tilde{H}$ do not exist at $\boldsymbol{x}$, if for some i, $s_{i}(\boldsymbol{x})=0$.

Let $\tilde{Z}=-\tilde{H}^{-1}$, propose the Sliding Algorithm (SA) for the FMP

$$\dot{\boldsymbol{x}}=-\tilde{H}^{-1}\nabla\tilde{U}(\boldsymbol{x},t)=Z[\nabla U(\boldsymbol{x})-\mathcal{G}^{T}\boldsymbol{u}].\tag{B.8}$$

The Sliding Algorithm is not defined where $s_{i}=0$. Note that the Sliding Algorithm is exactly the algorithm given in Section 4.3. The convergence of SA will be provided in next sections.

Now we define the Second-modified-problem (SMP). Let

$$\bar{U}_{\delta}(\boldsymbol{x},t)=U(\boldsymbol{x})-\sum f_{\delta,i}(s_{i})=U(\boldsymbol{x})-f_{\delta}(\boldsymbol{x},\boldsymbol{c}),$$

and for $f_{\delta}(\boldsymbol{x},\boldsymbol{c})=\sum f_{\delta,i}(s_{i})$, $H_{f_{\delta}}\geq 0$ exists

$$f_{\delta,i}(s_{i})=\begin{cases}s_{i}\alpha_{i} & \text{if}\quad s_{i}\geq\delta\\ \\ 0 & \text{if}\quad s_{i}\leq-\delta\end{cases}$$

$$\frac{df_{\delta,i}(s_{i})}{ds_{i}}=\begin{cases}\alpha_{i} & \text{if}\quad s_{i}\geq\delta\\ \\ 0 & \text{if}\quad s_{i}\leq-\delta\end{cases}$$

$$\frac{d^2 f_{\delta,i}(s_i)}{ds_i^2} = \begin{cases} 0 & \text{if } s_i \notin [-\delta, \delta] \\ \geq 0 & \text{if } -\delta \leq s_i \leq \delta \end{cases}$$

The gradient of $\bar{U}_\delta(\boldsymbol{x}, t)$ is given by

$$\nabla \bar{U}_\delta(\boldsymbol{x}, t) = \nabla U(\boldsymbol{x}) - \frac{df_\delta(\boldsymbol{x}, \boldsymbol{c})}{d\boldsymbol{x}} = \nabla U(\boldsymbol{x}) - \sum \frac{df_{\delta,i}(s_i)}{d\boldsymbol{x}}.$$

The Hessian matrix of $\bar{U}_\delta(\boldsymbol{x}, t)$, $\bar{H}_\delta = H - H_{f_\delta} < 0$.

The Second-modified-problem (SMP) is defined as

$$\max \bar{U}_\delta(\boldsymbol{x}, t). \tag{B.9}$$

The objective function of the SMP, $\bar{U}_\delta(\boldsymbol{x}, t)$, is continuous, strictly concave. It has a unique time-varying optimal solution $\bar{\boldsymbol{x}}_{\delta,opt}(t)$.

Let $\bar{Z}_\delta = -\bar{H}_\delta^{-1}$, propose the Cont-algorithm (CA) for the SMP

$$\dot{\bar{\boldsymbol{x}}}_\delta = -\bar{H}_\delta^{-1}[\nabla \bar{U}_\delta(\boldsymbol{x}, t) + \frac{d\nabla \bar{U}_\delta}{dt}] = \bar{Z}_\delta[\nabla U(\boldsymbol{x}) - \frac{df_\delta(\boldsymbol{x}, \boldsymbol{c})}{d\boldsymbol{x}} - \frac{d^2 f_\delta(\boldsymbol{x}, \boldsymbol{c})}{d\boldsymbol{x} dt}]. \tag{B.10}$$

Note that $\lim_{\delta \to 0} \|\bar{U}_\delta(\boldsymbol{x}, t) - \tilde{U}(\boldsymbol{x}, t)\| = 0$, moreover, due to the fact that $\bar{U}_\delta(\boldsymbol{x}, t)$ is continuous, strictly concave, it has a unique time-varying optimal solution $\bar{\boldsymbol{x}}_{\delta,opt}(t)$, moreover, one has

$$\lim_{\delta \to 0} \bar{\boldsymbol{x}}_{\delta,opt}(t) = \tilde{\boldsymbol{x}}_{opt}(t). \tag{B.11}$$

In the next sections, the convergence of the Cont-algorithm is provided, then the Sliding Algorithm is proved to converge.

B.2.2 Convergence of the Cont-algorithm

The SMP is an unconstrained time-varying optimization problem. It has a unique time-varying optimal solution $\bar{\boldsymbol{x}}_{\delta,opt}(t)$. We now prove that the Cont-algorithm $\dot{\bar{\boldsymbol{x}}}_{\delta}$ converges. Propose the following Lyapunov function

$$L_{\delta}(t) = (1/2)\nabla\bar{U}_{\delta}^{T}(\boldsymbol{x},t)\nabla\bar{U}_{\delta}(\boldsymbol{x},t).$$

Due to the fact, which is proved in Section B.1, that $\dot{\boldsymbol{x}}$ is fast to follow the time-varying boundary, the CA $\dot{\bar{\boldsymbol{x}}}_{\delta}$ and the SA $\dot{\boldsymbol{x}}$ are only different in the δ-layer, so $\dot{\bar{\boldsymbol{x}}}_{\delta}$ is fast to follow the time-varying δ-layer. The time-derivative is

$$\begin{aligned}
dL_{\delta}(t)/dt &= \nabla\bar{U}_{\delta}^{T}(\boldsymbol{x},t)\bar{H}_{\delta}\dot{\bar{\boldsymbol{x}}}_{\delta} + \nabla\bar{U}_{\delta}^{T}(\boldsymbol{x},t)\frac{d\nabla\bar{U}_{\delta}}{dt} \\
&= \nabla\bar{U}_{\delta}^{T}(\boldsymbol{x},t)\bar{H}_{\delta}\bar{Z}_{\delta}[\nabla\bar{U}_{\delta}(\boldsymbol{x},t) + \frac{d\nabla\bar{U}_{\delta}}{dt}] + \nabla\bar{U}_{\delta}^{T}(\boldsymbol{x},t)\frac{d\nabla\bar{U}_{\delta}}{dt} \\
&= -\nabla\bar{U}_{\delta}^{T}(\boldsymbol{x},t)\nabla\bar{U}_{\delta}(\boldsymbol{x},t) \leq 0,
\end{aligned}$$

it is zero if and only if $\nabla\bar{U}_{\delta}(\boldsymbol{x},t) = 0$; i.e., $\bar{\boldsymbol{x}}_{\delta}(t) = \bar{\boldsymbol{x}}_{\delta,opt}(t)$. So the CA converges to the optimal solution of the SMP.

B.2.3 Relationship between SA and CA

Assume $\boldsymbol{x}_{\delta}(t)$ is close to m constraints; i.e.,

$$s_i = g_i^T\boldsymbol{x} - c_i \in (-\delta,\delta), \quad i = 1,...,m$$

Let $G^T = \begin{bmatrix} g_1 & ... & g_m \end{bmatrix}$, without loss of generality, assume $m \leq n$, G has linear independent rows, and the first m columns of G are linear independent. Construct an $n \times n$ matrix Ω as the following

$$\Omega = \begin{bmatrix} G_0 & G_0\Lambda \\ 0 & I \end{bmatrix},$$

where $G = \begin{bmatrix} G_0 & G_0\Lambda \end{bmatrix}$, G_0 is an $m \times m$ invertible matrix, Λ is an $m \times (n-m)$ matrix, I is an $(n-m) \times (n-m)$ identity matrix, so Ω is invertible. Do a linear transformation $\boldsymbol{y} = \Omega\boldsymbol{x}$, and we use the index y to distinguish the terms in $\boldsymbol{x}$-space and $\boldsymbol{y}$-space, for example, U is the objective function in $\boldsymbol{x}$-space, and $U_y = U(\Omega^{-1}\boldsymbol{y})$ in $\boldsymbol{y}$-space. The Sliding Algorithm given in $\boldsymbol{y}$-space

$$\dot{\boldsymbol{y}} = Z_y[\nabla U_y(\boldsymbol{y}) - \mathcal{G}_y^T \boldsymbol{u}], \tag{B.12}$$

the Cont-algorithm given in $\boldsymbol{y}$-space

$$\dot{\bar{\boldsymbol{y}}}_\delta = \bar{Z}_{y,\delta}[\nabla U_y(\boldsymbol{y}) - \frac{df_{y,\delta}(\boldsymbol{y},\boldsymbol{c})}{d\boldsymbol{y}} - \frac{d^2 f_{y,\delta}(\boldsymbol{y},\boldsymbol{c})}{d\boldsymbol{y}dt}]. \tag{B.13}$$

Note that the following equation holds for any $\delta > 0$

$$\bar{H}_{y,\delta}\dot{\bar{\boldsymbol{y}}}_\delta + \frac{d\nabla\bar{U}_{y,\delta}}{dt} = -\nabla\bar{U}_{y,\delta}^T(\boldsymbol{y},t),$$

$$H_y\dot{\bar{\boldsymbol{y}}}_\delta - \begin{bmatrix} \frac{d^2 f_{y,\delta,1}}{dy_1^2}\dot{\bar{y}}_1 \\ \vdots \\ \frac{d^2 f_{y,\delta,m}}{dy_m^2}\dot{\bar{y}}_m \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \frac{d^2 f_{y,\delta,1}}{dy_1^2}\dot{c}_1 \\ \vdots \\ \frac{d^2 f_{y,\delta,m}}{dy_m^2}\dot{c}_m \\ \mathbf{0} \end{bmatrix} = -\nabla U_y + \frac{df_{y,\delta}}{d\boldsymbol{y}},$$

also note that $\lim_{\delta\to 0}\frac{d^2 f_{y,\delta,i}}{dy_i^2} = \infty$, one has

$$\lim_{\delta\to 0}\| \begin{bmatrix} \frac{d^2 f_{y,\delta,1}}{dy_1^2}(\dot{\bar{y}}_1 - \dot{c}_1) & \ldots & \frac{d^2 f_{y,\delta,m}}{dy_m^2}(\dot{\bar{y}}_m - \dot{c}_m) & \mathbf{0}^T \end{bmatrix} \| = 0,$$

$$\dot{\bar{\boldsymbol{y}}}_\delta = Z_y[\nabla U_y(\boldsymbol{y}) + \frac{df_{y,\delta}}{d\boldsymbol{y}}] + \mathcal{O}(\delta),$$

that means in the δ-neighborhood of the set of points of discontinuity of the Sliding Algorithm $\dot{\boldsymbol{y}}$, the Cont-algorithm $\dot{\bar{\boldsymbol{y}}}$ differs, for almost all t by not more than some value $\xi(\delta)$, from some mean (with any nonnegative weights) values of the function $\dot{\boldsymbol{y}}$ in the δ-neighborhood of the point $(t, \bar{y}_{\delta}(t))$, and $\lim_{\delta \to 0} \xi = 0$. More specifically,

$$\|\dot{\bar{\boldsymbol{y}}}_{\delta}(t) - \sum_{i=1,...,k} \eta_i \dot{\boldsymbol{y}}_i(t)\| \leq \xi, \tag{B.14}$$

$$\sum_{i=1,...,k} \eta_i = 1, \eta_i > 0, \|\boldsymbol{y}_i(t) - \bar{\boldsymbol{y}}_{\delta}(t)\| \leq \delta, i = 1, ..., k,$$

where the numbers η_i and the vectors $\boldsymbol{y}_i(t)$ may depend arbitrarily on $(t, \bar{\boldsymbol{y}}_{\delta}(t))$.

Moreover, condition (C.3) also holds in $\boldsymbol{x}$-space.

B.2.4 Convergence of the Sliding Algorithm

For any given $\epsilon_1 > 0$, one may construct a sequence $\{\epsilon_i\}_{i \geq 1}$, such that

$$\epsilon_1 > \epsilon_2 > ... > \epsilon_i > \epsilon_{i+1} > ...$$

Since condition (B.11) holds, for $\epsilon_1 > 0$, there exists $\delta_1(\epsilon_1) > 0$, such that if $0 < \delta < \delta_1(\epsilon_1)$

$$\|\bar{\boldsymbol{x}}_{\delta,opt}(t) - \tilde{\boldsymbol{x}}_{opt}(t)\| < \epsilon_1, \quad \forall t, \tag{B.15}$$

Note that the Cont-algorithm $\dot{\bar{\boldsymbol{x}}}_{\delta}$ converges to the time-varying optimal solution $\bar{\boldsymbol{x}}_{\delta,opt}(t)$, then for any initial point $(t_0, \boldsymbol{x}_0)$ and $\delta < \delta_1(\epsilon_1)$, there exists $t_1(\epsilon_1) < \infty$, such that

$$\|\bar{\boldsymbol{x}}_{\delta}(t) - \bar{\boldsymbol{x}}_{\delta,opt}(t)\| < \epsilon_1, \quad t > t_1(\epsilon_1), \tag{B.16}$$

Moreover, for $\epsilon_1 > 0$, and any interval $t_0 \leq t \leq T_1$, where $t_1 < T_1 < \infty$, there exist $\xi_1(\epsilon_1) > 0$ and $\delta_2(\epsilon_1) > 0$, such that if $\delta < \delta_2$, condition (C.3) holds, then with

any same initial point $(t_0, \boldsymbol{x}_0)$ [8], $\bar{\boldsymbol{x}}_\delta(t)$, the motion given by the CA $\dot{\bar{\boldsymbol{x}}}_\delta$, and $\boldsymbol{x}(t)$, the motion given by the SA $\dot{\boldsymbol{x}}(t)$, satisfy

$$\|\bar{\boldsymbol{x}}_\delta(t) - \boldsymbol{x}(t)\| < \epsilon_1, \quad t_0 \le t \le T_1. \tag{B.17}$$

Then by the conditions (B.15, B.16, B.17), one has

$$\|\boldsymbol{x}(t) - \tilde{\boldsymbol{x}}_{opt}(t)\| < 2\epsilon_1, \quad t_1(\epsilon_1) < t \le T_1.$$

Similarly to the above steps, for the sequence $\{\epsilon_i\}$, one may construct a time sequence $\{t_i, T_i\}_{i\ge 1}$

$$t_1 < t_2 < T_1 < t_3 < T_2 < ... < t_i < T_{i-1} < t_{i+1} < T_i < ...$$

such that

$$\|\boldsymbol{x}(t) - \tilde{\boldsymbol{x}}_{opt}(t)\| < 2\epsilon_i, \quad t_i < t \le T_i,$$

which implies

$$\|\boldsymbol{x}(t) - \tilde{\boldsymbol{x}}_{opt}(t)\| < 2\epsilon_i, \quad t_i < t \le t_{i+1},$$

so for $t > t_1$,

$$\|\boldsymbol{x}(t) - \tilde{\boldsymbol{x}}_{opt}(t)\| < 2\epsilon_1, \quad t > t_1.$$

Due to the fact that ϵ_1 can be arbitrarily small, $\boldsymbol{x}(t)$ given by the Sliding Algorithm converges to $\tilde{\boldsymbol{x}}_{opt}(t)$. Note that the Sliding Algorithm is exactly the algorithm given in Section 4.3 and $\tilde{\boldsymbol{x}}_{opt}(t) = \boldsymbol{x}_{opt}(t)$, so the algorithm given in Section 4.3 converges to the time-varying optimal solution $\boldsymbol{x}_{opt}(t)$.

Appendix C

Proof of Results in Chapter 5

This Appendix is devoted to present the proofs of the Theorem 5.1, Theorem 5.2, Theorem 5.3 and Theorem 5.4 in Chapter 5.

C.1 Proof of Theorem 5.1

Use the descent function defined by (5.13)

$$(1/2)L^{2}[\boldsymbol{x}(t),t] = (1/2)\nabla U^{T}[\boldsymbol{x}(t),t]\nabla U[\boldsymbol{x}(t),t].$$

Given the Continuous First Order Algorithm (CFoA) (5.3), the time-derivative of the descent function is

$$\frac{d\left[(1/2)L^{2}[\boldsymbol{x}(t),t]\right]}{dt} = \nabla U^{T}[\boldsymbol{x}(t),t]H(t)\dot{\boldsymbol{x}} + \nabla U^{T}[\boldsymbol{x}(t),t]\frac{\partial \nabla U[\boldsymbol{x}(t),t]}{\partial t}$$

$$= -\nabla U^{T}[\boldsymbol{x}(t),t]H(t)K[\boldsymbol{x}(t)]H(t)\nabla U[\boldsymbol{x}(t),t]$$

$$\leq -zp^{2}\nabla U^{T}[\boldsymbol{x}(t),t]\nabla U[\boldsymbol{x}(t),t]$$

$$\leq 0.$$

This derivative is zero if and only if $\nabla U[\boldsymbol{x}(t),t] = 0$, so $\|\nabla U[\boldsymbol{x}(t),t]\|$ converges to zero.

C.2 Proof of Theorem 5.2

In Continuous Second Order Algorithm (CSoA) (5.9)

$$\dot{\boldsymbol{x}} = -H^{-1}[\boldsymbol{x}(t),t]\left[K[\boldsymbol{x}(t)]\boldsymbol{y}(t) + \frac{\partial \nabla U[\boldsymbol{x}(t),t]}{\partial t}\right],$$

it implies

$$\frac{d}{dt}\nabla U[\boldsymbol{x}(t),t] = H(t)\dot{\boldsymbol{x}} + \frac{\partial \nabla U[\boldsymbol{x}(t),t]}{\partial t} = -K[\boldsymbol{x}(t)]\boldsymbol{y}.$$

Use the vector $\left[\nabla U[\boldsymbol{x}(t),t]^T \quad \boldsymbol{y}^T\right]^T$ as the state variable, the CSoA implies the following algorithm

$$\begin{cases} \dfrac{d}{dt}\nabla U[\boldsymbol{x}(t),t] = -K[\boldsymbol{x}(t)]\boldsymbol{y}, \\ \\ \qquad\qquad \dot{\boldsymbol{y}} = K[\boldsymbol{x}(t)]\nabla U[\boldsymbol{x}(t),t] - q(\boldsymbol{y}). \end{cases}$$

Propose the following descent function

$$V(t) = (1/2)\boldsymbol{y}^T\boldsymbol{y} + (1/2)\nabla U^T[\boldsymbol{x}(t),t]\nabla U[\boldsymbol{x}(t),t].$$

Its time-derivative is

$$dV(t)/dt = \boldsymbol{y}^T\dot{\boldsymbol{y}} + \nabla U^T[\boldsymbol{x}(t),t]\frac{d}{dt}\nabla U[\boldsymbol{x}(t),t] = -\boldsymbol{y}^T q(\boldsymbol{y}).$$

Due to the fact that each $q_i(y_i)$ is a continuous function of y_i which satisfies

$$\begin{aligned} & q_i(0) = 0, \\ & y_i q_i(y_i) > 0, \quad \text{for} \quad y_i \neq 0, \end{aligned}$$

then $dV(t)/dt < 0$ for $\boldsymbol{y} \neq 0$, it is zero if and only if $\boldsymbol{y} = 0$. By LaSalle's theorem [17], one has

$$\begin{aligned} & \lim_{t\to\infty} \boldsymbol{y} = 0, \\ & \lim_{t\to\infty} \dot{\boldsymbol{y}} = K[\boldsymbol{x}(t)]\nabla U[\boldsymbol{x}(t),t] - q(\boldsymbol{y}) = 0. \end{aligned}$$

Note that $K(\boldsymbol{x}) > zI$, one has

$$\lim_{t\to\infty} \nabla U[\boldsymbol{x}(t),t] = 0.$$

C.3 Proof of Theorem 5.3

Now we will provide the proof of Theorem 5.3. The proof has following steps:

- In Appendix C.3.1: (1) Define a Second Modified Problem (SMP), which is an approximation of the original problem (5.10); (2) To the SMP problem, apply the Continuous First Order Algorithm which is also referred as the Continuous Algorithm (CA) throughout the proof; (3) Achieve the convergence of the Continuous Algorithm.

- Given the structure of the Sliding Algorithm, for some time t, $\boldsymbol{x}(t)$ may not be able to "follow" some hyperplane $s_i(\boldsymbol{x}, t) = 0$. A precise definition of such case, called Not Follow Condition (NFC), is given in Appendix C.3.2.

- In Appendix C.3.3, discuss the case NFC does not hold for all time t, show that with the Sliding Algorithm (5.12), the descent function $L[\boldsymbol{x}(t), t]$ converges to zero, moreover, it converges no slower than exponentially.

- In Appendix C.3.4, discuss the case NFC holds, and prove Theorem 5.3.

Throughout this proof, recall that $U(\boldsymbol{x}, t)$ is said to be convex (concave) if $U(\boldsymbol{x}, t)$ is convex (concave) with respect to $\boldsymbol{x}$ at time t, and that $U(\boldsymbol{x}, t)$ is said to be (not be) differentiable if $U(\boldsymbol{x}, t)$ is (is not) differentiable with respect to $\boldsymbol{x}$ at time t. Moreover, define $Z(\boldsymbol{x}, t)$ as

$$Z(\boldsymbol{x}, t) = -H^{-1}(\boldsymbol{x}, t),$$

where $H(\boldsymbol{x}, t)$ is the Hessian matrix of $U(\boldsymbol{x}, t)$ with respect to $\boldsymbol{x}$, correspondingly, if $H(\boldsymbol{x}, t)$ is not defined for some $\boldsymbol{x}$, so is $Z(\boldsymbol{x}, t)$.

Also note that many quantities depend on $(\boldsymbol{x}(t), t)$, $(\boldsymbol{x}, t)$ or t, and for simplicity, unless needed for clarity, the $(\boldsymbol{x}(t), t)$, $(\boldsymbol{x}, t)$ and t are omitted.

C.3.1 Second Modified Problem, Continuous Algorithm and Its Convergence Analysis

Given problem (5.10) defined in Section 5.3, let

$$\bar{U}_{\delta}(\boldsymbol{x},t) = U(\boldsymbol{x},t) - \sum f_{\delta,i}(s_i) = U(\boldsymbol{x},t) - f_{\delta}(\boldsymbol{x},\boldsymbol{c}),$$

where $f_{\delta,i}(s_i)$ is a twice differentiable function which satisfies the following conditions:

$$f_{\delta,i}(s_i) = \begin{cases} s_i\alpha_i, & \text{if } \ s_i \geq \delta, \\ 0, & \text{if } \ s_i \leq -\delta, \end{cases}$$

$$\frac{df_{\delta,i}(s_i)}{ds_i} = \begin{cases} \alpha_i, & \text{if } \ s_i \geq \delta, \\ 0, & \text{if } \ s_i \leq -\delta, \end{cases}$$

$$\frac{d^2 f_{\delta,i}(s_i)}{ds_i^2} = \begin{cases} 0, & \text{if } \ s_i \notin [-\delta,\delta], \\ \geq 0, & \text{if } \ -\delta \leq s_i \leq \delta. \end{cases}$$

The gradient of $\bar{U}_{\delta}(\boldsymbol{x},t)$ is given by

$$\nabla\bar{U}_{\delta}(\boldsymbol{x},t) = \nabla U(\boldsymbol{x},t) - \frac{df_{\delta}(\boldsymbol{x},\boldsymbol{c})}{d\boldsymbol{x}} = \nabla U(\boldsymbol{x},t) - \sum \frac{df_{\delta,i}(s_i)}{d\boldsymbol{x}}.$$

The function $\bar{U}_{\delta}(\boldsymbol{x},t)$ is twice differentiable, and the Hessian matrix of $\bar{U}_{\delta}(\boldsymbol{x},t)$, $\bar{H}_{\delta}(\boldsymbol{x},t) = H(\boldsymbol{x},t) - H_{f_{\delta}}(\boldsymbol{x},t) < 0$.

The Second Modified Problem (SMP) is defined as following

$$\max \bar{U}_{\delta}(\boldsymbol{x},t). \tag{C.1}$$

For any given $\delta > 0$, the objective function of SMP, $\bar{U}_{\delta}(\boldsymbol{x},t)$, satisfies Condition 1. Note that the function $f_{\delta,i}(s_i)$ uniformly converges to the function $f_i(s_i)$ (5.10) as δ goes to zero, and that M is finite, then $\bar{U}_{\delta}(\boldsymbol{x},t)$ converges to $\tilde{U}(\boldsymbol{x},t)$ uniformly as δ goes to zero;

i.e., the SMP is an approximation problem of the form (5.8) defined in Section 5.2.1. One can apply the Continuous First Order Algorithm (5.3) to SMP, which is referred as the Continuous Algorithm (CA) for SMP in the reminder of this proof

$$\dot{\boldsymbol{x}}_{\delta} = -\bar{H}_{\delta}^{-1}(t)\left[\zeta\nabla U[\boldsymbol{x}(t),t] + \frac{\partial\nabla U[\boldsymbol{x}(t),t]}{\partial t} - \zeta\frac{df_{\delta}(\boldsymbol{x},\boldsymbol{c})}{d\boldsymbol{x}} - \partial(\frac{df_{\delta}(\boldsymbol{x},\boldsymbol{c})}{d\boldsymbol{x}})/\partial t\right]. \quad \text{(C.2)}$$

By applying Theorem 5.1, the Continuous Algorithm (C.2) converges; i.e.,

$$\lim_{t\to\infty}\|\nabla\bar{U}_{\delta}[\boldsymbol{x}(t),t]\| = 0.$$

C.3.2 Not Follow Condition (NFC)

In the time-varying optimization problem (5.10), the set of $\boldsymbol{x}$ for which the objective function is not differentiable, is a function of $\boldsymbol{c}(t)$. And, given the structure of SA, $\boldsymbol{x}$ may not be able to "follow" some hyperplane $s_i(\boldsymbol{x},t) = 0$ at some time. In this section, we provide a precise definition of such case referred to as the Not Follow Condition (NFC). Then, in the following sections, we will discuss the cases where NFC holds and does not hold. We start with the following proposition.

PROPOSITION 2. *[8] If in the δ-neighborhood of the set of points of discontinuity of the SA $\dot{\boldsymbol{x}}$, and for an interval $t_0 \le t \le t_1$, the CA $\dot{\boldsymbol{x}}_\delta$ and SA $\dot{\boldsymbol{x}}$ satisfy the following conditions*

$$\begin{cases} (i) \quad \text{for any } \epsilon, \text{ there exists } \delta_0, \text{ such that if } \delta < \delta_0, \\ \qquad \left\| \dot{\boldsymbol{x}}_\delta(t) - \sum_{i=1,\dots,k} \vartheta_i \dot{\boldsymbol{x}}_i(t) \right\| \le \epsilon, \quad \text{for a.e. } t_0 \le t \le t_1 \\ \qquad \sum_{i=1,\dots,k} \vartheta_i = 1, \vartheta_i > 0, \|\boldsymbol{x}_i(t) - \boldsymbol{x}_\delta(t)\| \le \delta, i = 1, \dots, k, \\ \qquad \text{where the numbers } k, \vartheta_i \text{ and the vectors } \boldsymbol{x}_i(t) \\ \qquad \text{may depend arbitrarily on } (t, \boldsymbol{x}_\delta(t)). \\ (ii) \quad \lim_{\epsilon \to 0} \delta_0 = 0 \end{cases} \tag{C.3}$$

Then given any $\xi > 0$, there exist $\delta_1 > 0$, and $\delta_2 > 0$, such that if

$$\delta < \delta_1, \quad \text{and} \quad \|\boldsymbol{x}_\delta(t_0) - \boldsymbol{x}(t_0)\| < \delta_2,$$

one has

$$\|\boldsymbol{x}_\delta(t) - \boldsymbol{x}(t)\| < \xi, \quad t_0 \le t \le t_1.$$

DEFINITION 1. *The Not Follow Condition holds for hyperplane $s_m(\boldsymbol{x}, t) = 0$ at t_m, if:*

- *There exists $dt_1 > 0$, such that, for the interval $[t_m - dt_1, t_m]$, condition (C.3) in Proposition 2 holds.*
- *At time t_m, $s_m[\boldsymbol{x}(t_m), t_m] = 0$.*
- *There exists $dt_2 > 0$, such that, for the interval $(t_m, t_m + dt_2]$, with $\boldsymbol{x}_\delta(t_m) = \boldsymbol{x}(t_m)$,*

$$\lim_{\delta \to 0} \sup_{(t_m, t_m + dt_2]} |s_m[\boldsymbol{x}_\delta(t), t]| = 0, \quad \text{and} \quad s_m[\boldsymbol{x}(t), t] \neq 0.$$

Remark: If NFC 1 holds at t_m for hyperplane $s_m(\boldsymbol{x}, t) = g_m^T \boldsymbol{x}(t) - c_m(t) = 0$, without loss of generality, we only consider the case

$$\dot{c}_m(t_m) > 0. \tag{C.4}$$

The proof for the case $\dot{c}_m(t_m) < 0$ is very similar, and the result is the same.

Throughout the rest of this proof, given $\boldsymbol{x}(t_0)$, a hyperplane $s_i(\boldsymbol{x}, t) = g_i^T \boldsymbol{x} - c_i(t) = 0$ being active at time t_0, means that $s_i(\boldsymbol{x}(t_0), t_0) = g_i^T \boldsymbol{x}(t_0) - c_i(t_0) = 0$. Let the active hyperplane set $G(t)$ be a matrix such that, if $s_i = 0$ is active at t, then g_i^T is a row of $G(t)$. Note that $G(t)$ depends on $\boldsymbol{x}(t)$, and is an $m(t) \times n$ matrix, where $m(t)$ is the number of active hyperplanes. And for simplicity, some times we omit t and use G as the active hyperplane set when there is no ambiguity.

C.3.3 NFC Does Not Hold

In this section, we address the case that NFC 1 does not hold for all t. It shows that if the NFC does not hold for all t, the descent function $L[\boldsymbol{x}(t), t]$ converges to zero no slower than exponentially.

- In Appendix C.3.3.1, more detailed forms of the descent function $L[\boldsymbol{x}, t]$ of problem (5.10) and the descent function $L_\delta[\boldsymbol{x}_\delta, t]$ of SMP (C.1) are provided.
- In Appendix C.3.3.2, a relationship between $L[\boldsymbol{x}, t]$ and $L_\delta[\boldsymbol{x}_\delta, t]$ is given.
- The convergence of SA is established by showing that $L[\boldsymbol{x}(t), t]$ does not increase for all t in Appendix C.3.3.3, and that $L[\boldsymbol{x}(t), t]$ decreases no slower than exponentially at t if $L[\boldsymbol{x}(t), t]$ is differentiable with respect to t in Appendix C.3.3.4.

C.3.3.1 Descent Function $L[\boldsymbol{x}, t]$ of Problem (5.10) and Descent Function $L_\delta[\boldsymbol{x}_\delta, t]$ of SMP (C.1)

The definition of the descent function $L[\boldsymbol{x}, t]$ of problem (5.10) is given by (5.13), here we will recall it and give a more detailed form of it for problem (5.10).

For problem (5.10) if at $\boldsymbol{x}$, $\tilde{U}(\boldsymbol{x},t)$ is differentiable,

$$L(\boldsymbol{x},t) = \|\nabla\tilde{U}(\boldsymbol{x},t)\|.$$

If at $\boldsymbol{x}$, $\tilde{U}(\boldsymbol{x},t)$ is not differentiable, assume that,

$$\begin{aligned} s_{i}(\boldsymbol{x},t) &= g_{i}^{T}\boldsymbol{x} - c_{i}(t) = 0, \quad \text{for } \ i = 1,...,m, \\ s_{i}(\boldsymbol{x},t) &= g_{i}^{T}\boldsymbol{x} - c_{i}(t) \neq 0, \quad \text{for } \ i = m+1,...,M. \end{aligned}$$

Due to the linear structure of $s_{i}(\boldsymbol{x},t)$, $i = 1,...,M$, for problem (5.10), the descent function defined by (5.13) has the following form

$$L(\boldsymbol{x},t) = \min_{\gamma_i\in[0,\alpha_i],\ i=1,...,m} \|\nabla U(\boldsymbol{x},t) - G^{T}\gamma - G^{'T}\boldsymbol{u}'\|, \tag{C.5}$$

where

$$\begin{aligned} G^{T} &= \begin{bmatrix} g_1 & ... & g_m \end{bmatrix}, \\ G^{'T} &= \begin{bmatrix} g_{m+1} & ... & g_M \end{bmatrix}, \\ \gamma &= \begin{bmatrix} \gamma_1 & ... & \gamma_m \end{bmatrix}^{T}, \\ \boldsymbol{u}' &= \begin{bmatrix} u_{m+1} & ... & u_M \end{bmatrix}^{T}, \end{aligned}$$

the u_{i} is defined in problem (5.10).

The descent function $L_{\delta}[\boldsymbol{x}_{\delta},t]$ for SMP (C.1) is

$$L_{\delta}[\boldsymbol{x}_{\delta},t] = \|\nabla\bar{U}_{\delta}(\boldsymbol{x}_{\delta},t)\|.$$

Note that it has been proven in Appendix C.1 that, for any $\delta > 0$, with the Continuous Algorithm (CA) (C.2), $L_{\delta}[\boldsymbol{x}_{\delta}(t),t]$ decreases at any t.

C.3.3.2 Relationship between $L(\boldsymbol{x},t)$ and $L_{\delta}(\boldsymbol{x}_{\delta},t)$

The following Lemma provides a relationship between $L(\boldsymbol{x},t)$ and $L_{\delta}(\boldsymbol{x}_{\delta},t)$.

LEMMA C.1. *Given any $\epsilon > 0$, any $\boldsymbol{x}_0$, and any t, there exist $\xi(\boldsymbol{x}_0) > 0$, and $\delta_0[\xi(\boldsymbol{x}_0)] > 0$ such that for any $\delta < \delta_0[\xi(\boldsymbol{x}_0)]$,*

$$\inf_{\|\boldsymbol{x}_\delta - \boldsymbol{x}_0\| < \xi(\boldsymbol{x}_0)} L_\delta(\boldsymbol{x}_\delta, t) > L(\boldsymbol{x}_0, t) - \epsilon.$$

Proof of Lemma C.1.

1. Given $\boldsymbol{x}_0$, if $\tilde{U}(\boldsymbol{x}, t)$ is not differentiable at $\boldsymbol{x}_0$,

$$L(\boldsymbol{x}_0, t) = \min_{\gamma_i \in [0, \alpha_i],\ i=1,\ldots,m} \|\nabla U(\boldsymbol{x}_0, t) - G^T \gamma - G'^T \boldsymbol{u}'\|,$$

and let $\gamma^{**}(\boldsymbol{x}_0, t)$ be a vector such that[1]

$$\gamma^{**}(\boldsymbol{x}_0, t) \in \{\arg \min_{\gamma_i \in [0, \alpha_i],\ i=1,\ldots,m} \|\nabla U(\boldsymbol{x}_0, t) - G^T \gamma - G'^T \boldsymbol{u}'\|\}.$$

Also note that

$$L_\delta(\boldsymbol{x}_\delta, t) = \|\nabla U(\boldsymbol{x}_\delta, t) - G^T \gamma_\delta(\boldsymbol{x}_\delta, t) - G'^T \gamma'_\delta(\boldsymbol{x}_\delta, t)\|,$$

where $\gamma_\delta(\boldsymbol{x}_\delta, t) = \begin{bmatrix} \gamma_{\delta,1} & \ldots & \gamma_{\delta,m} \end{bmatrix}$, and $\gamma'_\delta(\boldsymbol{x}_\delta, t) = \begin{bmatrix} \gamma_{\delta,m+1} & \ldots & \gamma_{\delta,M} \end{bmatrix}$ for some $\gamma_{\delta,i} \in [0, \alpha_i]$, $i = 1, \ldots M$. Given any $\xi(\boldsymbol{x}_0) > 0$, there exists $\delta_0[\xi(\boldsymbol{x}_0)] > 0$ such that if $\delta < \delta_0[\xi(\boldsymbol{x}_0)]$, one has

$$L_\delta(\boldsymbol{x}_\delta, t) = \|\nabla U(\boldsymbol{x}_\delta, t) - G^T \gamma_\delta(\boldsymbol{x}_\delta, t) - G'^T \boldsymbol{u}'\|, \quad \text{if} \quad \|\boldsymbol{x}_\delta - \boldsymbol{x}_0\| \le \xi(\boldsymbol{x}_0).$$

Let

$$\boldsymbol{x}_\delta^{**}(t) = \arg \min_{\|\boldsymbol{x}_\delta - \boldsymbol{x}_0\| \le \xi(\boldsymbol{x}_0)} L_\delta(\boldsymbol{x}_\delta, t),$$

[1] Note that if G^T has linear independent columns, the argument which minimize $\|\nabla U(\boldsymbol{x}_0, t) - G^T \gamma - G'^T \boldsymbol{u}'\|$ may be not unique.

and $d\nabla U = \nabla U[\boldsymbol{x}_{\delta}^{**}(t), t] - \nabla U(\boldsymbol{x}_0, t)$. Since $-H(t) < PI$ (Condition 1), then there exists $\xi(\boldsymbol{x}_0)$ such that $\|d\nabla U\| < \epsilon$, and one has

$$\begin{aligned}
&\inf_{\|\boldsymbol{x}_\delta - \boldsymbol{x}_0\| < \xi(\boldsymbol{x}_0)} L_\delta(\boldsymbol{x}_\delta, t) \\
\geq &\min_{\|\boldsymbol{x}_\delta - \boldsymbol{x}_0\| \leq \xi(\boldsymbol{x}_0)} L_\delta(\boldsymbol{x}_\delta, t) \\
= &\|\nabla U[\boldsymbol{x}_{\delta}^{**}(t), t] - G^T \gamma_\delta[\boldsymbol{x}_{\delta}^{**}(t), t] - G'^T \boldsymbol{u}'\|
\end{aligned}$$

$$\begin{aligned}
= &\|\nabla U(\boldsymbol{x}_0, t) - G^T \gamma_\delta[\boldsymbol{x}_{\delta}^{**}(t), t] - G'^T \boldsymbol{u}' + d\nabla U\| \\
\geq &\|\nabla U(\boldsymbol{x}_0, t) - G^T \gamma_\delta[\boldsymbol{x}_{\delta}^{**}(t), t] - G'^T \boldsymbol{u}'\| - \|d\nabla U\| \\
\geq &\min_{\gamma_i \in [0, \alpha_i],\ i=1,\ldots,m} \|\nabla U(\boldsymbol{x}_0, t) - G^T \gamma - G'^T \boldsymbol{u}'\| - \|d\nabla U\| \\
= &\|\nabla U(\boldsymbol{x}_0, t) - G^T \gamma^{**}(\boldsymbol{x}_0, t) - G'^T \boldsymbol{u}'\| - \|d\nabla U\| \\
> &L(\boldsymbol{x}_0, t) - \epsilon.
\end{aligned}$$

2. If $\tilde{U}(\boldsymbol{x}, t)$ is differentiable at $\boldsymbol{x}_0$, there exists $\xi(\boldsymbol{x}_0) > 0$ and $\delta_0[\xi(\boldsymbol{x}_0)] > 0$, such that if $\delta < \delta_0[\xi(\boldsymbol{x}_0)]$, one has

$$L_\delta(\boldsymbol{x}_\delta, t) = L(\boldsymbol{x}_\delta, t), \quad \text{if} \quad \|\boldsymbol{x}_\delta - \boldsymbol{x}_0\| < \xi(\boldsymbol{x}_0),$$

moreover

$$\inf_{\|\boldsymbol{x}_\delta - \boldsymbol{x}_0\| < \xi(\boldsymbol{x}_0)} L_\delta(\boldsymbol{x}_\delta, t) > L(\boldsymbol{x}_0, t) - \epsilon.$$

By the above two steps, one has

$$\inf_{\|\boldsymbol{x}_\delta - \boldsymbol{x}_0\| < \xi(\boldsymbol{x}_0)} L_\delta(\boldsymbol{x}_\delta, t) > L(\boldsymbol{x}_0, t) - \epsilon.$$

C.3.3.3 Convergence of SA - $L[\boldsymbol{x}(t), t]$ does not increase for all t

We now show that $L[\boldsymbol{x}(t), t]$ is a non-increasing function for all t if the NFC does not hold. Proceeding by contradiction, assume the function $L[\boldsymbol{x}(t), t]$ increases at some time t_0; i.e., there exist $dt > 0$ and $\epsilon > 0$ such that

$$L[\boldsymbol{x}(t_0 + dt), t_0 + dt] = L[\boldsymbol{x}(t_0), t_0] + \epsilon.$$

1. If at $\boldsymbol{x}(t_0)$, $\tilde{U}(\boldsymbol{x}, t)$ is differentiable, there exist $\xi[\boldsymbol{x}(t_0)] > 0$ and $\delta_a\left[\xi[\boldsymbol{x}(t_0)]\right] > 0$, such that if $\delta < \delta_a\left[\xi[\boldsymbol{x}(t_0)]\right]$, one has

$$L_\delta(\boldsymbol{x}_\delta, t_0) = L(\boldsymbol{x}_\delta, t_0), \quad \text{if} \quad \|\boldsymbol{x}_\delta - \boldsymbol{x}(t_0)\| < \xi[\boldsymbol{x}(t_0)].$$

   Due to the fact that $L_\delta(\boldsymbol{x}_\delta, t)$ is continuous with respect to $\boldsymbol{x}_\delta$, one can pick $\xi[\boldsymbol{x}(t_0)]$, such that

$$L_\delta(\boldsymbol{x}_\delta, t_0) < L[\boldsymbol{x}(t_0), t_0] + \epsilon/3, \quad \text{if} \quad \|\boldsymbol{x}_\delta - \boldsymbol{x}(t_0)\| < \xi[\boldsymbol{x}(t_0)].$$

   Hence, one may pick some $\boldsymbol{x}_\delta(t_0)$ such that

$$\|\boldsymbol{x}_\delta(t_0) - \boldsymbol{x}(t_0)\| < \xi[\boldsymbol{x}(t_0)], \quad \text{and} \quad L_\delta[\boldsymbol{x}_\delta(t_0), t_0] < L[\boldsymbol{x}(t_0), t_0] + \epsilon/3.$$

2. If at $\boldsymbol{x}(t_0)$, $\tilde{U}(\boldsymbol{x}, t)$ is not differentiable, at time t_0, $L[\boldsymbol{x}(t_0), t_0]$ has the form (C.5), and one has

$$\gamma^{**}[\boldsymbol{x}(t_0), t_0] \in \{\arg \min_{\gamma_i \in [0, \alpha_i],\ i=1,\ldots,m} \|\nabla U(\boldsymbol{x}(t_0), t_0) - G^T \gamma - G'^T \boldsymbol{u}'\|\}.$$

   Since for all t, $pI < -H(t) < PI$ (Condition 1), given $\epsilon/3$, there exists $\xi[\boldsymbol{x}(t_0)] > 0$, such that

$$\|\nabla U(\boldsymbol{x}_\delta, t_0) - \nabla U[\boldsymbol{x}(t_0), t_0]\| < \epsilon/3, \quad \text{if} \quad \|\boldsymbol{x}_\delta - \boldsymbol{x}(t_0)\| < \xi[\boldsymbol{x}(t_0)],$$

and there exists $\delta_b\left[\xi[\boldsymbol{x}(t_0)]\right] > 0$, such that if $\delta < \delta_b\left[\xi[\boldsymbol{x}(t_0)]\right]$, there exists some $\boldsymbol{x}_\delta(t_0)$ satisfying

$$\|\boldsymbol{x}_\delta(t_0) - \boldsymbol{x}(t_0)\| < \xi[\boldsymbol{x}(t_0)],$$
$$\nabla\bar{U}[\boldsymbol{x}_\delta(t_0), t_0] = \nabla U[\boldsymbol{x}_\delta(t_0), t_0] - G^T\gamma^{**}[\boldsymbol{x}(t_0), t_0] - G'^T\boldsymbol{u}'.$$

Then

$$\begin{aligned}
&L_\delta[\boldsymbol{x}_\delta(t_0), t_0] \\
=&\|\nabla\bar{U}[\boldsymbol{x}_\delta(t_0), t_0]\| \\
=&\|\nabla U[\boldsymbol{x}_\delta(t_0), t_0] - G^T\gamma^{**}[\boldsymbol{x}(t_0), t_0] - G'^T\boldsymbol{u}'\| \\
=&\|\nabla U[\boldsymbol{x}(t_0), t_0] - G^T\gamma^{**}[\boldsymbol{x}(t_0), t_0] - G'^T\boldsymbol{u}' + \nabla U[\boldsymbol{x}_\delta(t_0), t_0] - \nabla U[\boldsymbol{x}(t_0), t_0]\| \\
<&\|\nabla U[\boldsymbol{x}(t_0), t_0] - G^T\gamma^{**}[\boldsymbol{x}(t_0), t_0] - G'^T\boldsymbol{u}'\| + \epsilon/3 \\
=&L[\boldsymbol{x}(t_0), t_0] + \epsilon/3.
\end{aligned}$$

So at time t_0, for any given ϵ, there exists $\xi_1[\boldsymbol{x}(t_0)] > 0$, and $\delta_1\left[\xi_1[\boldsymbol{x}(t_0)]\right]$, such that if $\delta < \delta_1\left[\xi_1[\boldsymbol{x}(t_0)]\right]$, there exists $\boldsymbol{x}_\delta(t_0)$ satisfying

$$\|\boldsymbol{x}_\delta(t_0) - \boldsymbol{x}(t_0)\| < \xi_1[\boldsymbol{x}(t_0)],$$
$$L_\delta[\boldsymbol{x}_\delta(t_0), t_0] < L[\boldsymbol{x}(t_0), t_0] + \epsilon/3.$$

At time $t_0 + dt$, by applying Lemma C.1, given $\epsilon/3$, for all $\boldsymbol{x}(t_0 + dt)$, there exist $\xi_2[\boldsymbol{x}(t_0 + dt)]$ and $\delta_2\left[\xi_2[\boldsymbol{x}(t_0 + dt)]\right]$, such that if $\delta < \delta_2\left[\xi_2[\boldsymbol{x}(t_0 + dt)]\right]$,

$$\inf_{\|\boldsymbol{x}_\delta - \boldsymbol{x}(t_0+dt)\| < \xi_2[\boldsymbol{x}(t_0+dt)]} L_\delta(\boldsymbol{x}_\delta, t_0 + dt) > L[\boldsymbol{x}(t_0 + dt), t_0 + dt] - \epsilon/3.$$

Apply Proposition (2), so that given $\xi_2[\boldsymbol{x}(t_0+dt)]$, there exist δ_3, $\xi_1[\boldsymbol{x}(t_0)]$, and $\delta_1\left[\xi_1[\boldsymbol{x}(t_0)]\right]$, such that if $\delta<\delta_3$ and $\|\boldsymbol{x}_\delta(t_0)-\boldsymbol{x}(t_0)\|<\xi_1[\boldsymbol{x}(t_0)]$ [8],

$$\|\boldsymbol{x}_\delta(t_0+dt)-\boldsymbol{x}(t_0+dt)\|<\xi_2[\boldsymbol{x}(t_0+dt),t_0+dt].$$

Let $\delta=\min(\delta_1,\delta_2,\delta_3)$, then

$$\begin{aligned}
&L_\delta[\boldsymbol{x}_\delta(t_0+dt),t_0+dt]\\
>&L[\boldsymbol{x}(t_0+dt),t_0+dt]-\epsilon/3\\
=&L[\boldsymbol{x}(t_0),t_0]+\epsilon-\epsilon/3\\
=&L[\boldsymbol{x}(t_0),t_0]+(2/3)\epsilon,
\end{aligned}$$

and

$$L_\delta[\boldsymbol{x}_\delta(t_0),t_0]<L[\boldsymbol{x}(t_0),t_0]+\epsilon/3,$$

which implies that $L_\delta[\boldsymbol{x}_\delta(t),t]$ increases at t_0. This contradicts the fact that $L_\delta[\boldsymbol{x}_\delta(t),t]$ is a decreasing function of t for any $\delta>0$. Hence one concludes that $L[\boldsymbol{x}(t),t]$ is a non-increasing function of t.

C.3.3.4 Convergence of SA - $L[\boldsymbol{x}(t),t]$ decreases no slower than exponentially at t if $L[\boldsymbol{x}(t),t]$ is differentiable with respect to t

Now we assume that for some time interval, the set of active hyperplanes is invariant; i.e., $\boldsymbol{x}$ is sliding on hyperplanes $s_i=0$, $i=1,2,...,m$, which have gradient $G^T=\begin{bmatrix}g_1 & ... & g_m\end{bmatrix}$, and the rest hyperplanes $s_i=0$, $i=m+1,m+2,...,M$, have gradient $G'^T=\begin{bmatrix}g_{m+1} & ... & g_M\end{bmatrix}$. By [33] the equivalent motion is

$$\dot{\boldsymbol{x}}_{eq}=Z(t)\left[\zeta\nabla U[\boldsymbol{x}(t),t]-\zeta G^T\boldsymbol{u}_{eq}-\zeta G'^T\boldsymbol{u}'+\frac{\partial\nabla U[\boldsymbol{x}(t),t]}{\partial t}\right],$$

where $\boldsymbol{u}_{eq} = \left[u_{1,eq}, u_{2,eq}, ..., u_{m,eq}\right]$, and $u_{i,eq} \in [0, \alpha_i]$, $i = 1, 2, ..., m$. Note that

$$L[\boldsymbol{x}(t), t] = \min_{\gamma_i \in [0,\alpha_i],\ i=1,...,m} \|\nabla U[\boldsymbol{x}(t), t] - G^T \gamma - G'^T \boldsymbol{u}'\|,$$

let

$$\Gamma^{**}[\boldsymbol{x}(t), t] = \{\arg \min_{\gamma_i \in [0,\alpha_i],\ i=1,...,m} \|\nabla U[\boldsymbol{x}(t), t] - G^T \gamma - G'^T \boldsymbol{u}'\|\},$$

let vector $\gamma^{**}[\boldsymbol{x}(t), t] \in \Gamma^{**}[\boldsymbol{x}(t), t]$ be

$$\gamma^{**}[\boldsymbol{x}(t), t] = \left[\gamma_1^{**} \quad \gamma_2^{**} \quad ... \quad \gamma_m^{**}\right],$$

and let k be

$$k = \min_{\gamma^{**}[\boldsymbol{x}(t),t] \in \Gamma^{**}[\boldsymbol{x}(t),t]} card(\{\gamma_i^{**} : \quad \gamma_i^{**} \in (0, \alpha_i], i = 1, ..., m\}), \tag{C.6}$$

the k is the smallest number of positive entries in any solution $\gamma^{**}[\boldsymbol{x}(t), t]$, and we only consider the solutions with k positive entries. For a given such solution $\gamma^{**}[\boldsymbol{x}(t), t]$, without loss of generality

$$\gamma_i^{**} \in (0, \alpha_i), \quad i = 1, 2, ..., k_+,$$

$$\gamma_i^{**} = \alpha_i, \quad i = k_+ + 1, k_+ + 2, ..., k,$$

$$\gamma_i^{**} = 0, \quad i = k + 1, k + 2, ..., m,$$

then G^T can be rewritten as

$$G^T = \left[G_+^T \quad G_+'^T \quad G_-^T\right], \tag{C.7}$$

where the columns of G_+^T are the gradients of $s_i = 0$, $i = 1, 2, ..., k_+$, the columns of $G_+^{'T}$ are the gradients of $s_i = 0$, $i = k_+ + 1, k_+ + 2, ..., k$, and columns of G_-^T are the gradients of $s_i = 0$, $i = k + 1, k + 2, ..., m$.

LEMMA C.2. *There exists at least one solution* $\gamma^{**}[\boldsymbol{x}(t), t]$ *such that* G_+^T *has linear independent columns.*

Proof of Lemma C.2. Assume that $\gamma^{**}[\boldsymbol{x}(t), t]$ is a solution such that G_+^T has linear dependent columns. Without loss of generality, assume that

$$g_{k_+} = \sum_{i=1,...,k_+-1} g_i \beta_i.$$

According to

$$G^T = \begin{bmatrix} G_+^T & G_+^{'T} & G_-^T \end{bmatrix},$$

the vector $\gamma^{**}[\boldsymbol{x}(t), t]$ can be written as

$$\gamma^{**T}[\boldsymbol{x}(t), t] = \begin{bmatrix} \gamma_+^{**T} & \gamma_+^{**'T} & \gamma_-^{**T} \end{bmatrix}.$$

For simplicity, in the rest of the proof of Lemma C.2, let $\gamma = \gamma_+^{**}$. Then

$$\begin{aligned} G_+^T \gamma &= \sum_{i=1,...,k_+} g_i \gamma_i \\ &= g_{k_+} \gamma_{k_+} + \theta g_{k_+} (\alpha_{k_+} - \gamma_{k_+}) - \theta (\alpha_{k_+} - \gamma_{k_+}) (\sum_{i=1,...,k_+-1} g_i \beta_i) + \sum_{i=1,...,k_+-1} g_i \gamma_i \\ &= g_{k_+} [\theta (\alpha_{k_+} - \gamma_{k_+}) + \gamma_{k_+}] + \sum_{i=1,...,k_+-1} g_i [\gamma_i - \theta \beta_i (\alpha_{k_+} - \gamma_{k_+})], \end{aligned}$$

where $\theta \geq 0$ is a scalar. Let

$$\hat{\theta} = \min\{\frac{r_i - \alpha_i}{\beta_i (\alpha_{k_+} - \gamma_{k_+})}, 1 : i = 1, ..., k_+ - 1\},$$

then

$$\gamma_{new} = \left[\gamma_1 - \hat{\theta}\beta_1(\alpha_{k_+} - \gamma_{k_+}) \quad ... \quad \gamma_{k_+-1} - \hat{\theta}\beta_{k_+-1}(\alpha_{k_+} - \gamma_{k_+}) \quad \hat{\theta}(\alpha_{k_+} - \gamma_{k_+}) + \gamma_{k_+}\right]$$

is also a part of a solution with respect to G^T_+. Note that $\hat{\theta} \in (0, 1]$, it is the smallest θ such that there is at least one $\hat{i}$ satisfying

$$\gamma_{\hat{i}} - \theta\beta_{\hat{i}}(\alpha_{k_+} - \gamma_{k_+}) = \alpha_{\hat{i}}.$$

Remark: due to the fact that γ has the smallest cardinality (C.6), so there is no $\theta \in (0, \hat{\theta}]$ such that for some i, $r_{new,i} = 0$.

So for γ_{new}, there is at least one $\gamma_{new,\hat{i}} = \alpha_{\hat{i}}$, $\hat{i} \in \{1, ..., k_+\}$. With the $\gamma^{**T}_{+,new} = \gamma^T_{new}$, one has γ^{**}_{new}, $G^T_{+,new}$, and $G'^T_{+,new}$, note that $G^T_{+,new}$ has less columns than G^T_+. Then, by repeating the above procedure, one will get a solution such that G^T_+ has linear independent columns.

Now, by applying Lemma C.2, there is one solution $\gamma^{**}[\boldsymbol{x}(t), t]$, such that G^T can be written as the following partition where G^T_+ has linear independent columns

$$G^T = \begin{bmatrix} G^T_+ & G'^T_+ & G^T_- \end{bmatrix},$$

correspondingly

$$\gamma^{**T}[\boldsymbol{x}(t), t] = \begin{bmatrix} \gamma^{**T}_+ & \gamma^{**'T}_+ & \gamma^{**T}_- \end{bmatrix},$$
$$\boldsymbol{u}^T_{eq} = \begin{bmatrix} \boldsymbol{u}^T_{eq,+} & \boldsymbol{u}'^T_{eq,+} & \boldsymbol{u}^T_{eq,-} \end{bmatrix}.$$

Moreover G'^T_+ and G^T_- can be written as

$$G'^T_+ = G^T_+\Lambda'_+ + \bar{G}'^T_+, \quad G^T_- = G^T_+\Lambda_- + \bar{G}^T_-,$$

such that $G_{+}\bar{G}^{'T}_{+} = 0$ and $G_{+}\bar{G}^{T}_{-} = 0$. Now

$$\left[(1/2)L^{2}[\boldsymbol{x}(t),t]\right] = (1/2)\left[\nabla U[\boldsymbol{x}(t),t] - G^{'T}_{+}\gamma^{**'}_{+} - G^{'T}\boldsymbol{u}'\right]^{T}(I - G^{T}_{+}(G_{+}G^{T}_{+})^{-1}G_{+})$$
$$\left[\nabla U[\boldsymbol{x}(t),t] - G^{'T}_{+}\gamma^{**'}_{+} - G^{'T}\boldsymbol{u}'\right],$$

$$\begin{aligned}
&d\left[(1/2)L^{2}[\boldsymbol{x}(t),t]\right]/dt \\
&= \left[\nabla U[\boldsymbol{x}(t),t] - G^{'T}_{+}\gamma^{**'}_{+} - G^{'T}\boldsymbol{u}'\right]^{T}(I - G^{T}_{+}(G_{+}G^{T}_{+})^{-1}G_{+})[H(t)\dot{\boldsymbol{x}}_{eq} + \frac{\partial \nabla U[\boldsymbol{x}(t),t]}{\partial t}] \\
&= \left[\nabla U - G^{'T}_{+}\gamma^{**'}_{+} - G^{'T}\boldsymbol{u}'\right]^{T}(I - G^{T}_{+}(G_{+}G^{T}_{+})^{-1}G_{+}) \\
&\quad \left[H(t)Z(t)\left[\zeta\nabla U - \zeta G^{T}\boldsymbol{u}_{eq} - \zeta G^{'T}\boldsymbol{u}' + \frac{\partial \nabla U[\boldsymbol{x}(t),t]}{\partial t}\right] + \frac{\partial \nabla U[\boldsymbol{x}(t),t]}{\partial t}\right] \\
&= \left[\nabla U - G^{'T}_{+}\gamma^{**'}_{+} - G^{'T}\boldsymbol{u}'\right]^{T}(I - G^{T}_{+}(G_{+}G^{T}_{+})^{-1}G_{+}) \\
&\quad \left[-\zeta\nabla U + \zeta G^{T}\boldsymbol{u}_{eq} + \zeta G^{'T}\boldsymbol{u}' - \frac{\partial \nabla U[\boldsymbol{x}(t),t]}{\partial t} + \frac{\partial \nabla U[\boldsymbol{x}(t),t]}{\partial t}\right] \\
&= -\zeta\left[\nabla U - G^{'T}_{+}\gamma^{**'}_{+} - G^{'T}\boldsymbol{u}'\right]^{T}(I - G^{T}_{+}(G_{+}G^{T}_{+})^{-1}G_{+})\left[\nabla U - G^{'T}_{+}\gamma^{**'}_{+} - G^{'T}\boldsymbol{u}'\right] \\
&\quad + \zeta\left[\nabla U - G^{'T}_{+}\gamma^{**'}_{+} - G^{'T}\boldsymbol{u}'\right]^{T}(I - G^{T}_{+}(G_{+}G^{T}_{+})^{-1}G_{+}) \\
&\quad \left[G^{T}_{+}\boldsymbol{u}_{eq,+} + G^{'T}_{+}\boldsymbol{u}'_{eq,+} - G^{'T}_{+}\gamma^{**'}_{+} + G^{T}_{-}\boldsymbol{u}_{eq,-}\right] \\
&= -\zeta\left[\nabla U - G^{'T}_{+}\gamma^{**'}_{+} - G^{'T}\boldsymbol{u}'\right]^{T}(I - G^{T}_{+}(G_{+}G^{T}_{+})^{-1}G_{+})\left[\nabla U - G^{'T}_{+}\gamma^{**'}_{+} - G^{'T}\boldsymbol{u}'\right] \\
&\quad + \zeta\left[\nabla U - G^{'T}_{+}\gamma^{**'}_{+} - G^{'T}\boldsymbol{u}'\right]^{T}(I - G^{T}_{+}(G_{+}G^{T}_{+})^{-1}G_{+}) \\
&\quad \left[G^{T}_{+}\boldsymbol{u}_{eq,+} + (G^{T}_{+}\Lambda'_{+} + \bar{G}^{'T}_{+})(\boldsymbol{u}'_{eq,+} - \gamma^{**'}_{+}) + (G^{T}_{+}\Lambda_{-} + \bar{G}^{T}_{-})\boldsymbol{u}_{eq,-}\right] \\
&= -\zeta\left[\nabla U - G^{'T}_{+}\gamma^{**'}_{+} - G^{'T}\boldsymbol{u}'\right]^{T}(I - G^{T}_{+}(G_{+}G^{T}_{+})^{-1}G_{+})\left[\nabla U - G^{'T}_{+}\gamma^{**'}_{+} - G^{'T}\boldsymbol{u}'\right] \\
&\quad + \zeta\left[\nabla U - G^{'T}_{+}\gamma^{**'}_{+} - G^{'T}\boldsymbol{u}'\right]^{T}(I - G^{T}_{+}(G_{+}G^{T}_{+})^{-1}G_{+})\left[\bar{G}^{'T}_{+}(\boldsymbol{u}'_{eq,+} - \gamma^{**'}_{+}) + \bar{G}^{T}_{-}\boldsymbol{u}_{eq,-}\right].
\end{aligned}$$

Proceeding by contradiction, assume that for some $i \in \{k+1, k+2, ..., m\}$,

$$\left[\nabla U[\boldsymbol{x}(t), t] - G_{+}^{'T}\gamma_{+}^{**'} - G^{'T}\boldsymbol{u}'\right]^T (I - G_{+}^{T}(G_{+}G_{+}^{T})^{-1}G_{+})\bar{g}_{i,-} > 0,$$

where $\bar{g}_{i,-}$ is a column of $\bar{G}_{-}^{T}$. This implies $\gamma_{i}^{**} > 0$ (γ_{i}^{**} is the entry in γ^{**} with respect to g_i). It is contradiction, so

$$\left[\nabla U[\boldsymbol{x}(t), t] - G_{+}^{'T}\gamma_{+}^{**'} - G^{'T}\boldsymbol{u}'\right]^T (I - G_{+}^{T}(G_{+}G_{+}^{T})^{-1}G_{+})\bar{g}_{i,-} \leq 0.$$

Hence,

$$\left[\nabla U[\boldsymbol{x}(t), t] - G_{+}^{'T}\gamma_{+}^{**'} - G^{'T}\boldsymbol{u}'\right]^T (I - G_{+}^{T}(G_{+}G_{+}^{T})^{-1}G_{+})\bar{G}_{-}^{T} \leq 0.$$

Note that $\boldsymbol{u}_{eq,-} \geq 0$, then

$$\left[\nabla U[\boldsymbol{x}(t), t] - G_{+}^{'T}\gamma_{+}^{**'} - G^{'T}\boldsymbol{u}'\right]^T (I - G_{+}^{T}(G_{+}G_{+}^{T})^{-1}G_{+})\bar{G}_{-}^{T}\boldsymbol{u}_{eq,-} \leq 0.$$

Similarly, proceeding by contradiction, assume that for some $i \in \{k_{+}+1, k_{+}+2, ..., k\}$,

$$\left[\nabla U[\boldsymbol{x}(t), t] - G_{+}^{'T}\gamma_{+}^{**'} - G^{'T}\boldsymbol{u}'\right]^T (I - G_{+}^{T}(G_{+}G_{+}^{T})^{-1}G_{+})\bar{g}_{i,+}^{'} < 0,$$

where $\bar{g}_{i,+}^{'}$ is a column of $\bar{G}_{+}^{'T}$. This implies $\gamma_{i}^{**} < \alpha_i$ (γ_{i}^{**} is the entry in γ^{**} with respect to g_i). It is contradiction, so

$$\left[\nabla U[\boldsymbol{x}(t), t] - G_{+}^{'T}\gamma_{+}^{**'} - G^{'T}\boldsymbol{u}'\right]^T (I - G_{+}^{T}(G_{+}G_{+}^{T})^{-1}G_{+})\bar{g}_{i,+}^{'} \geq 0.$$

Hence,

$$\left[\nabla U[\boldsymbol{x}(t), t] - G_{+}^{'T}\gamma_{+}^{**'} - G^{'T}\boldsymbol{u}'\right]^T (I - G_{+}^{T}(G_{+}G_{+}^{T})^{-1}G_{+})\bar{G}_{+}^{'T} \geq 0.$$

Note that $\boldsymbol{u}'_{eq,+} - \gamma^{**'}_{+} \leq 0$, then

$$\left[\nabla U[\boldsymbol{x}(t),t] - G'^{T}_{+}\gamma^{**'}_{+} - G'^{T}\boldsymbol{u}'\right]^{T}(I - G^{T}_{+}(G_{+}G^{T}_{+})^{-1}G_{+})\bar{G}'^{T}_{+}(\boldsymbol{u}'_{eq,+} - \gamma^{**'}_{+}) \leq 0.$$

By the above steps, one has

$$\begin{aligned}
& d\left[(1/2)L^{2}[\boldsymbol{x}(t),t]\right]/dt \\
=& -\zeta\left[\nabla U - G'^{T}_{+}\gamma^{**'}_{+} - G'^{T}\boldsymbol{u}'\right]^{T}(I - G^{T}_{+}(G_{+}G^{T}_{+})^{-1}G_{+})\left[\nabla U - G'^{T}_{+}\gamma^{**'}_{+} - G'^{T}\boldsymbol{u}'\right] \\
& +\zeta\left[\nabla U - G'^{T}_{+}\gamma^{**'}_{+} - G'^{T}\boldsymbol{u}'\right]^{T}(I - G^{T}_{+}(G_{+}G^{T}_{+})^{-1}G_{+})\left[\bar{G}'^{T}_{+}(\boldsymbol{u}'_{eq,+} - \gamma^{**'}_{+}) + \bar{G}^{T}_{-}\boldsymbol{u}_{eq,-}\right] \\
\leq& -\zeta\left[\nabla U - G'^{T}_{+}\gamma^{**'}_{+} - G'^{T}\boldsymbol{u}'\right]^{T}(I - G^{T}_{+}(G_{+}G^{T}_{+})^{-1}G_{+})\left[\nabla U - G'^{T}_{+}\gamma^{**'}_{+} - G'^{T}\boldsymbol{u}'\right] \\
=& -\zeta L^{2}[\boldsymbol{x}(t),t],
\end{aligned}$$

and that $d\left[(1/2)L^{2}[\boldsymbol{x}(t),t]\right]/dt$ is zero if and only if $L[\boldsymbol{x}(t),t]$ is zero. Moreover, note that $L[\boldsymbol{x}(t),t]$ is a non-increasing function of t (this is proved in Appendix C.3.3.3), and that $L[\boldsymbol{x}(t),t]$ is not differentiable only at isolated time points, then if $L[\boldsymbol{x}(t),t]$ is not differentiable at t,

$$L[\boldsymbol{x}(t^{-}),t^{-}] \geq L[\boldsymbol{x}(t^{+}),t^{+}].$$

So $L[\boldsymbol{x}(t),t]$ is a decreasing function of t, and SA (5.12) converges; i.e.,

$$\lim_{t\to\infty} L[\boldsymbol{x}(t),t] = 0.$$

Moreover, $L[\boldsymbol{x}(t),t]$ decreases no slower than exponentially almost everywhere; i.e.,

$$dL[\boldsymbol{x}(t),t]/dt \leq -\zeta L[\boldsymbol{x}(t),t].$$

C.3.4 NFC Holds

In this section, we will address the case that $\boldsymbol{x}$ can not follow some hyperplanes; i.e., NFC 1 holds for some hyperplane $s_i(\boldsymbol{x}, t) = 0$ at t_i, and prove Theorem 5.3.

- In Appendix C.3.4.1, it shows that if NFC holds for hyperplane $s_i(\boldsymbol{x}, t) = 0$ at time t_i, the increment of the descent function $L[\boldsymbol{x}(t), t]$ is upper bounded.
- In Appendix C.3.4.2, it defines the "s_i-leaving" event and its pair event "s_i-returning".
- In Appendix C.3.4.3, according to the original problem, it defines a problem referred to as the *Base*-problem, and discuss its properties.
- The proof of Theorem 5.3 is given in Appendix C.3.4.4.

C.3.4.1 Bound on Descent Function Increment

Without loss of generality, assume that at some t_m, the hyperplanes $s_i(\boldsymbol{x}, t) = g_i^T \boldsymbol{x} - c_i(t) = 0, i = 1, ..., m$ are active, and at t_m, $\boldsymbol{x}$ can not follow hyperplane $s_m(\boldsymbol{x}, t) = 0$; i.e., NFC 1 holds for hyperplane $s_m = 0$ at time t_m. Then, the descent function $L[\boldsymbol{x}(t), t]$ may have discontinuities and may increase; i.e., one may have

$$L[\boldsymbol{x}(t_m^-), t_m^-] < L[\boldsymbol{x}(t_m^+), t_m^+].$$

In this section, we will give an upper bound on the increment of the descent function caused by not following just one hyperplane $s_m(\boldsymbol{x}, t) = 0$.

At time t_m^-

$$L[\boldsymbol{x}(t_m^-), t_m^-] = \min_{\boldsymbol{\gamma}_m} \|v_m\|,$$

$$\text{where} \quad v_m = \nabla U[\boldsymbol{x}(t_m), t_m] - G_m^T \boldsymbol{\gamma}_m - G'^T \boldsymbol{u}',$$

$$\boldsymbol{\gamma}_m = \begin{bmatrix} \underline{\gamma}_1 & ... & \underline{\gamma}_{m-1} & \underline{\gamma}_m \end{bmatrix}^T, \qquad \underline{\gamma}_i \in [0, \alpha_i], \quad i = 1, ..., m-1, m,$$

$$G_m^T = \begin{bmatrix} g_1 & ... & g_{m-1} & g_m \end{bmatrix}.$$

At time t_m^+

$$L[\boldsymbol{x}(t_m^+), t_m^+] = \min_{\boldsymbol{\gamma}_{m-1}} \|v_{m-1}\|,$$

$$\text{where} \quad v_{m-1} = \nabla U[\boldsymbol{x}(t_m), t_m] - G_{m-1}^T \boldsymbol{\gamma}_{m-1} - g_m u_m - G'^T \boldsymbol{u}', \tag{C.8}$$

$$\boldsymbol{\gamma}_{m-1} = \begin{bmatrix} \underline{\gamma}_1 & ... & \underline{\gamma}_{m-1} \end{bmatrix}^T, \qquad \underline{\gamma}_i \in [0, \alpha_i], \quad i = 1, ..., m-1,$$

$$G_{m-1}^T = \begin{bmatrix} g_1 & ... & g_{m-1} \end{bmatrix}.$$

here $\boldsymbol{u}' = \begin{bmatrix} u_{m+1} & ... & u_M \end{bmatrix}^T$, u_i, $i = m+1, ..., M$, is defined in problem (5.10). Also recall that when NFC 1 holds, we can concentrate on the case $\dot{c}_m > 0$; i.e., c_m increasing such that $\boldsymbol{x}$ "can not follow" the hyperplane $s_m = 0$. Hence, $s_m[\boldsymbol{x}(t_m^+), t_m^+] < 0$, it implies

$$u_m = 0.$$

Then, one has

$$L[\boldsymbol{x}(t_m^+), t_m^+] - L[\boldsymbol{x}(t_m^-), t_m^-] = \min_{\boldsymbol{\gamma}_{m-1}} \|v_{m-1}\| - \min_{\boldsymbol{\gamma}_m} \|v_m\| \leq \min_{\boldsymbol{\gamma}_{m-1}} \|v_{m-1}\|.$$

Recall that $L[\boldsymbol{x}(t_m^+), t_m^+] = \min_{\boldsymbol{\gamma}_{m-1}} \|v_{m-1}\|$ (C.8), and let[2]

$$\boldsymbol{\gamma}_{m-1}^{**} \in \{\arg\min_{\boldsymbol{\gamma}_{m-1}} \|v_{m-1}\|\}.$$

Note that the feasible set of $\boldsymbol{\gamma}_{m-1}$ is an $m-1$ dimension hypercube, and the extreme points (vertexes) $\bar{\boldsymbol{u}}_{m-1}^{j}$, $j = 1, 2, ..., 2^{m-1}$, are of the following form

$$\bar{\boldsymbol{u}}_{m-1}^{j} = \left[\begin{array}{ccc} \bar{u}_1^j & ... & \bar{u}_{m-1}^j \end{array}\right], \quad \bar{u}_i^j \in \{0, \alpha_i\}, \quad i = 1, ..., m-1,$$

and one has, for any $j = 1, 2, ..., 2^{m-1}$,

$$\begin{aligned} &\|\nabla U[\boldsymbol{x}(t_m), t_m] - G_{m-1}^T \boldsymbol{\gamma}_{m-1}^{**} - g_m u_m - G'^T \boldsymbol{u}'\| \\ \leq &\|\nabla U[\boldsymbol{x}(t_m), t_m] - G_{m-1}^T \bar{\boldsymbol{u}}_{m-1}^{j} - g_m u_m - G'^T \boldsymbol{u}'\|. \end{aligned}$$

Now, consider the motion of $\boldsymbol{x}(t)$ in a small neighborhood of the intersection of hyperplanes $s_i = 0$, $i = 1, ..., m-1$. For $j = 1, 2, ..., 2^{m-1}$, let

$$\dot{\boldsymbol{x}}^j = \zeta Z(t_m) \left[\nabla U[\boldsymbol{x}(t_m), t_m] - G_{m-1}^T \bar{\boldsymbol{u}}_{m-1}^{j} - g_m u_m - G'^T \boldsymbol{u}' + (1/\zeta) \frac{\partial \nabla U[\boldsymbol{x}(t_m), t_m]}{\partial t}\right],$$

which are the motions "above" and "below" the discontinuity surfaces $s_i = 0$, $i = 1, 2, ..., m-1$.

We now show that the largest increment in $L[\boldsymbol{x}(t), t]$ occurs when the hyperplane $s_m = 0$ is the only active one prior to this hyperplane becoming non-active at time t_m.

[2]The argument that minimizes the $\|v_{m-1}\|$ may not be unique if the G_{m-1}^T has linear dependent columns.

To show this, note that if the $\boldsymbol{x}$ can not follow $s_m = g_m \boldsymbol{x} - c_m = 0$, and sliding mode occurs on $s_i = 0$, $i = 1, ..., m-1$, it means

$$0 < g_m^T \dot{\boldsymbol{x}}_{eq} < \dot{c}_{max},$$

where the equivalent motion is

$$\begin{aligned}&\dot{\boldsymbol{x}}_{eq}\\ =&\zeta Z(t_m)\left[\nabla U[\boldsymbol{x}(t_m), t_m] - G_{m-1}^T \boldsymbol{u}_{m-1,eq} - g_m u_m - G'^T_{eq} \boldsymbol{u}' + (1/\zeta)\frac{\partial \nabla U[\boldsymbol{x}(t_m), t_m]}{\partial t}\right].\end{aligned}$$

The equivalent motion $\dot{\boldsymbol{x}}_{eq}$ is a convex linear combination of $\dot{\boldsymbol{x}}^j$, the motions around the discontinuity surfaces are [33]

$$\dot{\boldsymbol{x}}_{eq} = \sum_{j=1,2,...,2^{m-1}} \lambda_j \dot{\boldsymbol{x}}^j, \quad \text{where} \quad \sum_{j=1,2,...,2^{m-1}} \lambda_j = 1, \quad \lambda_j \in [0,1], \quad j = 1, 2, ..., 2^{m-1}.$$

Proceeding by contradiction, assume that for all $j = 1, 2, ..., 2^{m-1}$,

$$g_m^T \dot{\boldsymbol{x}}^j \geq \dot{c}_{max}.$$

Then

$$g_m^T \dot{\boldsymbol{x}}_{eq} = g_m^T \sum_{j=1,2,...,2^{m-1}} \lambda_j \dot{\boldsymbol{x}}^j \geq \dot{c}_{max}.$$

This is a contradiction. So there exists at least one $\bar{j}$ such that

$$g_m^T \dot{\boldsymbol{x}}^{\bar{j}} < \dot{c}_{max}.$$

Take any one such $\bar{j}$, then one can consider a point close to the intersection of hyperplanes $s_i = 0$, $i = 1, ..., m-1$, where the motion $\dot{\boldsymbol{x}}^r(t_m)$ is given by

$$\begin{aligned}
&\dot{\boldsymbol{x}}^r(t_m)\\
=&\zeta Z(t_m)\left[\nabla U[\boldsymbol{x}(t_m), t_m] - G^T_{m-1}\bar{\boldsymbol{u}}^{\bar{j}}_{m-1} - g_m u_m - G'^T\boldsymbol{u}' + (1/\zeta)\frac{\partial \nabla U[\boldsymbol{x}(t_m), t_m]}{\partial t}\right]\\
=&\dot{\boldsymbol{x}}^{\bar{j}},
\end{aligned}$$

it implies

$$g^T_m \dot{\boldsymbol{x}}^r(t_m) < \dot{c}_{max}.$$

Let $L^r[\boldsymbol{x}^r(t), t]$ be the descent function of trajectory $\boldsymbol{x}^r(t)$, also note that at time t_m, $\boldsymbol{x}^r(t)$ can not follow the hyperplane $s_m = 0$, then

$$\begin{aligned}
&L^r[\boldsymbol{x}(t^+_m), t^+_m] - L^r[\boldsymbol{x}(t^-_m), t^-_m]\\
\leq&\|\nabla \tilde{U}^r[\boldsymbol{x}(t_m), t_m]\|\\
=&\|\nabla U[\boldsymbol{x}(t_m), t_m] - G^T_{m-1}\bar{\boldsymbol{u}}^{\bar{j}}_{m-1} - g_m u_m - G'^T\boldsymbol{u}'\|.
\end{aligned}$$

Note that

$$\begin{aligned}
&L[\boldsymbol{x}(t^+_m), t^+_m] - L[\boldsymbol{x}(t^-_m), t^-_m] \leq \|\nabla U[\boldsymbol{x}(t_m), t_m] - G^T_{m-1}\boldsymbol{\gamma}^{**}_{m-1} - g_m u_m - G'^T\boldsymbol{u}'\|,\\
&L^r[\boldsymbol{x}(t^+_m), t^+_m] - L^r[\boldsymbol{x}(t^-_m), t^-_m] \leq \|\nabla U[\boldsymbol{x}(t_m), t_m] - G^T_{m-1}\bar{\boldsymbol{u}}^{\bar{j}}_{m-1} - g_m u_m - G'^T\boldsymbol{u}'\|,\\
&\|\nabla U[\boldsymbol{x}(t_m), t_m] - G^T_{m-1}\boldsymbol{\gamma}^{**}_{m-1} - g_m u_m - G'^T\boldsymbol{u}'\|\\
\leq&\|\nabla U[\boldsymbol{x}(t_m), t_m] - G^T_{m-1}\bar{\boldsymbol{u}}^{\bar{j}}_{m-1} - g_m u_m - G'^T\boldsymbol{u}'\|.
\end{aligned}$$

So the largest increment on the descent function occurs when just one hyperplane is active, and this will be the case we concentrate on. And we now give the upper bound

of such an increment. If $\boldsymbol{x}$ can not follow $s_{m} = g_{m}^{T}\boldsymbol{x} - c_{m} = 0$, note that

$$g_{m}^{T}\dot{\boldsymbol{x}} = g_{m}^{T}Z(t)\left[\zeta\nabla\tilde{U}(\boldsymbol{x},t) + \frac{\partial\nabla\tilde{U}(\boldsymbol{x},t)}{\partial t}\right] = \zeta g_{m}^{T}Z(t)\left[\nabla\tilde{U}(\boldsymbol{x},t) + (1/\zeta)\frac{\partial\nabla U(\boldsymbol{x},t)}{\partial t}\right],$$

then

$$0 < \zeta g_{m}^{T}Z(t)\left[\nabla\tilde{U}(\boldsymbol{x},t) + (1/\zeta)\frac{\partial\nabla U(\boldsymbol{x},t)}{\partial t}\right] < \dot{c}_{max}.$$

And recall that

$$\|g_{i}^{T}\| = 1,$$

$$pI < Z(t) < PI,$$

$$\left\|\frac{\partial\nabla U(\boldsymbol{x},t)}{\partial t}\right\| < Q.$$

Therefore, one has

$$\zeta p\left\|\nabla\tilde{U}(\boldsymbol{x},t) + (1/\zeta)\frac{\partial\nabla U(\boldsymbol{x},t)}{\partial t}\right\| < \dot{c}_{max},$$

$$\left\|\nabla\tilde{U}(\boldsymbol{x},t) + (1/\zeta)\frac{\partial\nabla U(\boldsymbol{x},t)}{\partial t}\right\| < \frac{\dot{c}_{max}}{\zeta p},$$

$$\left|\|\nabla\tilde{U}(\boldsymbol{x},t)\| - \left\|(1/\zeta)\frac{\partial\nabla U(\boldsymbol{x},t)}{\partial t}\right\|\right| < \frac{\dot{c}_{max}}{\zeta p},$$

$$\|\nabla\tilde{U}(\boldsymbol{x},t)\| < \frac{\dot{c}_{max}}{\zeta p} + \left\|(1/\zeta)\frac{\partial\nabla U(\boldsymbol{x},t)}{\partial t}\right\| < \frac{\dot{c}_{max}}{\zeta p} + \frac{Q}{\zeta}.$$

Hence, if $\boldsymbol{x}(t)$ can not follow hyperplane $s_{m}(\boldsymbol{x},t) = 0$, the worst case increment on $L[\boldsymbol{x}(t),t]$ is bounded from above

$$L[\boldsymbol{x}(t_{m}^{+}),t_{m}^{+}] - L[\boldsymbol{x}(t_{m}^{-}),t_{m}^{-}] < \frac{\dot{c}_{max}}{\zeta p} + \frac{Q}{\zeta}. \tag{C.9}$$

C.3.4.2 "s_i-leaving" Event and Its Pair Event "s_i-returning"

Consider the event $\boldsymbol{x}$ can not follow some hyperplane $s_i(\boldsymbol{x}, t) = 0$ at time t_i^l; i.e., NFC 1 holds at time t_i^l. This event is referred as "s_i-leaving" at t_i^l. We now define its pair event "s_i-returning". If there exists a finite t_i^r, such that the hyperplane $s_i(\boldsymbol{x}, t) = 0$ becomes active at time t_i^r, then such an event is referred to as "s_i-returning", also as the pair event of "s_i-leaving". Moreover, "s_i-leaving" is also referred to as pair event of "s_i-returning".

Given an initial condition t_0, $\boldsymbol{x}(t_0)$, and any finite $\hat{t}$, in the time interval $t_0 \leq t < \hat{t}$, there is an event sequence. For example

$$t_0 \leq t_1^{l,1} \leq t_2^l \leq t_1^{r,1} \leq t_3^l \leq t_1^{l,2} \leq t_2^r < \hat{t},$$

where at $t_1^{l,2}$, the event "s_1-leaving" happens the 2nd time in time interval $t_0 \leq t < \hat{t}$, etc. Moreover, there are several properties of such an event sequence:

- For a fixed hyperplane $s_i = 0$, and an event "s_i-leaving", it is not necessary to have its pair event "s_i-returning". For example, "s_3-leaving" event does not have its pair event in time interval $t_0 \leq t \leq \hat{t}$.
- The event "s_i-returning" must happen after its pair event "s_i-leaving"; i.e., $t_i^l < t_i^r$.
- For a fixed hyperplane $s_i = 0$, the event "s_i-leaving" may happen more than one time in time interval $t_0 \leq t < \hat{t}$, but there must have exactly one event "s_i-returning" in between any consecutive "s_i-leaving" events. For example, "s_1-leaving" event happens two times for the $s_1 = 0$ hyperplane, and there is one "s_1-returning" event in between the two "s_1-up" events; i.e., $t_1^{l,1} < t_1^{r,1} < t_1^{l,2}$

C.3.4.3 *Base*-problem $B(t_0, \hat{t})$

Given an initial condition t_0, $\boldsymbol{x}(t_0)$, and finite $\hat{t}$, we define $c_i^b(t)$ in the interval $[t_0, \hat{t}]$: for all k, such that at time $t_i^{l,k} \in [t_0, \hat{t}]$, the event "$s_i$-leaving" happens the kth time, and

at time $t_i^{r,k} \in [t_0, \hat{t}]$ its pair event "s_i-returning" kth time happens,

$$c_i^b(t) = g_i^T \boldsymbol{x}(t), \quad t_i^{l,k} \le t < t_i^{r,k},$$

$$c_i^b(t) = c_i(t), \quad \text{otherwise.}$$

For any given finite $\hat{t}$, we define the following optimization problem referred to as $B(t_0, \hat{t})$, the *Base*-Problem

$$\max \tilde{U}^b(\boldsymbol{x}, t) = U(\boldsymbol{x}, t) - \sum_{i=1,...,M} u_i s_i^b(\boldsymbol{x}, t), \tag{C.10}$$

$$s_i^b(\boldsymbol{x}, t) = g_i^T \boldsymbol{x} - c_i^b(t), \quad i = 1, ..., M,$$

$$u_i = \begin{cases} \alpha_i, & \text{if } s_i^b > 0, \\ 0, & \text{if } s_i^b < 0. \end{cases}$$

Note that the *Base*-problem depends on time t_0 and $\hat{t}$.

Recall that the original objective function $\tilde{U}(\boldsymbol{x}, t)$ is of this form

$$\tilde{U}(\boldsymbol{x}, t) = U(\boldsymbol{x}, t) - \sum_{i=1,...,M} u_i s_i(\boldsymbol{x}, t).$$

So in the *Base*-Problem, the hyperplanes $s_i^b(\boldsymbol{x}, t) = g_i^T \boldsymbol{x} - c_i^b(t) = 0$ are defined and take the place of $s_i(\boldsymbol{x}, t) = g_i^T \boldsymbol{x} - c_i(t) = 0$. Note that $s_i(\boldsymbol{x}, t)$ and $s_i^b(\boldsymbol{x}, t)$ have the same gradient, only $c_i(t)$ and $c_i^b(t)$ are different. Moreover, by the way the *Base*-problem is constructed, the trajectory of the original problem $\boldsymbol{x}(t)$ and the trajectory of the *Base*-problem $\boldsymbol{x}^b(t)$ satisfy

$$\boldsymbol{x}^b(t) = \boldsymbol{x}(t), \quad t_0 \le t \le \hat{t}.$$

Now we will compare the event sequences of the original problem and the *Base*-problem. Consider the event "s_i-leaving" kth time in the original problem. If its pair event happens in the original problem before time $\hat{t}$, then by the way the *Base*-problem is constructed, in the *Base*-problem the event "s_i-leaving" kth time and its pair event do not happen. Only if its pair event does not happen in the original problem before time $\hat{t}$, the event "s_i-leaving" kth time happens in the *Base*-problem. For example, the event sequence in the original problem is

$$t_0 \leq t_1^{l,1} \leq t_2^{l} \leq t_1^{r,1} \leq t_3^{l} \leq t_1^{l,2} \leq t_2^{r} < \hat{t},$$

correspondingly,the event sequence in the *Base*-problem is[3]

$$t_0 \leq t_3^{l} \leq t_1^{l,2} < \hat{t}.$$

So there is no pair event happening. Then given any initial condition t_0, $\boldsymbol{x}(t_0)$, $L[\boldsymbol{x}(t_0), t_0]$ and any finite $\hat{t}$, in the time interval $[t_0, \hat{t}]$, the event sequence of *Base*-problem has following properties:

- There are only "s_i-leaving" events.
- If "s_i-leaving" and "s_j-leaving" both happen, then $i \neq j$. Therefore, there are at most M "s_i-leaving" events.

Recall the definition of $c_i^b(t)$, at time $\hat{t}$, for any $i = 1, 2, ..., M$, $c_i^b(\hat{t}) = c_i(\hat{t})$, so one has

$$\boldsymbol{x}^b(\hat{t}) = \boldsymbol{x}(\hat{t}),$$

$$s_i^b(\boldsymbol{x}, \hat{t}) = s_i(\boldsymbol{x}, \hat{t}).$$

[3]Here the number 2 in the superscript $t_1^{l,2}$ implies $t_1^{l,2}$ is the time the event "s_1-leaving" 2nd time happening in the original problem, however it the *Base*-problem, the event "s_1-leaving" only happens once at time $t_1^{l,2}$.

It implies that at time $\hat{t}$, the descent function of the original problem $L[\boldsymbol{x}(\hat{t}), \hat{t}]$, and the descent function of the *Base*-problem $L^b[\boldsymbol{x}^b(\hat{t}), \hat{t}]$ satisfies[4]

$$L^b[\boldsymbol{x}^b(\hat{t}), \hat{t}] = L[\boldsymbol{x}(\hat{t}), \hat{t}].$$

C.3.4.4 Proof of Theorem 5.3

Recall that $T(\zeta, \epsilon, a)$ is given by (5.1), which is the time for system $\dot{e} = -\zeta e$ to decrease from $a + \epsilon$ to ϵ. It has the following property: Given any $a, a_1, a_2, \epsilon, \zeta > 0$, if $a = a_1 + a_2$, then

$$T(\zeta, \epsilon, a) < T(\zeta, \epsilon, a_1) + T(\zeta, \epsilon, a_2). \tag{C.11}$$

Given any initial condition t_0, $\boldsymbol{x}(t_0)$, $L[\boldsymbol{x}(t_0), t_0]$, recall the $\hat{T}$ defined in Theorem 5.3

$$\hat{T} = T(\zeta, \epsilon, L[\boldsymbol{x}(t_0), t_0] - \epsilon) + MT(\zeta, \epsilon, \frac{\dot{c}_{max}}{\zeta p} + \frac{Q}{\zeta}) < \infty.$$

Then for any $\hat{t} \geq \hat{T}$, one may define a *Base*-problem $B(t_0, \hat{t})$. Note that $B(t_0, \hat{t})$ has the property that in the time interval $[t_0, \hat{t}]$ there are at most M "s_i-leaving" events; i.e., there are at most M increments. Note that each increment is bounded (C.9), i.e,

$$L^b[\boldsymbol{x}^b(t_i^{l,+}), t_i^{l,+}] - L^b[\boldsymbol{x}^b(t_i^{l,-}), t_i^{l,-}] < \frac{\dot{c}_{max}}{\zeta p} + \frac{Q}{\zeta}.$$

Also note that $T(\zeta, \epsilon, a)$ is the time for system $\dot{e} = -\zeta e$ to decrease from $a+\epsilon$ to ϵ, which has property (C.11), and that $\hat{t} > \hat{T}$, then there exists time $t_\epsilon \in [t_0, \hat{t}]$ such that

$$L^b[\boldsymbol{x}^b(t_\epsilon^-), t_\epsilon^-] < \epsilon.$$

[4]The $L[\boldsymbol{x}(t), t]$ and $L^b[\boldsymbol{x}^b(t), t]$ are not non-increasing function, but we use them to measure the performance.

Moreover, in the time interval $[t_\epsilon, \hat{t}]$, there are at most M "s_i-leaving" events, so for any $\hat{t} > \hat{T}$,

$$\begin{aligned} L[\boldsymbol{x}(\hat{t}), \hat{t}] &= L^b[\boldsymbol{x}^b(\hat{t}), \hat{t}] \\ &< L^b[\boldsymbol{x}^b(t_\epsilon^-), t_\epsilon^-] + M\left[\frac{\dot{c}_{max}}{\zeta p} + \frac{Q}{\zeta}\right] \\ &< \epsilon + M\left[\frac{\dot{c}_{max}}{\zeta p} + \frac{Q}{\zeta}\right]. \end{aligned}$$

So for any given $\epsilon > 0$, and any initial condition t_0, $\boldsymbol{x}(t_0)$, $L[\boldsymbol{x}(t_0), t_0]$, apply the Sliding Algorithm, and one has that, for any $t \geq \hat{T}$,

$$L[\boldsymbol{x}(t), t] < \epsilon + M\left[\frac{\dot{c}_{max}}{\zeta p} + \frac{Q}{\zeta}\right].$$

C.4 Proof of Theorem 5.4

Throughout of this section, we will use following notation. Let G_1 be $m_1 \times n$, G_2 be $m_2 \times n$, and G_3 be $m_3 \times n$ matrix, such that $m_3 \leq m_1 \leq m_2 \leq n$,

$$G_1 \subseteq G_2$$

means that any row in G_1 is a row in G_2.

$$G_1 \cap G_2 = G_3$$

means any row of G_3 is a row in both G_1 and G_2. If G_1 has linear independent rows, the projection $\mathcal{P}(G_1)$ is defined as

$$\mathcal{P}(G_1) = (I - G_1^T(G_1 G_1^T)^{-1} G_1).$$

Recall that Section 5.3 focuses on the unconstrained problem (5.10) and Theorem 5.3, and Appendix C.3 provides proof of Theorem 5.3. Section 5.4 focuses on the

constrained problem (5.15), and addresses it by solving its unconstrained form, which is a special case of the unconstrained problem (5.10). Hence, all the definitions given in Appendix C.3 also work in this section. However, due to the fact that we have *constrained* problem in this section, we will refer to s_i as a constraint instead of a hyperplane. For example, we use "active constraint set" instead of "active hyperplane set", and both have the same meaning.

Also note that many quantities in this section depend on $(\boldsymbol{x}(t), t)$, $(\boldsymbol{x}, t)$ or t, for simplicity, unless needed for clarity, the $(\boldsymbol{x}(t), t)$, $(\boldsymbol{x}, t)$ and t are omitted. Moreover, $H_{opt} = H(\boldsymbol{x}_{opt})$ and $\nabla U_{opt} = \nabla U(\boldsymbol{x}_{opt})$.

C.4.1 Preliminary

Let G be any matrix with linear independent rows, whose rows are g_i, $i \in \{1, ..., M\}$. Note that one has finite number of constraints (the number M is finite), and that $pI < -H^{-1}(\boldsymbol{x}) < PI$, then there exist positive constants ψ and ϕ,

$$\psi = \max_{G} \|(GG^T)^{-1} G\| < \infty, \tag{C.12}$$

$$\phi = \max_{\boldsymbol{x}, G} \|(GH^{-1}(\boldsymbol{x})G^T)^{-1}\| < \infty. \tag{C.13}$$

Moreover, for any $n \times 1$ vector g being a column of G^T, rewrite $G^T = \begin{bmatrix} g & G^T_{rest} \end{bmatrix}$. Again, note that one has finite number of constraints, hence, there exists positive constant κ, such that for any G and g being a column of G^T,

$$\|\mathcal{P}(G_{rest})g\| > \kappa > 0. \tag{C.14}$$

Due to the condition (5.18), there exists a constant $\chi > 0$, such that

$$\chi = \max_{\boldsymbol{x} \in \mathcal{X}} \|\nabla U(\boldsymbol{x})\| < \infty. \tag{C.15}$$

Recall the constants P, $\dot{c}_{max}$, $\bar{\lambda}$, χ, ϕ, and ψ given by (5.16, 5.17, C.15, 5.19, C.12, C.13), and that $\dim \boldsymbol{x} = n$. For any given $\zeta > 0$, define

$$\alpha^* = \max(\bar{\lambda}, \psi\chi, n^{1/2}\phi P\chi + (1/\zeta)n^{1/2}\phi\dot{c}_{max}). \tag{C.16}$$

The time-varying optimization problem (5.15) given in Section 5.4 can be written as the following unconstrained form which is a special case of problem (5.10) addressed in Section 5.3

$$\max_{\boldsymbol{x}} \tilde{U}(\boldsymbol{x}, t) = U(\boldsymbol{x}) - \sum_{i=1,...,M} u_i s_i(\boldsymbol{x}, t). \tag{C.17}$$

Recall that

$$u_i = \begin{cases} \alpha_i, & \text{if } s_i > 0, \\ 0, & \text{if } s_i < 0. \end{cases}$$

Note that by (C.16), $\alpha^* \geq \bar{\lambda}$, and hence, if $\alpha_i > \alpha^* \geq \bar{\lambda}$, (i.e., the condition (5.21) holds), then the unconstrained problem (C.17) is equivalent to the original constrained problem (5.15) in the sense that they have the same solution for all t.

LEMMA C.3. *Let α^* be defined in (C.16). If $\alpha_i > \alpha^*$, $i = 1, ..., M$, and for some time t_0, $\boldsymbol{x}(t_0)$ is feasible (i.e., $s_i[\boldsymbol{x}(t_0), t_0] \leq 0$, $i = 1, ..., M$), then for $t \geq t_0$, $\boldsymbol{x}(t)$ is feasible.*

Proof of Lemma C.3. Assume at time t, for a constraint $s_m(\boldsymbol{x}, t) = g_m^T \boldsymbol{x} - c_m(t) \leq 0$, there exists feasible point $\boldsymbol{x}$ satisfying $s_m(\boldsymbol{x}, t) = g_m^T \boldsymbol{x} - c_m(t) = 0$, moreover, at feasible point $\boldsymbol{x}$, sliding mode occurs on hyperplanes $s_i = 0$, $i = 1, 2, ..., m-1$. If the vectors g_i, $i = 1, ..., m-1$ are linear dependent, let $\bar{G}$ be any matrix with linear independent rows, whose rows are g_i, $i \in \{1, ..., m-1\}$, and the column space of $\bar{G}^T$ equals the space spanned by vectors g_i, $i = 1, ..., m-1$, then the equivalent motion is [33]

$$\dot{\boldsymbol{x}}_{eq} = \zeta(Z - Z\bar{G}^T(\bar{G}Z\bar{G}^T)^{-1}\bar{G}Z)\nabla U + Z\bar{G}^T(\bar{G}Z\bar{G}^T)^{-1}\dot{\bar{\boldsymbol{c}}}.$$

The entries of vector $\bar{\boldsymbol{c}}$ are c_i, such that g_i^T is a row of $\bar{G}$. Note that the motion $\lim_{s_m \to 0^+} \dot{s}_m$ is

$$\lim_{s_m \to 0^+} \dot{s}_m = g_m^T \dot{\boldsymbol{x}}_{eq} - \zeta \alpha_m g_m^T Z g_m - \dot{c}_m.$$

Without loss of generality, assume that the vector g_m is not in the column space of $\bar{G}^T$. Let vector v be

$$\begin{aligned} v =& \left(\begin{bmatrix} \bar{G} \\ g_m^T \end{bmatrix} Z \begin{bmatrix} \bar{G}^T & g_m \end{bmatrix} \right)^{-1} \left(\begin{bmatrix} \bar{G} \\ g_m^T \end{bmatrix} Z \zeta \nabla U - \begin{bmatrix} \dot{\bar{\boldsymbol{c}}} \\ \dot{c}_m \end{bmatrix} \right) \\ =& \begin{bmatrix} * & * \\ -C_2^{-1} g_m^T Z \bar{G}^T (\bar{G} Z \bar{G}^T)^{-1} & C_2^{-1} \end{bmatrix} \begin{bmatrix} \bar{G} Z \zeta \nabla U - \dot{\bar{\boldsymbol{c}}} \\ g_m^T Z \zeta \nabla U - \dot{c}_m \end{bmatrix}, \end{aligned}$$

where $C_2^{-1} = (g_m^T Z g_m - g_m^T Z \bar{G}^T (\bar{G} Z \bar{G}^T)^{-1} G Z g_m)^{-1} > 0$. Then the last entry of vector v is

$$\begin{aligned} & (g_m^T Z g_m - g_m^T Z \bar{G}^T (\bar{G} Z \bar{G}^T)^{-1} G Z g_m)^{-1} (g_m^T \dot{\boldsymbol{x}}_{eq} - \dot{c}_m) \\ \leq & \|v\|_\infty \leq \|v\| \leq \zeta [\phi n^{1/2} P \chi + (1/\zeta) \phi n^{1/2} \dot{c}_{max}] = \zeta \alpha^*, \end{aligned}$$

then

$$g_m^T \dot{\boldsymbol{x}}_{eq} - \dot{c}_m \leq (g_m^T Z g_m - g_m^T Z \bar{G}^T (\bar{G} Z \bar{G}^T)^{-1} G Z g_m) \zeta \alpha^* \leq \|g_m^T Z g_m\|^2 \zeta \alpha^*.$$

It implies

$$\lim_{s_m \to 0^+} \dot{s}_m = g_m^T \dot{\boldsymbol{x}}_{eq} - \zeta \alpha_m g_m^T Z g_m - \dot{c}_m \leq \|g_m^T Z g_m\|^2 \zeta \alpha^* - \zeta \alpha_m g_m^T Z g_m < 0.$$

It means for any time t, the motion of $\boldsymbol{x}(t)$ close and just outside the feasible set is toward the inside of the feasible set. So if for some time t_0, $\boldsymbol{x}(t_0)$ is feasible, then for $t \geq t_0$, $\boldsymbol{x}(t)$ is feasible.

Assume that $\boldsymbol{x}$ is feasible and sliding on hyperplanes $s_i = 0, i = 1, 2, ..., m$, and the rows of the active hyperplane set G are g_i^T, $i = 1, 2, ..., m$. For the rest hyperplanes $s_i = 0, i = m+1, m+2, ..., M$, let the matrix G'^T be $\begin{bmatrix} g_{m+1} & ... & g_M \end{bmatrix}$. The equivalent motion is [33]

$$\dot{\boldsymbol{x}}_{eq} = Z(t) \left[\zeta \nabla U[\boldsymbol{x}(t), t] - \zeta G^T \boldsymbol{u}_{eq} - \zeta G'^T \boldsymbol{u}' \right].$$

By Lemma C.3, one has $\boldsymbol{u}' = 0$.

Recall that

$$L[\boldsymbol{x}(t), t] = \min_{\gamma_i \in [0, \alpha_i],\ i=1,...,m} \|\nabla U[\boldsymbol{x}(t), t] - G^T \gamma\|,$$

and that the partition of G^T is given by (C.7) in Appendix C.3.3.4,

$$G^T = \begin{bmatrix} G_+^T & G_+'^T & G_-^T \end{bmatrix},$$

we have following lemma.

LEMMA C.4. *Let α^* be defined in (C.16). If $\alpha_i > \alpha^*$, $i = 1, ..., M$, then the matrix $G_+'^T$ is empty.*

Proof of Lemma C.4. If the active constraint set G is not row linear independent, let G_{LI} be a matrix with linear independent rows such that $G_{LI} \subseteq G$ (any row in G_{LI} is a row in G). Note that such G_{LI} may be not unique. And one has

$$\begin{aligned} &\min_{\gamma_i \in [0, \alpha_i],\ i=1,...,m} \|\nabla U - G^T \gamma\| \\ \geq &\min_{G_{LI}: (G_{LI} G_{LI}^T)^{-1} G_{LI} \nabla U > 0} \|\nabla U - G_{LI}^T (G_{LI} G_{LI}^T)^{-1} G_{LI} \nabla U\|. \end{aligned}$$

Also note that for any G_{LI}^* such that

$$G_{LI}^* \in \{\arg \min_{G_{LI}: (G_{LI} G_{LI}^T)^{-1} G_{LI} \nabla U > 0} \|\nabla U - G_{LI}^T (G_{LI} G_{LI}^T)^{-1} G_{LI} \nabla U\|\},$$

one has, for all $i = 1, ..., M$,

$$\|(G^*_{LI} G^{*T}_{LI})^{-1} G^*_{LI} \nabla U\|_\infty \leq \|(G^*_{LI} G^{*T}_{LI})^{-1} G^*_{LI}\| \|\nabla U\| \leq \psi\chi \leq \alpha^* < \alpha_i.$$

Hence

$$\min_{\gamma_i \in [0,\alpha_i],\ i=1,...,m} \|\nabla U - G^T \gamma\| = \|\nabla U - G^{*T}_{LI} (G^*_{LI} G^{*T}_{LI})^{-1} G^*_{LI} \nabla U\|.$$

It implies $G_+ = G^*_{LI}$, and matrix G'^T_+ is empty.

By Lemma C.4, G^T can be written as

$$G^T = \begin{bmatrix} G^T_+ & G^T_- \end{bmatrix}.$$

And correspondingly

$$\gamma^{**T} = \begin{bmatrix} \gamma^{**T}_+ & \gamma^{**T}_- \end{bmatrix}.$$

Note that

$$\gamma^{**}_+ = (G_+ G^T_+)^{-1} G_+ \nabla U, \qquad \gamma^{**}_- = 0.$$

Moreover, for any time t, let

$$G_{+,opt} = G_+ \cap G_{opt},$$

then G^T can be rewritten as

$$G^T = \begin{bmatrix} G^T_{+,opt} & G^T_{+,rest} & G^T_- \end{bmatrix},$$

correspondingly

$$\gamma^{**T} = \begin{bmatrix} \gamma^{**T}_{+,opt} & \gamma^{**T}_{+,rest} & \gamma^{**T}_- \end{bmatrix}. \tag{C.18}$$

We now give a definition: $\boldsymbol{x}(t)$ being on the right active constraints set means that there exists $G_{+}(t)$ such that $G_{+,opt}(t) = G_{opt}(t)$.

C.4.2 Auxiliary Lemmas

Assume at some time t_g, $\boldsymbol{x}$ "reaches" the right active constraint set $G_{opt}(t_g)$; i.e., for some vector g being a column of $G^T_{opt}(t_g)$, one has

$$\begin{bmatrix} G^T_{+,opt}(t_g^-) & g \end{bmatrix} = G^T_{opt}(t_g), \quad \text{and} \quad G^T_{+,opt}(t_g^+) = G^T_{opt}(t_g).$$

And let

$$\left(\begin{bmatrix} G_+(t_g^-) \\ g^T \end{bmatrix} \begin{bmatrix} G^T_+(t_g^-) & g \end{bmatrix} \right)^{-1} \begin{bmatrix} G_+(t_g^-) \\ g^T \end{bmatrix} \nabla U = \begin{bmatrix} * \\ \gamma_g \end{bmatrix}, \tag{C.19}$$

the γ_g is the entry of the above vector, which is with respect to vector g.

LEMMA C.5. *Recall the definitions of κ and $\underline{\lambda}$ given in (C.14, 5.19), there exists $\bar{\sigma} > 0$, such that, if $L(t_g^-) < \bar{\sigma}$ and NFC 1 does not hold at time t_g, then $\gamma_g < (1/2)\underline{\lambda}$.*

Proof of Lemma C.5. NFC 1 does not hold at time t_g, then one has

$$G^T_+(t_g^-) \subseteq G^T_+(t_g^+) = \begin{bmatrix} G^T_+(t_g^-) & g \end{bmatrix}.$$

For simplicity, omit t_g^- and let $G_+ = G_+(t_g^-)$. One has

$$\left(\begin{bmatrix} G_+ \\ g^T \end{bmatrix} \begin{bmatrix} G^T_+ & g \end{bmatrix} \right)^{-1} \begin{bmatrix} G_+ \\ g^T \end{bmatrix} \nabla U = \begin{bmatrix} * \\ \gamma_g \end{bmatrix},$$

$$\begin{bmatrix} * & * \\ -C_2^{-1} g^T G^T_+ (G_+ G^T_+)^{-1} & C_2^{-1} \end{bmatrix} \begin{bmatrix} G_+ \nabla U \\ g^T \nabla U \end{bmatrix} = \begin{bmatrix} * \\ \gamma_g \end{bmatrix},$$

$$C_2^{-1} = (g^T g - g^T G_+^T (G_+ G_+^T)^{-1} G_+ g)^{-1}.$$

Note that $L(t_g^-) = \|(I - G_+^T (G_+ G_+^T)^{-1} G_+)\nabla U\| = \|\mathcal{P}(G_+)\nabla U\| < \bar{\sigma}$, then

$$\gamma_g = \frac{g^T \mathcal{P}(G_+)\mathcal{P}(G_+)\nabla U}{\|\mathcal{P}(G_+)g\|^2} \leq \frac{\|g^T \mathcal{P}(G_+)\|\|\mathcal{P}(G_+)\nabla U\|}{\|\mathcal{P}(G_+)g\|^2} = \frac{\|\mathcal{P}(G_+)\nabla U\|}{\|\mathcal{P}(G_+)g\|} \leq \bar{\sigma}/\kappa.$$

So if $\bar{\sigma} < (1/2)(\underline{\lambda}\kappa)$, then $\gamma_g < (1/2)\underline{\lambda}$.

LEMMA C.6. *Recall the definitions of p and P given in (5.16). Assume that $\boldsymbol{x}(t)$ is on the right active constraint set, i.e, $G_{+,opt}(t) = G_{opt}(t)$, and $L(t) < \hat{\sigma}$, then*

$$\|\nabla U - \nabla U_{opt}\| < \frac{P}{p}\hat{\sigma}. \tag{C.20}$$

Proof of Lemma C.6. G_+ is a $k \times n$ constant matrix with linear independent rows. There exist a matrix F and a vector $\hat{\boldsymbol{x}}$, such that

$$\boldsymbol{x} = F^T \boldsymbol{z} + \hat{\boldsymbol{x}} \quad \text{and} \quad U_z(\boldsymbol{z}) = U(F^T \boldsymbol{z} + \hat{\boldsymbol{x}}),$$

where $\boldsymbol{z} \in R^{n-k}$, F is an $(n-k) \times n$ matrix with linear independent rows, which satisfy

$$FF^T = I \quad \text{and} \quad G_+ F^T = 0.$$

Then for any vector $v \in R^n$, there exists a unique vector pair (v_G, v_F), such that

$$v = G_+^T v_G + F^T v_F,$$

and one has

$$v^T \mathcal{P}(G_+)v = (v_G^T G_+ + v_F^T F)\mathcal{P}(G_+)(G_+^T v_G + F^T v_F) = v_F^T v_F = v^T F^T F v,$$

i.e.,

$$\|\mathcal{P}(G_{+})v\| = \|Fv\|. \tag{C.21}$$

Note that

$$\nabla U_{z}(\boldsymbol{z}) = \frac{dU_{z}}{d\boldsymbol{z}} = F\nabla U(F^{T}\boldsymbol{z} + \hat{\boldsymbol{x}}) \text{ and } H_{z} = \frac{d^{2}U_{z}}{d\boldsymbol{z}^{2}} = FHF^{T}.$$

Moreover, one has

$$pI < -FHF^{T} < PI.$$

Let $\boldsymbol{z}_{opt}$ be the vector such that

$$\nabla U_{z}(\boldsymbol{z}_{opt}) = F\nabla U(F^{T}\boldsymbol{z}_{opt} + \hat{\boldsymbol{x}}) = 0,$$

then

$$\|\boldsymbol{z} - \boldsymbol{z}_{opt}\| \leq \frac{\|\nabla U_{z}(\boldsymbol{z})\|}{p}.$$

Moreover, the vector $F^{T}\boldsymbol{z}_{opt} + \hat{\boldsymbol{x}}$ satisfies

$$\|\mathcal{P}(G_{+})\nabla U(F^{T}\boldsymbol{z}_{opt} + \hat{\boldsymbol{x}})\| = \|F\nabla U(F^{T}\boldsymbol{z}_{opt} + \hat{\boldsymbol{x}})\| = 0.$$

It implies

$$\boldsymbol{x}_{opt} = F^{T}\boldsymbol{z}_{opt} + \hat{\boldsymbol{x}}.$$

Also note that

$$\|\boldsymbol{x} - \boldsymbol{x}_{opt}\|^{2} = (\boldsymbol{x} - \boldsymbol{x}_{opt})^{T}(\boldsymbol{x} - \boldsymbol{x}_{opt}) = (\boldsymbol{z} - \boldsymbol{z}_{opt})^{T}FF^{T}(\boldsymbol{z} - \boldsymbol{z}_{opt}) = \|\boldsymbol{z} - \boldsymbol{z}_{opt}\|^{2},$$

$$\|\mathcal{P}(G_{+})\nabla U(F^{T}\boldsymbol{z} + \hat{\boldsymbol{x}})\| = \|F\nabla U(F^{T}\boldsymbol{z} + \hat{\boldsymbol{x}})\| = \|\nabla U_{z}(\boldsymbol{z})\|,$$

then one has

$$\|\boldsymbol{x} - \boldsymbol{x}_{opt}\| < \frac{1}{p}\hat{\sigma},$$

and hence

$$\|\nabla U - \nabla U_{opt}\| < \frac{P}{p}\hat{\sigma}.$$

LEMMA C.7. *Recall the definitions of p, P, $\underline{\lambda}$, and ψ given in (5.16, 5.19, C.12), and the vector $\gamma^{**}_{+,opt}$ is defined by (C.18). Furthermore, assume that $\boldsymbol{x}(t)$ is on the right active constraint set; i.e., $G_{+,opt}(t) = G_{opt}(t)$. Then, there exists a constant $\hat{\sigma}$, such that, if $L(t) < \hat{\sigma}$,*

$$\gamma^{**}_{+,opt} > (1/2)\underline{\lambda}. \tag{C.22}$$

Proof of Lemma C.7. Let

$$v = (G_+ G_+^T)^{-1} G_+ (\nabla U - \nabla U_{opt}),$$

and let $\boldsymbol{\lambda}$ be the Lagrangian Multiplier vector. By assumption one has

$$G_{+,opt} = G_{opt}.$$

Therefore,

$$(G_+ G_+^T)^{-1} G_+ \nabla U_{opt} = \begin{bmatrix} (G_{+,opt} G_{+,opt}^T)^{-1} G_{+,opt} \nabla U_{opt} \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\lambda} \\ 0 \end{bmatrix}.$$

Also note that $G_+^T (G_+ G_+^T)^{-1} G_+$ is a projection matrix, then

$$\|G_+^T v\| = \|G_+^T (G_+ G_+^T)^{-1} G_+ (\nabla U - \nabla U_{opt})\| \leq \|\nabla U - \nabla U_{opt}\| \leq (P/p)\hat{\sigma},$$

and

$$\|v\| = \|(G_{+}G_{+}^{T})^{-1}G_{+}G_{+}^{T}v\| \leq \|(G_{+}G_{+}^{T})^{-1}G_{+}\|\|G_{+}^{T}v\| \leq \psi(P/p)\hat{\sigma}.$$

Hence

$$\|(G_{+}G_{+}^{T})^{-1}G_{+}\nabla U - (G_{+}G_{+}^{T})^{-1}G_{+}\nabla U_{opt}\|_{\infty}$$

$$= \left\| \begin{bmatrix} \gamma_{+,opt}^{**} \\ \gamma_{+,rest}^{**} \end{bmatrix} - \begin{bmatrix} \boldsymbol{\lambda} \\ 0 \end{bmatrix} \right\|_{\infty} = \|v\|_{\infty} \leq \|v\| = \psi(P/p)\hat{\sigma}.$$

Recall that

$$\boldsymbol{\lambda} = (G_{+}G_{+}^{T})^{-1}G_{+}\nabla U_{opt} > \underline{\lambda},$$

therefore, if $\psi(P/p)\hat{\sigma} < (1/2)\underline{\lambda}$, i.e., $\hat{\sigma} < \frac{\underline{\lambda}}{2(P/p)\psi}$, one has

$$\gamma_{+,opt}^{**} > (1/2)\underline{\lambda}.$$

Given an initial condition $\boldsymbol{x}(t_0)$ and $\boldsymbol{c}(t_0)$, assume that for $t \geq t_0$, the vector $\dot{\boldsymbol{c}}(t) = 0$; i.e., all constraints are time-invariant after time t_0. This case is a time-invariant convex optimization problem and a special scenario of the time-varying problem (5.15). And this time-invariant problem has a unique bounded time-invariant optimal solution $\boldsymbol{x}_{opt}$.

LEMMA C.8. *For the above time-invariant problem, there exists finite time $\bar{t}$, such that, at time $\bar{t}$, $\boldsymbol{x}(t)$ "reaches" the right active constraint set; i.e., for some vector g being a column of G_{opt}^{T}, one has*

$$\left[G_{+,opt}^{T}(\bar{t}^{-}) \quad g \right] = G_{opt} \quad \textit{and} \quad G_{+,opt}(\bar{t}^{+}) = G_{opt}.$$

Proof of Lemma C.8. Without loss of generality, assume that at the optimal $\boldsymbol{x}_{opt}$, the Hessian matrix satisfies $-H_{opt} = I$[5]. Also assume the G_{opt} is an $m \times n$ matrix ($m \leq n$). For any constraint $g_i^T \boldsymbol{x} - c_i \leq 0$, such that g_i^T is not a row of G_{opt}; i.e., $g_i^T \boldsymbol{x}_{opt} - c_i > 0$, there exists a finite time t_i such that $g_i^T \boldsymbol{x}(t) - c_i > 0$ for $t > t_i$. So without loss of generality, we only consider the constraints $g_i^T \boldsymbol{x} - c_i \leq 0$ such that g_i^T is a row of G_{opt}. Let G_k be any $k \times n$ matrix such that $G_k \subset G_{opt}$, moreover, recall that G_{opt} is assumed to have linear independent rows, then G_k has linear independent rows. Let

$$\varrho = \min_{G_k, k=0,1,\ldots,m-1} \{\min_{\gamma_k \geq 0} \|\nabla U_{opt} - G_k^T \gamma_k\|\}, \quad \text{where} \quad \gamma_k = \begin{bmatrix} \gamma_1 & \ldots & \gamma_k \end{bmatrix}^T.$$

Note that there exists Lagrangian Multiplier vector $\boldsymbol{\lambda} > 0$ such that $\nabla U_{opt} = G_{opt}^T \boldsymbol{\lambda}$, one has

$$\varrho > 0.$$

Consider the set $\mathcal{X}^\epsilon$, a small neighborhood of $\boldsymbol{x}_{opt}$[6],

$$\mathcal{X}^\epsilon = \{\boldsymbol{x} : \|\nabla U(\boldsymbol{x}) - \nabla U_{opt}\| < \epsilon, \text{ and } \boldsymbol{x} \text{ is feasible}\}.$$

Assume that at $\boldsymbol{x} \in \mathcal{X}^\epsilon$, $G_k \boldsymbol{x} - c = 0$, where $G_k \subset G_{opt}$; i.e., $\boldsymbol{x}$ is not on the right constraint set, one has

$$L(\boldsymbol{x}) = \min_{\gamma_k \geq 0} \|\nabla U - G_k^T \gamma_k\|.$$

Moreover, note that

$$\lim_{\boldsymbol{x} \to \boldsymbol{x}_{opt}} L(\boldsymbol{x}) = \min_{\gamma_k \geq 0} \|\nabla U_{opt} - G_k^T \gamma_k\| > \varrho,$$

[5] If $-H_{x,opt}$ is not identity matrix, one may do a linear transformation $\boldsymbol{y} = (-H_{x,opt})^{1/2} \boldsymbol{x}$, then for the new problem, i.e. $\boldsymbol{y}$-problem, at optimal solution $\boldsymbol{y}_{opt} = (-H_{x,opt})^{1/2} \boldsymbol{x}_{opt}$, $-H_{y,opt} = I$.

[6] By Lemma C.3, without loss of generality, we only consider feasible point.

then for $(1/2)\varrho$, there exists a small enough $\epsilon > 0$ such that for any $\boldsymbol{x} \in \mathcal{X}^{\epsilon}$ not being on the right constraint set,

$$L(\boldsymbol{x}) > (1/2)\varrho.$$

Note that for $t > t_{0}$, all constraints are time-invariant, then NFC 1 does not hold for $t > t_{0}$, the function $L(t)$ decreases and converges to zero, so there exist finite time t_{ϱ} and t_{ϵ} such that,

$$L[\boldsymbol{x}(t)] < (1/2)\varrho, \quad t > t_{\varrho},$$

$$\|\nabla U[\boldsymbol{x}(t)] - \nabla U_{opt}\| < \epsilon, \quad t > t_{\epsilon}.$$

For $t > \max(t_{\varrho}, t_{\epsilon})$, if $\boldsymbol{x}(t)$ is not on the right active constraint set, there is a contradiction. So for $t > \max(t_{\varrho}, t_{\epsilon})$, $\boldsymbol{x}(t)$ is on the right active constraint set; i.e., there exists a finite time, such that $\boldsymbol{x}(t)$ "reaches" the right active constraint set.

LEMMA C.9. *Recall the definitions of p, P, $\underline{\lambda}$, ψ, and κ given in (5.16, 5.19, C.12, C.14). There exists a constant σ, such that, if $L(t_{0}) < \sigma$, then $\boldsymbol{x}(t_{0})$ is on the right active constraint set; i.e., $G_{+,opt}(t_{0}) = G_{opt}(t_{0})$.*

Proof of Lemma C.9.

- Recall the definitions of p, P, $\underline{\lambda}$, ψ and κ given in (5.16, 5.19, C.12, C.14), there exist a constant $\bar{\sigma}$ from Lemma C.5, and a constant $\hat{\sigma}$ from Lemma C.7, let $\sigma = \min(\bar{\sigma}, \hat{\sigma})$. Assume at some time t_{0}, $L(t_{0}) < \sigma$. Proceeding by contradiction, assume that $G_{+,opt}(t_{0}) \subset G_{opt}(t_{0})$; i.e., $\boldsymbol{x}(t)$ is not on the right active constraint set at time t_{0}.

- Assume for $t \geq t_{0}$, all constraints are time-invariant after time t_{0}. Then the optimal solution $\boldsymbol{x}_{opt}(t)$ is time-invariant after time t_{0}; i.e., $\boldsymbol{x}_{opt}(t) = \boldsymbol{x}_{opt}(t_{0})$ for $t \geq t_{0}$. This is a special scenario of problem (5.15). By Lemma C.8, there exists a finite time $\bar{t} > t_{0}$, such that $\boldsymbol{x}(t)$ "reaches" the right active constraint set at time $\bar{t}$.

- Note that for $t \geq t_0$, all constraints are time-invariant, NFC 1 does not hold, and $L[\boldsymbol{x}(t), t]$ is a decreasing function, one has $L[\boldsymbol{x}(\bar{t}), \bar{t}] < \sigma$. By Lemma C.5, $\gamma_g < (1/2)\underline{\lambda}$. Note that the γ_g is the entry with respect to g in vector $\gamma^{**}_{+,opt}(t^+_\sigma)$ given in (C.22), by Lemma C.7, $\gamma_g > (1/2)\underline{\lambda}$. There is a contradiction.
- So if $L(t_0) < \sigma$, then $G_{+,opt}(t_0) = G_{opt}(t_0)$, $\boldsymbol{x}(t_0)$ is on the right active constraint set.

C.4.3 *Follow*-problem $F(t_0)$

Recall that in Appendix C.3.4.2, one defines the "s_i-leaving" and "s_i-returning" events and discusses the event sequence properties. Then given an initial condition t_0, $\boldsymbol{x}(t_0)$, and $\boldsymbol{c}(t)$ for $t \geq t_0$, define $c^f_i(t)$:

$$c^f_i(t) = c_i(t), \quad t^{r,k-1}_i \leq t < t^{l,k}_i,$$
$$c^f_i(t) = g^T_i \boldsymbol{x}(t), \quad \text{otherwise.}$$

where $t^{r,0}_i = t_0$. And define an optimization problem referred to as $F(t_0)$, the *Follow*-Problem

$$\max \tilde{U}^f(\boldsymbol{x}, t) = U(\boldsymbol{x}) - \sum_{i=1,...,M} u_i s^f_i(\boldsymbol{x}, t), \tag{C.23}$$

$$s^f_i(\boldsymbol{x}, t) = g^T_i \boldsymbol{x} - c^f_i(t), \quad i = 1, ..., M,$$

$$u_i = \begin{cases} \alpha_i, & \text{if } s^f_i > 0, \\ 0, & \text{if } s^f_i < 0. \end{cases}$$

Note that the *Follow*-problem depends on time t_0. Recall that the original objective function $\tilde{U}(\boldsymbol{x}, t)$ is of this form

$$\tilde{U}(\boldsymbol{x}, t) = U(\boldsymbol{x}) - \sum_{i=1,\ldots,M} u_i s_i(\boldsymbol{x}, t).$$

So in the *Follow*-Problem, the hyperplanes $s_i^f(\boldsymbol{x}, t) = g_i^T \boldsymbol{x} - c_i^f(t)$ are defined and take the place of $s_i(\boldsymbol{x}, t) = g_i^T \boldsymbol{x} - c_i(t) = 0$. Note that $s_i(\boldsymbol{x}, t)$ and $s_i^f(\boldsymbol{x}, t)$ has the same gradient, only $c_i(t)$ and $c_i^f(t)$ are different. By the way the *Follow*-problem is constructed, the trajectory of the original problem $\boldsymbol{x}(t)$ and the trajectory of the *Follow*-problem $\boldsymbol{x}^f(t)$ satisfy

$$\boldsymbol{x}^f(t) = \boldsymbol{x}(t), \quad t_0 \leq t.$$

Moreover, due to the fact that NFC 1 never holds for the *Follow*-problem $F(t_0)$, the descent function of the *Follow*-problem $L^f[\boldsymbol{x}^f(t), t]$ is a decreasing function for $t \geq t_0$, and

$$\lim_{t \to \infty} L^f[\boldsymbol{x}^f(t), t] = 0.$$

C.4.4 Proof of Theorem 5.4

Let σ be given by Lemma C.9. By the result of Theorem 5.3, there exists a large enough ζ such that[7]

$$\frac{M \dot{c}_{max}}{\zeta p} \leq (1/2)\sigma.$$

Hence, there exits a finite t_σ such that $L(t) < \sigma$ for $t > t_\sigma$. Moreover, by Lemma C.9, one has $G_{+,opt}(t) = G_{opt}(t)$ for $t > t_\sigma$.

[7] Due to $U(\boldsymbol{x})$ only depends on $\boldsymbol{x}$, $Q = 0$.

Define the *Follow*-problem $F(t_\sigma)$ for $t \geq t_\sigma$. For any $t > t_\sigma$, let $G^f(t)$ be the active constraints set of the *Follow*-problem $F(t_\sigma)$. One has

$$G(t) \subseteq G^f(t).$$

LEMMA C.10. *Let $\boldsymbol{x}_{opt}(t)$ be the optimal solution of the original problem, and $\boldsymbol{x}^f_{opt}(t)$ be the optimal solution of the Follow-problem, then for $t \geq t_\sigma$*

$$\boldsymbol{x}^f_{opt}(t) = \boldsymbol{x}_{opt}(t).$$

Proof of Lemma C.10. Recall that the *Follow*-problem (C.23) is

$$\max \tilde{U}^f(\boldsymbol{x}, t) = U(\boldsymbol{x}) - \sum_{i=1,\ldots,M} u_i s^f_i(\boldsymbol{x}, t),$$

and the original problem is

$$\max \tilde{U}(\boldsymbol{x}, t) = U(\boldsymbol{x}) - \sum_{i=1,\ldots,M} u_i s_i(\boldsymbol{x}, t).$$

For any t, $\boldsymbol{\lambda}(t)$ is the Lagrangian Multiplier with respect to the optimal constraint set $G_{opt}(t)$; i.e.,

$$\nabla U[\boldsymbol{x}_{opt}(t)] = G_{opt}(t)^T \boldsymbol{\lambda}(t).$$

Note the following facts

$$G^f(t)\boldsymbol{x}_{opt}(t) = \boldsymbol{c}^f(t) \quad \text{and} \quad G_{opt}(t) \subseteq G(t) \subseteq G^f(t),$$

i.e., the point $\boldsymbol{x}_{opt}(t)$ belongs to the feasible set of *Follow*-problem, and for the *Follow*-problem, the KKT condition holds at $\boldsymbol{x}_{opt}(t)$, then

$$G^f_{opt}(t) = G_{opt}(t), \quad \text{and} \quad \boldsymbol{x}^f_{opt}(t) = \boldsymbol{x}_{opt}(t).$$

Note that the descent function of the *Follow*-problem $F(t_{\sigma})$, $L^{f}[\boldsymbol{x}^{f}(t), t]$ is a decreasing function for $t \geq t_{\sigma}$, and

$$\lim_{t\to\infty} L^{f}[\boldsymbol{x}^{f}(t), t] = 0,$$

it implies

$$\lim_{t\to\infty} \|\boldsymbol{x}^{f}(t) - \boldsymbol{x}^{f}_{opt}(t)\| = 0.$$

Note that

$$\boldsymbol{x}^{f}(t) = \boldsymbol{x}(t), \quad t_{\sigma} \leq t,$$

then

$$\lim_{t\to\infty} \|\boldsymbol{x}(t) - \boldsymbol{x}_{opt}(t)\| = 0 \quad \text{and} \quad \lim_{t\to\infty} L(t) = 0.$$

Bibliography

[1] *http://www.planet-lab.org*.

[2] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *ACM SIGOPS Operating Systems Review, Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 131–145, Oct 2001.

[3] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and Principles of Internet Traffic Engineering. In *RFC 3272*, May 2002.

[4] C. Chau. Policy-based Routing with Non-strict Preferences. In *ACM SIGCOMM2006*, Sept 2006.

[5] Z. Duan, Z. Zhang, and Y. T. Hou. Service Overlay Networks: SLAs, QoS and Bandwidth Provisioning. In *IEEE/ACM Transactions on Networking*, pages 870–883, 2003.

[6] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS Adaptive Traffic Engineering. In *IEEE INFOCOM 2001*, 2001.

[7] N. Feamster, J. Borkenhagen, and J. Rexford. Guidelines for interdomain traffic engineering. *ACM SIGCOMM Computer Communications Review*, Oct 2003.

[8] A. F. Filippov. *Differential Equation with Discontinuous Righthand Sides*. Kluwer Academic Publishers, 1988.

[9] S. J. Golestani and S. Bhattacharyya. A Class of End-to-End Congestion Control Algorithms for the Internet. In *IEEE International conference on Network Protocols (ICNP)*, pages 137–150, Oct 1998.

[10] M. Grötschel, S. O. Krumke, and J. Rambau, editors. *Online Optimization of Large scale Systems*. Springer, 2001.

[11] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley. Overlay TCP for Multi-Path Routing and Congestion Control. In *ENS-INRIA ARC-TCP Workshop*, Nov 2003.

[12] K. Ho, M. Howarth, N. Wang, G. Pavlou, and S. Georgoulas. Two Approaches to Internet Traffic Engineering for End-to-End Quality of Service Provisioning. In *poster in the Proceedings of the 1st EuroNGI Conference on Next Generation Internet Networks - Traffic Engineering*, Apr 2005.

[13] S. Kandula, D. Katabi, B. Davie, and A. Charney. TeXCP: Responsive yet Stable Traffic Engineering. In *ACM SIGCOMM*, 2005.

[14] K. Kar, S. Sarkar, and L. Tassiulas. A Simple Rate Control Algorithm for Max Total User Utility. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 133–141, Apr 2001.

[15] F. Kelly, A. Maulloo, and D. Tan. Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability. *Operations Research Society*, 1:237–252, Mar 1998.

[16] F. Kelly and T. Voice. Stability of End-to-End Algorithms for Joint Routing and Rate Control. *ACM SIGCOMM Computer Communication Review*, 2005.

[17] H. K. Khalil. *Nonlinear Systems*. Prentice Hall, 2002.

[18] R. La and V. Anantharam. Utility Based Rate Control in the Internet for Elastic Traffic. *IEEE/ACM Transactions on Networking*, 10:272–286, Apr 2002.

[19] C. Lagoa and H. Che. Decentralized Optimal Traffic Engineering for the Internet. *ACM SIGCOMM Computer Communication Review*, Oct 2000.

[20] C. M. Lagoa, H. Che, and B. A. Movsichoff. Adaptive Control Algorithms for Decentralized Optimal Traffic Engineering in the Internet. *IEEE/ACM Transactions on Networking*, 12(3):415–428, Jun 2004.

[21] Z. Li and P. Mohapatra. QRON: QoS-Aware Routing in Overlay Networks. *IEEE Journal on Selected Areas in Communications*, 22(1):29–40, Jan 2004.

[22] S. Low and D. Lapsley. Optimization Flow Control, I: Basic Algorithm and Convergence. *IEEE/ACM Transactions on Networking*, 7:861–874, Dec 1999.

[23] J. K. MacKie-Mason and H. R. Varian. Pricing Congestible Network Resources. *IEEE Journal on Selected Areas in Communications*, 13:1141 – 1149, Sept 1995.

[24] B. Movsichoff, C. Lagoa, and H. Che. Decentralized Optimal Traffic Engineering in Connectionless Networks. *IEEE Journal on Selected Areas in Communications*, 23:293–303, Feb 2005.

[25] B. Movsichoff, C. Lagoa, and H. Che. End-to-End Optimal Algorithms for Integrated QoS, Traffic Engineering, and Failure Recovery. *IEEE/ACM Transactions on Networking*, Oct 2007.

[26] B. A. Movsichoff. *Distributed Traffic Engineering: a Sliding Mode Approach*. PhD thesis, Dept. Electical Engineering, Pennsylvania State University, U.S.A, 2004.

[27] I. Okumus, J. Hwang, S. Chapin, and H. Mantar. Inter-Domain Traffic Engineering on a Bandwidth Broker Supported DiffServ Internet. *Computer Communications Journal*, 2004.

[28] B. Polyak. *Introduction to Optimization*. Optimization Software, Inc. Publication Division, New York, 1987.

[29] A. Y. Popkov. Gradient Methods for Nonstationary Unconstrained Optimization Problems. *Automation and Remote Control*, 6:883–891, 2005.

[30] J. Shu and P. Varaiya. Pricing Network Services. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1221–1230, 2003.

[31] W. Su, C. M. Lagoa, and H. Che. A Family of Optimization-Based Traffic Control Laws for Overlay Networks. In *Proceeding of the 46thIEEE Conference on Decision and Control*, Dec 2007.

[32] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz. Over QoS: Offering Internet QoS Using Overlays. In *Proceedings of HotNet-I Workshop*, Oct 2002.

[33] V. I. Utkin. *Sliding Modes in Control and Optimization*. Springer-Verlag, 1992.

[34] H. Wang, H. Xie, Y. Yang, L. E. Li, Y. Liu, and A. Silberschatz. Stable Egress Route Selection for Interdomain Traffic Engineering. In *13th International Conference on Network Protocols (ICNP '05)*, Oct 2005.

[35] W. Wang, M. Palaniswami, and S. H. Low. Optimal Flow Control and Routing in Multi-path Networks. *Performance Evaluation Journal*, 2003.

[36] W. Xu and J. Rexford. MIRO: Multi-path Interdomain ROuting. In *ACM SIGCOMM2006*, Sept 2006.

[37] W. Zangwill. Nonlinear Programming via Penalty Functions. *Manag. Sci*, 13:344–358, 1967.

[38] Y. Zhao and W. Lu. Training Neural Networks with Time-varying Optimization. In *Proceedings of 1993 International Joint Conference on Neural Networks*, pages 1693–1696, 1993.

Vita

Wenjing Su was born in Beijing, P.R. China. She received the B.S. and M.S. degrees from the Tsinghua University, P.R.China in 2000 and 2003 respectively. In Fall 2003, she joined the Ph.D. program in Electrical Engineering at The Pennsylvania State University and started her research under the supervision of Professor Lagoa. Her research interests include time-varying optimization and the application of control theory and optimization to computer networks.