Distributed Estimation and Control in Networked Systems

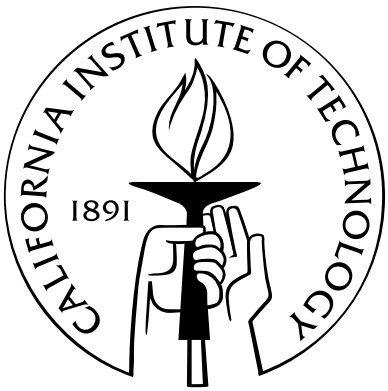
Thesis by

Vijay Gupta

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2006

(Submitted August 14, 2006)

Acknowledgements

Mere words cannot express the gratitude I owe to all the people who have made it possible for me to write this dissertation. Prof. Richard Murray, with his unquenchable enthusiasm and infectious spirit of hard work, has been a great advisor all along. Your ability to see the big picture and your emphasis on being aware of where one wants to go have been invaluable. Prof. Babak Hassibi not only taught me the fundamentals of estimation theory but also helped me broaden my horizons beyond the particular control/estimation problem I was considering. Your technical virtuosity and emphasis on the details have always amazed and inspired me.

The greatest joy of my academic life at Caltech was being able to collaborate with many brilliant people. Sometimes we were successful in solving the problem we set out for and sometimes not, but it was always a great learning experience. Prof. Cedric Langbort, now at UIUC, reignited my interest in distributed control and was an invaluable resource as we started traversing the path of robustness in multi-agent systems. Amir Farajidana not only collaborated with me on many problems, but was also on a constant lookout to prevent me from floating wild incorrect conjectures. Prof. Joao Hespanha, UCSB, was instrumental in my getting interested in the area of estimation over networks. Tim Chung and I spent many afternoons immersed in multi-sensor problems and he was a willing resource for any problem that I wanted to talk about. Demetri Spanos, with his ability to cut to the chase, was particularly helpful when I was formulating my initial ideas on the problem of information processing for estimation. Prof. Leonard Schulman was always ready to share his amazingly clear and insightful thoughts on any problem that I wanted to discuss. I would also like to thank Prof. Massimo Franceschetti, UCSD, and Dr. David Jeffcoat, AFRL, for their contribution to my rich academic experience at Caltech.

One reason I could survive the grind of PhD life year after year was the support network at Caltech that I could tap into. The International Students Program (especially Jim Endrizzi, Athena Trentin and Tina Lai), Caltech Y (especially Greg Fletcher and the many program leaders including Kai Shen, Dan Feldman and Spencer Mortensen), OASIS people (especially Meher Ayalasomayajula, Harish Manohara and Deepshikha Datta), TACIT, SEDS, OAT and the Caltech TSD club have all allowed me to view research as just one part of life. The many friends I met here are too numerous to mention, although some of them appear in different guises above. I think thanking you personally

would be more appropriate than simply listing your names here.

And finally, I express my appreciation for my family. It has always been a pillar of strength, inspiration and support. In particular, I would like to thank my parents: I will never be able to repay the unconditional love and patience that you have always showed towards me. As an inadequate but heartfelt token of gratitude, I dedicate this dissertation to you.

Abstract

Rapid advances in information processing, communication and sensing technologies have enabled more and more devices to be provided with embedded processors, networking capabilities and sensors. For the field of estimation and control, it is now possible to consider an architecture in which many simple components communicate and cooperate to achieve a joint team goal. This distributed (or networked) architecture promises much in terms of performance, reliability and simplicity of design; however, at the same time, it requires extending the traditional theories of control, communication and computation and, in fact, looking at a unified picture of the three fields. A systematic theory of how to design distributed systems is currently lacking.

This dissertation takes the first steps towards understanding the effects of imperfect information flow in distributed systems from an estimation and control perspective and coming up with new design principles to counter these effects. Designing networked systems is difficult because such systems challenge two basic assumptions of traditional control theory - presence of a central node with access to all the information about the system and perfect transmission of information among components. We formulate and solve many problems that deal with the removal of one, or both, of these assumptions. The chief idea explored in this dissertation is the joint design of information flow and the control law. While traditional control design has concentrated on calculating the optimal control input by assuming a particular information flow between the components, our approach seeks to synthesize the optimal information flow along with the optimal control law that satisfies the constraints of the information flow. Thus besides the question of '*What should an agent do?*', the questions of '*Whom should an agent talk to?*', '*What should an agent communicate?*', '*When should an agent communicate?*' and so on also have to be answered. The design of the information flow represents an important degree of freedom available to the system designer that has hitherto largely been ignored. As we demonstrate in the dissertation, the joint design of information flow and the optimal control input satisfying the constraints of that information flow yields large improvements in performance over simply trying to fit traditional design theories on distributed systems.

We begin by formulating a distributed control problem in which many agents in a formation need to cooperate to minimize a joint cost function. We provide numerical algorithms to synthesize the optimal constrained control law that involve solving linear equations only and hence are free from

numerical issues plaguing the other approaches proposed in the literature. We then provide and analyze a model to understand the issue of designing the topology according to which the agents interact. The results are very surprising since there are cases when allowing communication to happen between two agents may, in fact, be detrimental to the performance.

We then move on to consider the effects of communication channels on control performance. To counter such effects, we propose the idea of encoding information for the purpose of estimation and control prior to transmission. Although information theoretic techniques are not possible in our problem, we are able to solve for a recursive yet optimal encoder / decoder structure in many cases. This information flow design oriented approach has unique advantages such as being optimal for any packet drop pattern, being able to include the effect of known but random delays easily, letting us escape the limits set by reliability for transmission of data across a network by using intermediate nodes as 'repeaters' similar to a digital communication network and so on.

We finally take a look at combining the effects of multiple sources of information and communication channels on estimation and control. We look at a distributed estimation problem in which, at every time step, only a subset out of many sensors can transmit information to the estimator. This is also a representative resource allocation problem. We propose the idea of stochastic communication patterns that allows us to include the effects of communication channels explicitly during system design. Thus, instead of tree-search based algorithms proposed in the literature, we provide stochastic scheduling algorithms that can take into account the random packet drop effect of the channels. We also consider a distributed control problem with switching topologies and solve for the optimal controller. The tools that we develop are applicable to many other scenarios and we demonstrate some of them in the dissertation.

Along the way, we look at many other related problems in the dissertation. As an example, we provide initial results about the issue of robustness of a distributed system design to a malfunctioning agent. This notion is currently lacking in the control and estimation community, but has to be a part of any effective theory for designing networked or distributed systems.

Contents

List of Figures

Chapter 1

Introduction

In its traditional avatar, control theory deals with the following problem. As shown in Figure 1.1, consider a dynamical process whose state needs to be controlled; for instance, a mobile robot which needs to follow a given trajectory. The state of the dynamical process (e.g., the position and velocity of the robot) is observed by a sensor that generates possibly noisy measurements. The measurements are processed by an estimator that generates an estimate of the state of the process. This estimate is fed to a controller that aims at minimizing some cost function. In the mobile robot example, this can correspond to, say, deviation from a reference trajectory. The controller generates a control input that is applied to the process through an actuator. Over the last century or so, control theory has made spectacular advances both in understanding the underlying mathematical tools as well as in applying this model to many different problems and generating technological advances.

However, the very success of this model has meant that the many assumptions inherent in the above block-diagram are often forgotten. As an example, one can spell out such assumptions as

- measurements are generated periodically by the sensor and transmitted to the estimator;
- the estimator and controller are able to process all the data that is generated;
- all the components run on the same clock;

and many others. At the heart of most of these features is the implicit assumption that almost unlimited amounts of information can be communicated and processed by the system components. In most traditional control systems, this assumption largely held true. However, it is still somewhat surprising that more efforts have not been made to characterize and counter the effects of limited and imperfect communication sharing and processing, especially given the fact that, at its heart, control theory is an *information* science. The basic motif of feedback loop has as its underlying philosophy the use of information about the system to counter the effect of uncertainty in the environment[1].

[1]Admittedly, concepts like observability and measurement noises deal with the problem of limited information about the system. However, the assumptions of perfect information transmission and processing are commonly made.

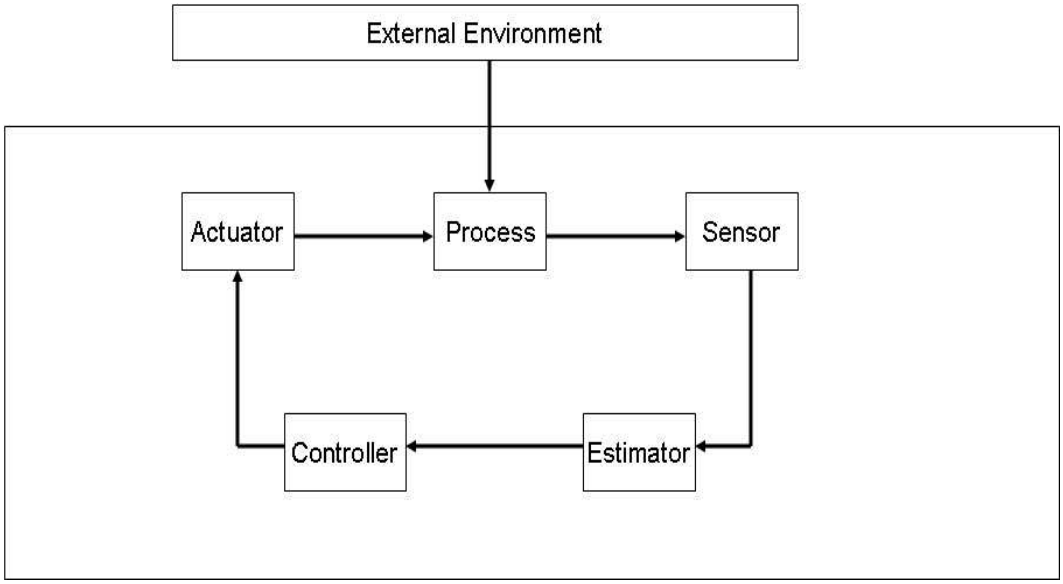


Figure 1.1: A basic control loop architecture.

In recent years, a compelling need has arisen to understand the effects of information flow on estimation and control more fully. Technological advances in hardware and software over the past few decades have enabled cheap and small, yet powerful, communication and computation devices. This development, in turn, has made it possible to envisage systems in which multiple simple components cooperate to achieve a joint team goal. One can imagine three possible design architectures for such *multi-agent* systems:

1. Every component may be asked to share all its information with every other component in a reliable fashion and all agents provided with ample computation power to process all this information to make (the same) globally optimal decisions. Even though multiple *physical* components may be present, this is still an example of a single agent system as far as the design is concerned. There is a rich theory already available to design such systems (modulo computational issues). We will refer to this architecture as a centralized system architecture. A simple example would be a group of aircrafts tasked with flying in a formation while either being allowed to communicate as much information as they desire with every other aircraft or communicating this information to a central ground unit that sends back the control inputs that they need to apply.

2. At the other end of the spectrum, all the components may be asked to depend only on the information that they possess to make their decisions. Thus, even though the components may have a joint team goal, they do not share information and instead try to coordinate their decisions by appealing to some correlated property of the environment that they are all observing. We will refer to this architecture as a completely decentralized architecture. Design

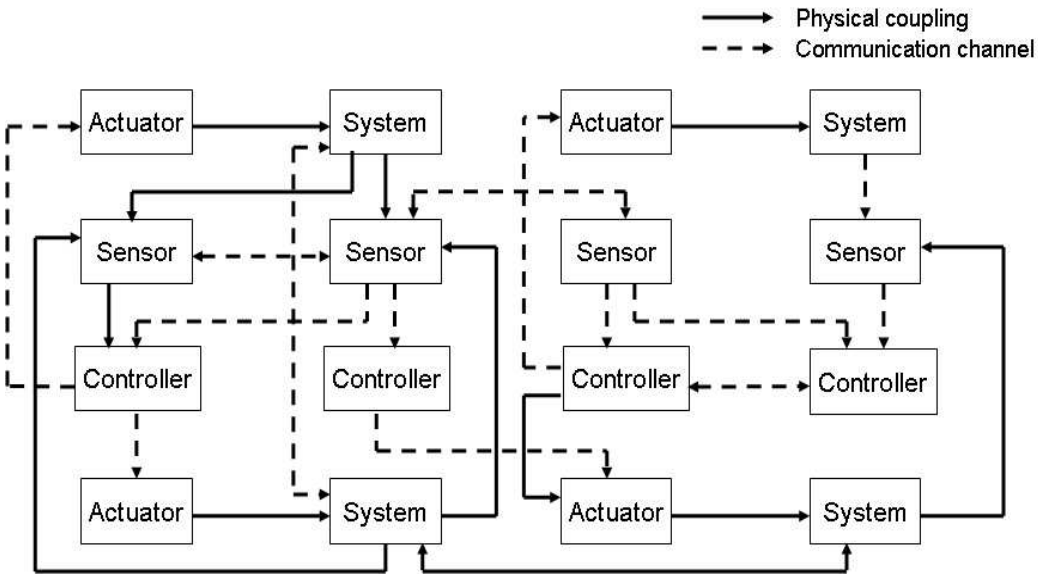


Figure 1.2: A possible control architecture for networked / distributed systems.

of such systems reduces to design of multiple single agent systems. An example would be a group of aircrafts tasked with monitoring a region for threats where every aircraft generates its trajectory as if it is the only aircraft present.

3. Every component may be allowed to share information with some other components. Thus, the components try to coordinate their behavior by transmitting some of their information (and possibly plans) to some other components. We will refer to this approach as a cooperative, networked or distributed system architecture. An example might be a group of aircrafts asked to achieve a common heading with every aircraft being allowed to observe the headings of the aircrafts only in its immediate vicinity[2].

The distributed system architecture, were it to work, is very powerful since it allows the design of the individual units or components to be much simpler, while not compromising too much on the performance. Additional benefits include increased robustness to component loss, increased flexibility in that the components can be reconfigured for many different tasks and so on. However, the design of such systems challenges the assumption of information flow being unrestricted and reliable, that, as mentioned above, is at the heart of the traditional systems and control theory.

A typical design diagram for a networked system is shown in Figure 1.2. Such systems consist of the same components as a conventional system — dynamic processes, sensors, controllers etc. — but there are many of them. Thus, there are many processes that need to be controlled. Their states are being sensed by many sensors. There is no one-to-one correspondence between processes and sensors. Many sensors might observe the same process and sensors may even switch from one

[2]This task is often referred to as a consensus or an agreement task.

process to another. The sensors may share information among themselves and also transmit it to the controllers. There might be many controllers with varying information and goals. A simple example is an air traffic control situation where the controller in the control tower can observe all the aircrafts but cannot micro-manage each one of them. On the other hand, the controller on-board the aircraft has limited information but has access to the actuators on-board the aircraft. All the components need to cooperate to achieve a team goal or to optimize a joint cost function and each entity can communicate with some other entities. The aspect of information as a central currency in the system is underlined by many new features that these systems present:

1. Information richness: Because of so many sensors being present and entities transmitting information to each other, the system as a whole is *information rich*. At the same time, however, every component has access to a different information set (or has a different *world-view*).

2. Information sharing: Since they need to cooperate, the components have to reach some level of agreement and hence they need to communicate and share information. Because of the information-rich aspect, sharing of *all* the information is not feasible. If indeed all information could be shared among all the components, the situation would revert to a centralized architecture.

3. Presence of imperfect communication links: Information sharing needs to be carried out over imperfect communication channels that have numerous limitations. As an example, such links may erase information stochastically, corrupt it, introduce random delays and so on.

4. Limited information processing capability: Components have limited information processing capabilities. Because of the huge amount of information being generated, components may need to decide which information to access and process at any time step.

These assumptions introduce information flow as both an enabler and a constraint in the system design. It is the sharing of information that allows components to cooperate and achieve better performance than that achieved by completely decentralized systems. However, the information flow is not perfect. There is too much information being generated so that all of it cannot be communicated or processed (information overload for the system); at the same time, components may not have access to the information they need to make globally optimal decisions because of constraints on the information flow (information paucity for the decision makers). These features are, by and large, not tackled in traditional systems and control theory and thus the design of such networked systems is quite different from conventional system design.

To fully exploit the benefits promised by such systems, there is, thus, a need to address the issues involved in networked system design and come up with a systematic theory for it. The design

procedures at the moment tend to be ad hoc and do not allow us to make intelligent choices through an understanding of the inherent trade-offs and capabilities of the system. This dissertation provides a new approach and new design principles to address some of the issues that are faced while designing networked systems from an estimation and control perspective.

Besides the motive stated above, studying control and estimation problems in networked systems is useful for another reason. Since networked systems couple *flow* and *processing* of information with its *use* to estimate and control dynamic processes, a theory for such systems can be expected to unify concepts from information and communication theory, distributed and parallel computation as well as estimation and control theory. In other words, by focusing on areas at the intersection of traditionally separately studied fields of communication, computation and control, networked systems can shed additional insight into all these fields.

As an instance, traditionally, information theory has grappled with the problem (as enunciated by Shannon [176]) of 'reproducing at one point either exactly or approximately a message selected at another point.' While this field has had fascinating success in dealing with that question, it is also useful to recall a central assumption often made in it. Once again, to quote Shannon, 'Frequently the messages have meaning ... these semantic aspects of communication are irrelevant to the engineering problem.' This divorce of meaning of information from the reward obtained by transmitting it allows us to treat all information content alike. Furthermore, it allows long periods of use of a communication channel and renders long delays irrelevant as long as the information is communicated. If we have to study the complexities that arise because of coupling between the flow of information and its use to estimate or control dynamical processes, we clearly cannot use results based on information theory in an off-the-shelf manner. As a simple instance, the same message may lose its value drastically if it is communicated to the controller with delay. This issue was eloquently raised by Witsenhausen [203] who used the term 'information pattern' to denote the flow of information between various components[3] and pointed out that the minimum cost achievable in a problem depends upon the prevailing information pattern. Thus, he suggested, information should be measured by its effect on optimal cost. This idea is a major extension of the traditional information theory, but is crucial if we want to fully grasp the interplay between information and dynamical systems. Networked control problems provide a natural source of problems in which such interplay happens.

Similarly, the fields of parallel processing and distributed computation have made amazing progress in dealing with problems such as speeding up computation by intelligently dividing it into operations that can be done concurrently by different processors or in calculation of functions of data held by many different processors. However, often these theories also make assumptions about all processors having access to a common memory unit or all processors being connected such that

[3]We will use either the term 'information flow' or 'information pattern' to denote this concept.

every processor can communicate with any other processor. Networked systems not only lack such features, they also pose problems of dynamics at the component level. Thus, components need to make decisions given whatever information they have without waiting for the coordination algorithm to finish. Traditional methods from parallel processing and distributed computation too need to be extended to fully understand the nexus between computation and control.

Thus, networked systems provide a playground where the traditional assumptions of communication, computation and control are challenged. Any progress made in understanding the analysis and synthesis of such systems would then necessarily expand our knowledge about all these fields. In this dissertation, we take a control and estimation oriented viewpoint and address some of the problems associated with the information flow.

1.1 Contributions of the Thesis

From the control and estimation perspective that we take in this dissertation, there are two reasons why including the effects of information flow makes the problem of designing a distributed system difficult.

1. Information is now diffused throughout the system. Different components have access to different information sets and must base their decisions on them. There might be heterogeneous processing capabilities as well. Since the decisions aim at optimizing a *joint* cost function, they have to be coordinated. Coordinating decisions when a central authority is lacking and the various decision makers have access to different information sets is a difficult and open problem.

2. Information flow is also now subject to the vagaries of communication channels. Thus, information may be lost, delayed or otherwise corrupted. How to use the received information optimally, as well as how to choose what information to transmit are open problems. These issues can have an immense effect on the estimation or control performance of the system.

In this dissertation, we take first steps towards attacking both these problems. We give specific contributions of each chapter in the corresponding introduction section later. Let us, however, briefly summarize the intellectual thrusts of the dissertation here.

The chief idea presented and explored in this dissertation is the *joint design of information flow and the control law* in cooperative multi-agent systems. Traditionally, control design has been limited to calculating the optimal control input given the dynamics of the agent and the cost function that needs to be minimized. Thus the focus has been on the question '*What should an agent do?*'. In systems that involve multiple agents and imperfect information flows, some attempts have also been made recently to analyze the performance hit incurred by the system. Our approach is fundamentally different in that it seeks to synthesize the optimal information flow and the control law jointly. Thus

besides the question of '*What should an agent do?*', the questions of '*Whom should an agent talk to?*', '*What should an agent communicate?*', '*When should an agent communicate?*' and so on also have to be answered. Of course, the information flow has to be designed within the constraints imposed by the environment, in particular, the effects introduced by the communication channel. However, the design of the information flow still represents an important degree of freedom available to the system designer that has hitherto largely been ignored. As we will demonstrate, the joint design of information flow and the optimal control input satisfying that information flow will yield large improvements in the system performance.

We begin in Chapter 2 by focussing on the diffuse nature of information in the system. We tackle two chief problems:

1. Given an information flow that specifies the topology according to which the components communicate, what is the optimal control / decision policy to be followed by the components? This is the question of '*What should an agent do?*'

2. How do we design the topology? This is the question of '*Whom should an agent communicate with?*'

Design of the optimal control law for arbitrary information flows is known to be NP hard [21, 159]. Thus, we are only able to provide numerical algorithms for the first problem. However, our algorithms involve solving linear equations only and hence avoid the convergence problems associated with other methods that exist in the literature. To be able to design the topology and the information flow, we need notions of value of an information flow. Such measures are highly non-additive and yield counter-intuitive results. We adopt a model where every communication edge that allows information to be shared between two components comes at an additional cost. While we prove that the recent results in the literature about loss of stability by adding new edges [59] are outcomes of the particular control law that those works assume (meaning that if the optimal control law is used, all information flow patterns have the same stabilizability properties), we also show that under this model, there are cases when adding edges is *harmful* for performance for any positive value of edge cost. In some other cases, we provide conditions on the edge cost that make adding or deleting edges beneficial. These results suggest that we need to take the folk-lore of 'cooperation is always better' with a pinch of salt.

In Chapter 3, we begin to focus on the malevolent effects of communication channels on the information flow, and hence, the performance. We apply the idea of encoding the idea prior to transmission, used in information theory, for estimation and control problems. Thus, moving away from the viewpoint prevailing in the literature about simply characterizing the performance loss when a communication channel is imperfect, we pose the question '*What should an agent communicate?*' to counter the effects introduced by the imperfect communication channel. However, while the *idea*

can be borrowed, the *techniques* from information theory are not applicable directly, because of the assumptions about delays and multiple uses of the channel mentioned already. For many cases, we develop encoding and decoding algorithms that satisfy the constraints imposed by networked control systems and are optimal for the purpose of estimation and control. Our approach differs drastically from the existing viewpoint of control theory, which assumes fixed and given transmission quantities and yields huge improvements both in terms of stability margins and performance. It also allows us to draw an analogy between the problem of transmitting information for control and digital communication. Thus, we can use any component that is relaying information in a similar manner as a repeater in digital communication to fight the degradation in the performance due to channels.

In Chapter 4, we begin to bring both the aspects together. We consider systems that involve both a large number of components as well as communication over imperfect channels. We begin by focussing on the problem of presence of multiple access channels that allow only one component to communicate its information at every time step. To answer the question of '*Which agent communicates when?*', we propose the idea of stochastic communication patterns that allow us to explicitly include the effect of communication channels during system design. We illustrate the use of such communication patterns in a distributed estimation problem and in a problem of distributed control with switching topologies. This approach also allows us to escape the curse of dimensionality, that will afflict us for systems comprising of multiple components in any resource allocation problem, if we use optimization procedures (such as tree search) that have been proposed in the literature.

Finally, in Chapter 5, we identify the next set of problems that need to be attacked and present some initial results on a few of these problems. For instance, we introduce the notion of robustness for networked systems. The notion is different from the usual concept of robustness in control theory which deals with imperfect knowledge of the process model. Although some features are common between the two notions, it is also different from the usual model of agent failure used in distributed computation. This is on account of the dynamics present in our problem as well as because of constraints on the information flow that we assume. We analyze some control algorithms proposed in the literature and show that this concept needs to be considered during the system / algorithm design.

1.2 Organization of the Thesis

The dissertation has been written so that each chapter can be read and understood in isolation. As discussed above, control and estimation in networked systems is difficult since such systems challenge two standard assumptions in estimation and control theory — presence of a central decision making unit with access to all the information being generated in the system; and ability to communicate any information reliably with an arbitrarily high degree of accuracy. Accordingly, in the dissertation,

we attack the problem on both the fronts. The approach we follow is a joint design of the control law and the information flow.

We begin in Chapter 2 by focusing on the first question. We consider the problem of distributed control of a formation of several dynamic agents[4] that need to cooperate with each other to minimize a joint cost function. The formation has a specified topology that determines the set of agents with which an agent can exchange information. Communication effects are largely ignored in this chapter with a communication link, if present, being modeled as perfect. In Section 2.4, we present a synthesis algorithm for the control law that minimizes a quadratic cost function for a finite time horizon case. The problem is known to be NP-hard in general, and hence we are able to present only a sub-optimal algorithm. However, the algorithm is interesting because it involves solving only a set of linear equations and hence is free from convergence issues plaguing other numerical approaches considered in the literature. This takes us to the question of role of the topology in distributed control in Section 2.6. We pose the question of jointly designing the topology and the control law to be followed by the agents. Obviously, if edges can be added at no extra cost, the optimal topology would be one in which every agent can communicate with every other agent. However, adding more edges may impose an additional cost to the system by, e.g., entailing more communication. We propose and analyze a new model for determining the influence of interconnection topology in distributed control. The results are surprising, as there are cases when adding any edge is detrimental for any link cost that is positive. In some other cases, we have a linear matrix inequality to find the link cost at which it ceases to be beneficial to add an extra edge. The chapter has two appendices in which we discuss related problems. In the first appendix, we discuss an alternative synthesis algorithm for the distributed control problem that is suitable for the infinite horizon version of the problem. In the second appendix, we discuss a distributed estimation problem. We consider the problem of active sensing using mobile sensor nodes that are jointly estimating the state of a dynamic target as a sensor network and provide a gradient search-based distributed algorithm for controlling their motion.

In Chapter 3, we shift gears to focus on the effects of communication channels on estimation and control. In most of the work presented in this dissertation, we model a communication channel using the packet erasure model. This model is discussed briefly at the beginning of the chapter in Section 3.2. The novelty in our approach is to pose the problem as an information transmission problem. This perspective of jointly designing the quantity to be transmitted along with the control law that uses this quantity optimally allows us to solve the Linear Quadratic Gaussian (LQG) problem in the face of packet drops. As the first step, in Section 3.3.2, we extend the familiar LQG separation principle to this problem. This principle allows us to solve the LQG control problem

[4]We refer to a dynamical process with a local sensor and a controller as an agent. A formation is simply a collection of such agents that are provided with a joint cost function that they need to minimize.

using a standard LQR state-feedback design along with an algorithm for propagating and using the information across the unreliable links. For many cases, we then provide such information processing algorithms that are recursive, yet optimal. Our design has many nice properties such as not assuming any statistical model of the packet drop events and being able to handle packet delays among others. Further, the solution is appealing from a practical point of view because it can be implemented as a small modification of an existing LQG control design. In Section 3.4, we consider the case of a single sensor transmitting information across a single link. In Section 3.5, we consider the problem of estimation across an arbitrary network of packet erasure channels. We propose the optimal information transmission algorithm and analyze its stability and performance. In Section 3.6, we extend the result to the case of multiple sensors with only one sensor transmitting over a packet erasure channel. We see that a naïve extension of the algorithm for a single sensor is not optimal. We provide an alternative algorithm that is optimal and analyze its properties. In the first appendix of the chapter, we consider quantization as another effect that communication channels may introduce. While stability of the plant under quantization strategies has been looked at, the performance of a control loop under quantization is largely ignored in the literature. We look at the performance aspects under the high rate assumption. In the second appendix, we look at random delays introduced by the channel.

In Chapter 4, we consider some problems that have aspects of both distributed estimation / control and communication links present. We begin by considering the situation when only one (or a subset) out of multiple sensors can take a measurement at every time step and communicate it over a packet erasure channel to an estimator. In keeping with our philosophy throughout the dissertation, we not only design the optimal estimator that utilizes the measurements available to it, but also look at the problem of constructing the optimal schedule for the sensors to communicate. We propose a stochastic selection strategy that, unlike the standard tree-based methods proposed in the literature, is explicitly able to take into account the packet dropping nature of the channel. We then consider a control problem in which the topologies are switching stochastically. This can model, e.g., stochastic information loss due to communication links. We solve for the optimal control law and analyze the performance of the system under it. The tools that we develop in the chapter are useful in various other problems. As an example, in the first appendix, we apply them to another resource allocation problem. We show how our algorithm can be used for the problem of sensor coverage in which multiple mobile sensors need to cover a geographical area. In the second appendix, we apply the tools to the problem of using Multiple Description (MD) coding to transmit data over a channel to improve control performance.

In the final Chapter 5 of the dissertation, we outline some directions in which more work is needed to obtain an effective theory of networked control / estimation systems. We begin by considering future directions from each of the three previous chapters. Then, we pose and partially analyze

the problem of robustness of a distributed estimation / control algorithm. This problem has not been considered in the literature so far. However, we argue that this problem is of fundamental importance and provide initial steps to tackle it.

Chapter 2

Distributed Estimation and Control in Formations of Dynamic Agents

This chapter deals with the problem of distributed estimation and control when there are many dynamic agents present. If agents share information among each other according to some topology, both the questions of optimal control law for a given topology ('*What should an agent do?*') and how to design the topology ('*Whom should an agent talk to?*') become important. We look at these questions in this chapter. The work presented in this chapter has partly appeared in [35, 82, 85, 123].

The chapter is organized as follows. We begin by introducing the problem and presenting the relevant prior work in Section 2.1. In Section 2.2, we present the notation we are going to use. Then, in Section 2.3, we set up the constrained controller synthesis problem. In Section 2.4, we present our solution to this problem for the finite horizon case. We present both an optimal but computationally intensive solution and a simpler sub-optimal solution. Section 2.5 contains some examples to illustrate the concepts and the algorithm. In Section 2.6, we move on to the problem of determining the influence of the topology of the graph. We first define the associated notions of value of a graph and our cost function model. Then, in Section 2.7, we analyze this model and present our results. Section 2.8 presents a short numerical example to explain our results. The first appendix presents an algorithm for synthesizing a sub-optimal controller to minimize an infinite-horizon LQ cost. In Appendix B, we consider the problem of distributed sensor motion control for estimation.

Contributions

The main contributions of the chapter are now summarized.

1. We present two algorithms for the finite-horizon version of the distributed control problem: one computationally expensive method for synthesizing the optimal controller and another more tractable method for synthesizing a sub-optimal controller. Both the algorithms rely only on

solving linear equations and hence are free from the numerical issues about convergence that afflict other approaches proposed in the literature.

2. We formally pose the problem of evaluating the role of topology in distributed control. We present a model for the problem and present initial analytic results. For instance, we prove that under certain assumptions, the optimal control topology according to our criterion is the fully decentralized one. We also present linear matrix inequalities (LMIs) to calculate the 'critical prices' at which it becomes detrimental to add new edges to an existing topology.

3. We present a simple gradient search based algorithm for the infinite-horizon version of the distributed control problem. Using the special structure of the problem, we are able to get rid of the problem of choosing the initial guess point prevalent in similar approaches proposed in the literature. Also, we prove the interesting result that a formation is controllable and/or stabilizable for any given information topology if and only if all the individual agents are controllable and/or stabilizable using their own inputs.

4. For the distributed target tracking problem, we propose a simple distributed algorithm for planning the motion of sensors in a network to achieve significantly better estimates of the target state.

2.1 Introduction

One of the major challenges in the problem of estimation and control of networked systems is the absence of a central controller that has access to information from all the agents and uses this information to compute the control signals that the agents should use. Presence of an information flow network so that an agent is allowed information only about some other agents to calculate its control input introduces fundamentally new features into the problem. The topology of the information flow can have many effects. On one hand, it may introduce instability if the information being fed through the network adds on constructively to the disturbance at a node [60]; on the other, for cooperative goals, it leads to a better performance than if agents do not share information.

Understanding and solving this problem is an important step in building a theory for control of networked systems. As a result, the problem of controlling a formation of dynamic agents where the information flow is specified by a topology has been garnering increasing attention (e.g., see [3] and the references therein). A Nyquist-like condition for stability of a formation using the individual plant transfer function and the Laplacian of the graph describing the topology of the information flow network was obtained by Fax and Murray in [59, 60]. Coordination of a group of autonomous agents when the graph topology changes over time was considered in Jadbabaie et al. in [107], who presented stability results for the case when the switching rule satisfies certain properties. These

results were expanded by Ren and Beard in [165]. A general framework for analysis of stability of interconnected systems where the topology can potentially be time-varying was presented in [81]. The effect of changing topology for the special case of consensus problems was also considered in [157].

However, most of the work in the literature, so far, has centered on stability analysis of the formation assuming certain control laws in place. A more general question is that of synthesis of the control law to be used by the agents in such a formation, such that some joint cost function is optimized. The defining feature of the problem is that while the cost function involves all the individual agents in the formation, the pre-specified topology of the formation imposes constraints on the form of the control law by limiting the information available to various agents at any time. Thus, it is not realistic to assume that an agent would know the state of all the other agents in the formation at any given time and be able to use it to calculate the control input. These features make the problem a distributed control problem with arbitrary information flow patterns, which is, in general, much harder to solve than the traditional optimal control problem.

Research in distributed control has a long history (see, e.g., [53]). Witsenhausen, in his seminal work [202, 203], showed that under distributed information constraints, a linear controller is not optimal in general and also that the cost function need not be convex in the controller variables. A discrete equivalent of Witsenhausen's counter-example was given in [21, 159] where it was also shown that the problem of finding a stabilizing controller under information pattern constraints is NP-complete. For particular information structures, the problem has been solved. Witsenhausen himself in [203] identified some cases where the standard LQG theory can be applied, and hence, the optimal controller is linear. Another important early contribution in this direction was the work of Ho and Chu [99], who looked at this problem in the context of team decision theory. That work identified a class of information patterns called *partially nested* structures under which the optimal controller is linear. Roughly speaking, under a partially nested structure, if the decision that controller A makes can affect the decision made by another controller B, then the controller B should have access to all the information that controller A has access to. Another information pattern for which the optimal controller is known to be linear is the one-step delayed information sharing pattern in which every controller has, at the current time, access to all the previously implemented control values and all the observations made by any other agent in the system through and including the previous time, and its own observations through the current time. Recursive solutions for this problem were provided for a quadratic cost by Sandell and Athans in [171], for an exponential cost by Fan et al. in [58] and for H_2, H_∞ and L_1 costs by Voulgaris in [195]. For some other tractable information patterns, see [8, 197, 206]. Some researchers have also studied this problem under the assumption of spatial invariance by using a multidimensional approach, e.g., see [7, 42]. Most of these special tractable information patterns were united by Rotkowitz and Lall in [169], who defined

a property called *quadratic invariance* and showed it to be necessary and sufficient for optimal distributed controllers to be synthesized efficiently via, e.g., convex programming.

Since the problem of synthesizing optimal distributed controllers for arbitrary information patterns is NP-hard, an alternative research thrust has been to synthesize sub-optimal controllers. Langbort et al. [122] presented sufficient LMI conditions that can be used for synthesizing a sub-optimal distributed controller. A convex programming approach to develop a sub-optimal controller for the H_2 performance criterion was considered in [45]. Receding horizon control for the problem has been explored, e.g., by Dunbar and Murray in [52] and by Keviczky in [115]. A different approach was inspired by the design of reduced-order controllers (see, e.g., [127]). This approach was used to obtain gradient descent based numerical algorithms for solving the optimal structured linear controller with an arbitrary number of free parameters for the infinite horizon case in many works, some examples being [181, 199]. In [82], this algorithm was explored for the case of vehicle formations and, in particular, it was proven that in this case, it is always possible to choose a feasible initial point. We will present this algorithm in Appendix A of the chapter. A similar algorithm can be applied to the finite horizon problem, as described by Anderson and Moore in [2], but the computational difficulties were pointed out by Kleinman et al. in [116]. Bemporad et al. in [14] considered the constrained LQR problem and came up with a numerical algorithm for the optimal piecewise affine controller. The algorithm was extended to the case of infinite-time horizon in [74]. A good survey of the attempts to solve the related fixed order and static output feedback problems can be found in [20, 44, 184] and the references therein.

As has been pointed out in many of the above mentioned works, the problem of finding the linear optimal controller that satisfies arbitrary constraints is very difficult. Even the question of stabilizability through a structured controller has not yet been solved in general [184]. Unless the problem has some special structure, finding the optimal controller with a prescribed structure remains open. Even the algorithms for finding sub-optimal controllers tend to involve numerical optimization and hence face convergence problems. In this chapter, we begin by setting up the finite-horizon LQR problem for the control of a network of autonomous agents with a given information flow topology. Even if the dynamics of the agents are not coupled and the only coupling present is due to the cost function, the optimal control law, in general, requires every agent to use knowledge about every other agent. We impose the constraint of a linear control law that satisfies a pre-specified topology in that any agent uses only the information about a pre-specified set of agents with which it can communicate. We solve for the optimal control law for a finite time horizon under these constraints. We see that calculation of the optimal control is computationally expensive and, instead, provide a sub-optimal solution that is computationally more tractable. Since both the algorithms we present involve solving only linear equations, they do not suffer from convergence problems encountered in many existing approaches that utilize, e.g., gradient descent algorithms.

In addition, we prove the interesting result that the information topology can neither help nor hinder controllability and stabilizability in the sense that a necessary and sufficient condition for any formation to be stable (controllable) is that each individual agent be stable (re. controllable) using its own inputs. This puts into perspective the recent results about a formation potentially being destabilized by addition of edges [59].

Evaluating the optimal controller for a given topology leads us naturally to the following question: 'Are there topologies that are more 'efficient' than others for distributed control?' The interconnection topology of the agents fixes the form of the control law and hence has a huge effect on the cost that the system has to suffer. In some applications (e.g., a power distribution network), an interconnection topology may already exist and the controller has to conform to this topology that has been decided *a priori*. However, if decentralization is not viewed as an external constraint, then both the topology and the controller have to be designed at the same time. For example, when deciding between a leader-follower and a fully decentralized architecture for cooperating multi-vehicle systems, the choice of the information pattern is an integral part of the control design process. Thus, it makes sense to find the minimal (with respect to some appropriate cost function) topology needed to achieve a particular control goal. Obviously, if there is no cost of adding a new communication link between two agents, the minimum cost will be achieved when the topology entails every agent communicating with every other agent. We present a model for evaluating the effect of topology in which we explicitly account for the cost of communication by making one of the weight matrices in the classical LQR cost function topology-dependent. This allows us to explore the trade-off between better control performance and higher communication cost incurred.

Similar questions have been asked for *static* agents in the theory of organizational efficiency and information cost, e.g., see Marschak and Radner [138]. However, the questions about efficiency of decision architectures has largely been ignored in the control literature. We are aware only of the following recent attempt at finding a minimal control interconnection structure. In [180], the authors show that when constructing a distributed controller from a set of observer-based controllers using different and parallel observations, the star interconnection topology is minimal, in the sense that the resulting control design problem has the minimal number of free parameters needed to ensure closed-loop stability. The model that we present has similarities to ideas in the field of Games over Networks (see, e.g., [106, 148]) and allows us to make rigorous statements regarding the optimal topology. For instance, we see that there are cases in which adding edges can actually be harmful from the view of the optimal cost. We also provide some conditions under which values of two graphs can be compared.

The appendices present complementary material to the main chapter. In the first appendix, we present an alternate method for synthesizing the optimal controller for distributed control problems that is suitable for the infinite horizon case. This algorithm is reminiscent of the method proposed

in [199]. However, we come up with stronger results by utilizing the structure of the problem. More importantly, we come up with a way to obtain the initial guess for the numerical algorithm, which was identified as the major problem in [199].

In the second appendix, we consider a distributed estimation problem. We consider the problem of controlling the motion of mobile sensor nodes that are jointly estimating the state of a dynamic target as a sensor network. Attempts have been made over the years to solve the problem of optimal motion control of sensors in various contexts. However, most of the optimal motion-planning and control techniques tend to be application-specific, e.g., [96, 151]. Particularly relevant to our area of interest in estimation/tracking and distributed sensing applications is the work of Mukai and Ishikawa [146, 147] who proposed a suboptimal centralized algorithm using the determinant of the estimated error covariance as a cost function. However, most of the approaches proposed in the literature assume that only one sensor's motion needs to be controlled. For systems where multiple sensors are present, a central computation device is assumed to be present that calculates the optimal trajectories of the sensors. In networked systems, where the number of sensing nodes may be large, it is obvious that this assumption needs to be relaxed. We propose a gradient search-based decentralized algorithm that demonstrates the benefits of distributed sensing. We also extend the algorithm to the task of tracking multiple targets and minimize the estimated error by forming sub-teams of the sensors appropriately.

2.2 Mathematical Preliminaries and Notation

By a network of interconnected dynamic agents, we mean a system of dynamically uncoupled agents in which every agent can use the information from a prescribed set of other agents (called its neighbors) for calculating its control input. The flow of information is thus described by identifying the set of neighbors for each agent and is referred to as the information flow topology. Consider a network of N agents. Together with the information flow topology, the network can be represented by a graph $\mathcal{G}$. Unless stated otherwise, in this chapter, we will be interested only in undirected graphs. The node set of the graph is denoted by $\mathcal{V} = \{v_i\}_{i=1}^{N}$ such that each agent i corresponds to a vertex v_i . The link (or edge) set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ comprises of edges such that the edge $e = (v_i, v_j)$ models the communication link between node v_i and node v_j. The set of neighbors of the node v_i is defined by

$$\mathcal{N}_i = \{v_j | (v_i, v_j) \in \mathcal{E}\}.$$

Thus, it corresponds to the set of all those nodes with which the node v_i can communicate. The degree of a node v_i is the cardinality of the set $\mathcal{N}_i$. We refer to the agents variously as vertices or nodes and the network as a graph or a formation. For graphs with the same vertex set $\mathcal{V}$, there is a natural partial order. If two graphs g and g' with the same vertex set are such that $\mathcal{E}(g) \subseteq \mathcal{E}(g')$,

we say that the graph g is a subgraph of g' and denote it by $g \preceq g'$. Equivalently we can say that g' is a supergraph of g and write $g' \succeq g$. A clique graph [91] is one each of whose subcomponents is a complete subgraph. Some examples and counter-examples of clique graphs are given in Figure 2.1.

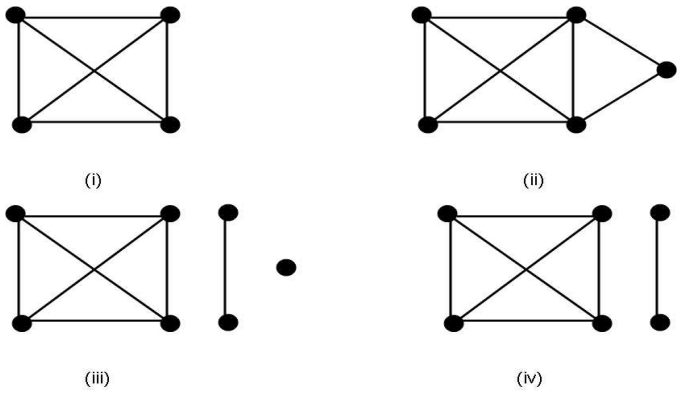


Figure 2.1: Some examples of graphs. All but (ii) are clique graphs.

Consider a graph with N nodes. The *adjacency matrix* (see, e.g., [18] for more details) denoted by $\mathcal{A}$ is a square matrix of size $N \times N$, defined as follows

$$\mathcal{A}_{ij} = \begin{cases} 1 & (v_i, v_j) \in \mathcal{E} \\ 0 & \text{otherwise.} \end{cases}$$

If we denote the degree of node v_i by O_i, then the *degree matrix* denoted by $\mathcal{D}$ is defined to be a square matrix of size $N \times N$, such that

$$\mathcal{D}_{ij} = \begin{cases} O_i & i = j \\ 0 & \text{otherwise.} \end{cases}$$

We define the *Laplacian* of a graph by the relation $\mathcal{L} = \mathcal{D} - \mathcal{A}$.

Consider the system evolving according to

$$x(k+1) = Ax(k).$$

We say that the system is (asymptotically) stable, or that A is (asymptotically) stable, if A has all its eigenvalues on or inside (strictly inside) the unit circle and all the eigenvalues on the unit circle are simple. A positive (semi-)definite matrix is a symmetric matrix with all eigenvalues (non-negative) strictly positive. For two matrices A and B, we write $A \geq B$ if $A - B$ is positive semi-definite.

We denote the expectation of a random variable X by $E[X]$. The covariance matrix of a random variable X with zero mean is defined by $E\left[XX^T\right]$, where X^T is the transpose of matrix X. The

covariance matrix is always a positive semi-definite matrix.

The trace of a square matrix X, denoted by $\text{trace}(X)$, is defined as the sum of its diagonal elements. The trace is also the sum of the eigenvalues of X. The trace operator satisfies the following properties (assume X, Y and Z to be compatible matrices; v is a column vector).

1. $\text{trace}(X + Y) = \text{trace}(X) + \text{trace}(Y)$.
2. $\text{trace}(XYZ) = \text{trace}(ZXY)$.
3. $E[v^T W v] = E[\text{trace}(W v v^T)]$.

In the last equation if W is a constant matrix, the right hand side can be further rewritten as $\text{trace}(W E[v v^T])$.

For an $m \times n$ matrix $X = [x_{ij}]$, the operation $\text{vec}(X)$ results in an $mn \times 1$ column vector with elements

$$\text{vec}(X) = \begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{m1} \\ x_{12} \\ \vdots \\ x_{mn} \end{bmatrix}.$$

The operation $A \otimes B$ denotes the Kronecker product (also called the direct product) between two matrices A and B (see [121] for details). It can be shown that for suitably dimensioned matrices A, X and B,

$$\text{vec}(AXB) = \left(B^T \otimes A\right) \text{vec}(X). \tag{2.1}$$

Let A, P and Q be real matrices. The equation

$$P = A^T P A + Q,$$

is called the discrete algebraic Lyapunov equation (DALE) if Q is symmetric. We will refer it as the Lyapunov equation in the following. We now summarize some results about the solutions of Lyapunov equation. For more details, see [64]. The Lyapunov equation has a unique solution P if and only if no eigenvalue of A is the reciprocal of an eigenvalue of A^T. If this condition is satisfied, the unique P is Hermitian. Furthermore, if A is asymptotically stable, then the unique Hermitian P is given by

$$P = \sum_{k=0}^{\infty} A^k Q (A^T)^k.$$

If A is stable and Q is positive definite (or semi-definite), then P is unique, symmetric and positive definite (or semi-definite).

2.3 Problem Formulation

Consider a formation of N agents, in which the i-th agent evolves according to the equation

$$x_i(k+1) = \bar{A}x_i(k) + \bar{B}u_i(k) + w_i(k),$$

where the control input $u_i(k)$ is given by

$$u_i(k) = F_{i,1}(k)x_i(k) + \sum_{j \in \mathcal{N}_i} F_{ij,2}(k)\left(x_i(k) - x_j(k)\right).$$

The state of the i-th agent is given by $x_i(k) \in \mathbf{R}^n$ and $u_i(k) \in \mathbf{R}^m$ denotes its control input. The form in which the control law is stated depends only on the relative state of the out-neighbors of agent i. However, because of the presence of the matrix $F_{i,1}(k)$, it is clear that this is not a limiting assumption. Assume that the noises $w_i(k)$'s are zero-mean, Gaussian and white. On stacking the states x_i of all the agents, we can obtain the system state vector x, whose evolution is described by

$$\begin{aligned} x(k+1) &= (I \otimes \bar{A})x(k) + (I \otimes \bar{B})u(k) + w(k) \\ u(k) &= (\mathrm{diag}\,(F_{i,1}(k)) + \mathcal{L}_{\mathrm{gen}}(k))x(k), \end{aligned} \tag{2.2}$$

where I is the identity matrix of suitable dimensions and diag $(F_{i,1}(k))$ is a block diagonal matrix with $F_{i,1}(k)$'s along the diagonal and zero matrices elsewhere. The vectors $u(k)$ and $w(k)$ are obtained by stacking the control laws and the noises for the individual agents, respectively. $\mathcal{L}_{\mathrm{gen}}(k)$ is a generalization of the Laplacian matrix of the graph and is formed as follows. Create the adjacency matrix $\mathcal{A}$ for the network. Then, replace each unity element that is at the (i,j)-th place by $-F_{ij,2}(k)$. Replace the diagonal element in the i-th row by a matrix which is the sum of the matrices $F_{i1,1}(k)$, $F_{i2,1}(k)$, $\cdots$, $F_{i(i-1)}(k)$, $F_{i(i+1)}(k)$, $\cdots$, $F_{iN,1}(k)$. The rest of the zero elements are replaced by zero matrices of appropriate dimensions. Note that the topological constraints on the form of control law are inherent in the structure of $\mathcal{L}_{\mathrm{gen}}(k)$.

In this chapter, we will assume that the topology of the network is known at any time step, although it may be time-varying. We ignore issues such as quantization error and message loss when agents communicate over the links. Note that if all the agents are not identical, relations similar to (2.2) can easily be obtained. The matrices $I \otimes \bar{A}$ and $I \otimes \bar{B}$ will be replaced by block diagonal matrices diag$(\bar{A}_i)$ and diag$(\bar{B}_i)$, but other details remain similar. We begin by discussing the questions of stabilizability and controllability of the formation under a specified topology constraint.

2.3.1 Stabilizability

Two questions arise immediately:

- Is it possible to stabilize a formation using information from other vehicles when the vehicles are individually not stable? In other words, if a vehicle is unstable, can the formation be stabilized by the exchange of information between different agents?
- Are some topologies inherently unstable in that even if the agents are stable, the information flow will always make it impossible to stabilize the formation?

We note the following result.

Proposition 2.1 *Consider a formation of interconnected dynamic agents as defined in Section 2.2.*

1. *A formation is controllable if and only if each individual agent is controllable.*
2. *A formation is stabilizable if and only if each individual agent is stabilizable.*

Proof We use the notation introduced above. Let the matrix $\bar{A}$ have dimensions $n \times n$ and there be N agents in the formation. As can be seen from (2.2), for controllability of the formation, we want the matrix M_1 to have rank nN [118], where

$$M_1 = \left[\begin{array}{cccccc} I \otimes \bar{B} & (I \otimes \bar{A})(I \otimes \bar{B}) & (I \otimes \bar{A})^2(I \otimes \bar{B}) & \cdots & (I \otimes \bar{A})^{nN-1}(I \otimes \bar{B}) \end{array} \right].$$

Using the standard property of Kronecker product

$$(a \otimes b)(c \otimes d) = ac \otimes bd,$$

we can rewrite M_1 as

$$M_1 = \left[\begin{array}{cccccc} I \otimes \bar{B} & (I \otimes \bar{A}\bar{B}) & (I \otimes \bar{A}^2\bar{B}) & \cdots & (I \otimes \bar{A}^{nN-1}\bar{B}) \end{array} \right].$$

This matrix has rank nN if and only if the following matrix has rank n.

$$M_2 = \left[\begin{array}{cccccc} \bar{B} & \bar{A}\bar{B} & \bar{A}^2\bar{B} & \cdots & \bar{A}^{nN-1}\bar{B} \end{array} \right].$$

Since $\bar{A}$ has dimensions $n \times n$, the equivalent condition is that the matrix

$$M_3 = \left[\begin{array}{cccccc} \bar{B} & \bar{A}\bar{B} & \bar{A}^2\bar{B} & \cdots & \bar{A}^{n-1}\bar{B} \end{array} \right],$$

be rank n. But M_3 being rank n is simply the condition for the individual agent being controllable. Thus, the formation is controllable if and only if each individual agent is controllable. This proves

the first part. The second part also follows from the above proof. The subspace not spanned by the columns of M_1 is stable if and only if the subspace not spanned by the columns of M_3 is stable. ∎

Note that although we have proved the theorem for the case of identical agent dynamics, the theorem holds for the case of agent dynamics being different as well. The terms $I \otimes \Phi$ and $I \otimes \Gamma$ would be replaced by block diagonal matrices but the condition for the controllability matrix of the formation having rank nN would still reduce to the condition that the controllability matrix of each agent have rank n.

2.3.2 Designing the Control Law

From (2.2), it can be seen that the problem of designing a control law that satisfies some topological constraints is equivalent to solving the control design problem for the system

$$\begin{aligned} x(k+1) &= \left(I \otimes \bar{A}\right) x(k) + \left(I \otimes \bar{B}\right) u(k) + w(k) \\ u(k) &= F(k)x(k), \end{aligned} \tag{2.3}$$

with the additional constraint that $F(k)$ should have those elements as 0 which correspond to zero entries in the $\mathcal{L}_{\text{gen}}(k)$ of the interconnection topology formed as above. $F(k)$ can then readily be cast in the form $\text{diag}(F_{i,1}(k)) + \mathcal{L}_{\text{gen}}(k)$ and the matrices $F_{i,1}(k)$ and $F_{ij,2}(k)$ obtained. Constraining the control $F(k)$ to have some elements zero forces us to consider only those matrices that live in a particular sub-space of the vector space of all matrices with the same dimensions as $F(k)$.

2.4 The Optimal Constrained Control Law

Denote $A = I \otimes \bar{A}$ and $B = I \otimes \bar{B}$ and rewrite (2.3) as

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + w(k) \\ u(k) &= F(k)x(k), \end{aligned} \tag{2.4}$$

where the initial condition $x(0)$ is random and is Gaussian with zero-mean and covariance $R(0)$. The noise $w(k)$ is also random white zero-mean Gaussian with covariance R_w. We wish to minimize the quadratic cost function

$$J_T = E\left[\sum_{k=0}^{T} \left(x^T(k)Qx(k) + u^T(k)Ru(k)\right)\right] + E\left[x^T(T+1)P^c(T+1)x(T+1)\right], \tag{2.5}$$

where Q and R are positive definite. If there were no constraints on the structure of the control law $F(k)$, this would be the classical LQR design problem. However, in our present problem, we wish

to additionally constrain the control law to lie within a space spanned by the basis vectors $\{\Theta_j, j = 1, 2, \ldots, p\}$. Thus, the problem is to find a control law of the form

$$F(k) = \sum_{j=1}^{p} \alpha_j(k)\Theta_j, \tag{2.6}$$

where $\alpha_j(k)$'s are scalars, that minimizes the cost function (2.5).

Remarks

1. The optimal constrained control law would not, in general, be the projection of the optimal control law on to the sub-space we are interested in. This is reminiscent of the fact that the optimal causal estimate for a random variable is not the same as the projection of the general optimal estimate on to the causal sub-space [112].

2. Requiring a priori that the controller be linear might be a non-trivial assumption. However, this commonly made assumption allows us to derive tractable algorithms for solving the problem and leads to sharper results.

2.4.1 A Preliminary Result

In this subsection we prove an intermediate result that we will use later. We begin with the following fact.

Lemma 2.2 *Suppose W is a positive semi-definite matrix and $P(K)$ denotes a matrix-valued function of the matrix argument K. If $P(K) > P(K_0)$, then trace $(P(K)W) \geq$ trace $(P(K_0)W)$.*

Proof Since $P(K) > P(K_0)$, we have $P(K) - P(K_0) > 0$. Also W is positive semi-definite, thus $W^{\frac{1}{2}}$ is defined. Hence, we note that

$$\begin{aligned} \text{trace}\left(W^{\frac{T}{2}}\left(P(K) - P(K_0)\right)W^{\frac{1}{2}}\right) &\geq 0 \\ \Rightarrow \text{trace}\left(\left(P(K) - P(K_0)\right)W\right) &\geq 0. \end{aligned}$$

But this means trace $(P(K)W) \geq$ trace $(P(K_0)W)$, which proves the assertion. ∎

Using this lemma we can prove the following.

Proposition 2.3 *Consider the cost function*

$$C = E\left[\begin{bmatrix} K_1Y_1 - X_1 \\ K_2Y_2 - X_2 \\ \vdots \\ K_lY_l - X_l \end{bmatrix}^T W \begin{bmatrix} K_1Y_1 - X_1 \\ K_2Y_2 - X_2 \\ \vdots \\ K_lY_l - X_l \end{bmatrix}\right],$$

where K_i's are arbitrary matrices while Y_i's and X_i's are vectors of suitable dimensions such that the cost function C is well-defined. Suppose that W is a symmetric positive-definite matrix that can be written in the form

$$W = \begin{bmatrix} W_{1,1} & W_{1,2} & \cdots & W_{1,l} \\ W_{2,1} & & \cdots & W_{2,l} \\ \vdots & & \ddots & \vdots \\ W_{l,1} & W_{l,2} & \dots & W_{l,l} \end{bmatrix},$$

where the blocks $W_{i,j}$ are of appropriate sizes so that the product $X_i^T W_{i,j} X_j$ is well defined. Finally assume that all diagonal blocks $W_{i,i}$'s are positive definite. Then, the optimal K_i's minimizing the cost function C are given by the solution to the coupled matrix equations

$$K_j = W_{j,j}^{-1}\left[\sum_i W_{j,i} R_{X_iY_j} - \sum_{i\neq j} W_{j,i} K_i R_{Y_iY_j}\right] R_{Y_j}^{-1}, \qquad \forall j = 1, 2, \cdots, l,$$

where $R_{Y_iY_j} = E\left[Y_iY_j^T\right]$ and $R_{X_iY_j} = E\left[X_iY_j^T\right]$.

Proof For each j, we can write the terms in the cost function C that depend on the matrix K_j as

$$C_j = \operatorname{trace}\left(K_j R_{Y_j} K_j^T W_{j,j} - K_j\Psi - \Psi^T K_j^T\right),$$

where

$$\Psi = \left[\sum_i R_{Y_jX_i} W_{i,j} - \sum_{i\neq j} R_{Y_jY_i} K_i^T W_{i,j}\right].$$

K_j needs to be chosen so as to minimize C_j. The minimization is of the form

$$\min_X \operatorname{trace}\left(XAX^TB + XC + C^TX^T\right),$$

where B is positive definite and hence invertible. This can be rewritten as

$$\begin{aligned}
&\min_X \operatorname{trace}\left(XAX^TB + XC + C^TX^T\right) \\
&= \min_X \operatorname{trace}\left(XAX^TB + XCB^{-1}B + C^TX^TBB^{-1}\right) \\
&= \min_X \operatorname{trace}\left(XAX^TB + XCB^{-1}B + B^{-1}C^TX^TB\right) \\
&= \min_X \operatorname{trace}\left(\left(XAX^T + XCB^{-1} + B^{-1}C^TX^T\right)B\right).
\end{aligned}$$

Now we use lemma 2.2. Thus, our problem reduces to that of determining X such that $XAX^T + XCB^{-1} + B^{-1}C^TX^T$ is minimized. We complete the squares to write

$$XAX^T + XCB^{-1} + B^{-1}C^TX^T = \left(X + B^{-1}C^TA^{-1}\right)A\left(X + B^{-1}C^TA^{-1}\right)^T - B^{-1}C^TA^{-1}C.$$

Thus, the minimizing X is seen to be given by $-B^{-1}C^TA^{-1}$. Applying this fact to our original problem of determining K_j, we see that

$$K_j = W_{j,j}^{-1}\left[\sum_i W_{j,i}R_{X_iY_j} - \sum_{i\neq j} W_{j,i}K_iR_{Y_iY_j}\right]R_{Y_j}^{-1}. \quad \blacksquare$$

Note that for calculation of the K_j's, we can use the identity (2.1). For each K_j we obtain the relation

$$\operatorname{vec}(K_j) = \operatorname{vec}\left(W_{j,j}^{-1}\sum_i W_{j,i}R_{X_iY_j}\right) - \sum_{i\neq j}\left[\left(R_{Y_iY_j}R_{Y_j}^{-1}\right)^T \otimes \left(W_{j,j}^{-1}W_{j,i}\right)\operatorname{vec}(K_i)\right].$$

We have one such equation for each K_j, $j = 1, \ldots, n$. These equations can readily be solved to obtain the values of $\operatorname{vec}(K_j)$ and, in turn, the matrices K_j can be determined.

2.4.2 The Optimal Control Law

From (2.5) we see that the cost function to be minimized is

$$J_T = E\left[\sum_{k=0}^{T} u^T(k)Ru(k) + \sum_{k=0}^{T} x(k)Qx(k)\right] + E\left[x^T(T+1)P^c(T+1)x(T+1)\right].$$

Using the equation (obtained from (2.4))

$$x(k) = A^kx(0) + \sum_{j=0}^{k-1} A^jBu(k-1-j) + \sum_{j=0}^{k-1} A^jw(k-1-j)$$

and the fact that the noise $w(k)$ is white and zero-mean allows us to rewrite the cost function as

$$J_T = E\left[\Gamma^T F \Gamma + \Gamma^T G \Lambda + \Lambda^T G^T \Gamma + \Lambda^T H \Lambda\right], \tag{2.7}$$

In the above equation

$$\Gamma = \begin{bmatrix} x(0)^T & w(0)^T & w(1)^T & \cdots & w(T)^T \end{bmatrix}^T$$

is the vector of all the random variables involved. Similarly,

$$\Lambda = \begin{bmatrix} u_0^T & u_1^T & \cdots & u_T^T \end{bmatrix}^T$$

is the control vector that is the optimization variable. Finally, the matrices F, G and H are functions of the known matrices A, B, R, Q and $P^c(T+1)$. The control vector Λ is constrained to be of the form

$$\Lambda = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(T) \end{bmatrix} = \begin{bmatrix} F(0)x(0) \\ F(1)x(1) \\ \vdots \\ F(T)x(T) \end{bmatrix}$$

where the matrices $F(i)$'s have some pre-specified elements zero because of the imposed topology. In particular, if we write

$$F(i)x(i) = \begin{bmatrix} F_1(i)x(i) \\ F_2(i)x(i) \\ \vdots \\ F_N(i)x(i) \end{bmatrix}$$

where $F_j(i)$ is the control law applied by the j-th agent at time step i, then those elements of $F_j(i)$ that correspond to the elements in the state vector $x(i)$ that the j-th agent does not have access to are zero. We can pull the constraints into the state vector and write

$$F_j(i)x(i) = K_j(i)y_j(i),$$

where $K_j(i)$ is now a matrix free of any constraints on its elements. The vector $y_j(i)$ is a stacked vector of the states of the agents that the j-th agent has access to at time i. These agents correspond

to the neighbors of the agent i. We can thus write

$$F(i)x(i) = \begin{bmatrix} K_1(i)y_1(i) \\ K_2(i)y_2(i) \\ \vdots \\ K_N(i)y_N(i) \end{bmatrix}.$$

In turn, Λ can be written as

$$\Lambda = \begin{bmatrix} F(0)x(0) \\ F(1)x(1) \\ \vdots \\ F(T)x(T) \end{bmatrix} = \begin{bmatrix} K_1(0)y_1(0) \\ K_2(0)y_2(0) \\ \vdots \\ K_N(0)y_N(0) \\ K_1(1)y_1(1) \\ \vdots \\ K_N(1)y_N(1) \\ \vdots \\ K_0(T)y_0(T) \\ \vdots \\ K_N(T)y_N(T) \end{bmatrix}. \tag{2.8}$$

The NT matrices $K_j(i)$'s are arbitrary and are now the optimization variables. From (2.7), we see that the cost function can be written as

$$\begin{aligned} J_T &= E\left[\Gamma^T F\Gamma + \Gamma^T G\Lambda + \Lambda^T G^T\Gamma + \Lambda^T H\Lambda\right] \\ &= E\left[\left(\Lambda + H^{-1}G^T\Gamma\right)^T H\left(\Lambda + H^{-1}G^T\Gamma\right)\right] + E\left[\Gamma^T\left(GH^{-1}G^T + F\right)\Gamma\right]. \end{aligned}$$

The choice of Λ only affects the first term. Thus, the optimization problem is

$$\min_{\Lambda} E\left[\left(\Lambda + H^{-1}G^T\Gamma\right)^T H\left(\Lambda + H^{-1}G^T\Gamma\right)\right],$$

where Λ is of the form (2.8). But this optimization problem is exactly in the form of Proposition 2.3. Thus, we can optimize the value of the cost function. This solves the optimal control law problem.

The solution involves the calculation of second order statistic terms which can be calculated off-line since the topology of the network is assumed to be known. The procedure holds even for the case when the topology is time-varying, as long as all the agents know the topology. However, note that we need to solve a total of NT coupled matrix equations. This is a formidable computational

burden. We now present a method that is computationally more tractable at the expense of being sub-optimal.

2.4.3 A Sub-optimal Control Law

Once again, we note from (2.5) that the T-horizon cost function to be minimized is

$$J_T = E\left[\sum_{k=0}^{T} u^T(k)Ru(k) + \sum_{k=0}^{T} x^T(k)Qx(k)\right] + E\left[x^T(T+1)P^c(T+1)x(T+1)\right].$$

We need to choose $u(0)$, $u(1)$, $\cdots$, $u(T)$ that minimize J_T. Following [95, Chapter 9], we gather terms that depend on the choice of $u(T)$ and $x(T)$ and write them as

$$\begin{aligned}
\Upsilon(T) &= E\left[u^T(T)Ru(T) + x^T(T)Qx(T)\right] + E\left[x^T(T+1)P^c(T+1)x(T+1)\right] \\
&= E\left[\begin{bmatrix} u^T(T) & x^T(T) \end{bmatrix} \Delta \begin{bmatrix} u(T) \\ x(T) \end{bmatrix}\right] + E\left[w^T(T)P^c(T+1)w(T)\right] \\
&= S(T) + O(T)
\end{aligned}$$

where

$$\begin{aligned}
\Delta &= \begin{bmatrix} R + B^TP^c(T+1)B & B^TP^c(T+1)A \\ A^TP^c(T+1)B & Q + A^TP^c(T+1)A \end{bmatrix} \\
S(T) &= E\left[\begin{bmatrix} u^T(T) & x^T(T) \end{bmatrix} \Delta \begin{bmatrix} u(T) \\ x(T) \end{bmatrix}\right] \\
O(T) &= E\left[w^T(T)P^c(T+1)w(T)\right].
\end{aligned}$$

To obtain the above equation, we have used the system dynamics given in (2.4) and the fact that the plant noise is zero mean. Thus, we can write

$$J_T = E\left[\sum_{k=0}^{T-1} u^T(k)Ru(k) + \sum_{k=0}^{T-1} x^T(k)Qx(k)\right] + S(T) + O(T). \tag{2.9}$$

We aim to choose $u(T)$ to minimize J_T. From (2.9), it is clear that the only term where the choice of $u(T)$ can make a difference is $S(T)$. On completing squares, $S(T)$ can be written as

$$S(T) = E\left[\left(u(T) - \bar{u}(T)\right)^T R_e^c(T)\left(u(T) - \bar{u}(T)\right)\right] + E\left[x^T(T)P^c(T)x(T)\right]$$

where

$$\begin{aligned} R_e^c(T) &= R + B^T P^c(T+1)B \\ P^c(T) &= Q + A^T P^c(T+1)A - A^T P^c(T+1)B \left(RB^T P^c(T+1)B\right)^{-1} B^T P^c(T+1)A \end{aligned}$$

and $\bar{u}(T)$ is the standard optimal LQ control given by

$$\bar{u}(T) = -\left(R_e^c(T)\right)^{-1} B^T P^c(T+1)Ax(T).$$

If $u(T)$ could depend arbitrarily on $x(T)$, its optimal value would have been given by $\bar{u}(T)$. However, owing to the topological constraints, that is not possible in our problem. Instead, $u(T)$ needs to be calculated so as to minimize $S(T)$ by using the information flow that satisfies the topological constraints. In other words, we need to find $u(T) = F(T)x(T)$ that minimizes $\Upsilon(T)$ where $F(T)$ has certain elements zero. The control problem, thus, reduces to an optimal estimation problem. To solve for $u(T)$, we rewrite it as

$$u(T) = \begin{bmatrix} u_1(T) \\ u_2(T) \\ \vdots \\ u_N(T) \end{bmatrix},$$

where each $u_i(T)$ is the control law that the i-th agent applies and is a linear function of the measurements the i-th agent has access to. Thus, we can write

$$u_i(T) = F_i(T)x(T),$$

where $F_i(T)$ has those elements 0 that correspond to the elements in the state vector $x(T)$ that the i-th agent does not have access to. Pulling the constraints into the state vector, we can write

$$u_i(T) = K_i(T)y_i(T),$$

where $K_i(T)$ does not have any constraint while the vector $y_i(T)$ is a stacked vector of the states of the agents that the i-th agent has access to at time T. Thus, the problem of choosing the control law $u(T)$ reduces to the problem of choosing $K_i(T)$'s so as to minimize the criterion

$$E\left[\begin{bmatrix} K_1(T)y_1(T) - \bar{u}_1(T) \\ K_2(T)y_2(T) - \bar{u}_2(T) \\ \vdots \\ K_N(T)y_N(T) - \bar{u}_N(T) \end{bmatrix}^T R_e^c(T) \begin{bmatrix} K_1(T)y_1(T) - \bar{u}_1(T) \\ K_2(T)y_2(T) - \bar{u}_2(T) \\ \vdots \\ K_N(T)y_N(T) - \bar{u}_N(T) \end{bmatrix}\right].$$

This is exactly the optimization problem discussed in Proposition 2.3. Thus, the matrices $K_i(T)$ can be easily obtained. Note that the solution involves only N coupled matrix equations and hence is computationally much less expensive than the optimal control law calculation discussed in section 2.4.2.

Denote the estimation error incurred due to the minimizing choice of $u(T)$ by $\Lambda(T)$. We have

$$S(T) = \Lambda(T) + E\left[x^T(T)P^c(T)x(T)\right].$$

We can, thus, write the cost function as

$$\begin{aligned} J_T &= E\left[\sum_{k=0}^{T-1} u^T(k)Ru(k) + \sum_{k=0}^{T-1} x^T(k)Qx(k)\right] + S(T) + O(T) \\ &= E\left[\sum_{k=0}^{T-1} u^T(k)Ru(k) + \sum_{k=0}^{T-1} x^T(k)Qx(k)\right] + E\left[x^T(T)P^c(T)x(T)\right] + \Lambda(T) + O(T) \\ &= J_{T-1} + \Lambda(T) + O(T). \end{aligned}$$

Thus, we now need to choose control inputs for time steps 0 to $T-1$ to minimize J_T. By scanning the terms on the right hand side of the equation, we see that $O(T)$ is independent of the choice of control laws from time 0 to $T-1$. However, unlike the standard case of control with imperfect observations [95, Chapter 9], we note that apart from J_{T-1}, the estimation error $\Lambda(T)$ is also a function of the state $x(T)$ and hence of the (unknown) control law $u(T-1)$. Moreover, it is a non-linear function of $u(T-1)$. The control $u(T-1)$ should be chosen to minimize the cost $J_{T-1}+\Lambda(T)$. Thus, the separation principle does not hold in our problem. This is not surprising since the information pattern in the problem is not classical [203] because the previous control inputs are not known fully to all the agents. We get across this problem by neglecting the estimation cost $\Lambda(T)$ and optimizing only J_{T-1}. For this purpose, we note that our argument, so far, was independent of time index T. Thus, we can recursively apply the argument for time steps $T-1$, $T-2$ and so on. We thus obtain a sub-optimal control law that is computationally more efficient.

Remarks

1. We have artificially enforced a separation principle that says that the controller synthesis problem can be separated into an estimation problem and the usual LQR control problem. At every time step, every agent tries to estimate the optimal control input (in the sense of Proposition 2.3) using the information it has access to and uses this estimate in the optimal LQR control law.

2. This method is in general sub-optimal since the separation principle does not hold in reality. However, since it replaces solution of NT coupled matrix equations by solving N coupled

matrix equations T times, it saves a lot on computational cost.

3. If needed, better performance can be achieved by including the estimation cost $\Lambda(T)$ in calculation of $u(T-1)$. It can be proven that this inclusion results in a convex problem that can be solved efficiently. However, this method would still not be optimal since for calculation of $u(T-2)$, we need to consider J_{T-2}, $\Lambda(T-1)$ and the cost incurred in imperfectly minimizing $\Lambda(T)$. Thus, the problem involves more and more terms to optimize over as we move backwards in time. The extent of sub-optimality can be reduced by including more terms in the optimization.

4. Intuitively, the approximation can be thought of as follows. At any time, the optimal control input of an agent will depend on the control inputs of all other agents at the previous time step. However, the agent is not allowed to observe these. We get around this problem by ignoring the direct dependence of the optimal control input on these terms. Instead, we use the fact that these terms will soon appear in the values of the states of the neighbors of the agent that are being observed. Thus, these terms will eventually be used in the calculation of control inputs.

5. For a fully connected topology, the sub-optimal algorithm yields the same result as the optimal algorithm since there is no estimation cost $\Lambda(T)$. In other words, since the control input of any agent can be calculated whenever needed, there is no approximation involved in ignoring its effect on each agent's control input.

2.5 Examples

We now consider two examples to illustrate the issues involved.

Example 1

Consider a network of four agents, each with single integrator dynamics. This case is of interest since single integrator dynamics can be used to solve consensus problems [157]. Let the agents be designated as v_i, $i = 1, 2, 3, 4$. The agent v_i has dynamics

$$\begin{aligned} x_i(k+1) &= x_i(k) - 0.2u_i(k) + w_i(k) \\ u_i(k) &= F_{i,1}x_i(k) + \sum_{j \in \mathcal{N}_i} F_{ij,2}(x_j(k) - x_i(k)). \end{aligned}$$

As before, denote $x(k)$ to be the state of the whole system and $u(k)$ to be the control vector obtained by stacking all the $x_i(k)$'s and $u_i(k)$'s respectively. Then, the evolution of the system is described

as

$$\begin{aligned} x(k+1) &= x(k) - 0.2u(k) + w(k) \\ u(k) &= F_1 x(k) + F_2 x(k), \end{aligned}$$

where F_1 is a diagonal matrix with $F_{1,1}, F_{2,1}, F_{3,1}, F_{4,1}$ as the diagonal elements; and the $(i,j)-$th element of the matrix F_2 is given by

$$[F_2]_{i,j} = \begin{cases} F_{ij,2} & i \neq j \text{ and } j \in \mathcal{N}_i \\ 0, & i \neq j \text{ and } j \notin \mathcal{N}_i \\ -\sum_j F_{ij,2}, & i = j. \end{cases}$$

The initial condition is random with zero mean and covariance as the identity matrix. Similarly the noise is white Gaussian with mean zero and covariance as the identity matrix. The cost function specified is

$$J = \sum_{k=0}^{T} E\left[x^T(k)Qx(k) + u^T(k)Ru(k)\right].$$

We present results for $T = 30$. We take the weighting matrices arbitrarily to be as follows:

$$Q = \begin{bmatrix} 1.6158 & 1.6884 & 1.2138 & 0.563 \\ 1.6884 & 2.798 & 1.2843 & 1.2528 \\ 1.2138 & 1.2843 & 0.9645 & 0.5147 \\ 0.563 & 1.2528 & 0.5147 & 0.7501 \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

We consider a constrained topology where we allow limited communication to happen. The vehicle v_1 can talk to v_2, the vehicle v_2 to v_1 and v_3, the vehicle v_3 to v_2 and v_4 and v_4 can talk to v_3. In this case, the evolution of the cost is as shown in Figure 2.2. The cost plotted in the figure is $E\left[x^T(k)Qx(k) + u^T(k)Ru(k)\right]$ at every time step k. We can see that the loss in performance from the sub-optimal algorithm is not huge. The savings in computational time, however, are considerable. Note that at the intermediate time values, the sub-optimal algorithm may perform better than the optimal algorithm. However, this can be easily explained by noting that the optimal algorithm is optimal for a time horizon of 30 steps and there is no guarantee that it is the optimal algorithm for a smaller time window as well.

In Figure 2.3, we show the steady state cost for the ring topology for a time horizon T of 100 time steps as we introduce delay into the system. The ring topology involves all communication links being present, except the (v_2, v_4) and (v_1, v_3) links. We assume that the state information is passed with some delay that is a multiple of the sampling time of the system but the agents calculate the

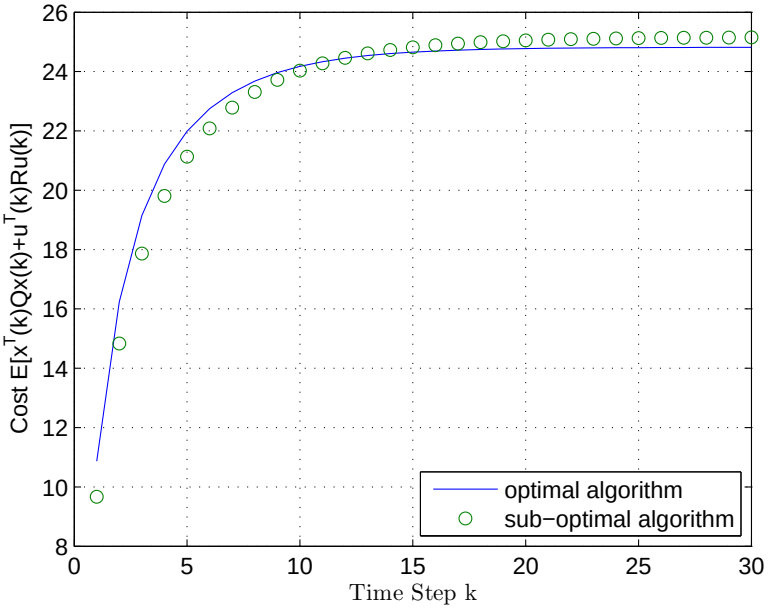


Figure 2.2: Comparison of the performance of the optimal and of the sub-optimal algorithm. The loss in performance due to the sub-optimal algorithm is not huge.

control law assuming there is no delay. It can be seen that the cost increases slowly and the system is reasonably robust to delay uncertainty. It becomes unstable only for a delay equal to or greater than 5 time steps.

Example 2

In this example, we use the dynamics of each agent as the dynamics of the Caltech Multi Vehicle Wireless Testbed vehicles, as described in [40, 198]. The non-linear dynamics are given by

$$\begin{aligned} m\ddot{x} &= -\mu\dot{x} + (F_L + F_R)\cos(\theta) \\ m\ddot{y} &= -\mu\dot{y} + (F_L + F_R)\sin(\theta) \\ J\ddot{\theta} &= -\psi\dot{\theta} + (F_R - F_L)r_f. \end{aligned}$$

F_L and F_R are the inputs, $m = 0.749$ kg is the mass of vehicle, $J = 0.0031$ kg m^2 is the moment of inertia, $\mu = 0.15$ kg-s is the linear frictional coefficient, $\psi = 0.005$ kgm^2/s is the rotational friction coefficient and $r_f = 0.089$ m is the distance from the center of mass of the vehicle to the axis of the fan. On linearizing the dynamics about the straight line $y = x$ at a velocity of 1 ms^{-1} along the x

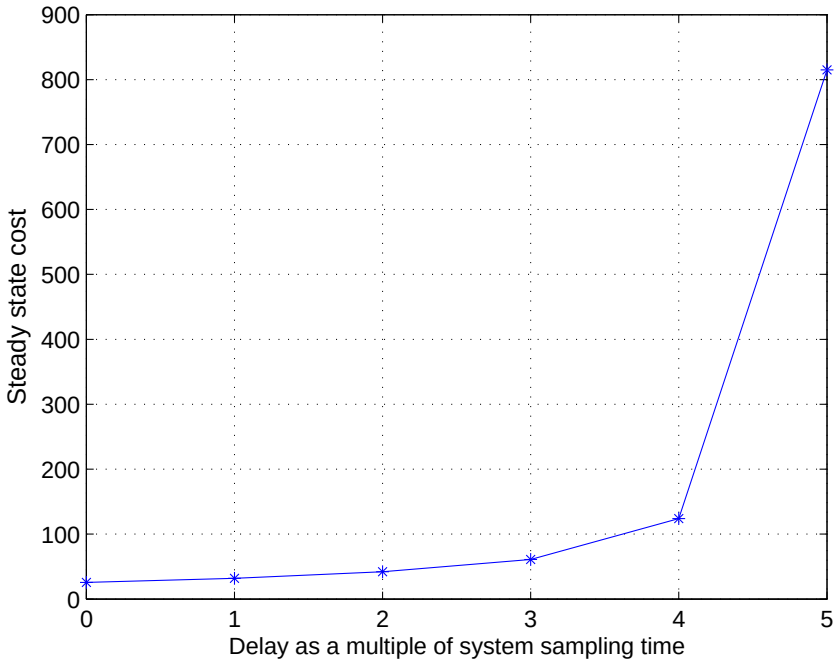


Figure 2.3: Variation of the steady state cost as delay is introduced in the system. The sub-optimal algorithm is robust to delays.

and y axes, we obtain the equations

$$\begin{aligned} \dot{X} &= AX + BU \\ U &= FX, \end{aligned}$$

where

$$X = \begin{bmatrix} x & y & \theta & \dot{x} & \dot{y} & \dot{\theta} \end{bmatrix}^T$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{-(F_L^{nom}+F_R^{nom})\sin(\theta^{nom})}{m} & \frac{-\mu}{m} & 0 & 0 \\ 0 & 0 & \frac{(F_L^{nom}+F_R^{nom})\cos(\theta^{nom})}{m} & 0 & \frac{\mu}{m} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{-\psi}{J} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{\cos(\theta^{nom})}{m} & \frac{\cos(\theta^{nom})}{m} \\ \frac{\sin(\theta^{nom})}{m} & \frac{\sin(\theta^{nom})}{m} \\ \frac{-r_f}{J} & \frac{-r_f}{J} \end{bmatrix}$$

$$\theta^{nom} = \frac{\pi}{4} \qquad F_L^{nom} = F_R^{nom} = \frac{\mu}{\sqrt{2}}.$$

We discretize the above equations with a step size $h = 0.2$. We consider 8 vehicles starting from an octagonal formation and consider the topologies possible as the communication radius of each vehicle is increased. By symmetry, there are 5 distinct topologies possible, with each vehicle talking to 0, 2, 4, 6 and 7 other vehicles respectively. The initial covariance matrix $R(0)$ is assumed to be the identity matrix. The cost function matrix R is also identity while the matrix Q is randomly generated. The cost function horizon is $T = 100$ time steps. A typical curve for the varying of the costs provided by the sub-optimal algorithm as the communication radius is increased is given in Figure 2.4. The cost plotted is the steady state error $E\left[x^T(k)Qx(k) + u^T(k)Ru(k)\right]$ for $k = 100$. Following general conclusions can be drawn for the example from the plot:

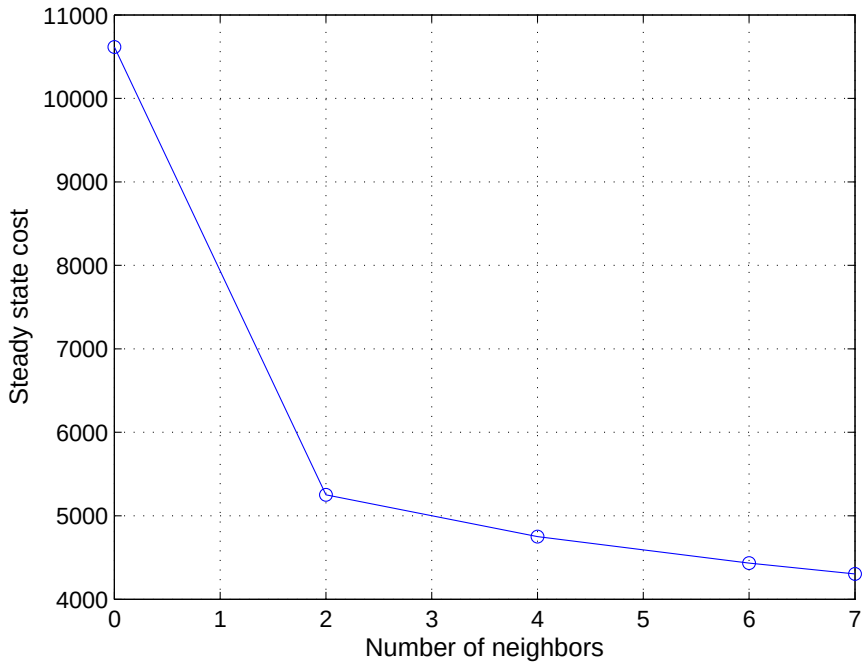


Figure 2.4: As the communication radius is increased, the cost obtained by the finite horizon sub-optimal control law goes down.

1. As more and more communication is allowed, the cost goes down.

2. The marginal utility of each communication link decreases as more and more links are added.

The difference in the performance between the sub-optimal and the optimal algorithms increases as the communication topology becomes sparser. Figure 2.5 shows the comparison of optimal and sub-optimal algorithms for a different value of the Q matrix. The cost considered is the steady state cost $E\left[x^T(k)Qx(k) + u^T(k)Ru(k)\right]$ for $k = 100$. It can be seen that even for the decentralized case, the error is of the order of only 30%.

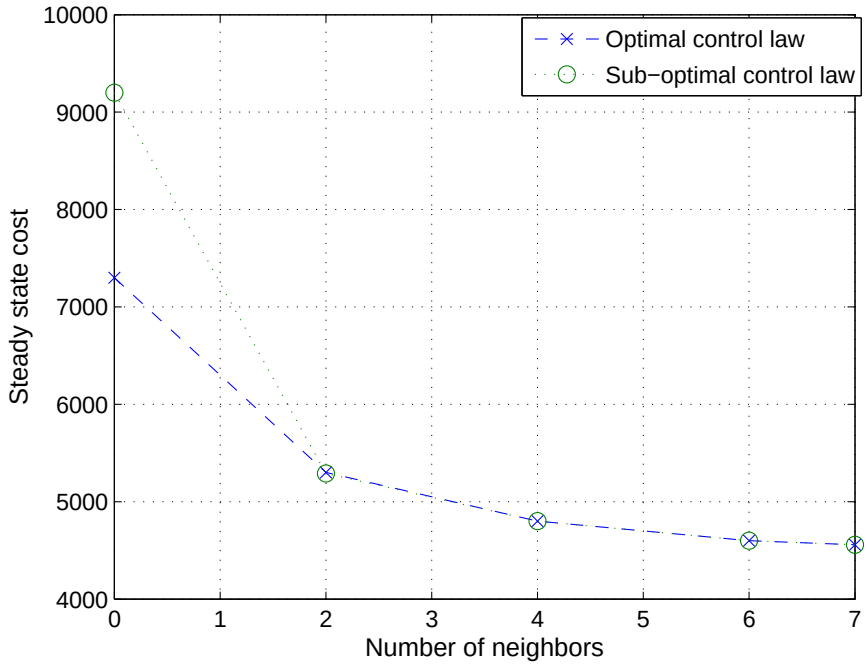


Figure 2.5: As the communication radius is increased, loss in performance due to the sub-optimal algorithm decreases.

2.6 Cost Functions and Value of a Graph

An implicit assumption in the method we considered above for distributed control synthesis (or, in fact, in almost any other method for synthesis proposed in the literature) is that the interaction topology is fixed *a priori* and provided to the control designer. In many applications, however, both the topology and the controller need to be designed simultaneously. Thus, in a multi-vehicle or a factory floor application, what topology to enforce is a valid design question. In this section, we propose a model to evaluate the *efficiency* of various topologies that achieve a particular control goal. We start with some notation and definitions.

Assume, once again, that there are N agents present with the i-th agent evolving as

$$x_i(k+1) = A_i x_i(k) + B_i u_i(k),$$

with the state $x_i(k) \in \mathbf{R}^{n_i}$ and the control input $u_i(k) \in \mathbf{R}^{m_i}$. Note that, for simplicity, we have assumed no noise to be present. Denote

$$n = \sum_{i=1}^{N} n_i \qquad\qquad m = \sum_{i=1}^{N} m_i.$$

Further, as before, denote the stacked vector of all the states of the agents $x_i(k+1)$ by $x(k)$ and the stacked vector of all the control inputs $u_i(k+1)$ by $u(k)$. Thus, the state of the entire system evolves as

$$x(k+1) = Ax(k) + Bu(k),$$

where the matrices A and B are block diagonal matrices with components A_i's and B_i's respectively. We assume that each pair (A_i, B_i) is controllable, which implies by a proof similar to that of Proposition 2.1, that the pair (A, B) is controllable.

Let $\mathcal{G}_N$ denote the set of all undirected graphs over N vertices. Every graph $g \in \mathcal{G}_N$ specifies a communication topology that defines the structure of specific control laws for the system. Consider a matrix F of dimensions $m \times n$ defined blockwise, each block F_{ij} being a $m_i \times n_j$ matrix such that F_{ij} is a zero matrix if there is no edge between vertices v_i and v_j in the graph g, i.e. $(v_i, v_j) \notin \mathcal{E}(g)$. Let us denote the set of all the matrices whose structure is described in this way by a graph g as $\mathcal{F}(g)$. For a control law $u(k) = Fx(k)$, if $F \in \mathcal{F}(g)$, then $u_i(k)$ involves the values of $x_j(k)$ only if (v_i, v_j) is an edge in the graph g. We will denote such a control law as *having structure* g. We will assume that the graph g has self-loops at each vertex, so that a node always has access to its own state value for the purpose of calculating its control input. We will make two assumptions before proceeding further:

1. We will assume that the control law is static. Thus, the control input at any time k is given by $u(k) = Fx(k)$.

2. We will consider the infinite horizon version of the cost function with the matrix R in the cost function as the identity matrix[1]. Thus, the cost is given by

$$J = \sum_{k=0}^{\infty} \left(x^T(k) Q x(k) + u^T(k) R u(k) \right). \tag{2.10}$$

We are interested in minimizing the cost function given in (2.10) over the topology and the control

[1]This assumption allows us to concentrate on the effect of the topology by just changing the weight matrix Q. If the structure of R also changes suitably along with that of Q in the discussion below, we do not need R to be identity.

law simultaneously. If there is no restriction on the structure of the control law, this optimization would typically yield a fully interconnected topology. In practice, however, building and maintaining each of the graph's communication edges has a cost which, if taken into account, may make this control law less attractive. To capture this trade-off between the closed loop performance and the controller topology, we introduce an *information cost* associated with each graph g. For a given matrix Q in the cost function (2.10), denote the cost achievable by the control law u that satisfies the structure according to the graph g as $J_g(Q,u)$. Further, define the minimal cost achievable this way as $J_g^\star(Q)$. Thus,

$$J_g^\star(Q) = \min_u J_g(Q,u).$$

Now we introduce the control cost of a graph through a mapping

$$\mathcal{Q} : \mathcal{G}_N \to \mathcal{S}_N,$$

where $\mathcal{S}_N$ is the set of all positive semidefinite matrices of size $N \times N$. Thus, the cost matrix Q in the cost function (2.10) is now a function of the graph g that decides the structure of the control law. Dependence of the cost function to be minimized on the topology that the control law needs to satisfy captures the information cost of the graph. In particular, our choice of a graph-dependent weight matrix may be seen as putting a price on the amount of energy used for communication, that fits naturally in the LQR framework that we are considering. There are, of course, many ways in which the mapping $\mathcal{Q}$ may be chosen. We give some possible ways below along with a physical interpretation.

1. Edge Separable Without Interference: In the case, the map $\mathcal{Q}$ satisfies

$$\mathcal{Q}(g) \mapsto Q_0 + \sum_{\substack{(v_i v_j)\in\mathcal{E}(g) \\ i\neq j}} P_{ij}, \tag{2.11}$$

   where $P_{ij} \geq 0$ is partitioned according to the subsystems and has all its blocks zero except (possibly) the (i,i)-th, (i,j)-th, (j,i)-th and (j,j)-th ones. This models the situation in which every node pays an energetic price to transmit its state value along every adjacent edge.

2. Edge Separable With Interference: In this case, the mapping $\mathcal{Q}(g)$ is still given as in (2.11), but each matrix P_{ij} merely belongs to the set $\mathcal{F}(g)$[2]. This corresponds to the case when the energetic cost paid for communication over each link depends on all the other links present in the graph. It is to capture this parasitic effect of edges on each other that we say that interference is present.

[2]Strictly speaking, any element of the set $\mathcal{F}(g)$ was defined to have dimensions $m \times n$ while the matrices P_{ij} have dimensions $n \times n$. However, the basic feature of the matrix being defined componentwise with the (i,j)-th block being 0 if $(v_i v_j) \notin \mathcal{E}(g)$ is still present. Further, in this case, we also impose the additional constraint $P_{ij} \geq 0$.

3. Non-Separable: In this case, there is no edge-by-edge contribution to the information cost of a graph and $\mathcal{Q}(g)$ can be a full matrix for any graph g. This can model a situation in which all the nodes agree to 'subsidize' the cost of any edge by reducing their own energies.

We define two general classes of the mapping $\mathcal{Q}(g)$. We say that the mapping $\mathcal{Q}(g)$ is

- non-decreasing: if $\forall g, g' \in \mathcal{G}_N, \qquad g \preceq g' \Rightarrow \mathcal{Q}(g) \leq \mathcal{Q}(g')$.
- structure-compatible: if[3] $\forall g \in \mathcal{G}_N, \qquad \mathcal{Q}(g) \in \mathcal{F}(g)$.

Note that edge separability in the mapping $\mathcal{Q}(g)$ is not sufficient for structure compatibility, since the matrix Q_0 in (2.11) may not be block diagonal to begin with.

In the next section, we present some results for specific classes of $\mathcal{Q}$ regarding the trade-off between the graph structure and the minimum cost J that is achievable.

2.7 Efficiency of a Graph

We define the *value* of a graph g as

$$V(g) = J_g^\star\left(\mathcal{Q}(g)\right).$$

Further, a graph $g^\star$ is optimal (or in the terminology of [106], *efficient*), if

$$\forall g \in \mathcal{G}_N, \qquad V(g^\star) \leq V(g).$$

The structure imposed by $g^\star$ corresponds to the minimal (in the sense of the quadratic cost (2.10)) communication requirements needed to control the N agents. Since there are only finitely many elements in $\mathcal{G}_N$, $g^\star$ always exists. We are interested in seeing if the mapping $\mathcal{Q}(g)$ provides us an indication of what $g^\star$ is, without having to explicitly calculate the values of all graphs g. Further we are also interested in seeing if there is any (partial) order imposed by the value function V.

2.7.1 Clique Graphs and the Efficient Graph

We begin with the following proposition that is easily proven.

Proposition 2.4 *1. If $g \preceq g'$, then $\forall Q \geq 0, \qquad J_{g'}^\star(Q) \leq J_g^\star(Q)$.*

2. If $Q \leq Q'$, then $\forall g \in \mathcal{G}_N, \qquad J_g^\star(Q) \leq J_g^\star(Q')$.

Proof 1. Let the optimizing control law with structure g that achieves $J_g^\star(Q)$ be given by $K(g)$. Since $g \preceq g'$, $K(g)$ satisfies the structure of g' as well. Thus, using the control law $K(g)$, the cost $J_1 = J_g^\star(Q)$ is achievable on g'. But, by definition, $J_{g'}^\star(Q) \leq J_1$. Combining the two relations, we see that $J_{g'}^\star(Q) \leq J_g^\star(Q)$. ∎

[3] See footnote 2.

2. If $Q \leq Q'$ then $Q' = Q + \bar{Q}$, where $\bar{Q} \geq 0$. Thus, it is obvious that $J_g(Q', u) = J_g(Q, u) + \Delta$, where $\Delta \geq 0$. Thus,

$$\begin{aligned} J_g^\star(Q') = \min_u J_g(Q', u) &\geq \min_u J_g(Q, u) + \min_u \Delta \\ &\geq \min_u J_g(Q, u) = J_g^\star(Q). \quad \blacksquare \end{aligned}$$

Corollary 2.5 *If the map $\mathcal{Q}(g)$ is non-decreasing and $g \preceq g'$, $J_{g'}^\star(\mathcal{Q}(g)) \leq V(g')$.*

Proof Applying the second part of the result in Proposition 2.4 to $Q = \mathcal{Q}(g)$ and $Q' = \mathcal{Q}(g')$ yields

$$J_{g'}^\star(\mathcal{Q}(g)) \leq J_{g'}^\star(\mathcal{Q}(g')).$$

Since by definition

$$J_{g'}^\star(\mathcal{Q}(g')) = V(g'),$$

the result holds. $\blacksquare$

As we saw in the initial sections of the chapter, determining the optimal structured controller for arbitrary topologies is a hard problem. Thus, determining the value of arbitrary graphs is also hard. However, we are able to compute the value of special graph classes. In particular, the following result holds for clique graphs.

Proposition 2.6 *Let $g \in \mathcal{G}_N$ be a clique graph and the mapping $\mathcal{Q}(g)$ be structure compatible. Then,*

$$V(g) = \min_u J(\mathcal{Q}(g), u),$$

i.e., provided that the cost matrix Q satisfies the structure of the graph g, the optimal unstructured LQ controller automatically satisfies the structure of g as well. Thus, in particular,

$$V(g) = x^T(0)P(g)x(0),$$

where $P(g)$ is the unique positive semi-definite solution of the Riccati equation

$$P(g) = A^T P(g)A + Q - A^T P(g)B\left(B^T P(g)B + I\right)^{-1} B^T P(g)A. \tag{2.12}$$

Proof It is standard that the unstructured control law minimizing $J(\mathcal{Q}(g), .)$ and the corresponding optimal value are given by the Riccati equation (2.12). We need to show that, under our assumptions, this optimal control law in fact has structure g.

Since g is a clique graph, there exists a permutation matrix π such that

- $\pi^{-1} = \pi^T$.
- $\pi^{-1}\mathcal{A}(g)\pi$ is block diagonal. Recall that $\mathcal{A}(g)$ is the adjacency matrix of the graph g.

This operation simply amounts to re-numbering the vertices of g so that vertices in the same clique have consecutive numbers and then partitioning the node set $\{v_1, \cdots, v_N\}$ as

$$\{v_1, \cdots, v_N\} = \bigcup_{i=1}^{c} \mathcal{I}_i,$$

where c is the number of cliques in g and $\mathcal{I}_i$ is the index set corresponding to clique i.

Let us define the $n \times n$ matrix Π_n by replacing every entry that is equal to '1' in the permutation matrix π by the identity matrix of proper dimensions. Similarly, define a $m \times m$ matrix Π_m. Both Π_n and Π_m are permutation matrices. Since $\mathcal{Q}$ is structure compatible, $\Pi_n^T \mathcal{Q} \Pi_n$ is block diagonal. Let us denote

$$\Pi_n^T \mathcal{Q} \Pi_n = \mathrm{diag}_{1 \le i \le c} \left(\mathcal{Q}_i \right).$$

Since the matrices A and B are block diagonal (and hence structure compatible), we can write

$$\Pi_n^T A \Pi_n = \mathrm{diag}_{1 \le i \le c} \left(\mathbf{A}_i \right) \qquad \Pi_n^T B \Pi_m = \mathrm{diag}_{1 \le i \le c} \left(\mathbf{B}_i \right),$$

where each block $\mathbf{A}_i$ and $\mathbf{B}_i$ is itself block-diagonal and corresponds to each clique in the graph. Thus,

$$\mathbf{A}_i = \mathrm{diag}_{j \in \mathcal{I}_i} \left(A_j \right) \qquad \mathbf{B}_i = \mathrm{diag}_{j \in \mathcal{I}_i} \left(B_j \right).$$

Since each pair (A_i, B_i) is controllable, so is the pair $(\mathbf{A}_i, \mathbf{B}_i)$ for each i. Hence, the Riccati equation

$$\begin{aligned} \bar{P} = (\Pi_n^T A \Pi_n)^T \bar{P} (\Pi_n^T A \Pi_n) + \Pi_n^T Q \Pi_n \\ - (\Pi_n^T A \Pi_n)^T \bar{P} (\Pi_n^T B \Pi_m) \left((\Pi_n^T B \Pi_m)^T \bar{P} (\Pi_n^T B \Pi_m) + I \right)^{-1} (\Pi_n^T B \Pi_m)^T \bar{P} (\Pi_n^T A \Pi_n) \end{aligned}$$

can be solved block-by-block and its unique positive semi-definite solution is $\Pi_n^T P(g) \Pi_n$. Further, it also has a block diagonal structure

$$\Pi_n^T P(g) \Pi_n = \mathrm{diag}_{1 \le i \le c} \left(\mathbf{P}_i \right).$$

Thus, the optimal control law for the original system is given by

$$\begin{aligned} F(g) &= -\left(B^T P(g) B + I\right)^{-1} B^T P(g) A \\ &= -\Big((\Pi_n \mathrm{diag}_{1\le i\le c}(\mathbf{B}_i)\Pi_m^T)^T \Pi_n \mathrm{diag}_{1\le i\le c}(\mathbf{P}_i)\Pi_n^T (\Pi_n \mathrm{diag}_{1\le i\le c}(\mathbf{B}_i)\Pi_m^T) \\ &\quad + \Pi_m \Pi_m^T \Big)^{-1} (\Pi_n \mathrm{diag}_{1\le i\le c}(\mathbf{B}_i)\Pi_m^T)^T \Pi_n \mathrm{diag}_{1\le i\le c}(\mathbf{P}_i)\Pi_n^T \Pi_n \mathrm{diag}_{1\le i\le c}(\mathbf{A}_i)\Pi_n^T \\ &= -\Pi_m^T \left((\mathrm{diag}_{1\le i\le c}(\mathbf{B}_i))^T \mathrm{diag}_{1\le i\le c}(\mathbf{P}_i) \mathrm{diag}_{1\le i\le c}(\mathbf{B}_i) + I \right)^{-1} \\ &\quad (\mathrm{diag}_{1\le i\le c}(\mathbf{B}_i))^T \mathrm{diag}_{1\le i\le c}(\mathbf{P}_i) \mathrm{diag}_{1\le i\le c}(\mathbf{A}_i) \Pi_n, \end{aligned}$$

where we have used the fact that both Π_n and Π_m are orthogonal matrices. Thus, the matrix $F(g)$ too has structure g. ∎

Proposition 2.6 is reminiscent of the results of [7] where it was proven that for spatially invariant systems, the optimal controller is itself spatially invariant. In our case, the optimal controller for a clique graph has the same structure as the graph, provided that the map $\mathcal{Q}$ is structure compatible. We can use this result to obtain the following result.

Proposition 2.7 *Let $\mathcal{Q}$ be non-decreasing and structure compatible and g be a clique graph. Then, $\forall g' \in \mathcal{G}_N$ such that $g \preceq g', V(g) \le V(g')$.*

Proof By Proposition 2.6, since g is a clique graph and $\mathcal{Q}$ is structure compatible,

$$V(g) = \min_u J\left(\mathcal{Q}(g), u\right).$$

Since the optimal controller has structure g, constraining the controller to structure g' cannot decrease the cost. Thus,

$$\min_u J\left(\mathcal{Q}(g), u\right) \le J^\star_{g'}(\mathcal{Q}(g)).$$

Finally, since the map $\mathcal{Q}$ is non-decreasing, Corollary 2.5 yields

$$J^\star_{g'}(\mathcal{Q}(g)) \le V(g').$$

Combining the three inequalities yields $V(g) \le V(g')$. ∎

This result can immediately be used to prove that the minimal control topology for the above assumptions is, in fact, the decentralized topology.

Corollary 2.8 *Let $\mathcal{Q}$ be non-decreasing and structure compatible. The graph $g^\star$ characterized by $\mathcal{E}(g^\star) = \Phi$, where Φ denotes the empty set, is efficient.*

Proof The result is obvious if we note that

1. $g^\star$ is a clique graph.

2. $g^\star \preceq g'$ for all $g' \in \mathcal{G}_N$.

Thus, $\forall g' \in \mathcal{G}_N, \qquad V(g) \leq V(g')$. ∎

Informally, this result states that cooperation can be detrimental for performance. Note that we know from Proposition 2.1 that for the purposes of stabilizability and controllability, all formations are equivalent. However, different formations will yield different performance levels, and this result states a model in which formations with more edges (hence more cooperation), in fact, perform worse. As an instance, consider the case of the mapping $\mathcal{Q}$ to be edge separable without interference. In this case, if the matrix Q_0 is block-diagonal, the map is structure-compatible. Then, the above result says that for any positive communication cost (i.e., no matter how small the matrices P_{ij} are) it is detrimental to include any edge in the controller's interconnection topology.

2.7.2 Pricing Edges

The previous section dealt with clique graphs. In this section, we give sufficient conditions for the addition of edges to *any* graph to be detrimental.

Proposition 2.9 Sufficiency: *Consider two graphs g and g'. If there exist $F \in \mathcal{F}(g')$ and $P > 0$ such that*

$$\begin{aligned} P &= (A+BF)^T P (A+BF) + \mathcal{Q}(g') + F^T F \qquad (2.13)\\ \mathcal{Q}(g) &\geq \mathcal{Q}(g') + \left(F + S^{-1}B^T P A\right)^T S \left(F + S^{-1}B^T P A\right) \qquad (2.14)\\ S &= B^T P B + I, \end{aligned}$$

then $V(g) \geq V(g')$.

Partial Converse: *Let either*

- *g be a clique graph and $\mathcal{Q}$ be structure compatible, or*
- *g be a complete graph,*

then $\mathcal{Q}(g) - \mathcal{Q}(g') \leq \mathcal{P} \Rightarrow V(g) \leq V(g')$, where

$$\begin{aligned} \mathcal{P} &= \left(\bar{F} + \bar{S}^{-1}B^T \bar{P} A\right)^T \bar{S} \left(\bar{F} + \bar{S}^{-1}B^T \bar{P} A\right),\\ \bar{S} &= B^T \bar{P} B + I, \end{aligned}$$

$\bar{F}$ is the optimal structured controller for the graph g' and $\bar{P}$ is the solution to (2.13) for $F = \bar{F}$.

Proof *Sufficiency*: Assume $P > 0$ satisfies (2.13). Then, since $\mathcal{Q}(g') + F^T F > 0$, the matrix $(A + BF)$ has all its eigenvalues inside the unit circle [64]. Thus,

$$\lim_{k\to\infty} x(k) = 0,$$

for the closed loop system

$$x(k+1) = (A + BF)x(k).$$

Also, from (2.13), we see that for every trajectory x of the closed loop system, we have

$$x^T(k)Px(k) - x^T(k+1)Px(k+1) = x^T(k)\mathcal{Q}(g')x(k) + u^T(k)u(k).$$

Writing one such equation for all $k = 0,\ 1,\ 2,\ \cdots$ and summing all these equations yields

$$x^T(0)Px(0) = \sum_{k=0}^{\infty} \left(x^T(k)\mathcal{Q}(g')x(k) + u^T(k)u(k)\right) = J_{g'}\left(\mathcal{Q}(g'), u\right), \tag{2.15}$$

for the structured controller $u(k) = Fx(k)$. Now from (2.13) we see that P satisfies

$$P = A^T PA - A^T PBS^{-1}B^T PA + \mathcal{Q}(g') + \left(F + S^{-1}B^T PA\right)^T S\left(F + S^{-1}B^T PA\right),$$

where $S = B^T PB + I$. This Ricatti equation is identical to the one obtained using LQ control theory if we were to look for a control law that minimizes a cost function of the form (2.10) with the cost matrix

$$Q = \mathcal{Q}(g') + \left(F + S^{-1}B^T PA\right)^T S\left(F + S^{-1}B^T PA\right).$$

Thus, we have

$$x^T(0)Px(0) = \min_u J\left(\mathcal{Q}(g') + \left(F + S^{-1}B^T PA\right)^T S\left(F + S^{-1}B^T PA\right), u\right).$$

Comparing this equation with (2.15) yields

$$J_{g'}\left(\mathcal{Q}(g'), u\right) = \min_u J\left(\mathcal{Q}(g') + \left(F + S^{-1}B^T PA\right)^T S\left(F + S^{-1}B^T PA\right), u\right). \tag{2.16}$$

Now, if inequality (2.14) holds, we have from the second part of Proposition 2.4

$$\min_u J\left(\mathcal{Q}(g') + \left(F + S^{-1}B^T PA\right)^T S\left(F + S^{-1}B^T PA\right), u\right) \leq \min_u J\left(\mathcal{Q}(g), u\right). \tag{2.17}$$

Finally, from the definition of the value function $V(g)$ we obtain

$$\begin{aligned} V(g') &\leq J_{g'}\left(\mathcal{Q}(g'), u\right) \\ \min_u J\left(\mathcal{Q}(g), u\right) &\leq V(g). \end{aligned} \tag{2.18}$$

Combining equations (2.16), (2.17) and (2.18) yields that if the conditions given in (2.13) and (2.14) hold, then

$$V(g') \leq V(g).$$

Partial Converse: As in the sufficiency proof, first note that for the control law $\bar{u} = \bar{F}x$, we have

$$J_{g'}\left(\mathcal{Q}(g'), \bar{u}\right) = \min_u J\left(\mathcal{Q}(g') + \mathcal{P}, u\right),$$

where $\bar{F}$ and $\mathcal{P}$ are as defined in the theorem statement. Since $\bar{F}$ is the optimal structured controller for g', we have

$$\begin{aligned} J_{g'}\left(\mathcal{Q}(g'), \bar{u}\right) &= V(g') \\ \Rightarrow \min_u J\left(\mathcal{Q}(g') + \mathcal{P}, u\right) &= V(g'). \end{aligned}$$

Now if $\mathcal{Q}(g) - \mathcal{Q}(g') \leq \mathcal{P}$, this yields

$$\min_u J\left(\mathcal{Q}(g), u\right) \leq V(g').$$

But if either g is complete or a clique graph with $\mathcal{Q}$ being structure compatible, the left hand side is simply $V(g)$. Thus, $V(g) \leq V(g')$. ∎

Proposition 2.9 can be used to design a map $\mathcal{Q}$ to enforce a particular design topology. Suppose a particular controller topology g' has been chosen. Then, one can ensure that adding any edge is unprofitable by choosing a stabilizing control gain $F \in \mathcal{F}(g')$, solving the Lyapunov equation (2.13) and picking the map $\mathcal{Q}$ such that (2.14) holds. However, the result is not very useful if given the map $\mathcal{Q}$ and a topology, one wants to see if it is (or not) advantageous to add new edges. In that case, the equations (2.13) and (2.14) need to be solved for both P and F. Even after using the Schur complement on (2.14) and rewriting (2.13) as two matrix inequalities, one ends up with a set of bilinear matrix inequalities, to solve which is, in general, NP-hard. Of course, if we are given a particular control law F (e.g., the consensus control law [157]) the inequalities can be solved easily. Similarly if we can find the value of P, a certificate whether or not adding edges is useful can be easily obtained. For instance, the following sufficient condition allows one, in some cases, to determine P and hence solve the above problem in a tractable fashion.

Proposition 2.10 *Let either*

- *g be a clique graph and $\mathcal{Q}$ be structure compatible, or*
- *g be a complete graph,*

then $V(g) \geq V(g')$ if there exists a matrix $F \in \mathcal{F}(g')$ such that the following Linear Matrix Inequality (LMI) is satisfied

$$\begin{bmatrix} -P(g) + \mathcal{Q}(g') & \begin{bmatrix} (A+BF)^T & F^T \end{bmatrix} \\ \begin{bmatrix} (A+BF) \\ F \end{bmatrix} & -\begin{bmatrix} P(g)^{-1} & 0 \\ 0 & I \end{bmatrix} \end{bmatrix} < 0,$$

where $P(g)$ is the positive semi-definite solution of the Riccati equation (2.12).

Proof Let $F \in \mathcal{F}(g')$ satisfy the given LMI. Then, using Schur complement, it also satisfies

$$P(g) - (A+BF)^T P(g)(A+BF) > \mathcal{Q}(g') + F^T F.$$

Pre-multiplying both sides by $x^T(k)$ and post-multiplying by $x(k)$ yields

$$\begin{aligned} x^T(k)\left(P(g) - (A+BF)^T P(g)(A+BF)\right)x(k) &> x^T(k)\left(Q + F^T F\right)x(k) \\ \Rightarrow x^T(k)P(g)x(k) - x^T(k+1)P(g)x(k+1) &> x^T(k)\left(Q + F^T F\right)x(k). \end{aligned}$$

Adding these inequalities for $k = 0, 1, \cdots$ yields

$$x^T(0)P(g)x(0) > J_{g'}\left(\mathcal{Q}(g'), u\right),$$

where $u = Fx$. Now if g satisfies either of the two assumptions of the theorem statement, the left hand side is simply $V(g)$. The right hand side, by definition, satisfies

$$V(g') \leq J_{g'}\left(\mathcal{Q}(g'), u\right).$$

Combining these facts yields $V(g) \geq V(g')$. ∎

2.8 Examples

The results of the previous section allow us to give a complete picture for the case when $N = 3$. We consider the case of mapping $\mathcal{Q}$ to be edge-separable with no interference with Q_0 as the identity matrix. It is easy to see that $\mathcal{Q}$ is non-decreasing as well as structure-compatible. The results are summarized in Figure 2.6 in the form of a lattice corresponding to the partial order '$\preceq$'. The

continuous lines indicate that for the map $\mathcal{Q}$ mentioned below, the bottom graph has value less than the top graph. The dashed lines indicate transition with possible 'critical prices' at which adding an edge becomes disadvantageous.

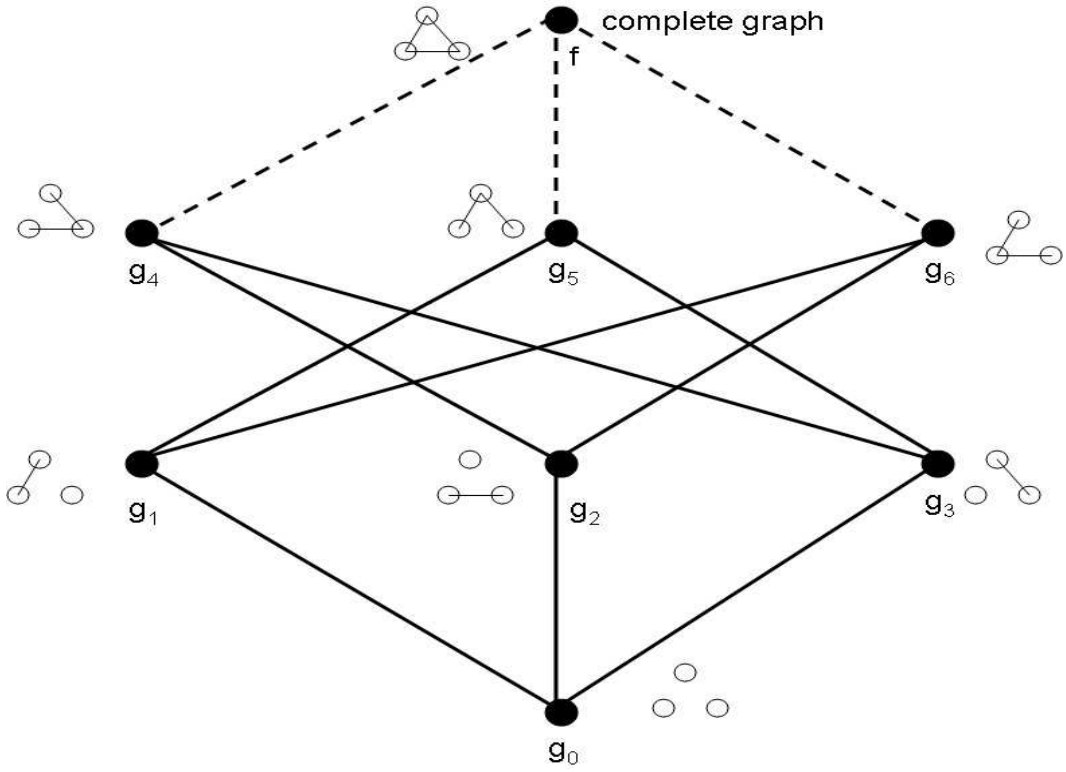


Figure 2.6: Comparison of the value of the graphs for the partially ordered set of all graphs on 3 vertices.

Using Proposition 2.7, we can immediately see that

$$\begin{aligned} V(g_0) &\leq V(g) \qquad \forall g \\ V(g_4) &\geq V(g_2), V(g_3) \\ V(g_5) &\geq V(g_1), V(g_3) \\ V(g_6) &\geq V(g_1), V(g_2) \\ V(g_f) &\geq V(g_0), V(g_1), V(g_2), V(g_3). \end{aligned}$$

However, we cannot a priori compare $V(g_f)$ with $V(g_4)$, $V(g_5)$ or $V(g_6)$. We use a gradient descent algorithm introduced in [82] to construct the optimal constrained controller for the infinite-horizon cost and then use Proposition 2.9 to compute the critical prices at which adding edges to obtain the fully connected graph becomes detrimental. The gradient descent algorithm is also described in Appendix 1 of the chapter.

We consider all subsystems to have a single state and the matrix values

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \qquad B = I$$

$$P_{12} = \begin{bmatrix} 10 & 20 & 0 \\ 20 & 50 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad P_{23} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 10 & 20 \\ 0 & 20 & 50 \end{bmatrix}.$$

We can obtain an approximate value for $V(g_5)$ and find that

$$\left(F(g_5) - P(g_5)\right)^T \left(F(g_5) - P(g_5)\right) = \begin{bmatrix} 0.0591 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.0591 \end{bmatrix},$$

after the algorithm in [82] converges. Thus, choosing

$$P_{13} = \begin{bmatrix} 0.04 & 0 & 0.01 \\ 0 & 0 & 0 \\ 0.01 & 0 & 0.04 \end{bmatrix} \leq \left(F(g_5) - P(g_5)\right)^T \left(F(g_5) - P(g_5)\right)$$

always yields $V(g_f) < V(g_5)$ while

$$P_{13} = 2 \times \begin{bmatrix} 0.04 & 0 & 0.01 \\ 0 & 0 & 0 \\ 0.01 & 0 & 0.04 \end{bmatrix} \leq \left(F(g_5) - P(g_5)\right)^T \left(F(g_5) - P(g_5)\right)$$

yields $V(g_f) > V(g_5)$.

2.9 Discussion

We began our discussion of distributed control with the problem of synthesis of a LQR optimal control law that is constrained to lie in a particular vector space. We saw that the problem was hard to solve in general. We presented a computationally expensive method for the optimal finite time horizon control and a computationally easier method to generate a sub-optimal control law. We presented examples which illustrated that the loss in performance due to the sub-optimal algorithm is not huge and that communication in general helps to bring down the cost. The methods involve the solution of linear equations and are thus free from convergence problems.

Although this work is a significant advance in the field in many respects, more work is needed to fully understand and solve the problem of optimal control of a network of dynamic agents. The optimal control law we have presented serves as a good benchmark to evaluate any other control strategy; however, it is computationally very expensive and good approximations that are more tractable will be useful in many situations. We have presented one such sub-optimal algorithm. From numerical examples it seems that the loss in performance is not huge. However, we have not been able to obtain an analytic expression for the loss in performance or a bound on it. Similarly, an analytic method for obtaining a relation between topology and optimal cost is needed. Our experience has been that not all topologies with same number of edges are equally good. The best topologies given the same number of edges tend to be the ones which are of the leader-follower type, in which there is one node in contact with all others. More work is needed to obtain analytic statements along these lines. Finally, work characterizing the effect of topology on the cost will also help in understanding the robustness of the algorithms to knowledge of the topology. Currently, we assume that either the control law is calculated off-line by a central processor, or each node knows the topology and calculates the control law for the entire system. To make this implementation more scalable, it will be useful to understand the effects of topology changes far from the individual agent, or equivalently, to imperfect knowledge of the topology far away.

In keeping with our central thesis of designing information flow, in this chapter, we also presented a model to evaluate the effect of topology on distributed control. The results that we presented are only a first attempt at quantifying the influence of topology on distributed control problems. The problem is hard partly because the underlying problem of obtaining the optimal constrained controller itself is largely unsolved. However, we have still been able to give some rigorous statements regarding optimal topologies and comparing some topologies to another. The results show that under some assumptions, any communication cost (no matter how small) can make adding edges detrimental in terms of performance. Thus, cooperation may not always be useful.

Clearly, there is much left to do in this promising direction. It would be nice to compare the values of two arbitrary graphs, not necessarily related by the partial order '$\preceq$'. Similarly, the conditions in Proposition 2.9 are currently tractable only if either the matrix F or P are given. Whether the conditions can be made tractable for arbitrary graphs is still an open problem. Another important long-term direction that emanates from this work is seeing if the synthesis *process* of the controller can be decentralized. Currently, calculation of the value of a graph (hence the determination whether or not a topology is to be implemented) is done by a central unit. A truly decentralized framework should have a mechanism in which the agents decide on the local links that wish to maintain based on some combination of local and global utility functions [148, 179]. This is similar, in spirit, to obtaining the optimal controller as a Nash equilibrium of a game. However, much more work is needed in this direction.

Appendix A: Optimal Constrained Control Law Synthesis for the Infinite Horizon Case

In this appendix, we provide an alternative gradient descent based algorithm for the synthesis of a constrained controller that is suitable when the quadratic cost function is infinite horizon. Consider once again the system

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ u(k) &= Fx(k), \end{aligned} \tag{2.19}$$

where the initial condition $x(0)$ is Gaussian with zero-mean and covariance $R(0)$. We wish to minimize the cost function

$$J = E\left[\sum_{k=0}^{\infty}\{x^T(k)Qx(k) + u^T(k)Ru(k)\}\right], \tag{2.20}$$

where $Q > 0$ and $R \geq 0$. In addition, we wish to constrain the control law to lie within a space spanned by the basis vectors $\{\Theta_j,\, j = 1, 2, \ldots, p\}$. Thus, the problem is to find a control law of the form

$$F = \sum_{j=1}^{p} \alpha_j \Theta_j, \tag{2.21}$$

where α_j's are scalars, that minimizes the above cost function. Assume that a F of the form (2.21) exists, such that $A + BF$ is stable. Then, for that F, we obtain from (2.20)

$$\begin{aligned} J &= E\left[x^T(0)\sum_{i=0}^{\infty}((A+BF)^T)^i(Q + F^TRF)(A+BF)^i x(0)\right] \\ &= E\left[x^T(0)Px(0)\right], \end{aligned}$$

where P is defined by

$$P = \sum_{i=0}^{\infty}((A+BF)^T)^i(Q + F^TRF)(A+BF)^i.$$

It is apparent from the definition of P, that it satisfies the Lyapunov equation

$$P = (Q + F^TRF) + (A+BF)^T P(A+BF). \tag{2.22}$$

Denote the covariance of the state at time step k by $R(k)$. Thus, the cost is given by $J = \text{trace}(PR(0))$. We wish to find the coefficients $\alpha_1, \alpha_2, \ldots, \alpha_p$ such that a control law of the form (2.21) minimizes the cost $J = \text{trace}(PR(0))$ where P satisfies (2.22) and $A+BF$ is stable. By the properties of the Lyapunov equation, if $(A + BF)$ is stable, P is positive-definite.

The Case where Noise is Present

Suppose that instead of (2.19), the system evolves as

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + w(k) \\ u(k) &= Fx(k), \end{aligned}$$

where the additional term $w(k)$ is stationary white Gaussian noise with zero mean and covariance R_w. Since the random noise goes on till infinity, the cost function considered in equation (2.20) will diverge. Thus, we need to reconsider what we mean by optimizing the cost function here. Following [118], we consider the cost function

$$J = \lim_{k\to\infty} E\left[x^T(k)Qx(k) + u^T(k)Ru(k)\right].$$

Since $(A+BF)$ is stable, J does not depend on the initial condition $x(0)$. Then, by a derivation similar to the one given above, we can write $J = \text{trace}\,(PR_w)$, where P satisfies (2.22). So we see that this problem reduces to the same formulation as above, if we redefine $R(0)$ to be R_w. From now on, we will study only the original formulation, assuming that such re-definitions have been carried out.

Algorithm 1

Differentiating (2.22) with respect to α_i, we obtain

$$\frac{\partial P}{\partial \alpha_i} = (A+BF)^T \frac{\partial P}{\partial \alpha_i}(A+BF) + \Phi_i^T[RF + B^T P(A+BF)] + [(A+BF)^T PB + F^T R]\Phi_i, \quad (2.23)$$

for all $i = 1, 2, \ldots, p$. We wish to find minima of trace$(PR(0))$. For a critical point,

$$\frac{\partial \text{trace}(PR(0))}{\partial \alpha_i} = 0 \qquad \Rightarrow \qquad \text{trace}\left(\frac{\partial P}{\partial \alpha_i} R(0)\right) = 0, \qquad \forall i = 1, 2, \ldots, p.$$

If we define $\Sigma = \Phi_i^T[RF + B^T P(A+BF)]$, we can write

$$\begin{aligned} \text{trace}\left(\frac{\partial P}{\partial \alpha_i} R(k)\right) &= \text{trace}\left((A+BF)^T \frac{\partial P}{\partial \alpha_i}(A+BF)R(k)\right) + \text{trace}\left(\Sigma R(k) + R(k)\Sigma^T\right) \\ &= \text{trace}\left(\frac{\partial P}{\partial \alpha_i}(A+BF)R(k)(A+BF)^T\right) + \text{trace}\left(\Sigma R(k) + R(k)\Sigma^T\right) \\ &= \text{trace}\left(\frac{\partial P}{\partial \alpha_i} R(k+1)\right) + \text{trace}\left(\Sigma R(k) + R(k)\Sigma^T\right). \end{aligned}$$

Using the above relation repeatedly, we can write

$$\text{trace}\left(\frac{\partial P}{\partial \alpha_i} R(0)\right) \;=\; \text{trace}\left(\Sigma\left(R(0)+\cdots+R(k)\right)+\Sigma^T(R(0)+\cdots+R(k))+\frac{\partial P}{\partial \alpha_i}R(k+1)\right).$$

Since $(A+BF)$ is stable, $R(k)$ is approximately a zero matrix for sufficiently large values of k. Thus, if we define $X = R_0 + R_1 + R_2 + \ldots$, so that X satisfies the Lyapunov equation

$$X = R_0 + (A+BF)X(A+BF)^T, \tag{2.24}$$

we can write the necessary condition for a critical point as

$$\text{trace}\Big(\Phi_i^T[B^TP(A+BF)+RF]X + [(A+BF)^TPB+F^TR]\Phi_i X\Big) = 0, \forall i = 1,\ldots,p, \tag{2.25}$$

where F satisfies (2.21), P satisfies (2.22) and X satisfies (2.24). Note that since we have only p free variables, we would not be able to satisfy more stringent conditions like

$$(A+BF)^TPB + F^TR = 0,$$

or even the somewhat relaxed condition

$$\Phi_i^T((A+BF)^TPB+F^TR) = 0, \forall i = 1,2,\ldots,p.$$

Also, note that if Φ_i denotes a matrix with all elements zero except the (j_i,k_i)th element being unity; the necessary condition given in (2.25) reduces to

$$\left[\left(B^TP(A+BF)+RF\right)X\right]_{j_i,k_i} = 0, \forall i = 1,2,\ldots,p,$$

where $[A]_{ij}$ denotes the (i,j)-th element of the matrix A. Thus, in the particular case when F has no restrictions on its structure we get back the usual condition $B^TP(A+BF)+RF = 0$.

One method to obtain the control law is to solve equation (2.25) iteratively. Alternatively, we can also use a gradient search algorithm for the minimization problem. The algorithm is given by

1. **Initialize:**
   (a) Start from an initial guess of the set $\{\alpha_i\}$.
   (b) Solve equation (2.22) for P using this value of the control law.
   (c) Check if P is positive semi-definite. If yes, proceed to the update step; else repeat initialization with another guess.
2. **Update:**

(a) Solve equation (2.23) for $\frac{\partial P}{\partial \alpha_i}$ using the set $\{\alpha_i\}$.

(b) Calculate the cost trace$(PR(0))$ and the gradient of the cost $\frac{\partial P}{\partial \alpha_i}R(0)$.

(c) Update the guess by changing the current guess $\{\alpha_i\}$ by some constant amount δ times the gradient of the cost function.

(d) Resolve equation (2.22) for P using this value of the control law.

(e) i. Check if P is positive semi-definite.

ii. Check if the cost is reduced by this value of P.

iii. If both i and ii are true, proceed to the next step. Otherwise, reduce δ by a half and again try to update. If stuck on this step for a long time, declare minima reached and terminate.

(f) Adopt the updated value of guess as the current guess and go through the update step again.

As discussed, e.g., in [199], generating suitable initial guesses for the general case is a non-trivial task. However, because of Proposition 2.1, for our application there exists a particularly simple way to generate the initial guess. We find the control law required by each vehicle to stabilize itself while using only its own information. The initial guess can always be the block diagonal matrix formed by stacking these laws along the diagonal of a matrix. This will always be a control law which stabilizes the formation, yet satisfies the topological constraints.

Algorithm 2

We present an alternate iterative algorithm in this sub-section that sheds additional light on the cost function. If we denote

$$Y = R^{1/2}(0)PR^{1/2}(0), \tag{2.26}$$

where $R^{1/2}(0)$ denotes the positive definite square-root of the positive definite matrix $R(0)$, and scale the matrices as[4]

$$A \leftarrow R^{-1/2}(0)AR^{1/2}(0) \qquad B \leftarrow R^{-1/2}(0)B$$
$$F \leftarrow FR^{1/2}(0) \qquad Q \leftarrow R^{1/2}(0)QR^{1/2}(0),$$

then our problem is to minimize trace(Y) subject to

$$Y = (A + BF)^T Y (A + BF) + (F^T RF + Q), \tag{2.27}$$

[4]Note that since we have scaled F, the new basis vectors Φ_i will themselves be scaled as $\Phi_i \leftarrow \Phi_i R^{1/2}(0)$.

and $A + BF$ being stable. Completing the squares in (2.27) and denoting

$$F = \sum_i \alpha_i \Phi_i, \qquad S = R + B^T Y B, \qquad F_c = S^{-1} B^T Y A,$$

we obtain the equation for the cost in the form

$$\text{trace}(Y) = \text{trace}\left((\sum_i \alpha_i \Phi_i + F_c)^T S (\sum_i \alpha_i \Phi_i + F_c) + A^T Y A + Q - A^T Y^T B S^{-1} B^T Y A \right). \tag{2.28}$$

Note that F_c is the centralized control law, in the sense that this is the optimal control law when there is no restriction on the form of F. Thus, we see from (2.28) that the cost function consists of two parts. There is a cost incurred even when F has no constraints on its structure. It is given by the second term on the right hand side of (2.28). When we impose constraints on the structure of F, we incur an additional cost $\text{trace}(Y_{dc})$ given by

$$\text{trace}(Y_{dc}) = \text{trace}\left((\sum_i \alpha_i \Phi_i + F_c)^T S (\sum_i \alpha_i \Phi_i + F_c) \right). \tag{2.29}$$

We rewrite (2.29) as

$$\begin{aligned} \text{trace}(Y_{dc}) &= \sum_i \alpha_i \sum_j \alpha_j \text{trace}(\Phi_i^T S \Phi_j) + 2 \sum_i \alpha_i \text{trace}(\Phi_i^T S F_c) + \text{trace}(F_c^T S F_c) \\ &= \begin{bmatrix} \alpha & 1 \end{bmatrix} M \begin{bmatrix} \alpha & 1 \end{bmatrix}^T, \end{aligned}$$

where

$$\begin{aligned} \alpha &= \begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_p \end{bmatrix} \\ M &= \begin{bmatrix} R_d & D \\ D^T & \text{trace}(F_c^T S F_c) \end{bmatrix} \\ R_d &= \begin{bmatrix} \text{trace}(\Phi_1^T S \Phi_1) & \text{trace}(\Phi_1^T S \Phi_2) & \dots & \text{trace}(\Phi_1^T S \Phi_N) \\ \text{trace}(\Phi_2^T S \Phi_1) & \ddots & & \vdots \\ \vdots & & & \\ \text{trace}(\Phi_N^T S \Phi_1) & \dots & \dots & \text{trace}(\Phi_N^T S \Phi_N) \end{bmatrix} \\ D &= \begin{bmatrix} \text{trace}(\Phi_1^T S F_c) & \text{trace}(\Phi_2^T S F_c) & \dots & \text{trace}(\Phi_N^T S F_c) \end{bmatrix}^T. \end{aligned}$$

On completion of squares, we obtain that the optimal value of the vector α is given by

$$\alpha = -D^T R_d^{-1}. \tag{2.30}$$

Thus, the iterative algorithm is given as follows.

1. Start from an initial guess F_0. Use this as the initial guess and solve for P by using (2.22). Calculate the initial guess for Y from this P by using (2.26) and denote it by Y_0.

2. Perform the following iteration

   (a) Solve for Y_{t+1} given the values Y_t and F_t, from the equation

   $$Y_{t+1} = (A + BF_t)^T Y_t (A + BF_t) + (F_t^T R F_t + Q).$$

   (b) From Y_{t+1} solve for S_{t+1} and $(SF_c)_{t+1}$ by using

   $$\begin{aligned} S_{t+1} &= R + B^T Y_{t+1} B \\ (SF_c)_{t+1} &= B^T Y_{t+1} A. \end{aligned}$$

   (c) Calculate R_d, D and the estimate of the control law as

   $$\begin{aligned} F_{t+1} &= \sum_j \alpha_{j,t+1} \Phi_j \\ \alpha_{t+1} &= \begin{bmatrix} \alpha_{1,t+1} & \alpha_{2,t+1} & \dots & \alpha_{N,t+1} \end{bmatrix} \\ \alpha_{t+1} &= -D^T R_d^{-1}. \end{aligned}$$

Note that the iteration in the above algorithm should be performed only as long as the cost is decreasing. Although we have no formal proof for it, numerical evidence suggests that the algorithm usually converges.

The Completely Decentralized Case

We consider the case where no agent has access to measurements of states of other agents. This case is interesting since we can analytically calculate the optimal control law and evaluate the cost. Moreover, a lower bound for the cost for any topology is given by the cost achievable for the completely decentralized case. For simplicity, we consider the case for the agent dynamics being a single integrator. We denote the (i, j)-th element of matrix P by $[P]_{i,j}$. If we let Φ_i be the matrix with zeros everywhere except the (i, i)-th element, we know that the optimal control law is

$$F = \sum_i \alpha_i \Phi_i,$$

such that F satisfies

$$\left[\left(B^T P (A + BF) + RF \right) X \right]_{i,i} = 0, \forall i.$$

The matrix X satisfies the Lyapunov equation

$$X = (I + hF)X(I + hF)^T + R(0).$$

If all the initial states are independent of each other and randomly chosen, $R(0)$ is a diagonal matrix with all diagonal entries positive. This, in turn, means that X is a diagonal matrix with

$$[X]_{i,i} = \frac{[R(0)]_{i,i}}{1 - (1 + h[F]_{i,i})^2}.$$

The condition for optimal F reduces to

$$\begin{aligned} \left([hP(I + hF)]_{i,i} + [RF]_{i,i}\right) X_{i,i} &= 0 \\ \Rightarrow h[P]_{i,i} + h^2[P]_{i,i}[F]_{i,i} + [R]_{i,i}[F]_{i,i} &= 0. \end{aligned}$$

Since

$$[P]_{i,i} = \sum_{k=0}^{\infty}(1 + h[F]_{i,i})^{2k}[F^T RF + Q]_{i,i} = \frac{([F]_{i,i})^2[R]_{i,i} + [Q]_{i,i}}{1 - (1 + h[F]_{i,i})^2},$$

the optimal control law and the resulting cost can be readily calculated. Thus, if we do not allow the vehicles to talk to each other, the off-diagonal terms of the cost matrices do not matter. Moreover, in this case, the cost simply turns out to be the sums of the costs incurred in controlling individual vehicles using their own state measurements with the cost function involving only the diagonal terms of the cost function matrices. Note that this analysis holds for the cases of all vehicles not being the same and also for general plant dynamics where we talk about block diagonal matrices. However, it breaks down if, e.g., the initial conditions are not all independent.

As a simple example, we consider a formation of MVWT vehicles, as described in the Example 2 of Section 2.5. We consider 8 vehicles starting from an octagonal formation and consider the topologies possible as the communication radius of each vehicle is increased. The initial covariance matrix R_0 is the identity matrix. The cost function matrix R is also identity while the matrix Q is randomly generated. A typical curve for the varying of the costs as the communication radius is increased is given in Figure 2.7. Once again we see that as more and more communication is allowed, the cost goes down. However, the marginal utility of each communication link decreases as more and more links are added. This may also be due to the fact that the edges added later bring the data of far-away vehicles which is not so important for stabilization.

To show the effect of the topology, we consider an example using 5 agents with single integrator

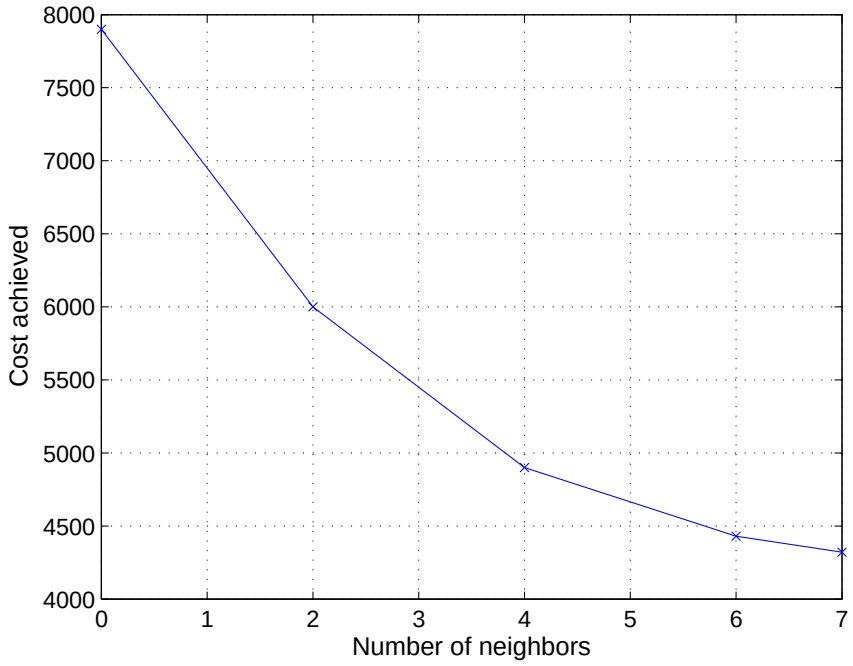


Figure 2.7: As the communication radius is increased, the cost goes down: the infinite horizon case.

dynamics described in Example 1 of Section 2.5. We use the cost function matrix Q as

$$Q = \begin{bmatrix} 4 & -1 & -1 & -1 & -1 \\ -1 & 4 & -1 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & -1 & -1 & 4 & -1 \\ -1 & -1 & -1 & -1 & 4 \end{bmatrix},$$

while R is taken to be identity matrix. R_0 is taken to be 10 times the identity matrix. We calculate the costs for all the graphs possible. Figure 2.8 shows the costs plotted as a function of the number of edges. As a general trend, we can see that there is a small variation in the cost for topologies with the same number of edges. The best topologies within the same equivalent class tended to be the ones which were of the leader-follower type, in which there was one node in contact with all others.

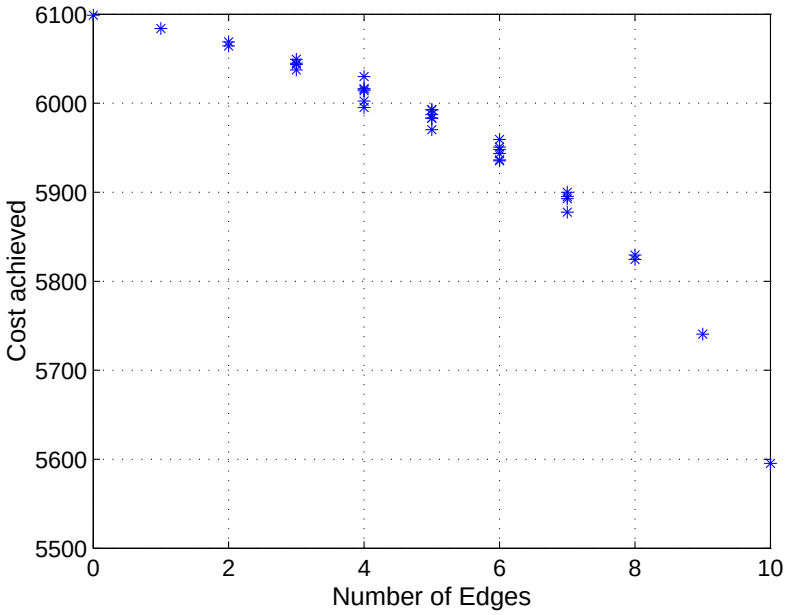


Figure 2.8: Cost achieved with all the graphs on 5 nodes. Not all topologies with same number of edges are equally good.

Appendix B: Distributed Motion Control for Estimation

In this appendix, we present a distributed motion control algorithm for optimal sensing by multiple mobile agents. Consider m targets doing a random walk in a plane. For simplicity, we assume that the targets are being modeled by a constant position model. Thus, the motion of the i-th target evolves according to

$$x_i(k+1) = x_i(k) + w_i(k), \tag{2.31}$$

where $x_i(k) \in \mathbf{R}^2$ is the state of the i-th target at time k consisting of the Cartesian x and y positions. $w_i(k)$ represents the process noise acting on the i-th target assumed zero-mean, Gaussian and white with covariance matrix $R_{w,i}$. Further, the process noises of different targets are assumed independent of each other.

The targets are tracked by N sensors that are modeled using a sonar sensor model [162, 189]. In such a model, each target j results in N measurements of the form

$$y_{ij}(k) = x_j(k) + \mathcal{T}(\theta_{ij})v_{ij}(k), \qquad i = 1, 2, \cdots, N \tag{2.32}$$

where $\mathcal{T}(\theta_{ij})$ is the rotation matrix that transforms the noise from the local sensor coordinates to the global coordinates. The noise $v_j(k)$ is again assumed zero-mean, Gaussian and white. The noises of various sensors and targets are assumed mutually independent. The covariance matrix $R_{ij}(k)$ of

the noise $v_{ij}(k)$ is assumed to be a diagonal matrix of the form

$$R_{ij}(k) = \begin{bmatrix} \left(\sigma_{ij}^{\text{range}}\right)^2 & 0 \\ 0 & \left(\sigma_{ij}^{\text{bearing}}\right)^2 \end{bmatrix}.$$

$\left(\sigma_{ij}^{\text{range}}\right)^2$ is the range noise variance and is a function $f(r_{ij})$ of the distance from the target j to the sensor i. A common model of $f(r_{ij})$ is as a quadratic dependence on range, with the minimum value being achieved at a particular distance from the target, namely the "sweet spot" of the sensor. The bearing noise variance $\left(\sigma_{ij}^{\text{bearing}}\right)^2$ is often modeled as a fixed multiple α of the range noise variance.

To process the observations and generate an estimate, a Kalman filter (KF) is used. However, a centralized KF for all the observations would be computationally very expensive. Instead, every node has a local Kalman filter that produces an estimate based only on local observations. Then, these estimates are exchanged and combined to yield a global estimate. The estimates are combined by assuming that there is no cross-covariance between local estimates[5] . Thus, we use the relations [11]

$$P_{\text{global}}^{-1}\hat{x}_{\text{global}} = \sum_{i=1}^{N} P_{\text{local}}^{-1}\hat{x}_{\text{local}}, \qquad\qquad P_{\text{global}}^{-1} = \sum_{i=1}^{N} P_{\text{local}}^{-1}.$$

P_{global} refers to the covariance of the error in the global estimate and hence is an indicator of the quality of the estimate. Since the sensor noise covariance matrix is a function of the distance between the sensor and the target, the quality of the estimate depends on the distances between the various sensors and the targets. Thus, by varying the positions of the sensors, we can vary the error covariance. The problem we pose is how to do so in a distributed way. As a cost function, we seek to minimize the determinant of the error covariance matrix P_{global}. This is referred to in the literature [141] as the D-optimal design.

Case I : Single Target

To begin with, assume that only a single target is present and every sensor is observing it. At every time step k, every sensor takes a measurement, calculates and transmits the local estimate and fuses information from all the sensors to obtain a global estimate. The task that remains for each sensor is to identify its optimal location for the next time step. As is discussed in [141], obtaining a policy in closed form (e.g. using dynamic programming) that minimizes the cost function over a long term is very difficult. Moreover, in many practical cases, the number of active sensors may keep on changing, making the problem even harder. Thus, we use a greedy gradient descent algorithm that defines the optimal control action (i.e., direction of motion) as one that will minimize the determinant of P_{global}

[5]Note that the assumption of no cross-covariance between the local estimates is not strictly true and hence the global estimate is sub-optimal. However, this algorithm is much simpler than its alternatives [10, 83] and seems not to incur huge performance penalties.

at the next time step. The algorithm proceeds as follows. Each sensor node, at every time step,

1. calculates the local estimate error covariance for all other nodes at the next time step by assuming that the nodes remain at their current position;

2. for each control action that it can take, calculates its own error covariance at the next time step and generates the corresponding global error covariance;

3. chooses the action that minimizes the cost (the determinant of P_{global}) at the next time step.

We assume that the set of possible control actions is finite, thus the gradient descent reduces to a discrete gradient search.

The algorithm is inherently distributed. Note that if a node transmits its local estimate to other sensors, it also implicitly transmits its current position. Also, if a node stops functioning, or a new node enters the field, the other agents can easily adapt to it.

Convergence Issues

Since the sensors use a distributed version of the descent algorithm, under the usual constraints of observability, the error estimates will reach a steady state and the sensor positions will be such that the cost function reaches *a* minimum (provided the step size α is small enough [16]). However, the minimum might only be local and not global. To take a look at the nature of minima, we need to take a closer look at the nature of the cost function. We look at a special case in which the local error covariance matrices are assumed to change only with position of the sensors and not with time. This can be the case, e.g., when the system has reached a steady state. We can relate changes in error covariance with changes in measurement noise covariance (hence the sensor positions) for examining the qualitative behavior of the cost function. Note that there are an infinite number of minima that are equivalent in the sense that we can generate the positions of sensor in one configuration from another configuration by simply rotating all the sensors about an axis perpendicular to the plane in which all the sensors lie. However, all these minima have equal value of the cost function and we can consider any particular one of them. We choose the one that has the sensor 1 on the x-axis. The other sensors $i = 2, \cdots, N$ are assumed to be at the position $(r_i \cos(\theta_i), r_i \sin(\theta_i))$. The cost function we are trying to minimize is

$$\det\left(P_{\text{global}}\right) = \det\left(\sum_i P_{i,\text{local}}^{-1}\right)^{-1}.$$

This is equivalent to maximizing the cost $\det\left(\sum_i P_{i,\text{local}}^{-1}\right)$. For the single target case, the function $f_{ij}(.)$ defined in respect to the measurement noise covariance does not depend on the argument j

and can be denoted by $f_i(.)$. For the i-th sensor, $P_{i,\text{local}}^{-1}$ is given by

$$\begin{bmatrix} \alpha f_i \sin^2(\theta_i) + f_i \cos^2(\theta_i) & (1-\alpha) f_i \cos(\theta_i)\sin(\theta_i) \\ (1-\alpha) f_i \cos(\theta_i)\sin(\theta_i) & f_i \sin^2(\theta_i) + \alpha f_i \cos^2(\theta_i) \end{bmatrix}.$$

Thus, the cost function to be maximized has the form $AD - B^2$, where

$$\begin{aligned} A &= \left[\frac{1}{f_1} + \frac{1}{\sigma}\sum\left[\frac{\sigma}{f_i} + \frac{(1-\sigma)\sin^2(\theta_i)}{f_i}\right]\right] \\ D &= \left[\frac{1}{f_1} + \frac{1}{\sigma}\sum\left[\frac{\sigma}{f_i} + \frac{(1-\sigma)\cos^2(\theta_i)}{f_i}\right]\right] \\ B &= \frac{(1-\sigma)^2}{\sigma^2}\left[\sum\frac{\cos(\theta_i)\sin(\theta_i)}{f_i}\right]^2 \end{aligned}$$

where the summation index i runs from 2 through N. This can be simplified to the cost function expression

$$\frac{1}{\sigma}\left[\sum_{i=1}^{N}\frac{1}{f_i}\right]^2 + \frac{(1-\sigma)^2}{\sigma^2 f_1}\left[\sum_{i=2}^{N}\frac{\sin^2(\theta_i)}{f_i}\right] + \frac{(1-\sigma)^2}{\sigma^2}\left[\sum_{2\le i<j\le N}\frac{\sin^2(\theta_i-\theta_j)}{f_i f_j}\right].$$

The following conclusions can readily be drawn from this form of the expression.

1. There are in general many local maxima.

2. Since all the terms are positive and f_i appear only in the denominator, the maxima are achieved when all the f_i are minimized. By assumption, the range noise covariance depends only on the distance from the sensor to the target. Thus, all the sensors would end up at particular distances from the target irrespective of the angles θ_i.

3. For the particular case of only two sensors, the angle θ_i would either be $\pi/2$ or $3\pi/2$ irrespective of the minimum values of f_1 and f_2. This agrees with our intuition of having the two sensors pointing in orthogonal directions.

4. For more than 2 sensors, if we assume that the minimum value of f_i are all the same, the different local maxima are found by solving for the angles θ_i, $i = 2, \cdots, N$ that maximize the expression

$$\sum_{i=2}^{N}\sin^2(\theta_i) + \sum_{2\le i<j\le N}\sin^2(\theta_i-\theta_j).$$

   Further, in such a case all local maxima have the same value.

Thus, at least in this simplified case, we need not worry about whether we are reaching the local minima or the global minima of the original cost function. Simulation examples seem to suggest that even in more general cases, the globally optimal performance is usually obtained.

Case II : Multiple Targets

The algorithm that we have proposed above can be readily extended to the case where there are multiple targets to be tracked. This situation arises frequently in surveillance, computer vision, signal processing etc. where a number of targets appear and disappear from the field. The sensors need to cooperatively obtain the best possible estimate of all these targets. The cost function to be minimized in this case can be a weighted sum of the determinants of error covariance matrices corresponding to the different targets. We assume that the sensors can identify the target from which the measurements originated. This allows us to bypass the problem of data association [33] and permits us to concentrate on the distributed active sensing based motion planning aspects.

If the sensors can take measurements related to all the targets at the same time, this case is exactly the same as the case considered above. We simply redefine the new target state to be a stacked vector of all the individual target states. The problem is much more interesting (and realistic) if the sensor can observe only one target at a time, with access to a low resolution observation (e.g. from overhead UAVs [193]) to indicate the presence of targets in the region. In this case, it is clear that assignment of sensors to targets that they will observe as sub-teams at each time step also affects the cost.

To solve this problem, we note that the optimal solution now involves optimization of the cost function over not only the sensor positions but also the assignments of sensors to targets. However, the only effect of adding new assignment variables to the optimization problem is to make the set of possible control actions larger. In other words, we can extend our algorithm in a simple way. In the optimization step, every sensor assumes that the positions as well as the target assignments of other nodes remain the same as the previous time step and optimizes its own position and target assignment to minimize the cost function. The algorithm that each sensor follows can be represented as below. At every time step k, every sensor does the following steps.

1. Local Observation:
   (a) Take local measurement.
   (b) Update local estimate $\hat{x}_{\text{local}}$;
2. Sensor Fusion:
   (a) Transmit local estimate $\hat{x}_{\text{local}}$ and error covariance matrix P_{local} and the sensor noise covariance matrix R_k^j to other sensors.
   (b) Receive information from other sensors.
   (c) Fuse all local estimates to get global estimate $\hat{x}_{\text{global}}$.
3. Optimization of target assignment and sensor position:

(a) Assume other sensors do not move.

(b) Assume other sensors do not change the target that they observe.

(c) Propagate P_{local} of other sensors by one time step.

(d) For all possible own target assignments and for all possible own allowable motion actions

i. Propagate own P_{local}.

ii. Fuse with propagated P_{local} of other sensors.

iii. Obtain cost function estimate.

(e) Identify cost-minimizing position and target assignment.

4. Update position and assignment:

(a) Update target assignment for next time step.

(b) Update position.

Examples

Let us illustrate the simple algorithm we have proposed with the help of some examples.

Single Target Tracking Problem

The first example is a comparison of a fully centralized algorithm in which the optimization is done by a central node with the decentralized method proposed in this paper. We seek to show that the performance loss from eliminating a central processing station is not substantial and well-worth the huge computational savings. Further, we contrast the proposed method against the algorithm proposed by Mukai and Ishikawa in [146] (referred to from now on as the MI algorithm). Work done in [146] implements the inverse covariance form of the Kalman filter to centrally fuse all of the measurement information from the individual sensor nodes, and uses the same cost function as this work. However, it requires communication between the sensors in the optimization step as well.

Our test case is constructed as follows. A single target must be localized and tracked by three sensing agents, where the target is subject to random walk in a plane, described by equation (2.31) where the covariance matrix of the noise $w(k)$ is given by

$$Q = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}$$

The j-th sensor observes the target through an equation of the form (2.32) with the j-th sensor's

measurement noise covariance matrix $R_j(k)$ being

$$R_j(k) = \begin{bmatrix} f & 0 \\ 0 & \gamma f \end{bmatrix},$$

where $\gamma = 5$ and the dependence on the range r of the quadratic function f for this example is $f(r) = 0.0008r^2 - 0.0250r + 0.3481$.

Comparing the MI algorithm with our decentralized algorithm, we see in Figure 2.9 that, in fact, the latter method *reduces* the cost by approximately 34% more than the former at steady state. This is understandable since the MI algorithm does not take into account the presence of other sensors at all while trying to optimize the position of higher-ranked sensors. Thus, the three sensors deviate from the optimal position of being stationed such that any two sensors subtend an angle of 120° at the target. The second sensor optimizes its position as if there are only two sensors and tries to move such that its viewing direction is orthogonal to the direction of the first sensor. The third sensor then tries to position itself to minimize the cost function given that the first two sensors have already positioned themselves. In contrast, our algorithm leads to every sensor taking into account the presence of other sensors while optimizing its position and thus leads to nearly optimal sensor positions.

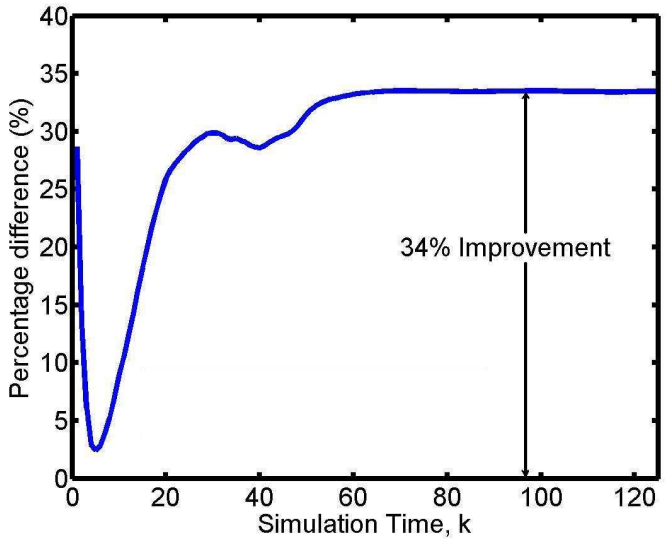


Figure 2.9: Improvement in performance with our algorithm compared to the one proposed in [146].

Moreover, even though our decentralized algorithm is sub-optimal, we see (refer to Figure 2.10) that in comparison to the completely centralized exhaustive search method over all possible control actions for all sensors, the performance loss is very little. In fact, at steady state, the loss is less than 2%. Thus, we see that the proposed decentralized algorithm offers not only the benefit of distributed-

ness, but also that of performance in the context of cost minimization as well as computation time.

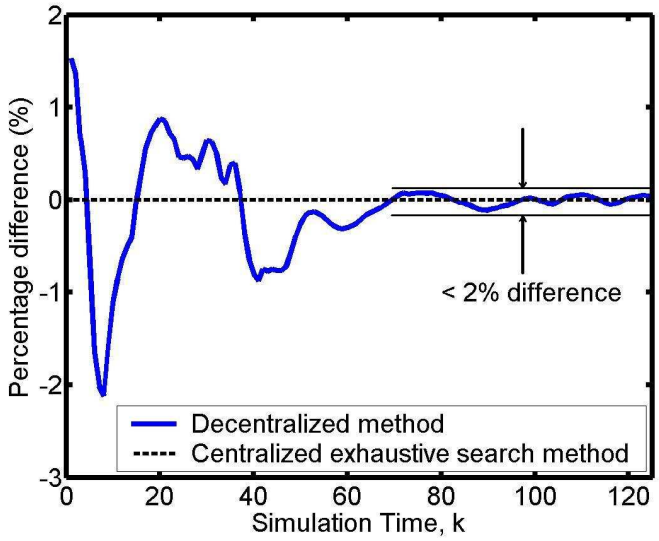


Figure 2.10: Performance loss if the proposed decentralized method instead of the exhaustive search optimization.

RoboFlag Patrolling Problem

The second example illustrates the use of our algorithms in a distributed surveillance type application and demonstrates the assignment methods of matching sensors with targets. We are inspired by the patrolling drill in RoboFlag. (For a detailed description of RoboFlag, see [43]).

The initial setup is illustrated in Figure 2.11(a). We assume sensors can observe only one target at a time. Multiple such sensors (depicted by circles) are initially assigned to a target (depicted by square), corresponding to vehicles patrolling a defense zone. Thus, initially, the sensors maneuver to maintain optimal observations of the target. Another target (e.g. an opponent vehicle) enters the playing field (Figure 2.11(b)), and is observed by a UAV or an arbiter [43]. A sub-team of sensors is automatically formed to track this second target while the remaining sensors maneuver themselves to optimally cover the first target. When the second target disappears, all the sensors return to the first target, as shown in Figure 2.11(c). Thus, sub-teams rejoin and all sensors re-position themselves for best measurements.

Thus, we observe several interesting behaviors exhibited by the system, some surprisingly complex given the simplicity of the algorithm. Firstly, we find that the division of the sensing task over multiple targets is a consequence of the distributed nature of our algorithm, rather than any prescribed method or heuristic approach. The sensors are able to optimally split into sub-teams without needing to explicitly address the issues of consensus, communication (except during the

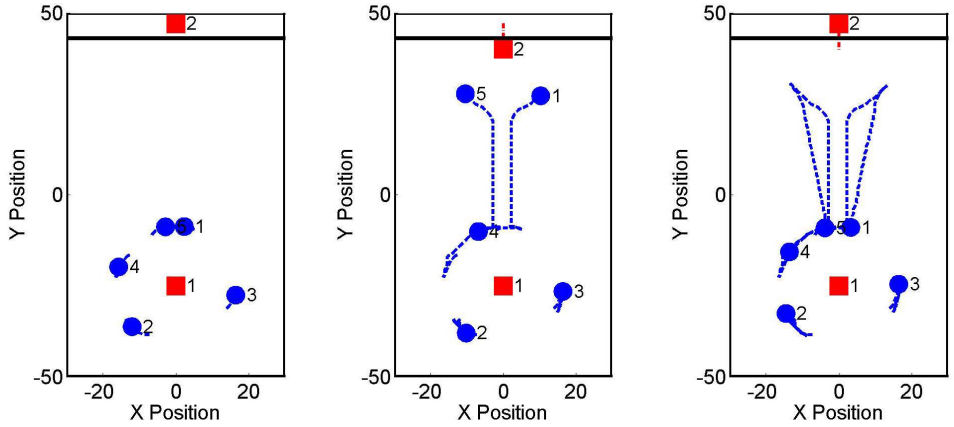


Figure 2.11: Sensor maneuvers in the roboflag patrolling problem considered in the text. The dotted lines represent the tracks made by the sensors.

data exchange step), and coordination of motion for formation control.

Additionally, each member of a sensor sub-team maneuvers optimally with respect to other members, modifying the formation dynamically with the addition/removal of sensing agents. For example, as the sub-team returns to re-join the original team of sensors, we witness the adjustment of the other sensors' positions to accommodate the reentering sub-team members. Again, what makes this behavior interesting is that it is simply due to the decentralized optimization of the cost function done in our algorithm. In other words, *a priori* designation of formations types, division of sensing tasks, and optimal estimation trajectories is not an input to the decentralized algorithm. Nevertheless, the behaviors and benefits of such a designation emerge.

Chapter 3

Countering Communication Channel Effects in Estimation and Control

We now proceed to model and study the second main source of complication in the design of networked control systems: presence of imperfect communication channels. Since the effects introduced by communication channels are not well-understood even for systems that require estimation and control of only one dynamic process (single agent systems), we will consider such systems in this chapter. We will take an information transmission oriented view (i.e., ask the question '*What should an agent communicate?*') that will allow us to concentrate on strategies that lead to optimal performance in spite of the presence of imperfect communication channels. The work presented in this chapter has partly appeared in [41, 78, 79, 80, 84, 88, 89].

This chapter is organized as follows. After a summary of the relevant literature, we begin in Section 3.2 by briefly discussing the packet erasure channel model that we use in the dissertation. In Section 3.3.2, we prove a separation principle between the control design and the optimal encoder / decoder. We then provide the optimal encoder / decoder design for the case of a single sensor transmitting information over a single channel in Section 3.4 and analyze its properties. We move on to the case of a single sensor transmitting data over an arbitrary network in Section 3.5. In Section 3.6, we consider the case of multiple sensors, only one of which transmits data over a communication link. We provide the optimal information transmission algorithm and show how the same algorithm can be used in some other situations. In the first appendix, we consider the LQ performance of a system with a digital memoryless channel inside the loop. In Appendix B, we present results about a jump linear Markov system in which the controller does not know the Markov system.

Contributions

The chief contributions of the work presented in this chapter are now summarized.

1. We pose and solve the problem of LQG control across communication links as an information transmission problem. We show that this viewpoint allows us to achieve the optimal LQG performance when the channel is dropping packets stochastically. We also prove a separation principle in the packet loss setting that allows us to solve the LQG problem using a state feedback controller in conjunction with an encoder and a decoder.

2. We provide optimal yet simple and recursive (Markovian) designs for the optimal encoder and decoder for many cases: a single sensor transmitting over a single link, a single sensor transmitting over an arbitrary network, multiple sensors transmitting over a single link and so on. Apart from yielding optimal performance for arbitrary sequences of packet drops, the algorithms have many other useful properties as well, such as the ability to handle delays, packet reordering and so on.

3. We analyze the stability and performance of the optimal algorithm for all these cases. As an instance, for a single sensor transmitting over a network, we show that the crucial property of the network that is important for stability is the max-cut probability. The performance analysis provides a lower bound for the performance of any causal estimation and control algorithm.

4. We analyze the LQ performance of a scalar system in the presence of a digital memoryless channel for many different types of quantizers. We also provide entropy based general lower bounds for the performance achievable by any quantizer.

5. We provide the analysis and synthesis results for a jump linear Markov system in which the controller is estimating the Markov state of the system as well.

3.1 Introduction

Traditional control theory usually assumes that the various components of the system (sensors, controllers and actuators) can exchange signals reliably and with an arbitrarily high precision. Historically this assumption has made sense since data was typically transmitted over small geographical distances using dedicated communication links. Thus, effects introduced by the communication media were minimal and could be ignored. In networked multi-agent systems, components are typically situated far away from each other and may communicate over wireless links or communication networks that are also used for transmitting other unrelated data. Thus, the effect of such imperfect communication links needs to be considered and accounted for. Communication links introduce

many phenomena that are potentially detrimental to the estimation / control performance, such as quantization error, random delays, data corruption and packet drops to name a few. In extreme cases, poor network performance can even destabilize a nominally stable control loop. Since such effects are not well-understood even for systems that comprise of a single dynamical process that needs to be estimated or controlled, we begin our investigations by focusing on such systems in this chapter.

We begin by considering a dynamic process that is being observed by a single sensor. The sensor communicates to a controller over a communication link. Recently, much attention has been directed towards such systems (see, e.g., [3, 119] and the references therein). As an example, quantization effects have been analyzed with increasing regularity since the seminal paper of Delchamps [46]. The problem of stabilization with finite communication bandwidth was considered by Wong and Brockett [204, 205]. Baillieul [6] also reported a tight bound on the data rate requirement for stabilizing a scalar system. Nair et al. [149, 150] considered the stabilization of stochastic linear systems and Markov jump linear systems with finite data rates. Tatikonda [185] studied stabilization of finite-dimensional discrete-time noiseless linear processes and also presented results about the optimal LQG control of linear systems across noisy feedback links (see also [23]). Elia and Mitter [54] considered the question of the optimal quantizer for stabilization. Various quantization and coding schemes for stabilization have been proposed in the literature, (see, e.g., [25, 57, 98, 104, 129, 160]). Similarly, the effects of delayed packet delivery have also been considered in many works, such as Nilsson [152], Blair and Sworder [19], Luck and Ray [135], Gupta et al. [87], Tsai and Ray [188], and Zhang et al. [208] to name a few, using various models for the network delay.

In this dissertation, we will mostly be interested in systems communicating over links that can be modeled as dropping packets randomly. The nominal system is shown in Figure 3.1, where the channels randomly drop packets being communicated from the plant to the controller and back. Preliminary work in this area has largely focused on the case of a single sensor transmitting information over a single channel and studied the stability of systems utilizing lossy packet-based communication, as in [94, 174, 208]. Performance of such systems as a function of packet loss rate was analyzed by Seiler in [174] and by Ling and Lemmon in [131] assuming certain statistical dropout models. Various approaches to compensate for the lost data have also been proposed. Nilsson [152] proposed two approaches for compensation for data loss in the link by the controller, namely keeping the old control or generating a new control by estimating the lost data, and presented an analysis of the stability and performance of these approaches. Hadjicostis and Touri [90] analyzed the performance when lost data is replaced by zeros. Ling and Lemmon [131, 133] proposed compensators for specific statistical data loss models in the case of single input single output (SISO) systems. In particular, in [131] they posed the problem of optimal compensator design for the case when data loss is independent and identically distributed (i.i.d.) as a nonlinear optimization. Azimi-Sadjadi [5] took an

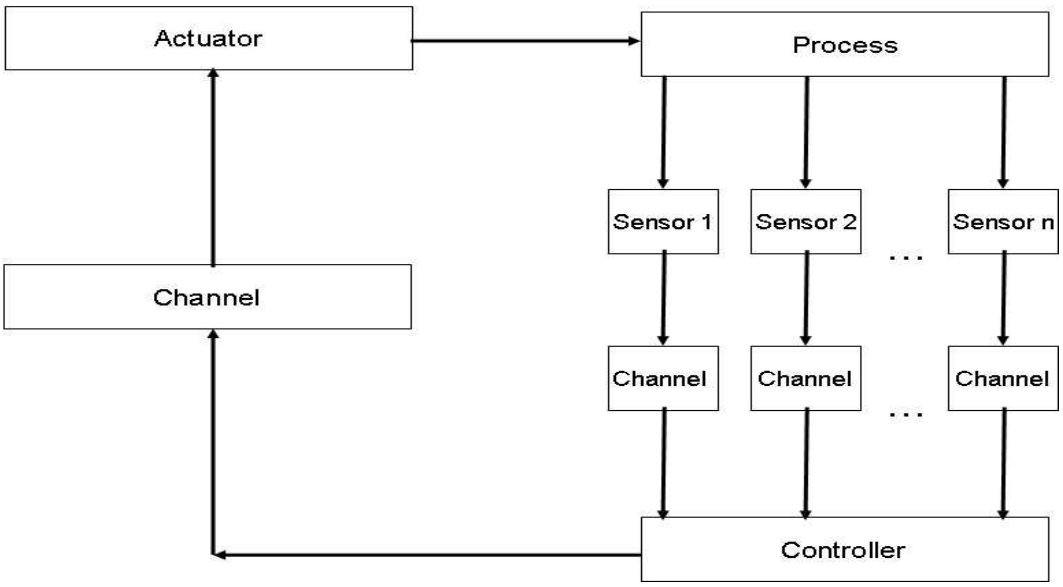


Figure 3.1: The architecture of a packet-based control loop. The links are unreliable and unpredictably drops packets. Most of the works in the literature look at the case of a single sensor transmitting information over a single channel.

alternative approach and proposed a sub-optimal estimator and regulator to minimize a quadratic cost. Sinopoli et al. [173] and Imer et al. [103] extended this approach further to obtain optimal controllers when the packet drops were i.i.d. The related problem of optimal estimation across a packet-dropping link was considered by Sinopoli et. al in [178] and extended by Gupta et al. in [77].

Most of the designs proposed in these references aim at designing a packet-loss compensator, as shown in Figure 3.2. The compensator accepts those packets that the link successfully transmits and comes up with an estimate for the time steps when data is lost. This estimate is then used by the controller. Our work takes a more general approach by seeking the LQG optimal control for this packet-based problem. In particular, our architecture is as shown in Figure 3.3. We will jointly design the controller, the encoder and the decoder to solve the optimal LQG problem. Even though the terminology reminds one of information theoretic designs, the encoder and the decoder can not be designed in our problem using information theoretic algorithms since the system has real-time constraints. The controller needs to generate a control input at every time step and thus, e.g., block coding based coding strategies cannot be used. We need to identify the optimal coding strategy for the purpose of estimation and control.

Based on a separation principle that we prove, the control problem is separated into one of designing a state-feedback optimal controller and another of transmitting information across unreliable links. This allows us to identify the information that needs to be made available to the controller for optimal performance. We then propose a simple recursive algorithm that ensures that this information is available to the controller for the case of a single sensor transmitting information over

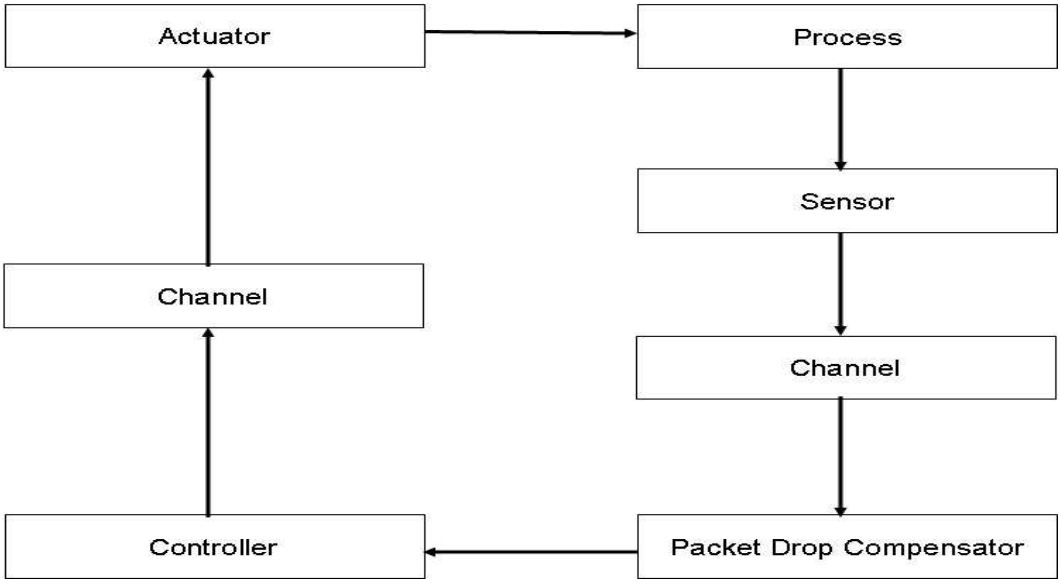


Figure 3.2: A common design for control over packet-based links. The compensator aims at mitigating the effects of packet losses. In most works, the controller-actuator channel is assumed to be absent.

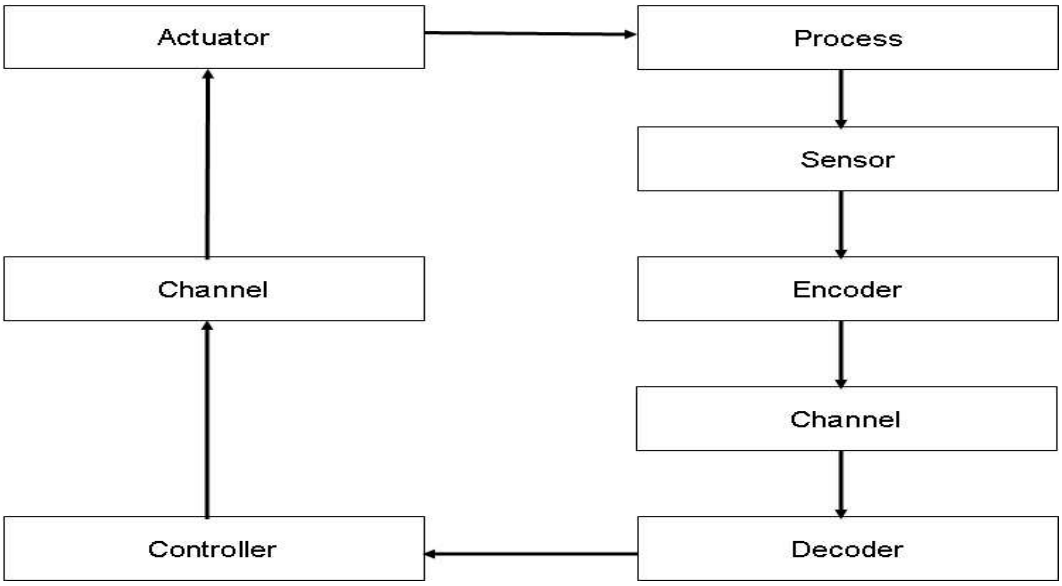


Figure 3.3: The structure of our optimal LQG control solution.

a single link. Even though the algorithm requires a constant amount of memory, transmission and processing at the sensor node, it is optimal for *any* packet drop pattern and has many additional desirable properties that we will illustrate later.

We then extend the optimal coding algorithm to the case of a single sensor transmitting information to the controller over a network of communication links that drop packets stochastically. Transmission of data over networks for the purpose of estimation and control is largely an open problem. In [186], Tatikonda studied some issues related to the quantization rates required for stability when data was being transmitted over a network of digital memoryless channels. Also relevant is the work of Robinson and Kumar [166], who consider the problem of optimal placement of the controller when the sensor and the actuator are connected via a series of communication links. They ignore the issue of delays over paths of different lengths (consisting of different number of links) and under a *Long Packet Assumption* come up with the optimal controller structure. There are two main reasons why the problem of encoding data for transmission is much more complicated in the case of transmission over a network:

1. If the intermediate nodes are allowed to process data, the network cannot be replaced by an erasure channel with the equivalent drop probability as the reliability of the network. Processing by intermediate nodes leads to an element of *memory*.

2. There are potentially multiple paths from the source to any node. These paths may offer data with varying amounts of delay and the processing algorithm needs to take care of this fact.

We again solve for the optimal encoder and decoder structures that are recursive in nature and hence require only a constant amount of memory, transmission and processing at every node. The analysis of the algorithm identifies a property of the network called the max-cut probability that is relevant for the purpose of stability of the control loop (or equivalently, that of the estimate error). We also provide a framework to analyze the performance of our algorithm. Our viewpoint allows us to view the intermediate nodes as repeaters in a digital communication channel that fight the degradation introduced by the channel.

Having solved the problem for the case of a single sensor, we move on to the case when multiple sensors are present. We start with the simplest case when only one of the sensors transmits data over a link that drops packets. This problem is also largely open. We encounter this case in our work on the multi-vehicle wireless testbed [40, 198]. In the testbed, each vehicle is equipped with an on-board gyro. In addition, each vehicle also obtains measurements from an overhead camera. While the gyro-controller link is hard-wired and hence does not drop packets, the camera communicates to the controller over a wireless link that drops packets randomly. Our solution to this problem again adopts the philosophy of processing information at the sensor end before transmission to combat the effects of the channels. Our architecture is as shown in Figure 3.4. We again provide recursive

yet optimal designs of the encoders, the decoder and the controller.

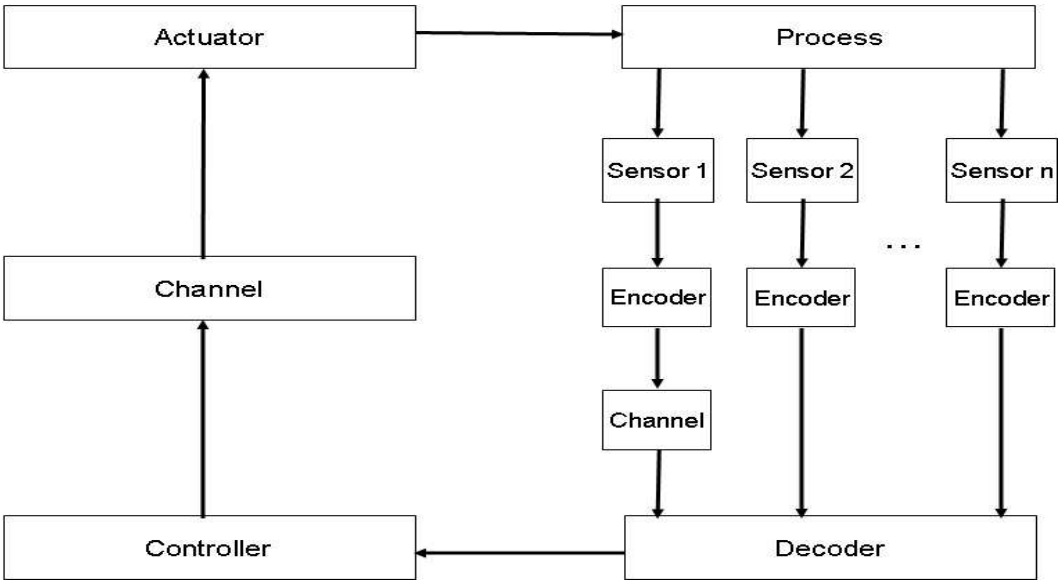


Figure 3.4: The structure of our optimal LQG control solution for the multiple-sensor case.

As an intermediate step, we also solve the following problem. Suppose, as shown in Figure 3.5, two sensors are estimating a process jointly while communicating over links that drop packets stochastically. What information should the sensors exchange so to obtain a good estimate? Related work to this problem has dealt with fusion of data from multiple sensors and track-to-track fusion. The classical Kalman filter is a centralized filter that assumes all observations coming to a central computing facility. The usual starting point is to come up with techniques to decentralize the filter computations. An early contribution was [200], where information obtained from the local sensors is combined to generate the global estimate. However, it required that data about the global estimate be sent from the fusion node to the local sensors. A similar requirement was imposed in the 'successive orthogonalization of measurement subspaces' algorithm proposed in [93]. This difficulty was first overcome in [30, 183], in which each local node sends its own local estimate based on its own data and communicates this estimate and the error covariance data to the fusion center. Similar results for continuous time systems were presented by Willsky et al. in [201]. These results were further extended by Hashemipour et al. in [92], where both the measurement and time update steps of the Kalman filter were decentralized. An alternative approach for data fusion from many nodes using the Federated filter was proposed by Carlson in [27]. A Bayesian method was used and some algorithms presented in [32, 101] which are optimal when there is no process noise. A scattering framework [128] and algorithms based on decomposition of the information form of the Kalman filter [15, 163] have also been proposed for data fusion. A scheme based on the concept of dynamic consensus was proposed in [182], but it assumes multiple communication rounds per time step of

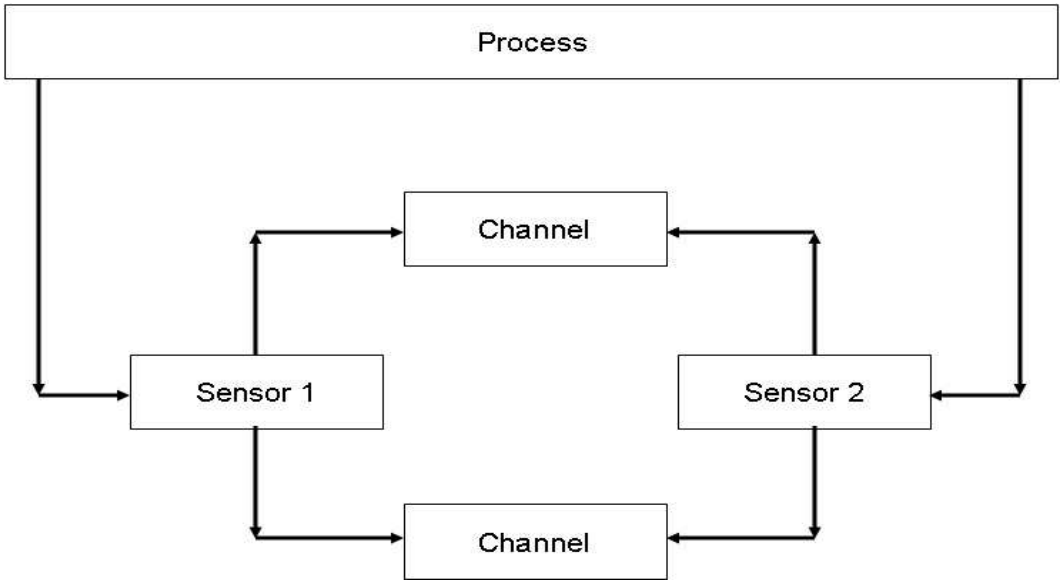


Figure 3.5: Structure of the joint estimation problem.

the system evolution. For some other approaches proposed in the literature (e.g. those based on tracklets [49]), see [31, 145].

However, these approaches assume a fixed communication topology among the nodes with a link, if present, being perfect. In our case, packets of information from one node to another will be dropped randomly by the communication channel present between them. This random loss of information reintroduces the problem of correlation between the estimation errors of various nodes [9] and renders the approaches proposed in the literature as sub-optimal. An approach to solve this problem was proposed by Bar-Shalom and Campo in [10] in the context of track-to-track fusion through exchange of state estimates based on each sensor's own local measurements, but the specific scheme that was used was not proven to be optimal. Moreover, as was found in [167], the algorithm for fusing the local state estimates that was proposed is not optimal in the mean square sense. It was subsequently proven in [28, 145] that the technique was based on an assumption that was not met in general and, in fact, calculation of global estimate using just the local estimates is possible only in very specific cases. We wish to address this problem of finding the optimal global estimate for each node in the case when there are communication channels present between the nodes and packets of information are being randomly dropped. Once again, we disallow approaches such as transmitting all the measurements taken by each node every time communication is possible because that can potentially entail transmitting arbitrarily large amounts of data. Instead, we will propose a recursive yet optimal strategy.

Our algorithm can also be extended to the case when there are multiple sensors that share a channel. Thus, at any time step, either all the sensors drop packets or all the transmissions are

successful. For the case of multiple sensors transmitting over multiple channels that can drop packets independently of each other, the problem is still open. In Chapter 5, we will discuss some partial results for the optimal algorithm and analyze some strategies using tools that we develop in the next chapter.

In the first appendix, we look at another effect that communication channels may introduce: quantization. As we saw above, many works have looked at the presence of digital memoryless channels in control loops. However, most of the work reported so far has focused on the effect of quantization on stability. It is worthwhile to also consider the question of performance of the system in the presence of quantization. This problem is much less well-studied. The performance of a scalar statically quantized system with delays was considered in [191]. Lemmon and Ling [126] presented an upper bound for the quantization noise for the case when dynamic uniform quantization is done over a channel that drops packets. They defined the performance in terms of signal to quantization ratio and presented some interesting trade-offs between the number of bits, locations of the system poles and the performance.

We study the effect of quantization on the LQR performance of the system. We consider a linear time-invariant scalar system with a control law in place and see how the performance degrades as less and less data is allowed to pass from the process to the controller. We come up with some interesting bounds for specific quantizers and some entropy-based general bounds on general centroid-based quantization and encoding schemes. We also consider extensions to dynamic quantization schemes and packet-dropping channels.

In the second appendix, we consider a system in which a channel between the sensor and the controller can exist in one of many states. These states can, e.g., correspond to different random delays applied or different noise powers that corrupt any signal transmitted over the channel. The channel transitions between the states according to a Markov chain. Thus, the system can be modeled as a jump linear Markov system. Such systems have been studied and analyzed extensively. As an example, Ji and Chizeck [108] studied the problem in detail and defined concepts like stability and controllability. Discrete-time versions of the jump-linear quadratic (JLQ) optimal control problem were solved for finite-time horizons in Blair and Sworder [19]. Nilsson and Bernhardsson [153] generalized the results of Ji and Chizeck to the case where the Markov chain determines the probability density function of the variables rather than the values of the variables themselves.

However, all the above approaches assumed the Markov state to be known. In the context of the work presented in this dissertation, this assumption means, e.g., that the receiver knows whether the link has dropped a packet or the delay it has introduced and so on. In the second appendix, we discuss the case when this assumption does not hold and a Markov state estimation algorithm is used to estimate the state of the channel as well. We analyze the case where the state estimate update depends only on the latest observation value. In particular, we consider a suboptimal version of the

causal Viterbi algorithm and show that a separation property need not hold between the control law and the state estimation algorithm.

3.2 The Packet Erasure Channel Model

In this dissertation, we are interested in channels that are discrete-time and packet-based in nature. At every discrete time step, a packet of data is created and transmitted over the channel. The receiver tries to estimate the data that was transmitted using the information it receives over the channel. A general model for a communication channel used in, for instance information theory, is as a probability matrix [39]. If the channel supports m input symbols and n output symbols, then the channel is completely described by an $m \times n$ matrix whose (i, j)-th element denotes the probability that the channel will output the j-th output symbol given that the i-th symbol was input[1]. While this characterization is appropriate for many information-theoretic purposes, for the purpose of analyzing the effect of communication links on estimation and control, this model can introduce too many details that can cloud the picture. An alternative is to summarize the effect of the communication link (and the concomitant physical layer information transmission and reception mechanism) in terms of various metrics or effects that the communication channels introduce into the system. Some of these effects are:

1. Time delay: Before data is transmitted over a channel, it is usually buffered, quantized and coded. All these operations consume some time. After a propagation delay, the data is decoded at the receiver end. If the data is not received properly, the communication protocol may specify a re-transmission of the data. Thus, by the time the information is used by the receiver, a delay has been introduced. Usually this delay is random and the probability distribution of the delay may change over time.

2. Data loss: In most communication protocols, if the receiver does not receive a data packet within a specified time limit, the packet is assumed to be lost. This data loss can happen due to a variety of reasons. For instance, in a shared multiple access medium such as the wireless channel, simultaneous transmission by two transmitters may lead to loss of data from both the transmitters. If transmission occurs over a network of communication channels, overflow of buffers can also lead to packet loss. Finally, if the data is extremely degraded by the time it reaches the receiver, the communication protocol may call for the packet to be dropped.

3. Quantization: Many communication channels and protocols are digital in nature. Thus, any data that needs to be transmitted over a channel needs to be quantized. The number of bits that can be transmitted at every time step is usually upper-bounded.

[1] In general this probability can be time varying.

4. Data corruption: The signal that the receiver regenerates may not be identical to the signal that the transmitter wished to communicate due to noise or attenuation introduced by the channel. While most communication protocols specify error detection and error correction codes, they might not be sufficient to provide immunity to particularly bad channel use instances.

Modeling a channel in terms of these metrics rather than in terms of a physical layer model that specifies the coding, transmission method, channel structure and so on simplifies the analysis and design of an estimation and control loop. It also separates the communication and control design parts of the problem, thus leading to a layered design approach. The control engineer can then demand a certain quality of service from the communication layer in terms of such metrics. In turn, the communication engineer can specify the range of values that the metrics can assume and the control goals are then computed based on these values. This notion of a layered architecture, even though sub-optimal in general, leads to simplicity and tractability in design and is often credited with the success of the Internet (due to the OSI model), serial computation (due to the von-Neumann bridge) and so on [72].

In this dissertation, for the most part, we will concentrate on the stochastic packet dropping effect of the channel. The time line for the operation of such a link is as follows. At every time step k,

- A packet containing some function of the information that the transmitter has access to is created at the transmitter side of the link.
- The packet is sent across the link.
- At time step $k+1$, the packet is either received without error, or dropped, probabilistically.

The information set that the transmitter has access to may have to satisfy some constraints. As an example, there is usually a limit on the memory at the transmitter that limits the amount of data that can be stored. We also impose the constraint that the function communicated over the link should be a finite vector. The packet dropping is a random process. If the packet drops are independent from one time step to the next and occur with the same probability at every time step, they are said to have occurred in an independent and identically distributed (i.i.d.) fashion. In some channels, drops are correlated from one time step to the next. This correlation can be captured by a more sophisticated model such as a Markov chain. In the classic Gilbert-Elliot channel model [56, 67] shown in Figure 3.6, the channel is assumed to exist in two states. The 'bad' state corresponds to the channel dropping packets and the 'good' state corresponds to a successful transmission. The channel switches between the two states according to a Markov chain. This model can capture the effect displayed, e.g., by a wireless channel in which packet drops occur in a bursty fashion. More sophisticated models comprising of multiple Markov states, each corresponding to a different

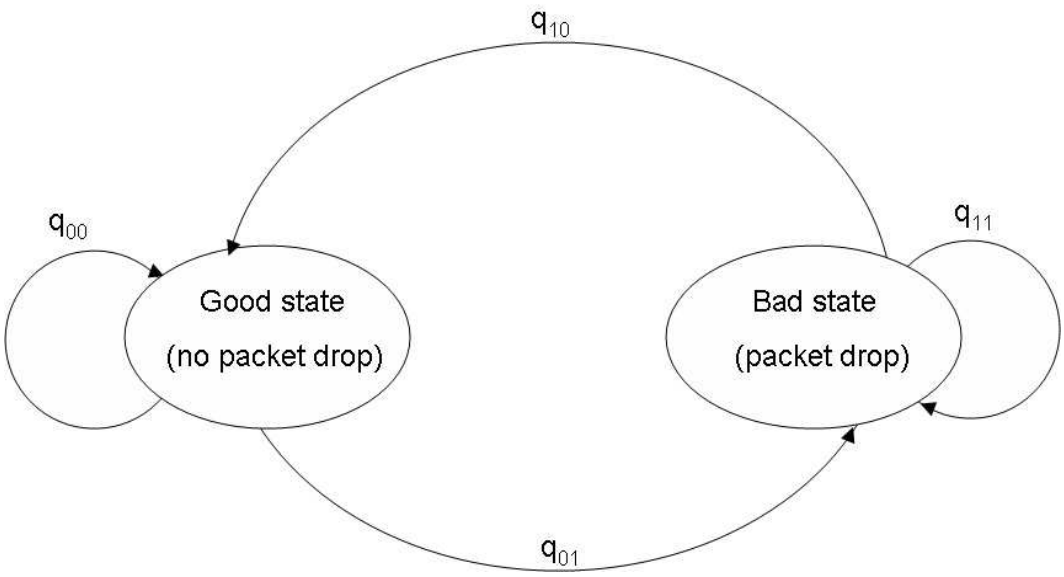


Figure 3.6: The classical Gilbert-Elliot channel model.

probability of packet drop are also available (see, e.g., [196, 207]). In any case, the packet dropping is a random process. We refer to individual (i.e. deterministic) realizations of this random process as *packet drop sequences*. A packet drop sequence is a binary sequence $\{\lambda(k)\}_{k=0}^{\infty}$ in which $\lambda(k)$ takes the value "*received*" if the link delivers the packet at time step k, and "*dropped*" if the packet is dropped.

This model is referred to as the *packet erasure model* of the channel. We assume sufficient bits per data packet and a high enough data rate so that quantization error is negligible. This assumption merely means that a sufficient number of bits are available so that the effect of the quantization error is swamped by the effect of the process and the measurement noises. We do not assume an infinite number of bits, so that strategies based on interleaving of bits to transmit an infinite amount of data are not admissible[2]. We also assume that enough error-correction coding is done within the packets so that the packets are either dropped or received without error. Finally, we will nominally consider the delays, if any, introduced by the channel to be less than one time step according to which the discrete-time dynamical process evolves. We will, however, revisit the issue of delays larger than one time step later in the chapter. We will, sometimes, also make an assumption of a one bit acknowledgement being available to the transmitter corresponding to whether or not the packet was received over the channel at the previous time step. Since the acknowledgement is just one-bit, it can be transmitted with a much higher reliability. We will assume that when an acknowledgement mechanism is present, acknowledgements are not dropped. Note that an implicit assumption in the model is that the receiver knows that a packet has been erased. Thus, it does not come up with a

[2]Moreover, the optimal encoder / decoder algorithms we present will achieve the same performance as if we were indeed transmitting an infinite amount of data.

faulty estimate of the data transmitted over the link.

3.3 Problem Formulation and Preliminary Results

3.3.1 Problem Setup

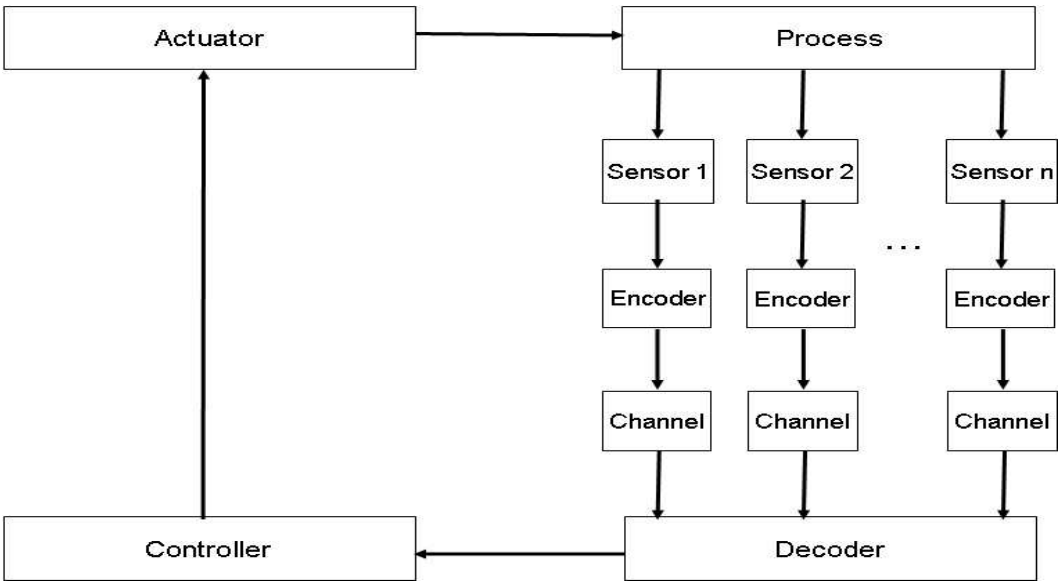


Figure 3.7: The set-up of the control across communication networks problem. Later in the chapter we also look at a channel present between the controller and the actuator.

Consider the arrangement in Figure 3.7. Let the discrete-time linear process evolve according to the equation

$$x(k+1) = Ax(k) + Bu(k) + w(k), \tag{3.1}$$

where $x(k) \in \mathbf{R}^n$ is the process state, $u(k) \in \mathbf{R}^m$ is the control input and $w(k)$ is the process noise assumed to be white, Gaussian, and zero mean with covariance matrix R_w[3]. The initial condition $x(0)$ is assumed to be independent of $w(k)$ and to have mean zero and covariance matrix $R(0)$. The state of the plant is measured by N sensors with the i-th sensor generating measurements according to the equation

$$y_i(k) = C_i x(k) + v_i(k). \tag{3.2}$$

The measurement noises $v_i(k)$'s are assumed white, zero-mean, Gaussian (with covariance matrix $R_{v,i}$) and independent of the plant noise $w(k)$ and of each other. Every sensor communicates its own measurements (or some function of the measurements) to the controller. For the moment, we ignore random delays and packet reordering in the channels and model them solely as packet erasure

[3]The results we present continue to hold for time-varying systems, but we consider the time-invariant case to simplify the presentation.

links with a fixed delay of one time step. For ease of presentation, we will denote the encoder as a node separate from the sensor and term it the source node.

The controller at every time step calculates a control input $u(k)$ and transmits it to the actuator. As shown in Figure 3.7, for the time being we ignore the channel between the controller and the actuator. We will revisit both the issue of delay as well as the presence of a controller-actuator channel later in the chapter and show how simple modifications to our design can take care of them. The controller aims at minimizing the quadratic cost function

$$J_T = E\left[\sum_{k=0}^{T}\left(x^T(k)Qx(k) + u^T(k)Ru(k)\right) + x^T(T+1)P^c_{T+1}x(T+1)\right], \tag{3.3}$$

where Q, R and P^c_{T+1} are all positive definite matrices. We make the usual assumptions of the pairs (A, B) and $(A, Q^{\frac{1}{2}})$ being stabilizable. The expectation in the cost function is taken over the initial condition $x(0)$ and the noise processes $\{w(k)\}$ and $\{v_i(k)\}$'s. Without the channels being present, this is the classical LQG control synthesis problem. The presence of communication channels that erase packets stochastically, however, alters the problem drastically.

The time-line of the operation of the channels is as follows. At every time step k,

1. The source node at each sensor computes a function of all the information it has access to at that time.

2. The source nodes transmit the functions on the communication links. The controller calculates the control $u(k)$ based on the information it possesses.

3. The controller observes the messages, if any, received on the links and updates its information set for the next time step. The source nodes update their information sets with the observations $y_i(k)$'s.

This time-line means that the there are two sources of delay even if the channel is not dropping any packets. First, the source nodes at time step k can only transmit a function of measurements till time step $k-1$. Further, there is a delay of one time step for transmission over the link. Thus, at time step k the controller can, at best, have access to measurements till time step $k-2$. Removal of any of these sources of delay will lead to only minor adjustments in the results given in the rest of the chapter.

The presence of packet erasure links warrants a discussion on the type of controllers that are allowed. The absolutely optimal LQG performance achievable is given by the classical LQR controller/Kalman estimator pair. However, this design does not respect the packetized nature of the communication. Specifically, the controller requires continual access to the Kalman filter output, which, in turn, requires continual access to the measurements from all the sensors. This access might not always be possible because of data loss in the communication links. In order to make

the class of controllers that are allowed more precise, we introduce the following terminology. Denote by $s_i(k)$ the finite vector transmitted from the sensor i to the controller at time step k. By causality, $s_i(k)$ can depend (possibly in a time-varying manner) on $y_i(0)$, $y_i(1)$, $\cdots$, $y_i(k-1)$, i.e., $s_i(k) = f_i(k)\left(y_i(0), y_i(1), \cdots, y_i(k-1)\right)$. Denote by the variable $\lambda_i(k)$ the (random) event whether or not a transmission was successful on the link i at time step k in the realization of the packet loss sequence $\mathcal{P}_i$ that is occurring in the link i. The *information set*, $\mathcal{I}(k)$ available to the controller at time k is the union of N sets $\mathcal{I}_i(k)$'s defined by

$$\mathcal{I}_i(k) = \{s_i(j) | \lambda_i(j) = \text{`}received\text{'}\}.$$

Also, denote by $t_i(k) < k$ the last time-step at which a packet was delivered over link i prior to time k. That is

$$t_i(k) = \max\{j < k \mid \lambda_i(j) = \text{`}received\text{'}\}.$$

The *maximal information set*, $\mathcal{I}^{\max}(k)$ at time-step k is then the union of sets $\mathcal{I}_i^{\max}(k)$ defined by

$$\mathcal{I}_i^{\max}(k) = \{y_i(j) \mid j < t_i(k)\}.$$

The maximal information set is the largest set of output measurements from sensor i on which the control at time-step k can depend. In general, the set of output measurements on which the control depends will be less than this set, since earlier packets, and hence measurements, may have been dropped. As stated earlier, we impose the restriction that the vectors $s_i(k)$ remain finite and thus, e.g., do not increase in size as k increases. We will call the set of $f_i(k)$'s which fulfill this requirement as $\mathcal{F}$. Without loss of generality, we will only consider information-set feedback controllers, i.e., controllers of the form $u(k) = u(\mathcal{I}(k), k)$. We denote the set of control laws allowed by $\mathcal{U}$. We shall assume perfect knowledge of the system parameters A, B, C, R_w and $R_{v,i}$'s at the controller. Moreover, we shall assume that the controller (and the decoder) have access to the previous control signals $u(0)$, $u(1)$, $\cdots$, $u(k-1)$ while calculating the control $u(k)$ at time k.

We can thus pose the packetized LQG problem as:

$$\min_{u \in \mathcal{U}, f_i \in \mathcal{F}} J_T(u, f_i, \mathcal{P}_i) = E\left[\sum_{k=0}^{T} \left(u^T(k) R u(k) + x^T(k) Q x(k)\right) + x^T(T+1) P^c(T+1) x(T+1)\right]. \tag{3.4}$$

Note that the cost functional J_T above depends on the random packet-drop sequences $\mathcal{P}_i$'s. However, we do *not* average across packet-drop processes; *the solution we will present is optimal for arbitrary realizations of the packet dropping processes.* That is, the controller, encoder and decoder we propose will minimize $J_T(u, f_i, \mathcal{P}_i)$ over the set of allowable controllers $\mathcal{U}$ and allowable functions $\mathcal{F}$ for any given packet-drop sequences $\mathcal{P}_i$'s. Because of this, we will occasionally suppress the packet-drop

dependence in the cost functional and merely write $J_T(u, f_i)$ or just J_T.

Our goal, then, is to solve the standard LQG problem with the additional complication of the packet-dropping links. While this may appear a small modification, it is unclear *a priori*, what the structure of the optimal control algorithm should be, and in what way the packetized links should be used through the design of the encoder and the decoder. We begin by presenting a separation principle.

3.3.2 A Separation Principle

Recall that we wish to construct the optimal control input based on the information set $\mathcal{I}^{\max}(k)$, but we have not yet specified how to design the functions $f_i(k)$'s that will allow the controller to compute that. If the links do not drop packets, sending the measurements $y_i(k)$'s in every packet is sufficient. When the links randomly drop packets, a naïve solution would be to send the entire history of the output variables at each time step. This would certainly be an optimal solution, however, as mentioned earlier, this is not allowed since it requires increasing data transmission as time evolves. Surprisingly, in a lot of cases, we can achieve performance equivalent to the naïve solution using a constant amount of transmission and memory. To this end, we first state the following separation principle.

Proposition 3.1 *Consider the packet-based optimal control problem posed in section 3.3.1. Suppose that all the sensors transmit all their previous measurements at every time step, so that the decoder has access to the maximal information set $\mathcal{I}^{\max}(k)$ at every time step k. Then, for an optimizing choice of the control, the control and estimation costs decouple. Specifically, the optimal control input at time k is calculated by using the relation*

$$u(k) = \hat{\bar{u}}(k|\mathcal{I}^{\max}(k), \{u(t)\}_{t=0}^{k-1}),$$

where $\bar{u}(k)$ is the optimal LQ control law while $\hat{\bar{u}}(k|\mathcal{I}^{\max}(k), \{u(t)\}_{t=0}^{k-1})$ denotes the minimum mean squared error (mmse) estimate of the random variable $\bar{u}(k)$ given the information set $\mathcal{I}^{\max}(k)$ and the previous control laws $u(0), \cdots, u(k-1)$.

Proof The proof is along the lines of the standard separation principle (see, e.g., [95, Chapter 9]). We seek to minimize the T-horizon cost function

$$J_T = E\left[\sum_{k=0}^{T} \big(u^T(k)Ru(k) + x^T(k)Qx(k)\big) + x^T(T+1)P^c(T+1)x(T+1)\right]$$

We need to choose $u(0), u(1), \cdots, u(T)$ that minimize J_T. We begin by gathering terms that depend

on the choice of $u(T)$ and $x(T)$ and writing them as

$$\begin{aligned}\Upsilon(T) &= E\left[u^T(T)Ru(T) + x^T(T)Qx(T)\right] + E\left[x^T(T+1)P^c(T+1)x(T+1)\right] \\ &= E\left[\begin{bmatrix} u^T(T) & x^T(T) \end{bmatrix} \Delta \begin{bmatrix} u(T) \\ x(T) \end{bmatrix}\right] + E\left[w^T(T)P^c(T+1)w(T)\right] \\ &= S(T) + O(T),\end{aligned}$$

where

$$\begin{aligned}\Delta &= \begin{bmatrix} R + B^T P^c(T+1)B & B^T P^c(T+1)A \\ A^T P^c(T+1)B & Q + A^T P^c(T+1)A \end{bmatrix} \\ S(T) &= E\left[\begin{bmatrix} u^T(T) & x^T(T) \end{bmatrix} \Delta \begin{bmatrix} u(T) \\ x(T) \end{bmatrix}\right] \\ O(T) &= E\left[w^T(T)P^c(T+1)w(T)\right].\end{aligned}$$

In the above equation, we have used the system dynamics given in (3.1) and the fact that the plant noise $w(k)$ is zero mean. Thus, we can write

$$J_T = E\left[\sum_{k=0}^{T-1} u^T(k)Ru(k) + \sum_{k=0}^{T-1} x^T(k)Qx(k)\right] + S(T) + O(T). \tag{3.5}$$

We aim to choose $u(T)$ to minimize J_T. From (3.5), it is clear that the only term where the choice of $u(T)$ can make a difference is $S(T)$. On completing squares, $S(T)$ can be written as

$$S(T) = E\left[\left(u(T) - \bar{u}(T)\right)^T R_e^c(T)\left(u(T) - \bar{u}(T)\right)\right] + E\left[x^T(T)P^c(T)x(T)\right]$$

where

$$\begin{aligned}R_e^c(T) &= R + B^T P^c(T+1)B \\ P^c(T) &= Q + A^T P^c(T+1)A - A^T P^c(T+1)B\left(RB^T P^c(T+1)B\right)^{-1} B^T P^c(T+1)A\end{aligned}$$

and $\bar{u}(T)$ is the standard optimal LQ control given by

$$\bar{u}(T) = -\left(R_e^c(T)\right)^{-1} B^T P^c(T+1)Ax(T).$$

If the controller had access to the entire state, it could simply use the standard optimal control $\bar{u}(T)$. However, that is not possible now. Instead, the controller needs to calculate $u(T)$ using the information it has access to. In other words, we need to find $u(T) = F(T)x(T)$ that minimizes

$\Upsilon(T)$ where $F(T)$ has to be a function of the information that the controller has access to. The control problem, thus, reduces to an optimal (in the sense of minimum mean squared error (mmse)) estimation problem. We can write the optimal control at time step T as

$$u(T) = \hat{\bar{u}}(T|\mathcal{I}^{\max}(T), \{u(t)\}_{t=0}^{T-1}) = -\left(R_e^c(T)\right)^{-1} B^T P^c(T+1) A\hat{x}(T|\mathcal{I}^{\max}(T), \{u(t)\}_{t=0}^{T-1}), \quad (3.6)$$

where $\hat{x}(T|\mathcal{I}^{\max}(T), \{u(t)\}_{t=0}^{T-1})$ is the mmse estimate of $x(T)$ given $\mathcal{I}^{\max}(T)$ and the previous control inputs $u(0)$, $u(1)$, $\cdots$, $u(T-1)$. Two things may be noted:

1. Since all the random variables are Gaussian, and the cost function to be optimized is quadratic, the optimal estimator $\hat{x}(T|\mathcal{I}^{\max}(T), \{u(t)\}_{t=0}^{T-1})$ is linear.

2. Suppose that instead of all the previous measurements, the sensors transmitted some other quantity so that the controller had access to the information set $\mathcal{I}(T)$. Since the information content in $\mathcal{I}(T)$ is upper bounded by the information contained in $\mathcal{I}^{\max}(T)$, the error covariance in calculating $\hat{x}(T|\mathcal{I}(T), \{u(t)\}_{t=0}^{T-1})$ is lower bounded by the error covariance in calculating $\hat{x}(T|\mathcal{I}^{\max}(T), \{u(t)\}_{t=0}^{T-1})$.

Thus, we only need to find the mmse estimate of $x(T)$, given the information $\mathcal{I}^{\max}(T)$ and the previous control inputs $\{u(t)\}_{t=0}^{T-1}$ available to the controller. Denote the estimation error incurred due to the minimizing choice of $u(T)$ by $\Lambda(T)$. We have

$$S(T) = \Lambda(T) + E\left[x^T(T)P^c(T)x(T)\right].$$

We can thus write the cost function as

$$\begin{aligned} J_T &= E\left[\sum_{k=0}^{T-1} u^T(k)Ru(k) + \sum_{k=0}^{T-1} x^T(k)Qx(k)\right] + S(T) + O(T) \\ &= E\left[\sum_{k=0}^{T-1} u^T(k)Ru(k) + \sum_{k=0}^{T-1} x^T(k)Qx(k)\right] + E\left[x^T(T)P^c(T)x(T)\right] + \Lambda(T) + O(T) \\ &= J_{T-1} + \Lambda(T) + O(T). \end{aligned}$$

Thus, we now need to choose control inputs for time steps 0 to $T-1$ to minimize J_T. By scanning the terms on the right hand side of the equation, we see that $O(T)$ is independent of the choice of control laws from time 0 to $T-1$. Similarly, $\Lambda(T)$ is also independent of all previous control laws since the control inputs are known while calculating $\hat{x}(T|\mathcal{I}^{\max}(T), \{u(t)\}_{t=0}^{T-1})$. Thus, the only term that is dependent on the control inputs till time step $T-1$ is J_{T-1}. But our argument so far was independent of time index T. Thus, we can recursively apply the argument above for time steps $T-1$, $T-2$ and so on. ∎

If we have a channel between the controller and the actuator, the separation principle would still hold, provided there is a provision for acknowledgment from the receiver to the transmitter for any packet successfully received over the channels. This observation has also been made in [103, 173].

There are two reasons this principle is useful to us:

1. The controller design part of the problem is now solved. The optimal controller is the solution to the LQ control problem.

2. We recognize that the optimal controller does not need to have access to the information set $\mathcal{I}^{\max}(k)$ at every time step k. The encoders and the decoder only need to ensure that the controller receives the quantity $\hat{\hat{u}}(k|\mathcal{I}^{\max}(k), \{u(t)\}_{t=0}^{k-1})$, or equivalently, $\hat{x}(k|\mathcal{I}^{\max}(k), \{u(t)\}_{t=0}^{k-1})$.

We now propose algorithms for various cases that require a constant amount of memory and transmission, yet allow the controller to have access to $\hat{x}(k|\mathcal{I}^{\max}(k), \{u(t)\}_{t=0}^{k-1})$ at every time step.

3.4 Single Sensor, Single Channel

We begin with the case depicted in Figure 3.8. A single sensor observes the process state $x(k)$ through measurements of the form

$$y(k) = Cx(k) + v(k),$$

where the noise $v(k)$ has covariance R_v and the pair (A, C) is detectable. The sensor transmits information over a communication link that erases the information stochastically. If a packet is received at time step k, the variable $\lambda(k)$ equals '*received*'; otherwise it equals '*dropped*'. At this moment, we make no assumption about the statistics of the packet dropping process.

For ease of notation, let $\hat{x}(k|l)$ denote the estimate of $x(k)$ based on all the measurements generated by the sensor up to time l and all control inputs till time $k-1$. Denote the corresponding error covariance by $P(k|l)$. Also, denote by $\bar{x}(k|l)$ the estimate of $x(k)$ based on all the measurements up to time l while assuming that no control input was applied as $x(k)$ evolved according to (3.1). $\bar{x}(k|l)$ can be evaluated through a filter that is identical to a Kalman filter except for the application of the control input during the time update step. We will call such a filter a modified Kalman filter. A modified Kalman filter requires the calculation of the quantity $P(k|l)$. Even though it does not stand for the estimate error covariance in the case of the modified Kalman filter, the calculation of $P(k|l)$ is identical for both the Kalman filter and the modified Kalman filter.

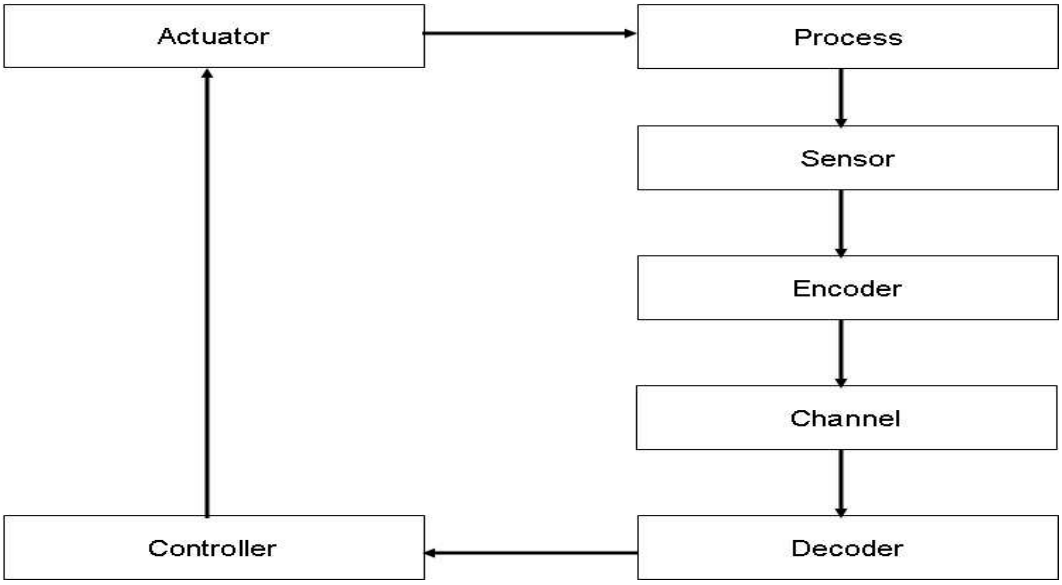


Figure 3.8: Problem setup for Section 3.4: A single sensor transmitting over a single link.

3.4.1 The Kalman Filter

For the sake of completeness, the Kalman filter is given below. For details on the derivation and properties, see, e.g., [112].

Measurement Update for the Kalman filter:

$$\begin{aligned} (P(k|k))^{-1} &= (P(k|k-1))^{-1} + C^T R^{-1} C \\ (P(k|k))^{-1}\,\hat{x}(k|k) &= (P(k|k-1))^{-1}\,\hat{x}(k|k-1) + C^T R^{-1} y(k). \end{aligned} \tag{3.7}$$

Time Update for the Kalman filter:

$$\begin{aligned} P(k|k-1) &= AP(k-1|k-1)A^T + R_w \\ \hat{x}(k|k-1) &= A\hat{x}(k-1|k-1) + Bu(k-1). \end{aligned} \tag{3.8}$$

If the initial state $x(0)$ of the process is Gaussian with zero mean and variance $R(0)$, the initial conditions for the Kalman filter are given by $\hat{x}(0|-1) = 0$ and $P(0|-1) = R(0)$. The equations for the modified Kalman filter are identical except for substituting $u(k) = 0$ in the time update step. Thus, while the measurement update step is still given by

$$\begin{aligned} (P(k|k))^{-1} &= (P(k|k-1))^{-1} + C^T R^{-1} C \\ (P(k|k))^{-1}\,\bar{x}(k|k) &= (P(k|k-1))^{-1}\,\bar{x}(k|k-1) + C^T R^{-1} y(k), \end{aligned} \tag{3.9}$$

the time update equations are given by

$$\begin{aligned} P(k|k-1) &= AP(k-1|k-1)A^T + R_w \\ \bar{x}(k|k-1) &= A\bar{x}(k-1|k-1). \end{aligned} \tag{3.10}$$

Note that calculation of the second order terms $P(k|k)$ and $P(k|k-1)$ does not require knowledge of either the measurements $y(j)$'s or the control inputs $u(j)$'s and these terms can even be calculated offline. Also, we recognize the following relation between $\hat{x}(k|k-1)$ and $\bar{x}(k|k-1)$.

$$\hat{x}(k|k-1) = \bar{x}(k|k-1) + \Psi(k), \tag{3.11}$$

where the term $\Psi(k)$ calculates the impact of the control input and can be calculated using the recursion

$$\Psi(k) = Bu(k-1) + \Gamma(k-1)\Psi(k-1),$$

with

$$\Gamma(k) = A\left(P(k-1|k-1)\right)^{-1} P(k-1|k-2)$$

and the initial condition $\Psi(0) = 0$. This observation allows us to separate the effect of the measurements and the control inputs.

3.4.2 Optimal Information Processing Algorithm

We propose the following design for the encoder and the decoder. At every time step k

- The encoder (at the sensor end) has access to measurements till time step $k-1$ in its information set. It runs the modified Kalman filter according to equations (3.9), (3.10) and transmits the output $\bar{x}(k|k-1)$ of this filter across the link.
- The decoder (at the controller end) maintains two variables: a variable $\psi(k)$ that takes into account the effect of the control inputs, and a local variable $\hat{x}^{dec}(k|k-1)$ (with initial condition $\hat{x}^{dec}(0|-1) = 0$). These variables are updated as follows.
  - The decoder can calculate the terms $P(k|k-1)$ and $P(k-1|k-1)$ as in the Kalman filter equations. It then calculates

    $$\psi(k) = Bu(k-1) + \gamma(k-1)\psi(k-1),$$

    where

    $$\gamma(k) = A\left(P(k-1|k-1)\right)^{-1} P(k-1|k-2)$$

and the initial condition is given by $\psi(0) = 0$.

– If $\lambda(k-1) =$ '*received*', the link successfully transmitted the packet containing the estimate $\bar{x}(k-1|k-2)$ at time step $k-1$. The decoder sets

$$\hat{x}^{dec}(k-1|k-2) = \bar{x}(k-1|k-2) + \psi(k-1).$$

– If $\lambda(k-1) =$ '*dropped*', the packet was dropped. The decoder implements the linear predictor

$$\hat{x}^{dec}(k-1|k-2) = A\hat{x}^{dec}(k-2|k-3) + Bu(k-2). \tag{3.12}$$

– Finally, the decoder outputs the estimate

$$\hat{x}^{dec}(k|k-2) = A\hat{x}^{dec}(k-1|k-2) + Bu(k-1).$$

The algorithm given above is optimal in the following sense.

Proposition 3.2 *In the algorithm described above,* $\hat{x}^{dec}(k|k-2) = \hat{x}(k|\mathcal{I}^{\max}(k), \{u(t)\}_{t=0}^{k-1})$.

Proof The proof is obvious given the relation (3.11). The information needed from the sensor at time step k for the calculation of $\hat{x}(k-1|k-2)$ is precisely $\bar{x}(k-1|k-2)$. The impact of the control input can be taken care of by the term $\Psi(k-1)$ which is the same as the term $\psi(k-1)$ calculated in the algorithm by the decoder. Now, for the case when $\lambda(k-1) =$ '*received*', the decoder in the algorithm has access to $\bar{x}(k-1|k-2)$ and $\Psi(k-1)$. Thus, it can calculate the centralized Kalman filter output $\hat{x}(k-1|k-2)$. Upon executing the time update step, it calculates $\hat{x}(k|k-2)$ which is $\hat{x}(k|\mathcal{I}^{\max}(k), \{u(t)\}_{t=0}^{k-1})$. For the case when $\lambda(k) =$ '*dropped*', the decoder propagates the best Kalman filter estimate $\hat{x}^{dec}(k-1|k-2)$ with the control inputs $u(k-2)$ and $u(k-1)$. Thus, in this case too, $\hat{x}^{dec}(k|k-2) = \hat{x}(\mathcal{I}^{\max}(k), \{u(t)\}_{t=0}^{k-1})$ ∎

Proposition 3.2 also solves the pure estimation problem in which the state of a dynamic process needs to be estimated across a packet erasure link. If the decoder sets the term $\psi(k)$ identically to 0, the optimal encoder and decoder structures are given as above. Moreover, taken together, Propositions 3.1 and 3.2 solve the packet-based LQG control problem posed in Section 3.3.1 for the case of a single sensor and a single link.

Proposition 3.3 *Consider the packet-based optimal control problem described in Section 3.3.1 for the case of only one sensor being present, For any packet dropping process* $\mathcal{P}$*, an LQR state feedback design together with the optimal transmission-estimation algorithm described above achieves the minimum of* $J_T(u, f, \mathcal{P})$.

Remarks

1. The information vector $\bar{x}(k|k-1)$ 'washes away' the effect of any previous packet losses. If $\lambda(k-1) =$ '*received*', $\hat{x}^{dec}(k|k-2)$ is identical to the case when all the previous measurements $\{y(0), y(1), \cdots, y(k-2)\}$ were available to the controller.

2. We have made no assumption about the packet dropping behavior. The algorithm provides the optimal estimate for an arbitrary packet drop sequence, irrespective of whether the packet drop can be modeled as an i.i.d. process (or a more sophisticated model like a Markov chain). We also have not made any assumption about the statistics of the packet drops being known.

3. We do not assume knowledge of the cost matrices Q and R at the sensor end. Thus, the cost function (and hence the optimal controller) can be changed at will without affecting the sensor/encoder operation. This is important, e.g., in our MVWT work where the matrices Q and R are user-specified while the encoder code is much harder to change.

Presence of delays

If we assume that there is a provision for time-stamping the packets sent by the encoder, the solution can readily be extended to the case when the channel applies a random delay to the packet so that packets might arrive at the decoder delayed or even out-of-order. At each time step, the decoder will face one of four possibilities, and will update its estimate as described below:

- It has access to $\bar{x}(k-1|k-2)$. It uses this to calculate the estimate $\hat{x}^{dec}(k|k-2)$ according to the algorithm given above.

- It does not receive anything. It uses the predictor equation (3.12) on $\hat{x}^{dec}(k-1|k-3)$.

- It receives $\bar{x}(m|m-1)$ while at a previous time step, it has already received $\bar{x}(n|n-1)$, where $n > m$. It discards $\bar{x}(m|m-1)$ and uses (3.12) on $\hat{x}^{dec}(k-1|k-3)$.

- It receives $\bar{x}(m|m-1)$ and at no previous time step has it received $\bar{x}(n|n-1)$, where $n > m$. It uses $\bar{x}(m|m-1)$ to calculate $\hat{x}^{dec}(m|m-2)$ and obtains $\hat{x}^{dec}(k|k-2)$ through the repeated application of (3.12).

Channel between the controller and the actuator

If we look at the proof of the separation principle above, the crucial assumption was that the controller knows what control input is applied at the plant. Thus, if we have a channel between the controller and the plant, the separation principle would still hold, provided there is a provision for an acknowledgment from the receiver to the transmitter for any packet successfully received over that channel[4]. Since the decoder can now have access to the control input applied at every time

[4]Note that we do not require acknowledgements for the sensor-controller channel.

step, it is apparent that our algorithm is optimal for this case as well. We can also ask the question of the optimal encoder-decoder design for the controller-actuator channel. The optimal decoding at the actuator end will depend on the information that is assumed to be known to the actuator (e.g. the cost matrices Q and R and the measurements from the sensor). Design of the decoder for various information sets is an interesting open problem.

3.4.3 Analysis of the Proposed Algorithm

In this section, we make some assumptions about the packet dropping random process and provide stronger results on the stability and performance of our algorithm. We model the channel erasures as occurring according to a Markov chain as discussed in Section 3.2, which includes the case of independent packet drops as a special case. Thus, the channel exists in either of two states, state 1 corresponding to a packet drop and state 2 corresponding to no packet drop and it transitions probabilistically between these states according to the transition probability matrix Q which is of the form

$$Q = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix}.$$

The (i, j)-th element of Q represents the probability of the state changing from i at the present time step to j at the next time step. We present the results for the case when the encoder has access to measurements till time step k while transmitting information about $x(k)$. The results when it has access to measurements only till time step $k-1$ are similar but more notation intensive.

3.4.3.1 Stability Analysis

We are interested in stability in the sense of bounded second moment. Thus, consider the infinite horizon cost

$$J_\infty = \lim_{T\to\infty} \frac{1}{T} J_T, \tag{3.13}$$

where the expectation to calculate J_T according to (3.4) is taken over the packet dropping process as well. We call the system stable if J_∞ is bounded; otherwise the system is unstable.

Consider the plant evolving as in (3.1). Consider also a Kalman filter that at time step $k+1$ has access to all the measurements and control inputs till time step k. Its state $\hat{x}(k|k-1)$ evolves as

$$\hat{x}(k+1|k) = A\hat{x}(k|k-1) + Bu(k) + K(k)\left(y(k) - C\hat{x}(k|k-1)\right),$$

where $K(k)$ is the Kalman gain given by

$$K(k) = P(k|k-1)C^T\left(CP(k|k-1)C^T + R_v\right)^{-1}.$$

Finally, the estimator at the decoder evolves according to the relation

$$\hat{x}^{dec}(k+1|k) = \begin{cases} A\hat{x}^{dec}(k|k-1) + Bu(k) & \text{channel in state 1} \\ \hat{x}(k+1|k) & \text{channel in state 2.} \end{cases}$$

Denote

$$\begin{aligned} e(k) &= x(k) - \hat{x}(k|k-1) \\ t(k) &= \hat{x}(k|k-1) - \hat{x}^{dec}(k|k-1). \end{aligned}$$

Since $u(k) = F(k)\hat{x}^{dec}(k|k-1)$, (3.1) can be rewritten as

$$x(k+1) = (A + BF(k))\, x(k) + w(k) - BF(k)\,(t(k) + e(k))\,.$$

If (A, B) is stabilizable, by construction $F(k)$ is the optimum control law and hence it stabilizes the system as long as the disturbances $w(k)$, $t(k)$ and $e(k)$ remain bounded in the second moment sense. We assume the noise $w(k)$ has a bounded covariance. Also, $e(k)$ has a bounded covariance by the assumption of detectability of (A, C). Finally, for $t(k)$, we see that it evolves according to

$$t(k+1) = \begin{cases} At(k) + K(k)v(k) - K(k)Ce(k) & \text{channel in state 1} \\ 0 & \text{channel in state 2.} \end{cases} \tag{3.14}$$

Again, note that $v(k)$ and $e(k)$ have bounded covariances. For $t(k)$ to be of bounded covariance, the Markov jump system of (3.14) needs to be stable. Further note that since our controller and encoder/decoder design is optimal, if the closed loop is unstable with our design, it is not stabilizable by any other design. We can summarize the above discussion and, following [152], write the stability condition as follows.

Proposition 3.4 *Consider the packet-based control problem defined in Section 3.3.1 for the single sensor, single link case in which the packet erasure channel is modeled as a Markov chain with transition probability matrix Q. Let the matrix pair (A, B) be stabilizable and the matrix pair (A, C) be detectable. The system is stabilizable, in the sense that the covariance of the state is bounded, if and only if the matrix*

$$\left(Q^T \otimes I\right) \begin{bmatrix} 0 & 0 \\ 0 & A \otimes A \end{bmatrix} \tag{3.15}$$

has eigenvalues strictly less than unity in magnitude, where I is identity matrix and 0 is the zero matrix of suitable dimensions. Further, if the system is stabilizable, one controller and encoder/decoder design that stabilizes the system is given in Proposition 3.3.

As a simple example, suppose the channel has two states between which it jumps independently. With a probability p at each time step, the channel drops the packet. Also, assume that the plant is scalar with the system matrix given by a. Then, the above condition reduces to the condition $pa^2 < 1$.

For the case of the channel dropping packets independently from one time step to the next, this case was also looked at in [178] if the sensor did not do any encoding and transmitted measurements, while the decoder did the optimal decoding. A condition similar to (3.15) was identified as a *necessary* condition for stability. In the above analysis, through viewing the problem as an information transmission problem we have not only obtained the optimal controller structure, but also proven that the condition is necessary for *any* algorithm to be stable. Further we have provided a recursive strategy that allows the condition to be sufficient as well.

3.4.3.2 Performance Analysis

We now calculate the total quadratic cost J_∞ incurred by the system for the infinite-horizon case as defined in (3.13). We will make the additional assumption that the Markov chain is stationary and regular [55] and that the stationary probability of channel being in state i is given by π_i. For a stable system, in the infinite horizon case, the cost J_∞ reduces to

$$\begin{aligned} J_\infty &= \lim_{T\to\infty} \frac{1}{T} J_T \\ &= \lim_{T\to\infty} E\left[x^T(T)Qx(T) + u^T(T)Ru(T)\right] \\ &= \text{trace}\left(P_x(\infty)Q\right) + \text{trace}\left(P_u(\infty)R\right), \end{aligned} \tag{3.16}$$

where

$$\begin{aligned} P_x(\infty) &= \lim_{T\to\infty} E\left[x(T)x^T(T)\right] \\ P_u(\infty) &= \lim_{T\to\infty} E\left[u(T)u^T(T)\right]. \end{aligned}$$

With the assumptions of stability and detectability, the control law matrix $F(k)$ and the Kalman gain matrix $K(k)$ can be considered as constant matrices F and K respectively. Similar to the discussion in the stability analysis above, we can write the evolution of the system in the following manner. Denote

$$\begin{aligned} z(k) &= \begin{bmatrix} x^T(k) & u^T(k) & e^T(k) & t^T(k) \end{bmatrix}^T \\ n(k) &= \begin{bmatrix} w^T(k) & v^T(k) \end{bmatrix}^T. \end{aligned}$$

Then, we have

$$z(k+1) = \begin{cases} \mathbf{A}_1 z(k) + \mathbf{B}_1 e(k) & \text{channel in state 1 at time } k \\ \mathbf{A}_2 z(k) + \mathbf{B}_2 e(k) & \text{channel in state 2 at time } k, \end{cases}$$

where

$$\mathbf{A}_1 = \begin{bmatrix} A & B & 0 & 0 \\ FA & FB & -FA & -FA \\ 0 & 0 & A-KC & 0 \\ 0 & 0 & -KC & A \end{bmatrix} \qquad \mathbf{A}_2 = \begin{bmatrix} A & B & 0 & 0 \\ FA & FB & -F(A-KC) & 0 \\ 0 & 0 & A-KC & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{B}_1 = \begin{bmatrix} I & 0 \\ 0 & 0 \\ I & -K \\ 0 & -K \end{bmatrix} \qquad \mathbf{B}_2 = \begin{bmatrix} I & 0 \\ 0 & FK \\ I & -K \\ 0 & 0 \end{bmatrix},$$

0 denotes the zero matrix and I the identity matrix of suitable dimensions. Define the stationary covariance

$$P(\infty) = \lim_{k\to\infty} E\left[z(k)z^T(k)\right].$$

Also, denote

$$A_1 = \mathbf{A}_1 \otimes \mathbf{A}_1 \qquad A_2 = \mathbf{A}_2 \otimes \mathbf{A}_2$$
$$G_1 = \mathbf{B}_1 R \mathbf{B}_1^T \qquad G_2 = \mathbf{B}_2 R \mathbf{B}_2^T$$
$$R = E\left[e_k e_k^T\right] \qquad G = \begin{bmatrix} \text{vec}(G_1)^T & \text{vec}(G_2)^T \end{bmatrix}^T.$$

Finally, define the conditional state covariance as

$$\tilde{P}_i = \pi_i \lim_{k\to\infty} E\left[z(k)z^T(k) | \text{ channel in state } i \text{ at time } k\right]$$

so that

$$P^\infty = \tilde{P}_1 + \tilde{P}_2.$$

Then, we can use the results of [152] to obtain the following result.

Proposition 3.5 *Define*

$$\tilde{P} = \begin{bmatrix} vec(\tilde{P}_1)^T & vec(\tilde{P}_2)^T \end{bmatrix}^T.$$

Then, $\tilde{P}$ is the unique solution to the linear equation

$$\tilde{P} = (Q^T \otimes I) \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \tilde{P} + (Q^T \otimes I) \left(\begin{bmatrix} \pi_1 & 0 \\ 0 & \pi_2 \end{bmatrix} \otimes I \right) G,$$

where I is the identity matrix, 0 is the zero matrix and other quantities have been defined above.

Once we calculate $\tilde{P}$, we can readily evaluate the cost in (3.16) by using the relations

$$\begin{aligned} P_x^\infty &= \begin{bmatrix} I & 0 & 0 & 0 \end{bmatrix} P^\infty \begin{bmatrix} I & 0 & 0 & 0 \end{bmatrix}^T \\ P_u^\infty &= F \begin{bmatrix} 0 & I & 0 & 0 \end{bmatrix} P^\infty \begin{bmatrix} I & 0 & 0 & 0 \end{bmatrix}^T F^T. \end{aligned}$$

3.4.4 Examples

In this section, we consider some examples to illustrate the performance of our algorithm. We consider the example system considered by Ling and Lemmon in [132]. The plant transfer function is

$$H(z) = \frac{z^{-1} + 2z^{-2}}{1 + z^{-1} + 2z^{-2}},$$

so that the system evolves as

$$\begin{aligned} x(k+1) &= \begin{bmatrix} 0 & -2 \\ 1 & -1 \end{bmatrix} x(k) + \begin{bmatrix} 2 \\ 1 \end{bmatrix} u(k) + \begin{bmatrix} 2 \\ 1 \end{bmatrix} w(k) \\ y(k) &= \begin{bmatrix} 0 & 1 \end{bmatrix} x(k). \end{aligned}$$

The process noise $w(k)$ is zero mean with unit variance and the packet drop process is i.i.d. The cost considered is the steady state output error $\lim_{T\to\infty} y^2(T)$. [132] assumes unity feedback when packets are delivered and gives an optimal compensator design when packets are being lost.

On analyzing the system with our algorithm, we observe that our algorithm allows the system to be stable up to a packet drop probability of 0.5 while the optimal compensator in [132] is stable only if the packet drop probability is less than 0.25. Also, if we analyze the performance, we obtain the plot given in Figure 3.9. The performance is much better throughout the range of operation for our algorithm. The performance of the two algorithms is not the same even at zero probability of packet drop since the optimal compensator presented in [132] assumes unity feedback. The performance, if we assume unity feedback in our algorithm, is also plotted in the figure. It can be seen that the difference in performance is mainly due to the novel encoding-decoding algorithm proposed.

Note that while the strategy in [132] entails transmitting only one number (the dimension of the measurement $y(k)$), our algorithm transmits two numbers (the dimension of the state $x(k)$). This is unavoidable since the strategy in [132] works only for single input single output plants. While the

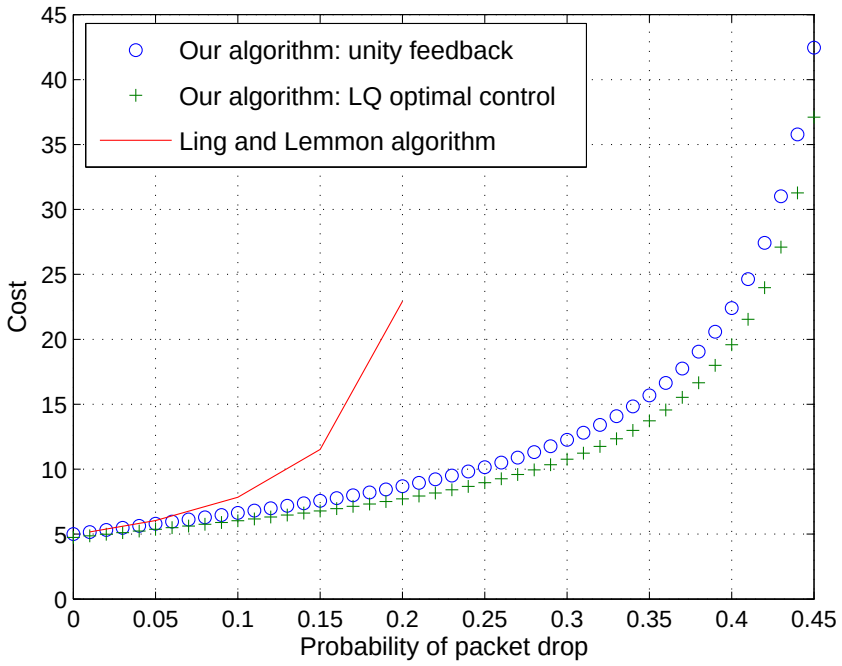


Figure 3.9: Comparison of performance of our algorithm with that obtained if no encoding was done (Ling and Lemmon algorithm).

problem of finding the optimal encoding when there is a limit on the dimension of the data vector that can be transmitted is open[5], it is of interest to compare the strategies when data transmission requirements are the same. Consider a scalar system

$$x(k+1) = 1.2x(k) + w(k),$$

where the noise $w(k)$ has mean zero and variance $R_w = 0.01$. The process is being observed through a sensor of the form

$$y(k) = x(k) + v(k),$$

where the measurement noise $v(k)$ has variance $R_v = 1$. We compare the performance obtained by transmitting measurements and doing the best decoding as suggested by [178] and that obtained by our algorithm. We do not close the loop and are interested only in the estimate error covariance at the receiver end of the channel. Since the analysis of [178] holds only for independent and identically distributed (i.i.d.) packet drops we assume the packet drops in the example to be i.i.d. as well. The results are depicted in Figure 3.10. The transmission frequency refers to the probability that the packet is successfully transmitted. We can see that even in this simple example, optimal encoding

[5]The limit should be less than the dimension of the state vector, otherwise our algorithm solves the problem. Any additional data transmission is redundant.

can lead to performance improvements.

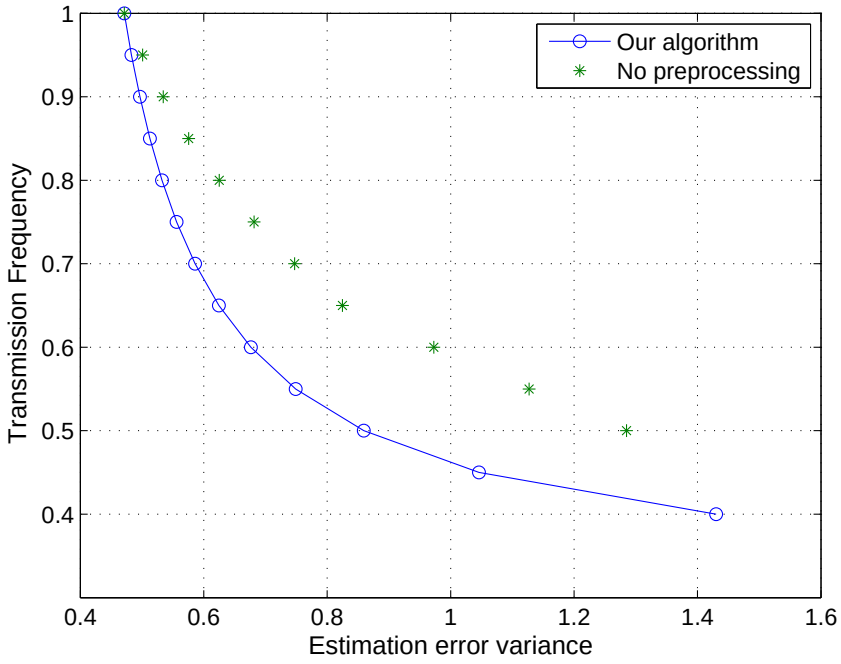


Figure 3.10: Comparison of performance for an estimation task when the measurement matrix is invertible.

3.5 Single Sensor, Network of Arbitrary Topology

We now move on to the problem setup shown in Figure 3.11. Consider once again a process of the form

$$x(k+1) = Ax(k) + Bu(k) + w(k)$$

being observed by a single sensor

$$y(k) = Cx(k) + v(k). \tag{3.17}$$

However, the sensor has to communicate to the controller or the decoder across a network of communication links that stochastically drop packets. The results presented in this section are additionally important since they can also be used for the problems of routing of data and synthesis of networks for the purpose of estimation and control. An analysis identical to that of Section 3.3.2 and Section 3.4.1 shows that a separation principle holds and, further, that the decoder can calculate the impact of the control input on the optimal estimate. Thus, it is sufficient for the algorithm to be

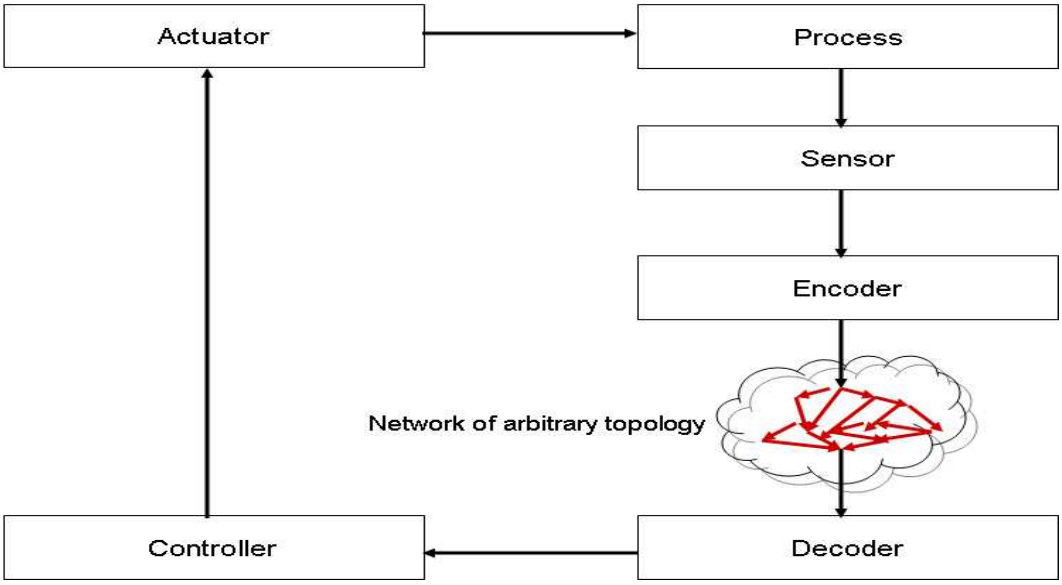


Figure 3.11: Problem setup for Section 3.5: Single sensor transmitting over a network of arbitrary topology. Every node in the network is able to communicate, and usually has similar memory and processing capabilities as the encoder at the sensor that is generating the measurements.

able to estimate the state of a process that evolves as

$$x(k+1) = Ax(k) + w(k). \tag{3.18}$$

Accordingly, from now on, we will restrict our attention to the task of estimating in the minimum mean squared error sense the state $x(k)$ of a process evolving as in (3.18) across a network of communication links.

3.5.1 Mathematical Notation

By an arbitrary network, we mean a network in which communication links are connected in an arbitrary topology. As before, the source node or the encoder is placed at the sensor end and is denoted by s. The decoder at the controller end needs to estimate the process state. It is designated as the sink or the destination node d. The source s and the destination d are connected via a network of communication links. We can model the communication network as a directed graph $\mathcal{G}$ in a natural way. Denote the node set of $\mathcal{G}$ by $\mathcal{V}$. Note that, in particular, $\mathcal{V}$ contains s and d. Also, denote the edge set of $\mathcal{G}$ by $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. The edges of the graph represent the communication links and are, in general, directed. Specifically, the link $e = (u, v)$ models a communication channel between node u and node v. We assume there are M edges or links present in the network. For any node $i \in \mathcal{V}$, the set of outgoing edges corresponds to the links along which the node can transmit messages while the set of incoming edges corresponds to the links along which the node receives

messages. We denote the set of in-neighbors of node v by $\mathcal{N}_v$. As stated above, the networks that we consider are arbitrary, so that we make no a priori assumptions on the topology of the graph depicting the network.

The communication links are once again modeled using a packet erasure model. If the packet dropping process is independent from one time step to the next (or, in other words, memoryless), the probability of dropping a packet on link $e \in \mathcal{E}$ is given by p_e independent of time. If the process is uncorrelated in space, each probability p_e is independent of other links as well.

The operation of the different nodes in the network at every time-step k can be described as follows:

1. Every node computes a function of all the information it has access to at that time.
2. It transmits the function on all the out-going edges. We allow some additional information in the message that tells us the time step j such that the function that the node transmits corresponds to the state $x(j)$. The sink node calculates the estimate of the current state $x(k)$ based on the information it possesses.
3. Every node observes the messages from all the incoming links and updates its information set for the next time step. For the source node, the message it receives at time step k corresponds to the observation $y(k)$.

As in the case with a single link, at time step k, the function that the source node transmits depends on measurements $y(0)$, $y(1)$, $\cdots$, $y(k-1)$. We can easily adapt the discussion presented below to the case when measurements till time step $y(k)$ are available. Furthermore, even if there were no packet drops, if the sink node is d hops away from the source node (i.e., the shortest path from the source node to the sink node involves d edges), its estimate for the state $x(k)$ at time k can only depend on measurements $y(0)$, $y(1)$, $\cdots$, $y(k-d-1)$ till time $k-d-1$. Thus, unlike the model in [166], every communication edge consumes one hop, or in other words, one time step as data is transmitted over it.

3.5.2 Optimal Encoding and Decoding

For the node i, denote by $\mathcal{I}^i(k)$ the information set that it can use to generate the message that it transmits at time step k. This set contains the aggregate of the information the node has received on the incoming edges at time steps $t = 0, 1, \cdots, k-1$. As an example, for the source node s,

$$\mathcal{I}^s(k) = \{y(0), y(1), \cdots, y(k-1)\}.$$

Based on the information set $\mathcal{I}^i(k)$, the i-th node can calculate its minimum mean squared error (mmse) estimate of the state $x(k)$. We denote the estimate calculated by the i-th node at time k as

$\hat{x}^i(k|\mathcal{I}^i(k))$ or more shortly as $\hat{x}^i(k)$, where it is understood that the estimate is based on the information set $\mathcal{I}^i(k)$. We denote the error covariance associated with this estimate by $P^i(k|\mathcal{I}^i(k))$ or more compactly as $P^i(k)$. We aim to minimize the error covariance $P^d(k)$. Clearly, for two information sets $\mathcal{I}^i(k,1)$ and $\mathcal{I}^i(k,2)$ related by $\mathcal{I}^i(k,1) \subseteq \mathcal{I}^i(k,2)$, we have $P^i(k|\mathcal{I}^i(k,1)) \leq P^i(k|\mathcal{I}^i(k,2))$.

The packet drops occur according to a random process. Let $\lambda_{uv}(k)$ be the binary random variable describing the packet drop event on link $(u,v) \in \mathcal{E}$ at time k. $\lambda_{uv}(k)$ assumes the value '*dropped*' if the packet is dropped on link (u,v) at time k and '*received*' otherwise. For a network with independent and memoryless packet drops, $\lambda_{uv}(k)$ has a Bernoulli distribution with parameter p_{uv}. We define $\lambda_{uu}(k) =$ '*received*'. Once again, we refer to instantiations of the packet drop process as packet drop sequences. Given a packet drop sequence, at time step k we can define a time stamp $t^i(k)$ for node i such that the packet drops did not allow any information transmitted by the source after $t^i(k)$ to reach the i-th node in time for it to be a part of $\mathcal{I}^i(k)$.

Now, consider an algorithm $\mathcal{A}_1$ as follows. At time step k, every node takes the following actions:

1. Calculate the estimate of state $x(k)$ based on the information set at the node.
2. Transmit its entire information set on the outgoing edges.
3. Receive any data successfully transmitted along the incoming edges.
4. Update its information set and affix a time stamp corresponding to the time of the latest measurement in it.

When this algorithm is executed for a particular drop sequence, the information set at node i will be of the form

$$\mathcal{I}^i(k) = \{y(0), y(1), \cdots, y\left(t^i(k)\right)\},$$

where $t^i(k) < k$ is the time stamp as defined above. This is the maximal information set $\mathcal{I}^{i,\max}(k)$ that the node i can possibly have access to with any algorithm. For any other algorithm, the information set will be smaller than this since earlier packets, and hence measurements, might have been dropped. The error covariance at any node when it calculates its estimate of the state $x(k)$ is the least when the information set it has access to is $\mathcal{I}^{i,\max}(k)$. The algorithm $\mathcal{A}_1$ requires an increasing amount of memory and transmission as time goes on. We will now describe an algorithm $\mathcal{A}_2$ that achieves the same performance at the expense of constant memory and transmission (modulo the transmission of the time stamp). The algorithm proceeds as follows. At each time step k, every node takes the following actions:

1. Calculate its estimate $\hat{x}^i(k)$ of the state $x(k)$ based on any data received at the previous time step $k-1$ and its previous estimate. The estimate can be computed using a switched linear filter, as shown later.

2. Affix a time stamp corresponding to the latest measurement used in the calculation of the estimate in step 1 and transmit the estimate on the outgoing edges.

3. Receive any data on the incoming edges and store it for the next time step.

To prove that algorithm $\mathcal{A}_2$ is indeed optimal, we need the following intermediate result.

Lemma 3.6 *Consider any edge (i,j) and any packet drop pattern. At time step k, let the node i transmit the measurement set*

$$S^{ij} = \{y(0), y(1), \cdots, y(l)\}$$

on the edge (i,j) if algorithm $\mathcal{A}_1$ is executed. If, instead, algorithm $\mathcal{A}_2$ is executed, the node i transmits the estimate

$$\hat{x}\left(k|S^{ij}\right) = \hat{x}(k|\{y(0), y(1), \cdots, y(l)\})$$

along the edge (i,j) at time step k.

Proof The proof readily follows by induction on the time step k. For time $k = 1$, the source node s transmits $\{y(0)\}$ along all edges of the form $(s,.)$ while following algorithm $\mathcal{A}_1$ and the estimate $\hat{x}(1|y(0))$ while executing algorithm $\mathcal{A}_2$. If any edge is not of the form $(s,.)$, there is no information transmitted along that edge in either algorithm. Thus, the statement is true for $k = 1$. Now, assume that the statement is true for $k = n$. Consider the node i at time $k = n+1$. If the node i is the source node, the statement is true by an argument similar to that at $k = 1$. Let us assume that node i is not the source node. Consider all edges that transmitted data at time step $k = n$ to node i. If algorithm $\mathcal{A}_1$ is being executed, suppose the edge (p,i) transmits the measurement set

$$S^{pi} = \{y(0), y(1), \cdots, y(t(p))\},$$

where $p \in \mathcal{N}_i$ is an in-neighbor of node i. Also, denote the measurement set that the node i has access to from time step $k = n-1$ as

$$S^{ii} = \{y(0), y(1), \cdots, y(t(i))\}.$$

Note that at time step $k = n$, the node i transmitted the set S^{ii} along all outgoing edges. Let v be the node for which

$$t(v) = \max\{t(i) \cup \{t(p)|p \in \mathcal{N}_i\}\}.$$

Then, at time $k = n+1$ under algorithm $\mathcal{A}_1$, the node i transmits along all outgoing edges the measurement set

$$\mathcal{S}^1 = \{y(0), y(1), \cdots, y(t(v))\}.$$

Now, consider the case when algorithm $\mathcal{A}_2$ is being executed. By the assumption of the statement being true at time step $k = n$, the edges (p, i) transmit the estimate

$$\hat{x}(n|S^{pi}) = \hat{x}(n|\{y(0), y(1), \cdots, y(t(p))\}),$$

for all $p \in \mathcal{N}_i$. Also, since at time $k = n$ the node transmitted S^{ii} on any edge $(i, .)$ in algorithm $\mathcal{A}_1$, it has access to the estimate $\hat{x}(n|S^{ii})$ when algorithm $\mathcal{A}_2$ is executed. Clearly, the set S^{vi} is the superset of all sets S^{ii} and S^{ji} where $j \in \mathcal{N}_i$ and v have been defined above. Thus, the estimate that the node i calculates at time $k = n+1$ is $\hat{x}(n+1|S^{vi})$. But the measurement set S^{vi} is simply the set $\mathcal{S}^1$. Hence, at time step $k = n+1$, the node i transmits along all outgoing edges the estimate $\hat{x}(n+1|\mathcal{S}^1)$. Thus, the statement is true at time step $k = n+1$ along all edges of the form $(i, .)$. Since the node i was arbitrary, the statement is true for all edges in the graph. Thus, we have proven that if the statement is true at time $k = n$, it is true at time $k = n+1$. But it is true at time $k = 1$. Thus, by the principle of mathematical induction, it is true at all time steps. ∎

Note that we have also shown that if at time step k, the node has access to the measurement set S^{ii} from time step $k-1$ when algorithm $\mathcal{A}_1$ is executed; it has access to the estimate $\hat{x}(k-1|S^{ii})$ from time step $k-1$ when algorithm $\mathcal{A}_2$ is executed. We can now state the following result.

Proposition 3.7 *The algorithm $\mathcal{A}_2$ is optimal in the sense that it leads to the minimum possible error covariance at any node at any time step.*

Proof Consider a node i. At time k, let $j \in \mathcal{N}_i \cup \{i\}$ such that $\lambda_{ji}(k-1) =$ '*received*'. Denote the measurement set that is transmitted from node j to node i at time step k under algorithm $\mathcal{A}_1$ by S^{ji}. As in the proof of Lemma 3.6, there is a node v, such that S^{vi} is the superset of all the sets S^{ji}. Thus, the estimate of node i at time k under algorithm $\mathcal{A}_1$ is

$$\hat{x}^{\mathcal{A}_1}(k) = \hat{x}(k|S^{vi}).$$

From Lemma 3.6, when algorithm $\mathcal{A}_2$ is executed, at time step k, the node i has access to the estimates $\hat{x}(k-1|S^{ji})$. Once again, since S^{vi} is the superset of all the sets S^{ji}, the estimate of node i at time step k is simply

$$\hat{x}^{\mathcal{A}_2}(k) = A\hat{x}(k-1|S^{vi}) = \hat{x}(k|S^{vi}).$$

Thus, we see that for any node i, the estimates $\hat{x}^{\mathcal{A}_1}(k)$ and $\hat{x}^{\mathcal{A}_2}(k)$ are identical for any time step k for any packet drop pattern. But algorithm $\mathcal{A}_1$ leads to the minimum possible error covariance at each node at each time step. Thus, algorithm $\mathcal{A}_2$ is optimal. ∎

Remarks

1. The step of calculating the estimate at each node in the algorithm $\mathcal{A}_2$ can be implemented using a switched linear filter as follows. The source node implements a Kalman filter and updates its estimate at every time step with the new measurement received. Every other node i checks the time-stamps on the data coming on the incoming edges. The time-stamps correspond to the latest measurement used in the calculation of the estimate being transmitted. Then, node i updates its time-stamp using the relation

$$t^i(k) = \max_{j \in \mathcal{N}_i \cup \{i\}} \lambda_{ji}(k-1) t^j(k-1). \tag{3.19}$$

   Suppose the maximum of (3.19) is achieved by node $v \in \mathcal{N}_i \cup \{i\}$. Then, the node i updates its estimate as

$$\hat{x}^i(k) = A\hat{x}^v(k-1),$$

   where $\hat{x}^v(k)$ denotes the estimate of the state $x(k)$ maintained by the node v.

2. As in the case of a single link, the algorithm is optimal for any packet drop pattern, i.e., irrespective of whether the packet drops are occurring in an i.i.d. fashion or are correlated across time or space or even adversarial in nature. We also do not assume any knowledge of the statistics of the packet drops at any of the nodes. Finally, if the communication links introduce finite delays and packet reordering, the algorithm can be modified along the lines of Section 3.4.2.

3. We have proved that the algorithm is optimal for any node. Thus, we do not need to assume only one sink. The algorithm is also optimal for multiple sources if all sources have access to measurements from the same sensor. For multiple sources with each source obtaining measurements from a different sensor, the problem remains open.

4. A priori we had not made any assumption about a node transmitting the same message along all the out-going edges. It turned out that in this optimal algorithm, the messages are the same along all the edges. Thus, the algorithm is suitable for broadcast channels such as the wireless channel.

5. The communication requirements can be reduced somewhat by adopting an event-based protocol in which a node transmits only if it updated its estimate based on data arriving on an incoming edge at the previous time step. This will not degrade the performance but reduce the number of transmissions, especially if packet drop probabilities are high.

In a sense our algorithm corresponds to communication of information over a digital channel while the strategy of using no encoding is an analog communication scheme. Our algorithm allows the intermediate nodes to play the role of repeaters that help to limit the effect of the channel by

decoding and re-encoding the data along the way. In analog channels, repeaters make no difference in the received Signal-to-Noise ratio and hence the signal quality. Similarly, in our setting, if raw measurements are being transmitted, presence of intermediate nodes does not help in improving the estimation performance.

3.5.3 Stability Analysis

For the rest of the chapter, we will assume that packets are dropped in each link in an i.i.d. fashion with the loss probability in link $e = (u, v)$ being p_e. We also assume packet drop processes in two different links to be independent of each other. We begin by computing the conditions for the estimate error at the sink node to be stable under algorithm $\mathcal{A}_2$ (or equivalently $\mathcal{A}_1$) when the source and the sink are connected with a network of arbitrary topology. Since the algorithm $\mathcal{A}_2$ is optimal, these form necessary conditions for the estimate error covariance to be stable under any other algorithm (in particular, for the case when the nodes do not do any processing and simply transmit measurements).

Once again, we are interested in stability in the bounded second moment sense. Thus, for the destination node trying to estimate a process of the form (3.18), denote the error at time step k as

$$e^d(k) = x(k) - \hat{x}^d(k),$$

where $\hat{x}^d(k)$ is the estimate of the destination node. We can compute the covariance of the error $e(k)$ at time k as

$$P^d(k) = E\left[e^d(k)(e^d(k))^T\right],$$

where the expectation is taken over the initial condition $x(0)$, the process noise $w(j)$, the measurement noise $v(j)$ and the packet dropping processes in the network. We consider the steady-state error covariance in the limit as k goes to infinity, i.e.,

$$P^d(\infty) = \lim_{k\to\infty} P^d(k). \tag{3.20}$$

If $P^d(\infty)$ is bounded, we will say that the estimate error is stable; otherwise it is unstable.

We can isolate the effect of the network by explicitly taking the expectation with respect to the packet dropping process. For node d and time k, $t^d(k)$ denotes the time-stamp of the most recent observation used in estimating $x(k)$ at the destination node d. This time-stamp evolves according

to (3.19). The expected estimation error covariance at time k at node d can thus be written as

$$\begin{aligned} P^d(k) &= E\left[e^d(k)(e^d(k))^T\right] \\ &= E\left[\left(x(k)-\hat{x}^d(k)\right)\left(x(k)-\hat{x}^d(k)\right)^T\right] \\ &= \sum_{l=0}^{k} E\left[\left(x(k)-\hat{x}^d(k|t^d(k)=l)\right)\left(x(k)-\hat{x}^d(k|t^d(k)=l)\right)^T\right] \text{Prob}\left(t^d(k)=l\right), \end{aligned}$$

where, in the final equation, we have explicitly taken the expectation with respect to the packet dropping process and $\hat{x}^d(k|t^d(k)=l)$ denotes the estimate of $x(k)$ at the destination node given all the measurements $\{y(0), y(1), \cdots, y(l)\}$. We see that the effect of the packet dropping process shows up in the distribution of the time-stamp of the most recent observation used in estimating $x(k)$[6]. For future use, we denote the *latency* for the the node d at time k as

$$l^d(k) = k - 1 - t^d(k).$$

Also, denote the mmse estimate of $x(k)$ given all the measurements $\{y(0),\ y(1),\ \cdots,\ y(k-1)\}$ by $M(k)$. Clearly $M(k)$ evolves in time according to a Riccati recursion. We can now rewrite the error covariance $P^d(k)$ as

$$\begin{aligned} P^d(k) &= \sum_{l=0}^{k-1} E\left[\left(x(k)-\hat{x}^d(k|l^d(k)=l)\right)\left(x(k)-\hat{x}^d(k|l^d(k)=l)\right)^T\right] \times \text{Prob}\left(l^d(k)=l\right) \\ &= \sum_{l=0}^{k-1}\left[A^l M(k-l)\left(A^l\right)^T + \sum_{j=0}^{l-1} A^j Q\left(A^j\right)^T\right] \text{Prob}\left(l^d(k)=l\right). \end{aligned} \tag{3.21}$$

The above equation gives the expected estimation error covariance for a general network with any packet dropping process. The effect of the packet dropping process appears in the distribution of the latency $l^d(k)$. As we can see from (3.21), the stability of the system depends on how fast the probability distribution of the latency decreases.

To analyze the stability, we use Proposition 3.4, which is restated here for the case of independent packet drops.

Proposition 3.8 *Consider a process of the form (3.18) being estimated using measurements from a sensor of the form (3.17) over a packet-dropping link that drops packets in an i.i.d. fashion with probability p. Suppose that the sensor calculates the mmse estimate of the measurements at every time step and transmits it over the channel. Then, the estimate error at the receiver is stable in the*

[6]Note that Prob $\left(t^d(k)=k\right)=0$.

bounded second moment sense if and only if

$$p|\rho(A)|^2 < 1,$$

where $\rho(A)$ is the spectral radius of the matrix A appearing in (3.18).

Network with Links in Parallel

We begin by considering a network consisting only of links in parallel. Consider the source and the sink node being connected by a network with m links in parallel with the probability of packet drop in link e being p_e. Since the same data is being transmitted over all the links, the distribution of the latency in (3.21) remains the same if the network is replaced by a single link that drops packets when all the links in the original network drop packets and transmits the information if even one link in the original network allows transmission. Thus, the packet drop probability of this equivalent link is $p_1 p_2 \cdots p_m$. The necessary and sufficient condition for the error covariance to diverge thus becomes

$$p|\rho(A)|^2 < 1,$$

where

$$p = p_1 p_2 \cdots p_m.$$

Necessary Condition for Stability in Arbitrary Networks

Using the result for parallel networks, we can obtain a necessary condition for stability for general networks as follows.

Proposition 3.9 *Consider a process of the form (3.18) being estimated using measurements from a sensor of the form (3.17) through an arbitrary network of packet dropping links with drop probabilities p_e's. Consider every possible division of the nodes of the network into two sets with the source and the sink node being in different sets (also called a cut-set). For any such division, let $p_1, p_2, \cdots, p_n$ denote the packet erasure probabilities of the edges that connect the two sets (equivalently, the edges that are in the cut). Define the cut-set erasure probability as*

$$p_{cut\ set} = p_1 p_2 \cdots p_n.$$

Then, a necessary condition for the error covariance to converge is

$$p_{network}|\rho(A)|^2 < 1,$$

where $p_{network}$ is the network erasure probability defined as

$$p_{network} = \max_{all\ possible\ cut\text{-}sets} p_{cut\ set}.$$

Proof Denote the given network by $\mathcal{N}_1$. Consider a cut set C of the network $\mathcal{N}_1$, with the source s being in set A and the destination node d in set B and the links 1, 2, $\cdots$, n joining the sets A and B. Form another network $\mathcal{N}_2$ by replacing all links within the sets A and B by perfect links, i.e., links that do not drop packets and further do not introduce any delay as data is transmitted across them. Now for any packet drop pattern, denote the information sets that the destination node has access to at any time step k over networks $\mathcal{N}_2$ and $\mathcal{N}_1$ by $\mathcal{I}^{d,\mathcal{N}_2}(k)$ and $\mathcal{I}^{d,\mathcal{N}_1}(k)$ respectively. It is obvious that

$$\mathcal{I}^{d,\mathcal{N}_1}(k) \subseteq \mathcal{I}^{d,\mathcal{N}_2}(k).$$

Thus, the estimate error covariances at the destination node for the two networks are related by

$$P^d(k|\mathcal{I}^{d,\mathcal{N}_1}(k)) \geq P^d(k|\mathcal{I}^{d,\mathcal{N}_2}(k)).$$

Hence, by considering the stability of error covariance over network $\mathcal{N}_2$, we can obtain a necessary condition for the stability of error covariance over network $\mathcal{N}_1$. Now $\mathcal{N}_2$ consists of the source and the sink joined by edges 1, 2, $\cdots$, n in parallel. The condition for the error covariance across $\mathcal{N}_2$ to converge is thus

$$p_{\text{cut set}}|\rho(A)|^2 < 1,$$

where

$$p_{\text{cut set}} = p_1 p_2 \cdots p_n.$$

This is thus a necessary condition for error covariance across $\mathcal{N}_1$ to be stable. One such condition is obtained by considering each cut-set. Thus, a necessary condition for the error covariance to converge is

$$p_{\text{network}}|\rho(A)|^2 < 1,$$

where

$$p_{\text{network}} = \max_{\text{all possible cut-sets}} p_{\text{cut set}}. \quad \blacksquare$$

Network with Links in Series

Consider now the case where the network consists of two links in series, with probability of packet drops p_1 and p_2. Denote the nodes as N_1, N_2 and N_3 with N_1 being the source node and N_3 the sink. Denote the estimate at node N_i at time k by $\hat{x}^i(k)$. Also, let $e^1(k)$ be the error between

$x(k)$ and $\hat{x}^2(k)$. Similarly let $e^2(k)$ be the error between $\hat{x}^2(k)$ and $\hat{x}^3(k)$. We are interested in the second moment stability of $e^1(k) + e^2(k)$. Clearly, a sufficient condition is that both $e^1(k)$ and $e^2(k)$ individually be second moment stable. Applying Proposition 3.8, this translates to the condition

$$\begin{aligned} p_1|\rho(A)|^2 &< 1 \\ p_2|\rho(A)|^2 &< 1. \end{aligned}$$

If p be the greater of the probabilities p_1 and p_2, the sufficient condition thus is

$$p|\rho(A)|^2 < 1.$$

But this is identical to the necessary condition stated in Proposition 3.9. Thus, the condition above is both necessary and sufficient. Clearly this argument can be extended to any number of links in series. If there are m links in series with the probability of drop of the i-th link being p_i, then a necessary and sufficient condition for the estimate error to diverge at the sink node is

$$p|\rho(A)|^2 < 1,$$

where

$$p = \max(p_1, p_2, \cdots, p_m).$$

Sufficient Condition for Arbitrary Networks

We now proceed to prove that the condition stated in Proposition 3.9 is sufficient as well for stability. We have the following result.

Proposition 3.10 *Consider the assumptions of Proposition 3.9 on the process and the network. Then, the algorithm $\mathcal{A}_2$ will stabilize the process in (3.18) given that*

$$p_{\text{network}}|\rho(A)|^2 < 1$$

Proof First note that if a packet dropping link between two nodes v and u with probability of drop p_e is replaced by two parallel links with drop probabilities $p_i^{(1)}$ and $p_i^{(2)}$ such that $p_i = p_i^{(1)} p_i^{(2)}$, the average error covariance of the estimation under algorithm $\mathcal{A}_2$ does not change at any node. This is true because the probability distribution of the latency in (3.21) does not change with this replacement.

Now, consider the set $\Gamma = \{\gamma_1, \gamma_2, \cdots \gamma_m\}$ of all simple directed paths from the source to the sink in the network graph. An edge i may be in more than one of these paths. If the edge i is in

path γ_j, we will denote that as $i \in \gamma_j$. Consider the following optimization problem

$$\min_{\beta_j} \prod_{j=1}^{m} \beta_j, \tag{3.22}$$

subject to the constraints

$$\begin{aligned} &\prod_{i \in \gamma_j} \beta_j \geq p_i \qquad \forall \text{ edges } i \text{ in the network} \\ &1 \geq \beta_j \geq 0 \qquad \forall j = 1, 2, \cdots, m. \end{aligned} \tag{3.23}$$

A simple change of variables

$$\psi_j = -\log \beta_j, \tag{3.24}$$

transforms the above optimization problem into the following linear program in the variables ψ_j's

$$\max_{\psi_j} \sum_{j=1}^{m} \psi_j \tag{3.25}$$

subject to

$$\begin{aligned} &\sum_{i \in \gamma_j} \psi_j \leq -\log p_i \qquad \forall \text{ edges } i \text{ in the network} \\ &\psi_j \geq 0 \qquad \forall j = 1, 2, \cdots, m. \end{aligned}$$

The solutions of the optimization problems (3.22) and (3.25), denoted by $\{\beta_j^\star\}$ and $\{\psi_j^\star\}$, are related through the relation

$$\psi_j^\star = -\log \beta_j^\star.$$

The structure of the linear program (3.25) is the same as the one used for finding the maximum flow possible in a fluid network [36, Page 59] with the same topology as our packet dropping network and the capacity of the link i being equal to $-\log p_i$. The solution to the problem of finding the maximum flow through a fluid network is well-known to be given by the max-flow min-cut theorem. Using this fact, we see that the solution to the optimization problem (3.25) is given by

$$\psi_j^\star = \min_{\text{all possible cut-sets}} \left(\sum_{i \in cut} -\log p_i \right).$$

Thus, for the optimization problem (3.22), the solution is given by

$$\begin{aligned} \beta_j^\star &= \max_{\text{all possible cut-sets}} \left(\prod_{i \in cut} p_i \right) \\ &= \max_{\text{all possible cut-sets}} p_{\text{cut set}} \\ &= p_{\text{network}}. \end{aligned} \tag{3.26}$$

Next, consider the paths in the set Γ. Assign to each path γ_j the value $\beta_j^\star$ obtained in (3.26). Form a new set $\mathcal{B}$ of all those paths γ_j's for which the associated optimal variable $\beta_j^\star$ is strictly less than one. The remaining paths in Γ have equivalent erasure probability as unity and can be ignored. Now form a new network $\mathcal{N}'$ as follows. The node set of $\mathcal{N}'$ is the union of those nodes of the original network $\mathcal{N}$ that are present on any path in $\mathcal{B}$. Each pair of nodes (u, v) in the node set of $\mathcal{N}'$ is connected by (possibly) multiple links. For all paths $\gamma_j \in \mathcal{B}$, if an edge i is present between two nodes u and v, we add an edge between nodes u and v in $\mathcal{N}'$ and associate with it an erasure probability $\beta_j^\star$. The following properties of $\mathcal{N}'$ are easily verified.

- By construction, $\mathcal{N}'$ can be presented as union of edge-disjoint paths. Each path in $\mathcal{N}'$ corresponds to one path in $\mathcal{B}$. Furthermore, for each path, the probabilities of packet drop on all the links of that path are equal (in fact, equal to $\beta_j^\star$).

- By virtue of (3.26) and the procedure followed to construct $\mathcal{N}'$, the product of the probabilities of packet drop of the different paths is equal to the equivalent probability of the network, p_{network}, for the network $\mathcal{N}$.

- For any pair of nodes that were connected by a link in $\mathcal{N}$, the product of the probabilities of packet dropping of the links in $\mathcal{N}'$ connecting these two nodes is greater than or equal to the drop probability of the link between the same pair of nodes in $\mathcal{N}$. This can be seen from the first inequality constraint of (3.23).

Therefore, the estimate error covariance at the sink by following algorithm $\mathcal{A}_2$ in the original network $\mathcal{N}$ is less than or equal to the error covariance by following $\mathcal{A}_2$ in the new network $\mathcal{N}'$. Thus, to obtain a sufficient condition on stability, we can analyze the performance of $\mathcal{A}_2$ in the network $\mathcal{N}'$. For this we consider another algorithm, which we denote as $\mathcal{A}_3$. In this algorithm we consider the disjoint paths in $\mathcal{N}'$ and assume that estimates on different paths are routed separately. Thus, if a node lies on many paths, on each path it forwards the packets it received on that path only. Clearly the performance $\mathcal{A}_3$ cannot be better than $\mathcal{A}_2$ since in $\mathcal{A}_2$ we send the most recent estimate received from different paths at any node compared to forwarding the estimates on different paths separately from each other.

Therefore, to prove the theorem we only need to show the stability of estimation using protocol $\mathcal{A}_3$ assuming that the condition of Proposition 3.9 holds. Since we do not mix the estimates obtained

from different paths in $\mathcal{A}_3$, the network can be considered as a collection of parallel paths, with each path consisting of links with equal drop probability. Therefore using the stability analysis of serial networks presented earlier, each path (from a stability point of view) can be viewed as an erasure channel with drop probability equal to the drop probability of one link in that path. Using the stability analysis of parallel networks, we see that the stability of the new network under protocol $\mathcal{A}_3$ operation is equivalent to the stability of a packet erasure link with probability of erasure equal to the product of the drop probabilities of different path, which, as mentioned earlier, is equal to the network erasure probability defined in Proposition 3.9. Therefore, assuming that the network erasure probability satisfies

$$p_{\text{network}}|\rho(A)|^2 < 1, \tag{3.27}$$

the network $\mathcal{N}'$ is stable under protocol $\mathcal{A}_3$. But the performance of $\mathcal{A}_3$ cannot be better than of $\mathcal{A}_2$. Thus, $\mathcal{N}'$ is stable under $\mathcal{A}_2$. Therefore the original network $\mathcal{N}$ is stable under protocol $\mathcal{A}_2$ assuming (3.27) is satisfied. ∎

Remarks

1. We have provided a necessary and sufficient condition for the expected error covariance to remain bounded for a network of arbitrary topology. For any other causal data processing algorithm, it provides a necessary condition for stability. For the special case of the network consisting of only one link with erasure probability p, $p_{\text{network}} = p$ and the condition reduces to the condition in Proposition 3.8, as it should.

2. We can also compare the stability condition we have derived for our algorithm with our information processing algorithms. In particular, let us compare this condition to a simpler algorithm $\bar{\mathcal{A}}$ in which the intermediate nodes do not have *any* memory. At each time step k, the source node forwards the measurement $y(k-1)$. The intermediate nodes compare the time stamps of the measurements they received at the previous time step along different incoming edges and forward the most recent one. If they did not receive any measurement on the last time step, they do not transmit anything. By definition, the probability that the destination node receives any particular measurement $y(k)$ from the source over the network is upper-bounded by the reliability of the network (see, e.g., [47]). Let us consider a simple example of a line network in which n edges each with drop probability p are combined in series. With our optimal algorithm, the necessary and sufficient condition for expected estimate error covariance to be stable is

$$p|\rho(A)|^2 < 1.$$

On the other hand, in algorithm $\bar{\mathcal{A}}$, the probability that any measurement is received by the

destination node is

$$q = 1 - (1-p)^n.$$

By a method similar to the one used in [77, 178], it can be proven that a *necessary* condition for stability is that

$$q|\rho(A)|^2 < 1.$$

As an example, for $n = 5$ links and drop probability $p = 0.2$, $q = 0.67$. Thus, our algorithm yields a huge improvement for the stability margin.

3.5.4 Performance Analysis

In this section we calculate the performance of the algorithm $\mathcal{A}_2$. We are interested in the steady-state expected estimate error covariance defined in (3.20). As in the stability analysis, we assume that the packet drops are memoryless and uncorrelated in space. For each link e and time k, let $Z_e(k)$ be the difference between k and the time at which the most recent successful transmission on link e before time k happened, i.e.,

$$Z_e(k) = \min\{j \geq 1 | \lambda_{uv}(k+1-j) = \text{`}received\text{'}\}.$$

We define $Z_{uu}(k) = 1$. Using the definition of $Z_e(k)$, the last time that any message is received at node v from link (u, v) is $k - Z_{uv}(k) + 1$ and that message has time-stamp $t^u(k - Z_{uv}(k))$. Then, (3.19) can be rewritten in terms of $Z_e(k)$ as

$$t^v(k) = \max_{u \in v \cup \mathcal{N}_v} t^v(k - Z_{uv}(k)).$$

$Z_e(k)$ is distributed as a truncated geometric random variable. For $1 \leq i < k$,

$$\text{Prob}\left(Z_e(k) = i\right) = (1-p_e)p_e^{i-1},$$

while for $i = k$,

$$\text{Prob}\left(Z_e(k) = i\right) = 1 - \sum_{i=1}^{k-1} \text{Prob}\left(Z_e(k) = i\right).$$

We can get rid of the truncation by extending the definition of $t^u(k)$ for the case $k < 0$. We define

$$t^u(k) = 0 \qquad \forall k < 0.$$

As an example, for the source node s, without extending the definition we have

$$t^s(k) = k - 1 \qquad \forall k \geq 1.$$

Using the extended definition,

$$t^s(k) = (k-1)^+ \qquad \forall k,$$

where $x^+ = \max\{0, x\}$. In general, using the extended definition of $t^u(k)$ for all k and for any node u, we can easily verify that

$$t^v(k) = \max_{u \in v \cup \mathcal{N}_v} t^u(k - Z_{uv}(k)), \tag{3.28}$$

where $Z_e(k)$'s are now independent random variables distributed according to a geometric distribution. Thus,

$$\text{Prob}\,(Z_e(k) = i) = (1 - p_e)p_e^{i-1} \quad \forall i \geq 1, \forall k.$$

Note that $Z_e(k)$'s do not depend on k anymore. Thus, from now on, we will omit the argument k and just write Z_e. We will refer to p_e as the *parameter* of the geometrically distributed random variable Z_e. Solving the recursive equation (3.28), we can write $t^v(k)$ in terms of the time-stamp at the source node (i.e., $(k-1)^+$) as

$$t^v(k) = \max_{P:\text{an s-v path}} (k - 1 - \sum_{e \in P} Z_e)^+, \tag{3.29}$$

where the maximum is taken over all paths P in the graph $\mathcal{G}$ from source s to the node v. Therefore the latency at node v can be written as

$$l^v(k) = k - 1 - t^v(k) = \min\{k-1, \min_{P:\text{an s-v path}} (\sum_{e \in P} Z_e)\}.$$

From the above equation, it can be seen that as $k \to \infty$ the distribution of $l^v(k)$ approaches a constant distribution of l^v defined as

$$l^v = \min_{P:\text{an s-v path}} (\sum_{e \in P} Z_e). \tag{3.30}$$

For the destination node d, we refer to l^d as the *steady-state latency* of the network. Let us now concentrate on the destination node[7]. From (3.21), the steady-state error covariance can now be rewritten as

$$P(\infty) = \sum_{l=0}^{\infty} \text{Prob}\,(l^d = l) \left[A^l P^\star A^l + \sum_{j=0}^{l-1} A^j Q A^j \right], \tag{3.31}$$

[7]If we are interested in the error covariance at some other node v, simply denote v as the destination node.

where $P^\star$ is the steady-state estimation error covariance of $x(k)$ based on the measurements $\{y(0), y(1), \cdots, y(k-1)\}$ and is the solution to the Discrete Algebraic Riccati Equation (DARE)

$$P^\star = AP^\star A^T + R_w - AP^\star C^T (CP^\star C^T + R_v)^{-1} CP^\star A^T.$$

Since the pair $\{A, R_w^{\frac{1}{2}}\}$ is stabilizable, the rate of convergence of $P(0)$ to $P^\star$ is exponential [112] and the substitution of $P^\star$ for $P(k-1)$ in (3.21) does not change the steady-state error covariance.

Let us define the generating function of the complementary density function $G(X)$ and the moment generating function $F(X)$ of the steady state latency l_d

$$\begin{aligned} G(X) &= \sum_{l=0}^{\infty} \text{Prob}\,(l^d \geq l+1) X^l \\ F(X) &= \sum_{l=0}^{\infty} \text{Prob}\,(l^d = l) X^l, \end{aligned} \tag{3.32}$$

where X is an arbitrary matrix. It can be readily verified that

$$F(X) = (X - I)G(X) + I. \tag{3.33}$$

On vectorizing (3.31) we obtain

$$\begin{aligned} \text{vec}\,(P(\infty)) &= F\,(A \otimes A)\,\text{vec}\,(P^\star) + G\,(A \otimes A)\,\text{vec}\,(Q) \\ &= ((A \otimes A - I)G(A \otimes A) + I)\,\text{vec}\,(P^\star) + G(A \otimes A)\text{vec}\,(Q)\,. \end{aligned} \tag{3.34}$$

We can see from (3.34) that the performance of the system depends on the value of $G(X)$ evaluated at $X = A \otimes A$. In particular, the system is stable if and only if $G(X)$ is bounded at $A \otimes A$. Since $G(X)$ is a power series, boundedness of $G(x)$ at $A \otimes A$ is equivalent to the boundedness of $G(x)$ (evaluated for a scalar x) at the square of the spectral radius of A. We summarize the result of the above arguments as follows.

Proposition 3.11 *Consider a process of the form (3.18) being observed using a sensor of the form (3.17) through an arbitrary network of packet dropping links with drop probabilities p_e's. Let the packet drops be independent from one time step to the next and across links. Then, the minimum expected steady-state estimation error covariance at the receiver is given by (3.34). Furthermore, the error covariance is stable, in the sense of bounded expected steady-state error, iff $|\rho(A)|^2$ lies in the region of convergence of $G(x)$ where $\rho(A)$ is the spectral radius of A.*

The above theorem allows us to calculate the steady state expected error covariance for any network as long as we can evaluate the function $G(X)$ for that network. We now consider some special

networks and evaluate the performance explicitly. We start with a network consisting of links in series, or a line network.

Line Networks

In this case, the network consists of only one path from the source to the destination. Thus,

$$F(X) = E\left[X^{l^d}\right] = E\left[X^{\sum_e Z_e}\right],$$

where the summation is taken over all the edges in the path. Since the drops across different links are uncorrelated, the variables Z_e's are independent. Since Z_e is a geometric random variable,

$$E\left[X^{Z_e}\right] = (1-p_e)X\left(I - p_e X\right)^{-1},$$

provided that $p_e\rho(X) < 1$, where $\rho(X)$ is the spectral radius of matrix X. Therefore,

$$F(X) = E\left[X^{\sum_e Z_e}\right] = \prod_e E\left[X^{Z_e}\right] = \prod_e \left[(1-p_e)X\left(I - p_e X\right)^{-1}\right].$$

Using partial fractions and the relation in (3.33), we can thus easily show that $G(X)$ is given by

$$G(X) = \sum_{i=0}^{n-1} X^i + X^n \sum_e c_e \frac{p_e}{1-p_e}(I - p_e X)^{-1},$$

where

$$c_e = (\prod_{e' \neq e}(1 - \frac{p_e}{p_{e'}}))^{-1}.$$

Therefore the cost can be written as

$$\mathrm{vec}\left(P(\infty)\right) = \prod_e \left[(A \otimes A)\left(\frac{I - p_e A \otimes A}{1-p_e}\right)^{-1}\right] \mathrm{vec}\left(P^\star\right) + G\left(A \otimes A\right)\mathrm{vec}\left(Q\right). \tag{3.35}$$

Remarks

1. We can see from the above argument that the system is stable if for every link e we have $p_e|\rho(A)|^2 < 1$ or equivalently $\max_e p_e|\rho(A)|^2 < 1$. This matches with the condition in section 3.5.3.

2. For the case that some of p_e's are equal, a different partial fraction expansion applies. In

particular for the case when there are n links all with the erasure probability p, we obtain

$$\begin{aligned}\mathrm{vec}\left(P(\infty)\right) = (A\otimes A)^n \left(\frac{I - pA\otimes A}{1-p}\right)^{-n} \mathrm{vec}\left(P^\star\right) \\ + \sum_{i=0}^{n-1}\left[\frac{p}{1-p}(A\otimes A)^n\left(\frac{I-pA\otimes A}{1-p}\right)^{-i-1}\right]\ \mathrm{vec}(Q) + \sum_{i=0}^{n-1}(A\otimes A)^i \mathrm{vec}(Q). \quad (3.36)\end{aligned}$$

3. When there is only one link between the source and the destination, the cost reduces to a particularly simple form. In that case, the steady state error covariance will be the solution to the Lyapunov equation

$$P(\infty) = \sqrt{p}AP(\infty)\sqrt{p}A + \left(Q + (1-p)AP^\star A\right).$$

This expression can alternately be derived using Markov jump linear system theory as in Proposition 3.5.

Network of Parallel Links

Consider a network with one sensor connected to a destination node through n links with probabilities of packet drop $p_1, \ldots, p_n$. Since the same data is being transmitted over all the links, using (3.30) the steady state latency can be written as

$$l_d = \min_{1\le i\le n}(Z_i).$$

Note that Z_i's are all independent geometrically distributed variables with parameters p_i's respectively. It is an easy excercise to show that their minimum is itself geometrically distributed with parameter

$$p_{eq} = \prod_i p_i.$$

Thus, $F(X)$ can be evaluated as

$$F(X) = (1-p_{eq})X(I - p_{eq}X)^{-1},$$

and $G(X)$ can thus be written as

$$G(X) = (I - p_{eq}X)^{-1} = (I - \prod_i p_i X)^{-1}.$$

Thus, the steady-state error can be evaluated using (3.34). Note that the region of convergence of $G(X)$ enforces for stability

$$\prod_i p_i |\rho(A)|^2 < 1,$$

which again matches with the condition in Section 3.5.3.

Arbitrary Network of Parallel and Serial Links

Using similar arguments as in previous sections, we can find the steady-state error covariance of any network of parallel and serial links. These networks are derived from the parallel and serial concatenations of sub-networks. The following two simple rules can give the generating function of the steady-state latency for any network of parallel and series links. Let $l_d(\mathcal{N})$ denote the steady-state latency function of network $\mathcal{N}$. Also, given two subnetworks $\mathcal{N}_1$ and $\mathcal{N}_2$, denote their series combination by $\mathcal{N}_1 \oplus \mathcal{N}_2$ and their parallel combination by $\mathcal{N}_1 \| \mathcal{N}_2$.

1. Suppose the network $\mathcal{N}$ can be decomposed as a series of two subnetworks $\mathcal{N}_1$ and $\mathcal{N}_2$. Then, from the definition of the steady-state latency, and the fact that packet erasures in the two subnetworks are independent of each other, we have

$$l^d(\mathcal{N}_1 \oplus \mathcal{N}_2) = l^d(\mathcal{N}_1) + l^d(\mathcal{N}_2).$$

Thus, we obtain

$$\begin{aligned} F_{\mathcal{N}_1 \oplus \mathcal{N}_2}(X) &= E\left[X^{l^d(\mathcal{N}_1 \oplus \mathcal{N}_2)}\right] \\ &= E\left[X^{l^d(\mathcal{N}_1)+l^d(\mathcal{N}_2)}\right] \\ &= E\left[X^{l^d(\mathcal{N}_1)}\right] E\left[X^{l^d(\mathcal{N}_2)}\right] \\ &= F_{\mathcal{N}_1}(X) F_{\mathcal{N}_2}(X). \end{aligned}$$

Finally, using (3.33), the complementary density function of the network $\mathcal{N}$ is given as

$$\begin{aligned} G_{\mathcal{N}_1 \oplus \mathcal{N}_2}(X) &= (X-I)^{-1}\left(F_{\mathcal{N}_1 \oplus \mathcal{N}_2}(X) - I\right) \\ &= G_{\mathcal{N}_1}(X)(X-I)G_{\mathcal{N}_2}(X) + G_{\mathcal{N}_1}(X) + G_{\mathcal{N}_2}(X) \\ &= (X-I)G_{\mathcal{N}_1}(X) G_{\mathcal{N}_2}(X) + G_{\mathcal{N}_1}(X) + G_{\mathcal{N}_2}(X), \end{aligned}$$

where in the last line we have used the fact that

$$G(X)(X-I) = (X-I)G(X).$$

2. If the network $\mathcal{N}$ can be decomposed as parallel combination of two sub-networks $\mathcal{N}_1$ and $\mathcal{N}_2$, we have

$$l^d(\mathcal{N}_1 \| \mathcal{N}_2) = \min\{l^d(\mathcal{N}_1), l^d(\mathcal{N}_2)\}.$$

Once again, the erasures in the two subnetworks are independent of each other. Thus,

$$\text{Prob}\left(l^d(\mathcal{N}_1 \| \mathcal{N}_2) \geq l\right) = \text{Prob}\left(l^d(\mathcal{N}_1) \geq l\right) \text{Prob}\left(l^d(\mathcal{N}_2) \geq l\right).$$

Thus, if we denote

$$G_{\mathcal{N}_1}(X) = \sum_{i=0}^{\infty} a_i X^i \qquad\qquad G_{\mathcal{N}_2}(X) = \sum_{i=0}^{\infty} b_i X^i,$$

then

$$G_{\mathcal{N}_1 \| \mathcal{N}_2}(X) = \sum_{i=0}^{\infty} a_i b_i X^i.$$

We can now easily derive the steady state error of any network consisting of links in series and parallel with each other using the above two rules. As an example consider the network depicted in Fig. 3.12. In this case

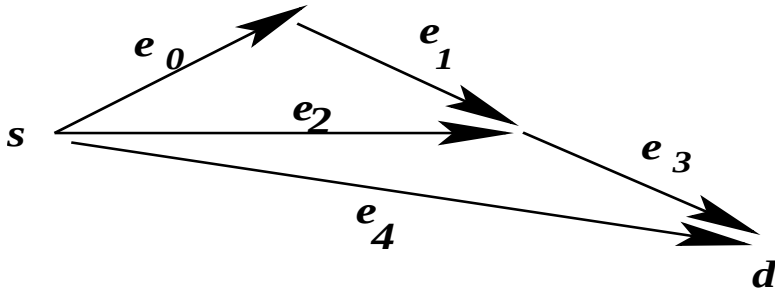


Figure 3.12: Example of a network of combination of parallel and serial links

$$\mathcal{N} = (((\mathcal{N}_0 \oplus \mathcal{N}_1) \| \mathcal{N}_2) \oplus \mathcal{N}_3) \| \mathcal{N}_4),$$

where each of the sub-networks $\mathcal{N}_i$ is just a link with probability of packet drop p. The generating function of a link with erasure probability p is given by

$$G(X) = (I - pX)^{-1}.$$

Suppose a subnetwork with generating function $G(X)$ is in parallel with a link with erasure probability p. Then, it is easy to see that the generating function of the entire network is given by $\mathcal{L}_p(G)(X)$, where $\mathcal{L}_p$ is an operator such that $\mathcal{L}_p(G)(X) = G(pX)$. Thus, the generating function

of the network can be written as

$$G(X) = \mathcal{L}_p(\mathcal{L}_p(G_0 * G_1) * G_3)(X)$$

where

$$G_i(X) = (I - pX)^{-1},\ i = 0, 1, 3$$

is the generating function for the i-th link and $G_i * G_j$ denotes the generating function of the series combination of link i and j. The steady state error covariance can thus be evaluated.

Networks with Arbitrary Topology

In this section, we consider the performance of arbitrary networks. Finding the distribution of the steady-state latency l_d of a general network is not an easy task, because different paths may overlap. This can introduce dependency in the delays incurred along different paths and the calculation of the minimum delay and hence the steady-state latency becomes quite involved. However, using a method similar to the one used in Section 3.5.3, we can provide upper and lower bounds on the performance. We first mention the following intuitive lemma without proof.

Lemma 3.12 *Let $P^{\infty}(\mathcal{N}, \{p_e,\, e \in \mathcal{E}\})$ denote the expected steady-state error of a system with communication network represented by graph $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ and probabilities of packet drop p_e, $e \in \mathcal{E}$. Then, the expected steady-state error is non-increasing in p_e's, i.e., if $p_e \leq q_e \quad \forall\, e \in \mathcal{E}$*

$$P^{\infty}(\mathcal{N}, \{p_e,\, e \in \mathcal{E}\}) \leq P^{\infty}(\mathcal{N}, \{q_e,\, e \in \mathcal{E}\}).$$

Lower Bound Using the above lemma, we can lower bound the steady-state error by making a subset of links erasure free. This is similar to the method we used to prove the necessity of the stability condition in section 3.5.3. Consider any cut-set of the network. Setting the probability of erasure equal to zero for every link except those forming the cut gives a lower bound on the error. Therefore,

$$P^{\infty}(\mathcal{N}, \{p_e,\, e \in \mathcal{E}\}) \geq P^{\infty}(\mathcal{N}, \{q_e,\, e \in \mathcal{E}\}),$$

where

$$q_e = \begin{cases} p_e & e \text{ is in the cut} \\ 0 & \text{otherwise.} \end{cases}$$

Now $P^{\infty}(\mathcal{N}, \{q_e,\, e \in \mathcal{E}\})$ can be evaluated using the results given above for a network of parallel links. By considering the maximum along all possible cut-sets, we obtain the closest lower bound.

Upper Bound Once again, we use a method similar to the one used to prove the sufficiency of the stability condition in Section 3.5.3. In the proof of Proposition 3.10, it is shown that the performance of the network $\mathcal{N}$ is lower bounded by the performance of another network $\mathcal{N}'$ (in other words, the error covariance in $\mathcal{N}$ is upper bounded by the error covariance in $\mathcal{N}'$ that has series and parallel links only and has the following properties:

- $\mathcal{N}$ and $\mathcal{N}'$ have the same node set.
- $\mathcal{N}'$ is the combination of edge-disjoint paths from the source to destination.
- The value of the max-cut in $\mathcal{N}'$ is the same as in the original network $\mathcal{N}$.

The performance of $\mathcal{N}'$ can be computed based on the results given above for arbitrary networks composed of subnetworks in series and parallel. This provides an upper bound on the steady-state error covariance of the original network.

3.5.5 Examples

In this section, we illustrate the above results using some simple examples. Consider a scalar process evolving as

$$x(k+1) = 0.8x(k) + w(k),$$

that is being observed through a sensor of the form

$$y(k) = x(k) + v(k).$$

The noises $w(k)$ and $v(k)$ are assumed zero-mean, white, independent and Gaussian with unit variances. To begin with, suppose that the source and the destination node are connected using two links in series, each with a probability of packet erasure p. Figure 3.13 shows the performance of our strategy as the probability p is varied. The simulation results refer to data generated by a random run averaged over 100000 time steps while the theoretical values refer to the value predicted by using (3.34). We can see that the two sets of values match quite closely.

We also carried out a similar exercise for the source and destination nodes connected by two links in parallel, each with packet erasure probability p. The results are plotted in Figure 3.14. We can once again see that the simulated values match quite closely with the theoretical values.

As the next example, we consider the source and destination nodes connected by a bridge network shown in Figure 3.15. We assume all the links in the network to have probability of erasure p. This network cannot be reduced to a series of series and parallel sub-networks. We can however, calculate the performance analytically in this particular case and compare it to the upper and lower bounds presented earlier. The networks used for calculating the bounds are also shown in Figure 3.15.

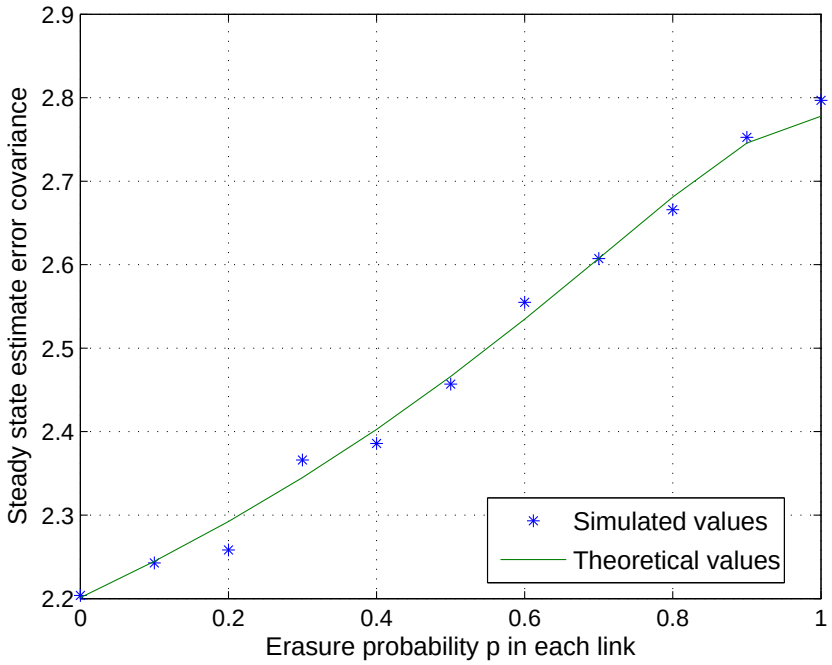


Figure 3.13: Simulated and theoretical results for a line network.

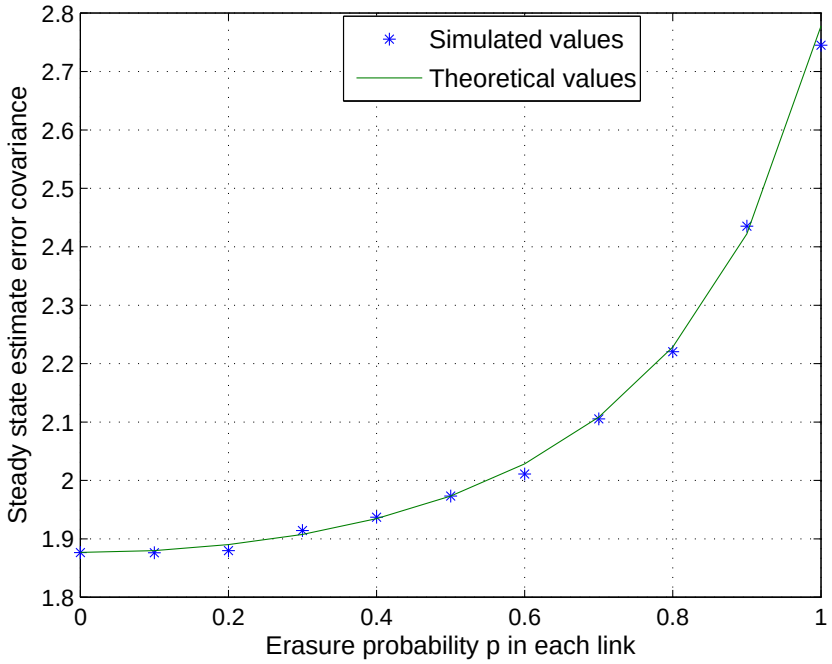
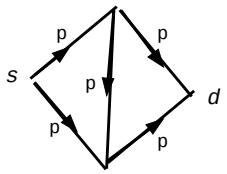
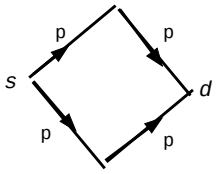
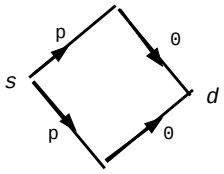


Figure 3.14: Simulated and theoretical results for a parallel network.



Bridge Network





Network used for upper bound

Network used for lower bound

Figure 3.15: Bridge network and the networks used for calculating lower and upper bounds.

Figure 3.16 shows a comparison of the analytical and simulated values with the lower and upper bounds. The simulated values do not fall below the upper bound every-time because of numerical issues; otherwise the bounds are tight.

We can also compare the performance of our algorithm with that obtained if no encoding were done and only measurements were transmitted. Consider the process

$$x(k+1) = \begin{bmatrix} 1.25 & 0 \\ 1 & 1.1 \end{bmatrix} x(k) + w(k)$$

being observed by a sensor of the form

$$y(k) = x(k) + v(k),$$

where $w(k)$ and $v(k)$ are white independent Gaussian noises with means zero and covariance identity. We consider transmission of data across a series of n channels. Figure 3.17 shows the difference in simulated performance of the algorithm in which no encoding is done and for our algorithm for various values of n. It can be seen that the there is considerable gain in performance even for moderate packet drop probabilities if the optimal encoding algorithm is followed. For each point we did 50000 simulations with each simulation being 1000 time steps long.

As a final example, we consider the process

$$x(k+1) = 1.2x(k) + u(k) + w(k)$$

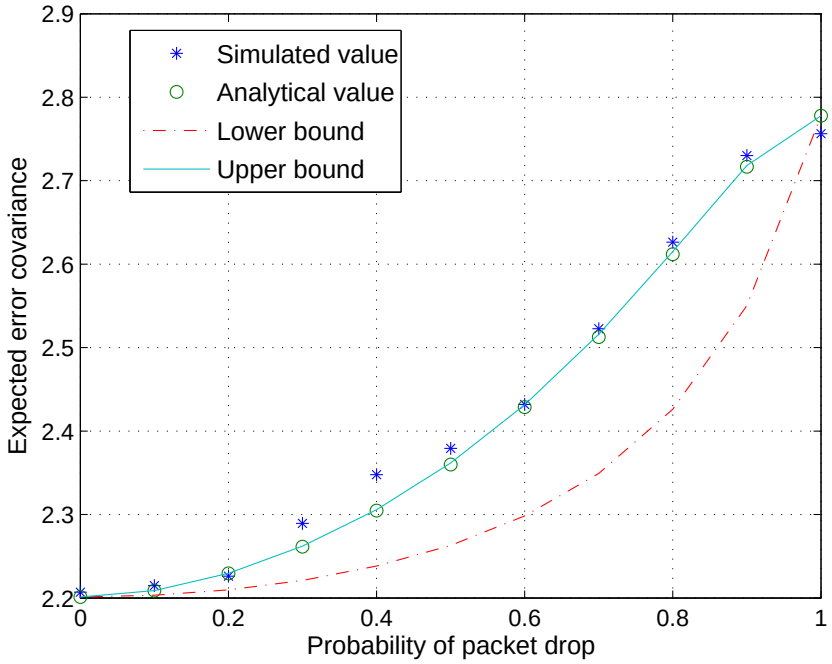


Figure 3.16: Simulated values and theoretical bounds for the bridge network.

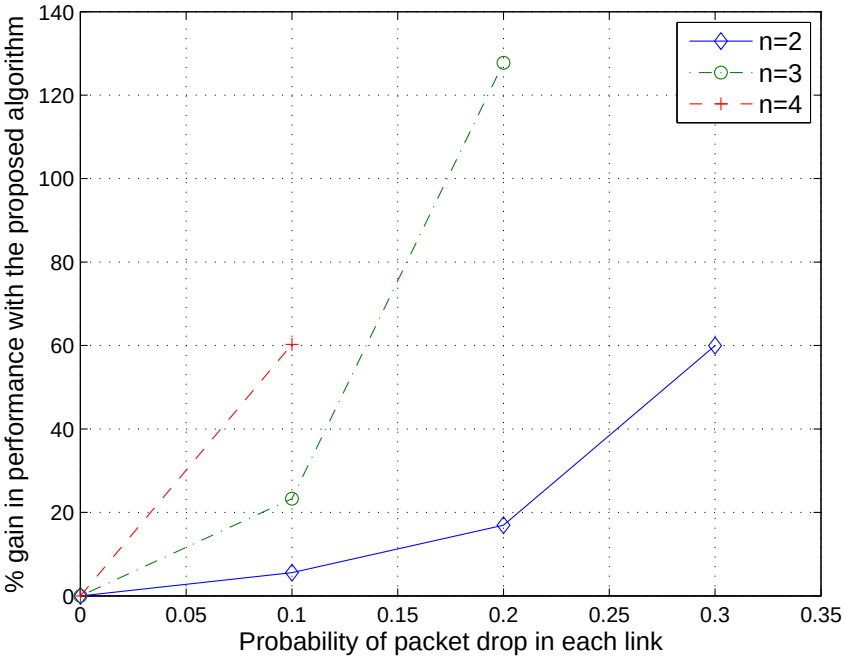


Figure 3.17: Simulated difference in performance of an algorithm in which no encoding is done and our optimal algorithm for a series connection of n links.

being observed through a sensor of the form

$$y(k) = x(k) + v(k),$$

which communicates with the controller over a series network of n links each with packet drop probability p. The controller is interested in minimizing the quadratic cost

$$J = \lim_{K\to\infty} \frac{1}{K} E\left[\sum_{k=0}^{K} x^T(k)Qx(k) + u^T(k)Ru(k)\right].$$

The cost matrices Q and R as well as the noise variances R_w and R_v are assumed to be unity. Figure 3.18 shows the variation of the cost with the probability for different number of links n. The loss in performance is very rapid with the increase in number of links for even moderate values of drop probability.

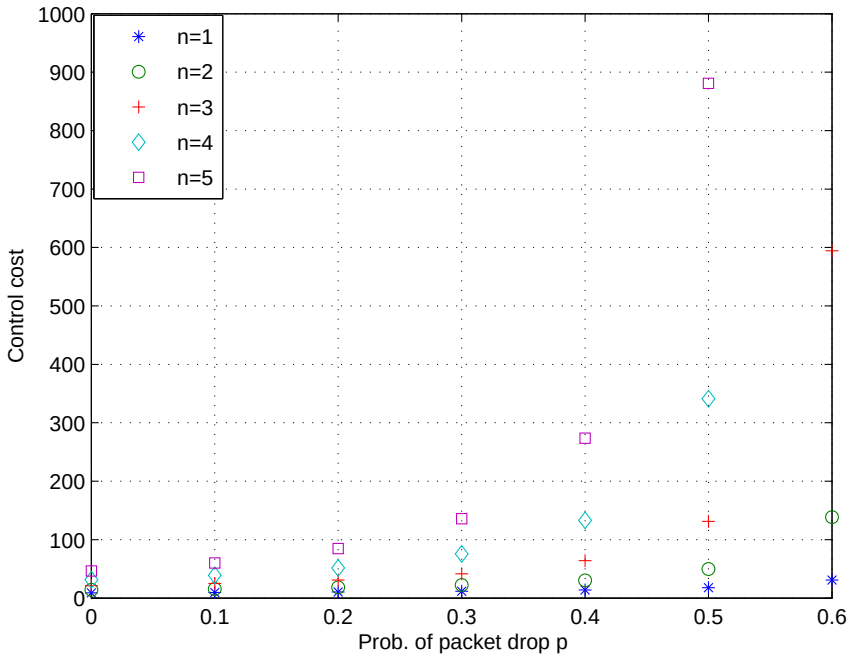


Figure 3.18: Loss in performance as a function of packet drop probability for n links in series.

3.5.6 Correlated erasure events

The analysis so far assumed that the erasure events are memoryless and independent across different links in the network. We could thus formulate the performance in terms of a generating function of the steady-state latency distribution as defined in (3.30) and also obtain stability conditions. We

now look at the effect of dropping these assumptions.

Markov events

If we assume that the drop events on each link are governed by a Markov chain (but are still independent of other links), we can obtain the performance as follows. Let us assume that the packet drop event on link (u, v) evolves according to a Markov chain with transition matrix M_{uv} where M_{uv} is irreducible and reversible. Let us first consider the case where the initial distribution of packet drop on each link is the stationary distribution of the Markov chain on that link. Then, we can rewrite (3.19) in a similar fashion as (3.29) where Z_l is a geometric random variable with distribution

$$\text{Prob}\,(Z_{uv} = l) = \begin{cases} \alpha_{uv} M_{uv}(1,2) M_{uv}(1,1)^{l-2} & \forall\, l \geq 2 \\ 1 - \alpha_{uv} & l = 1 \end{cases},$$

with α_{uv} as the probability of packet drop based on the stationary distribution of link (u, v) and $M_{uv}(i, j)$ as the (i, j)-th element of M_{uv}. Therefore, all the previous analysis goes through. In particular, the stability condition is

$$(\max_{\text{c:s-d cut}} \prod_{e \in \text{c}} M_e(1,1)) |\rho(A)|^2 < 1.$$

Now, if the initial distribution is not the stationary distribution, the variables $Z_{uv}(k)$ will not be time-independent and the analysis does not goes through. However, since for large k the Markov chains will approach their stationary distribution, the stability condition remains unchanged.

Spatially correlated events

Suppose that the packet drop events are correlated across the network but memoryless over time. Now $Z_e(k)$'s are not independent across the network and hence finding the steady-state error covariance does not seem to be tractable. However, we can find the condition for stability. For this, we define a generalized notion of equivalent probability of packet drop for correlated events. Consider a cut-set c, and let $\mathcal{B}(c)$ denote the set of edges crossing this cut. Then, the equivalent probability of packet drop for this cut is defined as

$$p_{eq}(c) = \text{Prob}\,(\lambda_{uv} = \text{`}\textit{dropped}\text{'}, \quad \forall\, (u, v) \in \mathcal{B}(c)).$$

The value of the max-cut for the network is the maximum of $p_{eq}(c)$ over all the cuts,

$$p_{\text{mc}}(\mathcal{N}) = \max_{\text{all cut-sets } c} p_{eq}(c).$$

We can then show that the condition for stability of the system is

$$p_{\mathrm{mc}}(\mathcal{N})|\rho(A)|^2 < 1.$$

To see this, consider the scenario when only one packet is to be routed from the source to destination starting at time t_0. For each time-step $t \geq t_0$ let $\mathcal{V}_r(t)$ denote the set of nodes that have received the packet at time t. Clearly $\mathcal{V}_r(t_0) = \{s\}$. We want to bound the probability that at time $t_0 + T$, destination node has not yet received the packet. For every time-step between t_0 and $t_0 + T$, $\mathcal{V}_r(t)$ clearly forms a cut-set since it contains s and not d. Now the size of $\mathcal{V}_r(t+1)$ does not increase with respect to time-step t iff all the links that cross the cut generated by $\mathcal{V}_r(t)$ drop packets. However, by the definition of $p_{\mathrm{mc}}(\mathcal{N})$ the probability of this event is at most $p_{\mathrm{mc}}(\mathcal{N})$. Therefore,

$$|\mathcal{V}_r(t+1)| \begin{cases} \geq |\mathcal{V}_r(t)| + 1 & \text{with prob. at most } p_{\mathrm{mc}}(\mathcal{N}) \\ = |\mathcal{V}_r(t)| & \text{with prob. at least } 1 - p_{\mathrm{mc}}(\mathcal{N}) \end{cases}$$

Thus, for large T, the probability that at time t_0+T the destination node has not received the packet is upper bounded by $n(1-p_{\mathrm{mc}}(\mathcal{N}))^n T^n p_{\mathrm{mc}}(\mathcal{N})^{T-n}$, where n is the number of nodes in the network. In the original scenario, a new packet is generated at the source at each time step. However, since the importance of the packets used in the estimate is increasing with time, we can upper bound the error by considering that the network is only routing the packet generated at time $k - l$. The probability that the latency is larger than l grows like $f(l)p_{\mathrm{mc}}(\mathcal{N})^l$, where $f(l)$ is polynomial in l with bounded degree and thus the sufficiency of the stability condition follows. The necessity part involves similar ideas and is omitted.

3.5.7 Synthesis of a Network

One can use the performance results above to design networks for the purpose of estimation and control. To consider a simple example, consider a scalar system observed by sensor s. Assume that the destination is located at distance d_0 from the sensor. The probability of dropping a packet on a link depends on its physical length. A reasonable model for probability of dropping packets is given by[8]

$$p(d) = 1 - \exp(-\beta d^{\alpha}),$$

where β and α are positive constants. α denotes the exponent of power decay in the wireless environment. We are interested in the optimal number n of relay nodes that we should place between the sensor and the destination so as to minimize the expected steady-state error covariance. Clearly, there is a trade-off involved since more nodes will reduce the probability of erasure but at

[8]This expression can be derived by considering the probability of outage in a Rayleigh fading environment.

the same time lead to a higher delay before the destination receives a packet. Assuming that n sensors are uniformly placed, there are $n+1$ links each with drop probability q. Thus, from (3.36), $P(\infty)$ can be written as

$$P(\infty) = \left(\frac{a^2(1-q)}{1-qa^2}\right)^{n+1}(P^\star + \frac{R_w}{a^2-1}) - \frac{R_w}{a^2-1}.$$

Thus, assuming that $a^2 > 1$, the optimal number of relay nodes is the solution to the problem

$$\min_n \left(\frac{a^2(1-p(\frac{d_0}{n+1}))}{1-p(\frac{d_0}{n+1})a^2}\right)^{n+1}.$$

If $a^2 < 1$, then the minimization in the above problem is replaced with maximization. As an example consider the system considered in Section 3.5.5. Suppose the sensor and the estimator are placed a distance $d_0 = 5$ units apart. We can calculate the optimal number of nodes to be placed between the source and the destination node using our synthesis results. The constant values we use are α=2, $\beta = 1$. In this case, the optimal number of relays turns out to be $n = 4$.

3.5.8 Unicast Networks

So far, we have assumed that the topology of the network, as given by the graph, was fixed. Any node could transmit a message on all the out-going edges. If there is a restriction on the number of edges that a node can transmit on, the same algorithm can easily be adapted. As an example, we can consider networks that are unicast in the sense that each node should choose one out of a set of possible edges to transmit the message on. Thus, there are two parts of the problem:

1. Choose the optimal path for data to flow from the source node to the sink node.

2. Find the optimal information processing scheme to be followed by each node along that path.

The two parts can clearly be solved separately in the sense that given any path, the optimal processing strategy is the algorithm described in Section 3.5.2. To choose the optimal path, we need to define a metric for the cost of a path. We can consider two choices:

1. If the metric is the condition for stability of the estimate error, then the problem can be recast as choosing the shortest path in a graph with the length of a path being given by its equivalent probability of packet drop. Thus, we need to find the path that has the minimum p_{path} among all the paths. Since each path is just a line network, this reduces to the problem

$$\min_{P:\text{s-d path}} \max_{e\in P} p_e.$$

The above problem is well studied in the computer science community and can be solved as a shortest-path problem over a min-max semi-ring in a distributed fashion [143].

2. If the metric is performance and hence the steady-state error then the problem is more complicated in general. We can consider the special case of a scalar system and no process noise. In this case, from (3.35), we have for path q,

$$\log P_q(\infty) = \sum_{e \in q} \log(\frac{(1-p_e)a^2}{1-p_e a^2})$$

Now the problem is equivalent to

$$\min_{q:\text{s-d path}} \sum_{e \in q} \log(\frac{(1-p_e)a^2}{1-p_e a^2})$$

This problem can also be solved in a distributed way [37].

3.6 Multiple Sensors

We have completely solved the LQG problem in the presence of packet erasure links for the case when only a single sensor is present. We now move on to the case of multiple sensors present. Consider thus a process of the form

$$x(k+1) = Ax(k) + Bu(k) + w(k),$$

being observed via N sensors of the form

$$y_i(k) = C_i x(k) + v_i(k) \qquad \forall i = 1, 2, \cdots N. \tag{3.37}$$

We assume that the measurement noises $v_i(k)$'s are independent of each other and of the process noise $w(k)$. Each sensor needs to communicate to the controller over a packet erasure communication link[9]. We again provide the optimal information processing strategy that needs to be followed by each node in the network to allow the controller to calculate the best possible estimate in the minimum mean square sense. As before, the algorithm is recursive, and thus requires a constant amount of memory, processing and transmission at every node in the network per time step, yet is optimal for any packet-dropping process.

An analysis identical to that of Section 3.4.1 shows that it is sufficient for the algorithm to allow

[9]Even though we present results only for the case when the sensors are communicating over *links*, it is apparent from the treatment of the single sensor case that the results extend to the case of sensors communicating over *networks*.

an estimator to estimate the state of a process that evolves as

$$x(k+1) = Ax(k) + w(k). \tag{3.38}$$

Accordingly, from now on, we will restrict our attention to the task of estimating in the minimum mean squared error sense the state $x(k)$ of a process evolving as in (3.38) through sensors of the form (3.37) when the sensors communicate over communication links that stochastically erase packets.

The problem is much more complicated for the case of multiple sensors than a single sensor. We start with a simple case depicted in Figure 3.19 in which only one of the N sensors transmits information over a packet-dropping link; the other sensors are able to communicate with the estimator at every time step. As we shall show later, this problem is related to the problem depicted in Figure 3.5 in which two sensors aim at obtaining a joint estimate of the state of a dynamic process while communicating over packet erasure links. While our first impulse would be to transmit state estimates that each sensor generates using local measurements, as mentioned in Section 3.1, this approach is sub-optimal. We will provide the optimal algorithm that identifies the correct quantity to transmit.

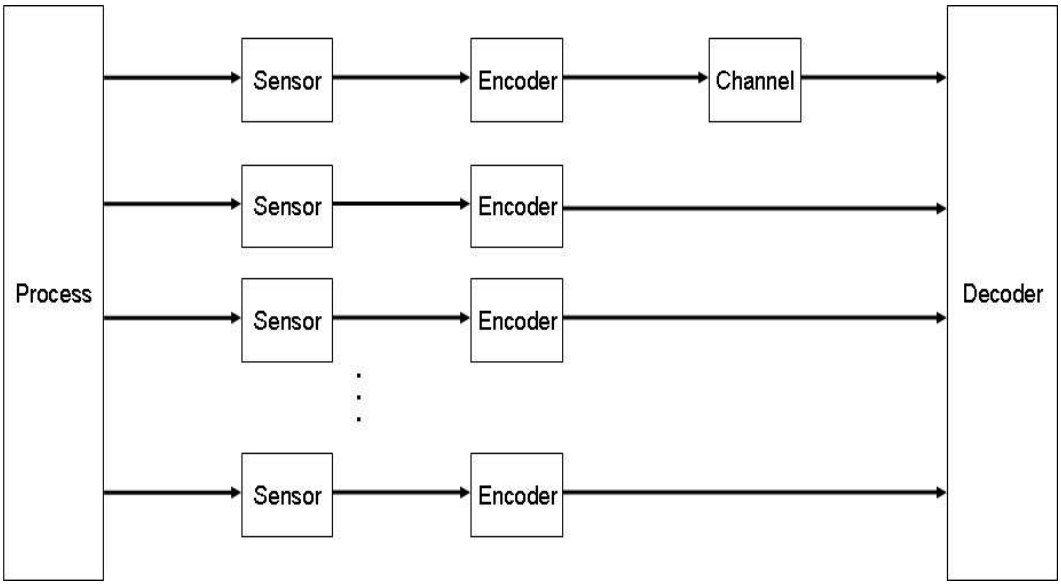


Figure 3.19: Estimation using information from multiple sensors. Only one sensor transmits over an imperfect communication link.

3.6.1 Optimal Information Transmission Algorithm

Let $\hat{x}_i(k|l)$ denote the estimate of $x(k)$ evolving as in (3.38) based on all the measurements of sensor i up to time l. Denote the corresponding error covariance by $P_i(k|l)$. We will refer to these quantities

as the *local* estimates and error covariances respectively. We can also define the corresponding *global* quantities $\hat{x}(k|l)$ and $P(k|l)$ that depend on measurements from all the N sensors up to time l. Without loss of generality, denote the sensor that transmits on the packet erasure link as sensor 1. The optimal information processing algorithm proceeds as follows.

1. *Encoder for sensor 1:* At each time step $k \geq 1$,

   - Obtain measurement $y_1(k-1)$ and run a local Kalman filter to obtain $\hat{x}_1(k-1|k-1)$ and $P_1(k-1|k-1)$.
   - Calculate

   $$\theta_1(k-1) = \left(P_1(k-1|k-1)\right)^{-1} \hat{x}_1(k-1|k-1) - \left(P_1(k-1|k-2)\right)^{-1} \hat{x}_1(k-1|k-2).$$

   - Calculate the global error covariance matrices $P(k-1|k-1)$ and $P(k-1|k-2)$ using

   $$\begin{aligned} \left(P(k-1|k-1)\right)^{-1} &= \left(P(k-1|k-2)\right)^{-1} + \sum_{i=1}^{N} (C_i)^T \left(R_{v,i}^1\right)^{-1} (C_i) \\ P(k-1|k-2) &= AP(k-2|k-2)A^T + R_w. \end{aligned}$$

   - Obtain $\gamma(k-1) = \left(P(k-1|k-2)\right)^{-1} AP(k-2|k-2)$.
   - Finally, calculate

   $$i_1(k) = \theta_1(k-1) + \gamma(k-1) i_1(k-2)$$

   with $i_1(-2) = i_1(-1) = 0$ and transmit it.

2. *Encoder for all other sensors 2, 3, $\cdots$, N:* At each time step $k \geq 1$, transmit the measurement $y_i(k-1)$.

3. *Decoder:* At each time step $k \geq 2$,

   - For all $j = 2, 3, \cdots, N$, use $y_j(k-2)$ to come up with $i_j(k-2)$ using an algorithm similar to the one followed by the encoder for sensor 1.
   - Maintain a local variable $\hat{x}^{dec}(k|k)$ which is updated as follows.
     (a) If $\lambda_1(k-1) =$ '*received*', all links successfully transmitted packets. In that case, set the estimate through

     $$\left(P(k-2|k-2)\right)^{-1} \hat{x}^{dec}(k-2|k-2) = \sum_{j=1}^{N} i_j(k-2).$$

(b) If $\lambda_1(k-1) =$ '*dropped*', sensor 1 could not transmit any information. In this case, propagate the estimate $\hat{x}^{dec}(k-3|k-3)$ using the measurements $y_i(k-2)$ for $i = 2, 3, \cdots, N$ through a Kalman filter.

- Finally, declare the estimate of the decoder $\hat{x}^{dec}(k|k-2)$ as

$$\hat{x}^{dec}(k|k-2) = A^2 \hat{x}^{dec}(k-2|k-2).$$

Proposition 3.13 *In the above algorithm,*

$$\hat{x}^{dec}(k|k-2) = \hat{x}\left(k|\mathcal{I}^{max}(k)\right).$$

Proof Consider a centralized Kalman filter that has access to measurements from a sensor of the form

$$y(k) = Cx(k) + v(k)$$

where

$$C = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_N \end{bmatrix} \qquad v(k) = \begin{bmatrix} v_1(k) \\ v_2(k) \\ \vdots \\ v_N(k) \end{bmatrix}. \tag{3.39}$$

Let R be the covariance matrix of the noise $v(k)$. Since the measurement noises $v_i(k)$'s are independent of each other, R is block-diagonal. Thus, the measurement update equations of the Kalman filter are

$$\begin{aligned}
(P(k|k))^{-1} &= (P(k|k-1))^{-1} + C^T R^{-1} C \\
&= (P(k|k-1))^{-1} + \sum_i \left[(P_i(k|k))^{-1} - (P_i(k|k-1))^{-1} \right] \\
(P(k|k))^{-1} \hat{x}(k|k) &= (P(k|k-1))^{-1} \hat{x}(k|k-1) + C^T R^{-1} y(k) \\
&= (P(k|k-1))^{-1} \hat{x}(k|k-1) \\
&\quad + \sum_i \Big[(P_i(k|k))^{-1} \hat{x}_i(k|k) - (P_i(k|k-1))^{-1} \hat{x}_i(k|k-1) \Big].
\end{aligned}$$

Recognizing that the time update equations are

$$\begin{aligned}
P(k|k-1) &= AP(k-1|k-1)A^T + R_w \\
\hat{x}(k|k-1) &= A\hat{x}(k-1|k-1),
\end{aligned}$$

we can write

$$\begin{aligned}(P(k|k))^{-1}\hat{x}(k|k) &= (P(k|k-1))^{-1}\hat{x}(k|k-1) \\ &\quad + \sum_i \left((P_i(k|k))^{-1}\hat{x}_i(k|k) - (P_i(k|k-1))^{-1}\hat{x}_i(k|k-1)\right) \\ &= \sum_i I_i(k),\end{aligned}$$

where the term $I_i(k)$ is the contribution of the measurements of the i-th sensor and is given by

$$I_i(k) = \Theta_i(k) + \Gamma(k)\Theta_i(k-1) + \Gamma(k)\Gamma(k-1)\Theta_i(k-2) + \cdots + (\Gamma(k)\Gamma(k-1)\cdots\Gamma(1))\,\Theta_i(0),$$

where

$$\begin{aligned}\Theta_i(k) &= (P_i(k|k))^{-1}\hat{x}_i(k|k) - (P_i(k|k-1))^{-1}\hat{x}_i(k|k-1) \\ \Gamma(k) &= (P(k|k-1))^{-1}AP(k-1|k-1).\end{aligned}$$

The covariance matrices do not involve any measurements and can be calculated off-line.

Note that because of our assumptions about the time-line, even if there were no packet drops, at time step k, the estimator can only calculate $\hat{x}(k|k-2)$. Thus, the information needed from sensor j to calculate the global estimate at time step k is precisely $I_j(k-2)$. Now for the case when $\lambda(k-1) =$ '*received*', the decoder in the algorithm has access to $i_j(k-2)$'s that are the same as $I_j(k-2)$'s. Thus, it can calculate the centralized Kalman filter output $\hat{x}(k|k-2)$ which is $\hat{x}(k|\mathcal{I}^{\max}(k))$. For the case when $\lambda(k-1) =$ '*dropped*', the decoder propagates the best Kalman filter estimate $\hat{x}^{dec}(k-3|k-3)$ with measurements from all other sensors except the 1st sensor. Thus, in this case too, $\hat{x}^{dec}(k|k-2) = \hat{x}(k|\mathcal{I}^{\max}(k))$ ∎

Proposition 3.13 also presents the solution to the estimation problem depicted in Figure 3.5. We can use an encoder and a decoder described in the algorithm at each sensor. The decoder has access to local measurements at every time step while the other sensor transmits information over a packet dropping channel. Thus, the situation is identical to the one considered above for the case $N = 2$.

Remarks

1. Note that the computation and memory required for calculating $I_i(k)$ does not grow with time since we can use the recursion

$$I_i(k) = \Theta_i(k) + \Gamma(k)I_i(k-1).$$

2. Once more, properties like 'washing away' of the effect of any previous packet losses, optimality

for any packet dropping process, ability to include large delays and packet re-ordering etc. hold.

3. The algorithm, as proposed, does not extend to multiple packet dropping channels. The crucial assumption used in the algorithm that prevents this extension is that the encoder for sensor 1 uses the fact that all other sensors will transmit their information at every time step. For multiple channels, this assumption will not be satisfied. Extension of the algorithm to such cases remains an open problem.

3.6.2 Analysis of the Proposed Algorithm

We now model the channel erasures as occurring according to a Markov chain and analyze the stability and performance of the estimation error covariance using our algorithm. Let the transition probability matrix of the Markov chain be denoted by Q. We are once again interested in stability in the sense of bounded second moment, i.e., the system is stable if the error

$$x(k) - \hat{x}^{dec}(k|k-2)$$

have bounded covariance as time k evolves. This is, in turn, equivalent to the condition that the error

$$x(k) - \hat{x}^{dec}(k|k-1)$$

have bounded covariance as time k evolves. For simplicity, we consider only the case $N = 2$. The extension to the general case is straight-forward.

Denote by $y(k)$ the vector formed by stacking $y_1(k)$ and $y_2(k)$. We have three dynamical systems. The plant state $x(k)$ evolves as in (3.38). The state $\hat{x}(k)$ of a centralized Kalman filter with access to measurements from both sensors at every time step would evolve as

$$\hat{x}(k+1) = A\hat{x}(k) + K^c(k)\left(y(k) - C\hat{x}(k)\right).$$

Finally, the state $\hat{x}^{dec}(k)$ of the estimator at the decoder evolves according to

$$\hat{x}^{dec}(k+1) = \begin{cases} A\hat{x}^{dec}(k) + K^d(k)\left(y_2(k) - C_2\hat{x}^{dec}(k)\right) & \text{channel in state 1} \\ \hat{x}(k+1) & \text{otherwise.} \end{cases}$$

Denote $e(k) = x(k) - \hat{x}(k)$ and $t(k) = \hat{x}(k) - \hat{x}^{dec}(k)$. Note that $t(k)$ evolves according to

$$t(k+1) = \begin{cases} \left(A - K^d(k)C_2\right)t(k) + L^1(e(k)) + L^2(v_1(k)) + L^3(v_2(k)) & \text{channel in state 1} \\ 0 & \text{otherwise,} \end{cases} \tag{3.40}$$

where $L^n(\beta)$ denotes a term linear in β. Now $e(k)$ has bounded covariance matrices by our detectability assumption. Also, the measurement noises $v_i(k)$'s have bounded covariance. For $t(k)$ to be of bounded variance, the Markov jump system of (3.40) needs to be stable. Further, note that since our encoder/decoder design is optimal, if the system is unstable with our design, it is not stabilizable by any other design. We can thus say the following.

Proposition 3.14 *Consider the estimation and control problem defined in Section 3.6 for the case of two sensors. Let the packet erasure channel is modeled as a Markov chain with transition probability matrix $Q = [q_{ij}]$. Let the matrix pair (A, B) be stabilizable and the matrix pair (A, C) be detectable. The system is stabilizable, in the sense that the variance of the state is bounded, if and only if $q_{22}|\lambda_{\max}\left(\bar{A}\right)|^2 < 1$, where $\lambda_{\max}\left(\bar{A}\right)$ is the maximum magnitude eigenvalue of the unobservable part of matrix A when (A, C_2) is put in the observer canonical form and q_{22} is the probability that the channel drops packets at two consecutive time steps. Further, if the system is stabilizable, one controller and encoder/decoder design that stabilizes the system is given in Proposition 3.13.*

We can also carry out a performance analysis along the lines of the single sensor case. We omit the details here.

3.6.3 Example

We consider a simple example now. Consider the system considered in Section 3.4.4. The system evolves as

$$\begin{aligned} x(k+1) &= \begin{bmatrix} 0 & -2 \\ 1 & -1 \end{bmatrix} x(k) + \begin{bmatrix} 2 \\ 1 \end{bmatrix} u(k) + \begin{bmatrix} 2 \\ 1 \end{bmatrix} w(k) \\ y(k) &= \begin{bmatrix} 0 & 1 \end{bmatrix} x(k). \end{aligned}$$

The process noise $w(k)$ is zero mean with unit variance and the packet drop process is i.i.d. The system is observed through two sensors of the form

$$\begin{aligned} y_1(k) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(k) + v_1(k) \\ y_2(k) &= \begin{bmatrix} 0 & 1 \end{bmatrix} x(k) + v_2(k). \end{aligned}$$

The sensor noises are zero mean with variance 10 and 1 respectively. We consider the cost function $\lim_{T\to\infty}\left(y^2(T)\right)$. Figure 3.20 shows the simulated performance of our algorithm as a function of the packet loss probability. We also plot the performance for a hypothetical sensor that receives information from both sensors without any packet drop and for an algorithm in which sensors exchange only measurements. It can be seen that even in this very simple case, our algorithm can lead to a performance gain of up to 40% over the strategy of using no encoder. For the purpose of

the numerical example, we have assumed that the estimate at the controller was calculated causally, i.e., to calculate an estimate of $x(k)$, measurements $y_1(k)$ and $y_2(k)$ could be used.

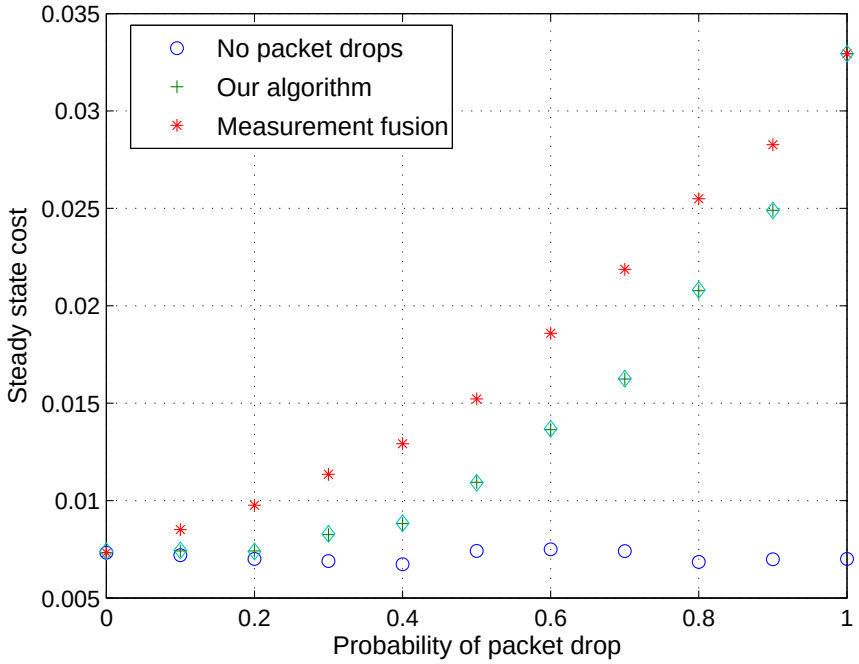


Figure 3.20: Comparison of performance for various algorithms for the two sensor case.

3.6.4 Extensions

The algorithm we have provided can be easily adapted for any situation in which a sensor that is communicating over a packet erasure link is able to predict the sensors that would be able to transmit their data to the estimator. As an example, consider the following situation. The system evolves according to

$$x(k+1) = Ax(k) + w(k).$$

The system is observed by N sensors with the i-th sensor being of the form

$$y_i(k) = C_i x(k) + v_i(k).$$

All the sensors communicate over a multiple access channel to the controller. Nominally, every sensor is able to transmit its information. However, sometimes the channel is in a deep fade and the packets from every sensor are dropped. We can adapt the above algorithm to identify the optimal encoder and decoder for this situation as well. The encoder design remains unchanged and the j-th sensor transmits the quantity $i_j(k)$. If the decoder receives the packets, it calculates its estimate

through

$$(P(k-2|k-2))^{-1}\,\hat{x}^{dec}(k-2|k-2) = \sum_{j=1}^{N} i_j(k-2).$$

If the decoder does not receive anything, it time updates its own estimate from the previous time step.

3.7 Discussion

In this chapter, we considered the problem of optimal LQG control when sensors and controller are communicating across a channel or a network. We modeled the links as switches that drop packets randomly and proved that a separation exists between the optimal estimate and the optimal control law. For the optimal estimate, we identified the information that the sensors should provide to the controller in a lot of cases. This can be viewed as constructing an encoder for the channel. We also designed the decoder that uses the information it receives across the link to construct an estimate of the state of the plant. The proposed algorithms are recursive yet optimal irrespective of the packet drop pattern. For the case of packet drops occurring according to a Markov chain, we carried out stability and performance analysis of our algorithm.

Viewing communication links as a means of transmitting information and constructing encoders and decoders for the purpose of estimation and control is a very powerful idea. As we have seen, it leads to huge performance gains. We have identified simple recursive yet optimal structures for many cases. However, more work is needed. For the case of multiple packet-dropping links, the optimal design is still unknown. Another intriguing possibility is considering the effect of allowing only a limited number of bits in the packet. The works of Sahai [170] and Ishwar et al. [105] seem relevant in this direction. However, from the view of optimal control, this issue has to be examined in greater detail.

Appendix A: Effect of Quantization on the Performance at High Rates

In this appendix, we consider a different effect of the channel: quantization. We study the impact of quantization on the performance of a scalar dynamical system in the high rate regime.

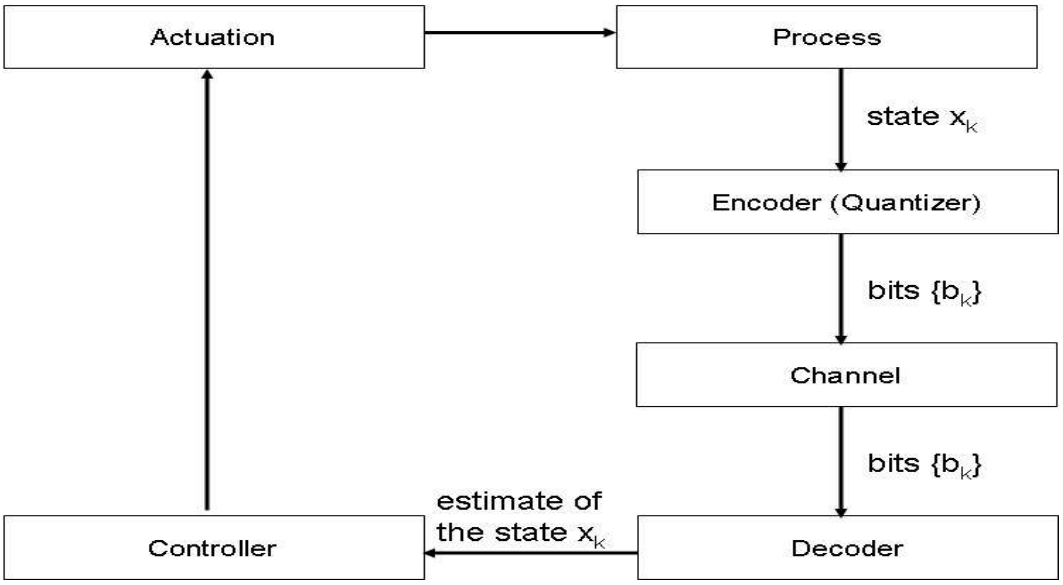


Figure 3.21: A system in which a quantizer quantizes the state and transmits it across a digital channel.

Consider the situation described in Figure 3.21. We have a linear time-invariant process evolving in discrete time according to the relation

$$x(k+1) = ax(k) + u(k) + w(k) \tag{3.41}$$

with $x(k) \in \mathbf{R}$ as the process state, $u(k)$ as the control input and $w(k)$ as zero-mean white noise[10]. The noise $w(k)$ is assumed to be bounded in an interval $[-M, M]$ with variance σ^2. The initial condition $x(0)$ is assumed to be distributed in the interval $[-s, s]$ and is assumed to be independent of $w(k)$. The assumption of bounded noise and initial condition uncertainty is essential to prevent quantizer overflow issues.

The process state is observed in a noiseless fashion by the encoder. The encoder denotes a mapping from the state $x(k)$ to a stream of bits $\{b(k)\}$. The encoder has access to all the previous states $\{x(j)\}_{j=0}^{k}$ and the previous control signals $\{u(j)\}_{j=0}^{k-1}$ when it encodes $x(k)$. For the most part we will restrict our attention to encoders that merely perform the action of quantization and ignore the possibility of other source coding.

[10]To emphasize the point that our results in this appendix hold only for scalar systems, we will use lower case letters, thus a instead of the system matrix A. The same comment holds for the control law and the cost function defined later.

The bit-stream $\{b(k)\}$ is transmitted over a channel. We will chiefly be concerned with noiseless digital channels, thus, the decoder receives the bit stream without any erasures or flips. Based on the bit stream, the decoder outputs an estimate of the state $\hat{x}(k)$. The decoder has access to all the previous bit streams $\{b(j)\}_{j=0}^{k}$, the previous decoded estimates $\{\hat{x}(j)\}_{j=0}^{k-1}$ and the previous control signals $\{u(j)\}_{j=0}^{k-1}$ when it decodes $\{b(k)\}$. We assume a linear control law of the form $u(k) = f\hat{x}(k)$. This control signal $u(k)$ is then used in the further evolution of the process according to (3.41). We assume that there is no channel present between the controller and the process. In the absence of any encoder, channel and decoder, we have $\hat{x}(k) = x(k)$. In general the two quantities would not be equal.

As discussed in Section 3.1, a lot of works have analyzed the stability of the above design. However, the performance of the system is less well understood. We consider the finite-time and infinite time horizon LQ costs given by

$$\begin{aligned} J_T &= E\left[\sum_{k=0}^{T} x^2(k)q + u^2(k)r\right] \\ J_\infty &= \lim_{T\to\infty} \frac{1}{T} J_T, \end{aligned} \tag{3.42}$$

where, as usual, q is positive and r is non-negative. We assume that system has been sufficiently well-designed so that it remains stable and inside the range of operation of the quantizer at all times to avoid quantizer over-flow. We wish to consider the effect of various quantizers on J_K and J_∞.

We will denote the probability density function of a continuous random variable X by $f_X(x)$ and its expectation by $E[X]$. The differential entropy of X is denoted by $h(X)$ and defined according to (see, e.g., [39])

$$h(X) = -\int_{-\infty}^{\infty} f_X(x) \log f_X(x) dx,$$

where $0 \log 0$ is interpreted as 0 and the log is taken to the base 2. A scalar quantizer Q of size N is a mapping from an interval on the real number line into a finite set $\mathcal{C}$ containing N reproduction points called codewords. The interval is partitioned into N cells where the i-th cell, denoted by R_i, is an interval for which each point maps to the i-th codeword y_i as

$$R_i = \{x \in \mathbf{R} : Q(x) = y_i\},$$

where y_i is the i-th codeword. The quantizer is said to be mid-point based if the reconstruction level y_i is the mid-point of the cell R_i. It is said to be centroid-based if y_i is the centroid (conditional expectation of x in region R_i) of region R_i. We call a quantizer static or fixed if the mapping does not change with time, otherwise we call it dynamic. The rate of the quantizer is defined as $r = \log_2(N)$. We will be interested in the mean squared distortion of the quantizer defined as $D = E\left[|X - \hat{X}|^2\right]$

when the scalar random variable X is quantized and reproduced as $\hat{X}$.

Analysis

We begin by considering the case when the quantizer is fixed. In general, the cost function is not easy to calculate analytically. This is because the quantization error at time k depends on the probability density function of $x(k)$ which is not easy to calculate as time k evolves. To obtain a handle on the performance of different quantizers, we make the high rate approximation, which says that the rate of the quantizer is high (and hence the distortion is low). The results we obtain can thus be treated as approximations which become better as the rate of the code increases.

Theorem 3.15 (From [73]) *Given a scalar quantizer with a mean square based distortion measure $d(x,y) = ||x-y||^2$, the expected distortion of the random variable X being quantized can be bounded as follows*

$$\bar{d} \geq \bar{d}_L = \frac{1}{12N^2} E\left[\lambda(X)^{-2}\right],$$

where $\lambda(X)$ is the asymptotic quantizer density normalized to unit integral, obtained as we keep on increasing the number of quantization levels while N refers to the total number of quantization cells. Further, the lower bound becomes tighter as the rate of the code gets high.

Uniform quantizer We first consider a mid-point based uniform quantizer, which is a very simple and commonly used quantizer. If the region to be quantized is $[-t,t]$, the asymptotic quantizer density is given by

$$\lambda(x) = \frac{1}{2t}.$$

Since $N = \frac{2t}{\delta}$, the distortion measure evaluates to

$$\bar{d} = \frac{\delta^2}{12}.$$

In addition we note from [137] that at high rates, for a mid-point based uniform quantizer, if we denote the variable being quantized as x_k and the quantization error by δ_k, then $E\left[x_k\delta_k\right] \ll E\left[\delta_k^2\right]$ and thus can be approximated to be zero[11]. Thus, we can now evaluate the cost function to be

$$J_T = (q+rf^2)E[x_0^2]\sum_{k=0}^{T}(a+f)^{2k} + rf^2(T+1)\frac{\delta^2}{12} + \frac{q+rf^2}{1-(a+f)^2}\left(\frac{f^2\delta^2}{12}+\sigma^2\right)\sum_{k=0}^{T}(1-(a+f)^{2k}).$$

For calculation of J_∞ we need to find conditions such that $\lim_{T\to\infty} J_T/T$ does not diverge. To this end, we assume that $(a+f)^2 < 1$ and that $E[x^2(k+1)] < E[x^2(k)]$. The first condition means f is

[11] The result is true only under some technical conditions listed in [137] that, however, hold in our case.

stabilizing while the second condition places a limit on the size of the quantization cell. Assuming that there are N quantization cells, this condition implies that

$$N^2 \geq \frac{f^2 s^2}{3\left(1-(a+f)^2\right)E[x^2(k)] - 3\sigma^2}.$$

In particular for $k=0$, this condition implies

$$N^2 \geq \frac{f^2 s^2}{\left(1-(a+f)^2\right)s^2 - 3\sigma^2}.$$

Note that for the case when there is no noise and the control law $f=-a$, this reduces to the results derived in, e.g., [185]. With these assumptions, the infinite-horizon cost is

$$J_\infty = rf^2\frac{\delta^2}{12} + \frac{q+rf^2}{1-(a+f)^2}\left(\frac{f^2\delta^2}{12}+\sigma^2\right).$$

Logarithmic Quantizer We now calculate the cost for a mid-point based logarithmic quantizer that has been shown to be the most optimal quantizer for stabilization [54]. To apply Theorem 3.15 for a logarithmic quantizer that is operating over the union of the regions $[-t,-\epsilon]$ and $[\epsilon,t]$, we note that the asymptotic quantizer density is given by

$$\lambda(x) = \frac{1}{|x|}|_{\text{normalized to unit integral}} = \frac{1}{2|x|\ln(\frac{t}{\epsilon})}.$$

Thus, the distortion measure approximately evaluates to

$$\bar{d} = \frac{1}{3}\left(\frac{\ln(\frac{t}{\epsilon})}{N}\right)^2 E\left[x^2\right].$$

Now, consider a logarithmic quantizer with ratio g. Thus, the quantization cells are given for the positive axis by the intervals $[\epsilon, g\epsilon]$, $[g\epsilon, g^2\epsilon]$, $\cdots$, $[g^{p-1}\epsilon, g^p\epsilon]$, where p and N are related by $2p=N$. Since $g^p\epsilon = t$,

$$\bar{d} = \frac{1}{3}\left(\frac{\ln(g)}{2}\right)^2 E\left[x^2(k)\right] = \frac{(\ln g)^2}{12}E\left[x^2(k)\right] \approx E\left[\Delta^2(k)\right].$$

Using the Cauchy-Schwarz inequality

$$-\sqrt{E[\Delta^2(k)]E[x^2(k)]} \leq E[\Delta(k)x(k)] \leq \sqrt{E[\Delta^2(k)]E[x^2(k)]},$$

we can obtain that

$$h_1 E[x^2(0)]\frac{1-g_1^{T+1}}{1-g_1} + \frac{h_1\sigma^2(T-1+g_1^{T+1})}{1-g_1} \leq J_T \leq h_2 E[x^2(0)]\frac{1-g_2^{T+1}}{1-g_2} + \frac{h_2\sigma^2(T-1+g_2^{T+1})}{1-g_2},$$

where

$$\begin{aligned} g_1 &= (a+f)^2 + cf^2 - 2 \mid f(a+f) \mid \sqrt{c} \\ g_2 &= (a+f)^2 + cf^2 + 2 \mid f(a+f) \mid \sqrt{c} \\ h_1 &= q + rf^2 + rcf^2 - 2rf^2\sqrt{c} \\ h_2 &= q + rf^2 + rcf^2 + 2rf^2\sqrt{c} \end{aligned}$$

and $c = \frac{(\ln g)^2}{12}$. Thus, a necessary condition for J_∞ to exist is $g_1 \leq 1$ and a sufficient condition is $g_2 \leq 1$. Assuming these conditions exist, we obtain

$$\frac{h_1\sigma^2}{1-g_1} \leq J_\infty \leq \frac{h_2\sigma^2}{1-g_2}.$$

Lower Bound for Centroid-based Quantizers It is well-known that the optimal quantizer minimizing the mean-square distortion error is a centroid-based quantizer [73]. Since the density function of $x(k)$ depends on the densities of all previous quantization errors, it is difficult to compute a priori and the optimal quantizer has to be obtained at every step through an iterative algorithm such as the Lloyd-Max algorithm [66, 73] or through a dynamic programming based algorithm [26]. We now consider such quantizers. To begin with, we note that for centroid-based quantizers (see, e.g., [137]) $E[\Delta(k)\hat{x}(k)] = 0$ for every time step k. Thus,

$$E[x^2(k+1)] = (a+f)^2 E[x^2(k)] - (f^2+2af)E[\Delta^2(k)] + \sigma^2.$$

Thus, the cost can be evaluated as

$$\begin{aligned} J_T = \sum_{k=0}^{T} \Big[& -rf^2 E[\Delta_k^2] + (q+rf^2)(a+f)^{2k} E[x^2(0)] + \\ & \sigma^2(q+rf^2)\sum_{j=0}^{k-1}(a+f)^{2j} - (f^2+2af) \times (q+rf^2)\sum_{j=0}^{k-1}(a+f)^{2j} E[\Delta^2(k-1-j)] \Big]. \end{aligned}$$

The cost can easily be evaluated for specific quantizers such as uniform or logarithmic. Instead, we lower bound the cost function for any centroid-based quantizer using entropy arguments that do not require high-rate approximations. We note the following [39]

- Given n bits to describe a random variable X with differential entropy $h(X)$, the error can have differential entropy no less than $h(X) - n$.
- Given a random variable X with differential entropy $h(X)$, the lowest possible variance of X is $\frac{1}{2\pi e} 2^{2h(X)}$.

- *The Entropy-Power Inequality*: Given two independent random variables X and Y with differential entropy $h(X)$ and $h(Y)$ respectively,

$$2^{2h(X+Y)} \geq 2^{2h(X)} + 2^{2h(Y)}.$$

- Entropy of a random variable X is no less than the entropy of X given additional information about another random variable Y.

At time step $k = 0$, the entropy is simply $h(x(0))$, thus the entropy of $\Delta(0)$ is at least $h(x(0)) - n$. At time step $k = 1$, we have

$$h(x(1)) \geq h\left(x(1)|\hat{x}(0)\right) = h\left(ax(0) + w(0)|\hat{x}(0)\right).$$

Now, $x(0)$ and $w(0)$ are independent (even given $\hat{x}(0)$). Denote the entropy of the noise by $h(w)$. Then,

$$\begin{aligned} 2^{2h(x(1))} &\geq 2^{2h(ax(0)|\hat{x}(0))} + 2^{2h(w|\hat{x}(0))} \\ &= 2^{2\log(a)+2h(x(0)|\hat{x}(0))} + 2^{2h(w)} \\ &\geq 2^{2\log(a)} 2^{2h(x(0))-2n} + 2^{2h(w)}. \end{aligned}$$

Let $c = a^2 2^{-2n}$. Thus, we obtain

$$\begin{aligned} h(x(1)) &\geq \frac{1}{2}\log\left[c2^{2h(x(0))} + 2^{2h(w)}\right] \\ \Rightarrow h(\Delta(1)) &\geq \frac{1}{2}\log\left[c2^{2h(x(0))} + 2^{2h(w)}\right] - n. \end{aligned}$$

In a similar manner, we can prove in general that

$$h(\Delta(k)) \geq \frac{1}{2}\log\left[c^k 2^{2h(x(0))} + \sum_{j=0}^{k-1} c^j 2^{2h(w)}\right] - n.$$

Finally, the error variance at time step k is bounded by

$$E[\Delta^2(k)] \geq \frac{1}{2\pi e} 2^{-2n}\left[c^k 2^{2h(x(0))} + \sum_{j=0}^{k-1} c^j 2^{2h(w)}\right].$$

Thus, we can evaluate the lower bound on cost function as

$$J_T \geq \sum_{k=0}^{T}\Bigg[-rf^2\frac{1}{2\pi e}2^{-2n}\left(c^k 2^{2h(x(0))}+\sum_{j=0}^{k-1}c^j 2^{2h(w)}\right)+(q+rf^2)(a+f)^{2k}E[x^2(0)]$$
$$-(f^2+2af)(q+rf^2)\frac{1}{2\pi e}2^{-2n}\sum_{j=0}^{k-1}(a+f)^{2j}\left(2^{2h(x(0))}c^{k-j-1}+\sum_{i=0}^{k-j-2}2^{2h(w)}c^i\right)$$
$$+\sigma^2(q+rf^2)\sum_{j=0}^{k-1}(a+f)^{2j}\Bigg].$$

Further, if we assume

$$(a+f)^2 \leq 1, \qquad a^2 2^{-2n} \leq 1, \tag{3.43}$$

we obtain

$$J_\infty \geq -rf^2\frac{1}{2\pi e}2^{-2n}2^{2h(w)}\frac{1}{1-c}+\frac{\sigma^2(q+rf^2)}{1-(a+f)^2}$$
$$-(f^2+2af)(q+rf^2)\frac{1}{2\pi e}2^{2(h(w)-n)}\frac{1}{1-(a+f)^2}\frac{1}{1-c}.$$

We do not yet have an analytic expression for the tightness of the bound. Note that the condition given in (3.43) is similar to the condition obtained for stability of a scalar unstable system in, e.g., [185]. Also, since we are interested in quantizing the current state only, we escape the complexity of having to define terms like average conditional entropy power as in [150].

So far, we have assumed that the quantization is not followed by any noiseless coding. Moreover, we have concentrated on the case of fixed rate quantization. Thus, we defined the rate of the quantizer as $\log(N)$, where N is the number of quantization levels. If we assume that noiseless coding is permitted, it makes more sense to consider the entropy of the output vector as a measure of the rate. In such a case, we note the following result.

Theorem 3.16 (From [73]) *The constrained entropy high rate quantizer bound is given by*

$$\bar{d}_L \geq \frac{1}{12}e^{-2\left(H(q(X))-h(X)\right)},$$

where $h(X)$ is the differential entropy of the random variable X while $H(q(X))$ is the entropy of quantized variable $q(X)$. Furthermore, equality is achieved if and only if the asymptotic quantizer density $\lambda(x)$ is a constant, that is, the quantizer reproduction vectors are uniformly distributed over some set having probability 1. Thus, the bound is achieved by high rate lattice vector quantizers since they have a uniform density of quantization levels.

If we define rate as $R = H(q(X))$ (which gives the average codeword length achievable using noiseless coding and hence the average rate), we obtain the above bound. For a fixed R, it can be proved that this distortion is lower than the one achieved for a given code rate. However, actually achieving this rate might require long codewords and hence might not be practical in a real-time system. There also have been some works in the information theory literature, e.g. [177], that provide a way to bound the output entropy of a quantizer given the number of levels of the quantizer. Such works may provide an interesting way to achieve a trade-off between the two notions of rate.

Dynamic Quantization It is apparent that only the region corresponding to the uncertainty that the decoder has about $x(k)$ needs to be quantized and the information sent. We now consider this case of dynamic quantization in which the number of quantization levels N remains fixed; however, the range over which quantization is being done varies with time. Thus, the meaning of the bits $\{b(k)\}$ changes as time index k progresses. This is similar to schemes like prediction based encoding outlined in [98] and yields better performance, at the cost of added complexity due to a time-varying quantizer. Moreover, it assumes some level of synchronization between the encoder and the decoder so that both agree on the specific quantizer to which the bits at time k pertain.

For simplicity, we consider only the infinite-time horizon cost function J_∞. For the case of a uniform quantizer with N levels, the quantization step size at time k is given by

$$N\delta(k) = l(k) = a\delta(k-1) + 2M.$$

The variance of the quantization error at time k, $E[\Delta^2(k)]$ can be evaluated as before to be $\frac{\delta^2(k)}{12}$. Thus,

$$J_\infty = \frac{q + rf^2}{1 - (a+f)^2}\left(\sigma^2 + \frac{f^2M^2}{3(N^2 - a^2)}\right) + \frac{rf^2M^2}{3(N^2 - a^2)}.$$

The conditions for existence of J_∞ are

$$(a+f)^2 \leq 1, \qquad \frac{a}{N} < 1.$$

Also, note that this cost is equivalent to that of a static uniform quantizer with step size $\delta = \frac{4M^2}{N^2 - a^2}$. Since the cost function for a static uniform quantizer is an increasing function in the step size δ, this gives us a relation between the parameters M, N and a for determining which of the two quantizers, static or dynamic, is better.

Stochastic Packet Drops So far, we have assumed a perfect noiseless digital channel, in that the bits $\{b(k)\}$ were transmitted to the decoder without fail. We can also incorporate the effect of stochastic data loss. Thus, at any time step, with a probability p the packet containing the bits $\{b(k)\}$ is erased. The results we obtain can be extended to packet drops happening according to a

Markov chain as well. Note that the expectation in the cost function (3.42) is now also over the probability of packet drops at each time step. For simplicity, we consider only the infinite horizon cost J_∞ for the case of a midpoint-based uniform quantizer. In this case, the state evolves according to the equation

$$x(k+1) = \begin{cases} (a+f)x(k) + f\Delta(k) + w(k) & \text{with prob } 1-p \\ ax(k) + w(k) & \text{with prob } p. \end{cases} \tag{3.44}$$

For a uniform quantizer, $E[\Delta^2(k)] = \frac{\delta^2}{12}$ while $\Delta(k)$ and $w(k)$ are independent of each other. Thus, we can evaluate the steady-state covariance as

$$P_\infty = \frac{(1-p)f^2\frac{\delta^2}{12} + \sigma^2}{1-(a+f)^2(1-p) - a^2p}.$$

Since the cost function is given by

$$J_\infty = (q + rf^2)P_\infty + rf^2\frac{\delta^2}{12},$$

it can be easily evaluated. For convergence, we have the additional condition

$$(1-p)(a+f)^2 + pa^2 < 1.$$

Examples We now consider some simple examples to illustrate the above results. We consider the system parameter $a = 2$. The initial condition $x(0)$ is assumed to be uniformly distributed in the range $[-20, 20]$ while the white noise $w(k)$ is assumed to be uniformly distributed in the range $[-1, 1]$. The cost function we consider is

$$J_\infty = \lim_{k\to\infty} E\left[x^2(k) + u^2(k)\right].$$

For this cost function, the optimal control law without quantization turns out to be $f = -1.618$. We use this control law to consider the performance of various quantization schemes considered above. For the quantizers that operate on a fixed range, the minimum region to be quantized is $[-20, 20]$. We will assume that the control law does not allow the system to go outside this range, thus avoiding quantizer overflow.

Figures 3.22 and 3.23 show a comparison of our theoretical approximations with simulation results for uniform and logarithmic quantizers respectively. The simulation results refer to the cost in steady state averaged over 10000 runs for a system using the particular quantizer. The initial condition and the noise driving the system were chosen randomly for each run. It can be seen that

the approximations are quite good, at least in this example. Also, even for this simple system, logarithmic quantizer yields much better performance for the same number of bits. However, for convergence, the uniform quantizer requires 2 or more bits while for $\epsilon = 0.1$, the necessary and sufficient conditions for convergence for the logarithmic quantizer require 3 and 4 bits respectively. Figure 3.24 shows a comparison of the performance achieved by the dynamic uniform quantizer with the performance bound derived using entropy arguments. We see that the bound is reasonably tight in this example. Of course, static quantizers perform much worse than the dynamic quantizers, especially when a small number of bits are used. Figure 3.25 shows the performance of the system as a function of the packet loss probability across a channel that drops packets in an i.i.d. fashion. A uniform quantizer with 6 bits is used. The system becomes unstable at the theoretical value of $p = 0.22$.

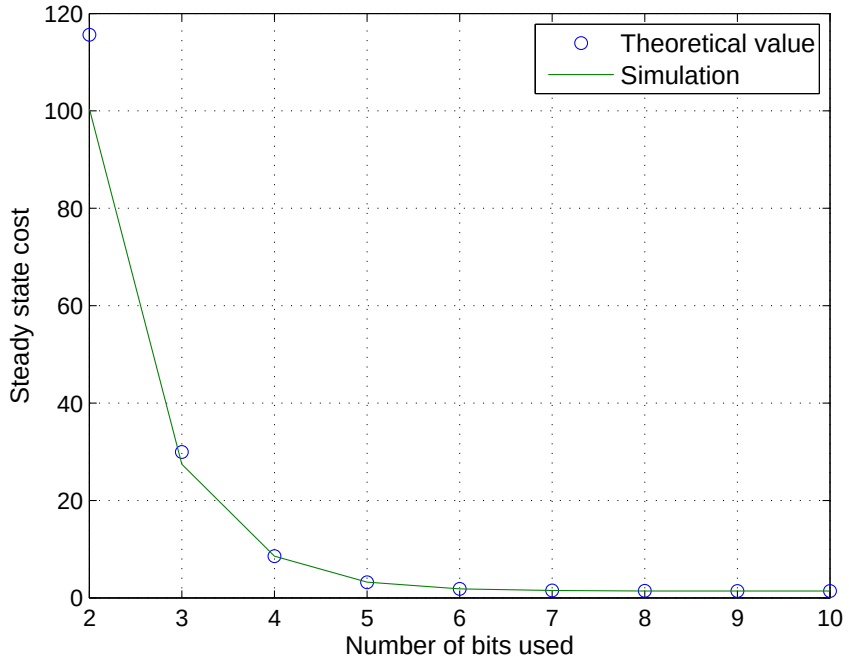


Figure 3.22: Comparison of performance of approximations for uniform quantizer calculated using our approach with simulation results.

Thus, we see that quantization can impact the performance of a closed loop system quite a lot. We have considered the problem of evaluating performance for a given control law under the high rate regime. Clearly, there are many interesting directions in which this work can be extended. So far, we have only considered scalar processes. A more general case is when the process state $x(k)$ and measurement vector $y(k)$ are vectors. Studying quantization issues for such plants takes us into the realm of vector quantization theory, which is less well-developed than its scalar counterpart and hence the extension is not trivial. The basic difficulty is that each component of the vector

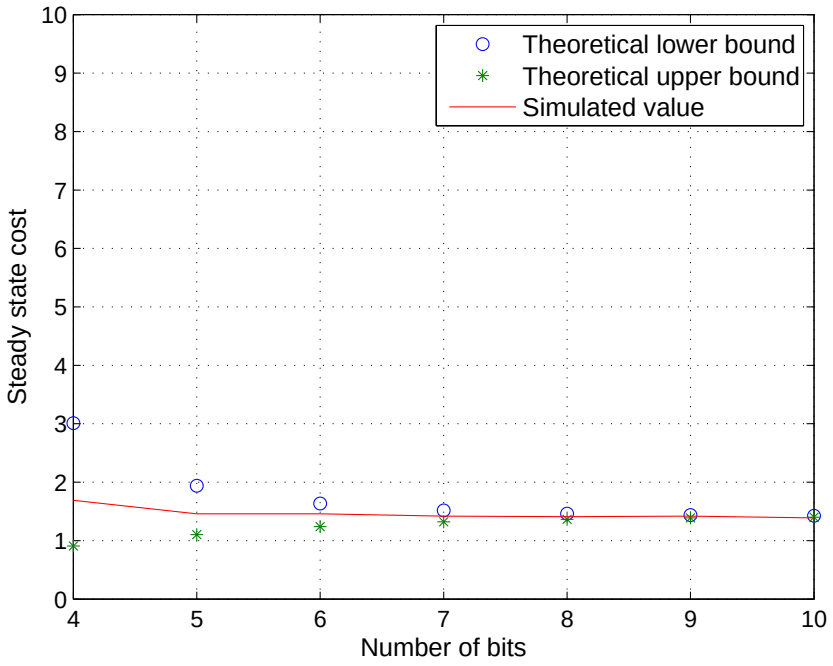


Figure 3.23: Comparison of performance of approximations presented in the text for logarithmic quantizer with simulation results.

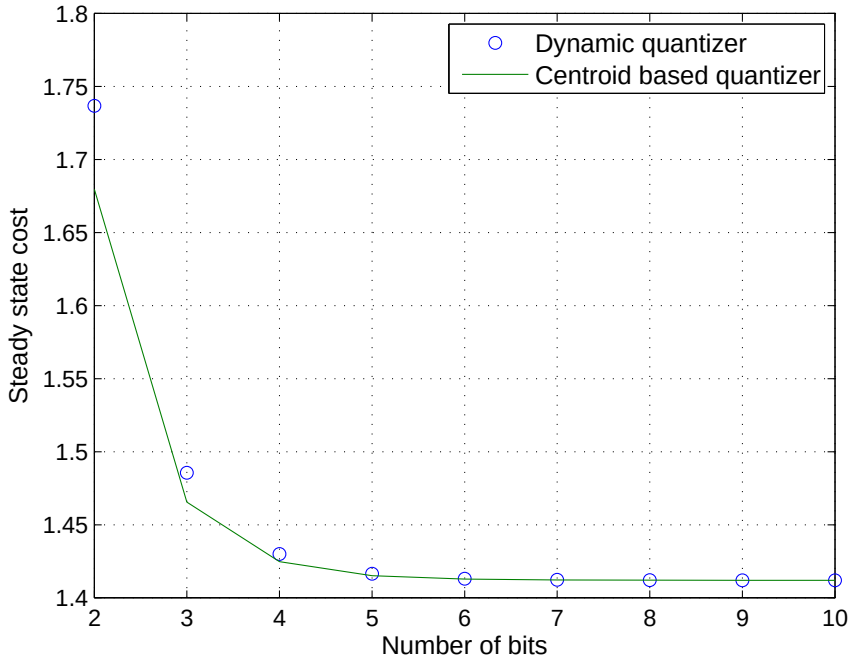


Figure 3.24: Comparison of the performance of the dynamic quantizer with the lower bound.

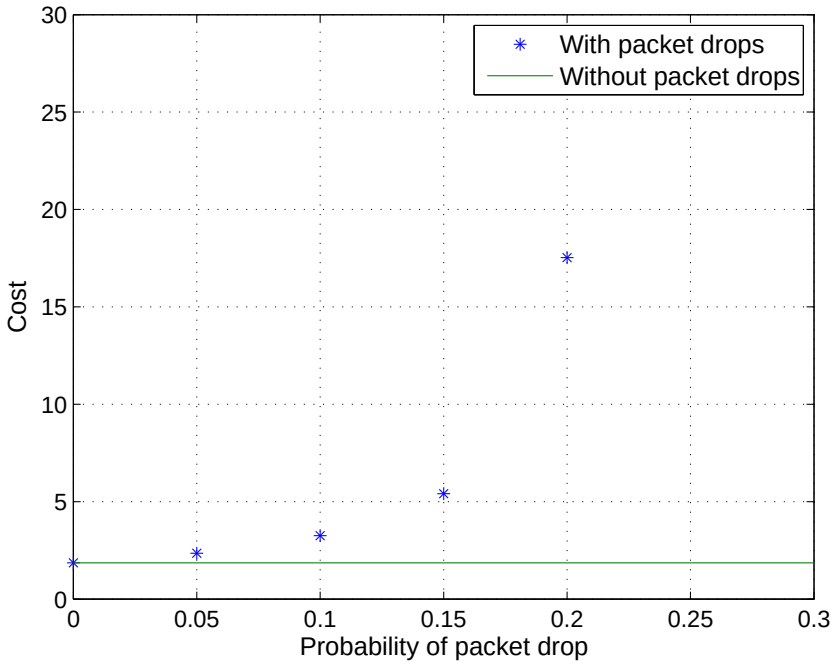


Figure 3.25: Performance of the system with a uniform quantizer across a packet dropping channel.

$y(k)$ carries information about other components and hence it is extremely wasteful to do scalar quantization on each component separately. If the system matrix A is diagonal (or diagonalizable) and the matrix C is invertible, and hence this dependence is not present, the results from scalar quantization that we derive above can be used on each component. Another important question is designing an optimal controller in presence of quantization.

Appendix B: Control of Jump Linear Markov Systems with Markov State Estimation

So far in the chapter, we assumed that the receiver knew whether or not the packet was lost and what the delay value, if any, was. In this appendix, we loosen this assumption. We consider a jump linear Markov system being stabilized using a linear controller with the Markov state associated with the probability distribution of a measured variable. We assume that the Markov state is not known, but is being estimated based on the observations of the variable.

Jump linear Markov systems are of independent interest, e.g., in fault-prone dynamic systems [29], tracking [154] and so on. We are interested in them primarily because they can be used to model the presence of a communication channel that exists in many states. We saw some such examples in the main part of the chapter. To consider another example, consider the system shown in Figure 3.26. The figure represents a system in which the sensor and the controller communicate over a medium which introduces random delays. The medium can be a bus shared with other devices, or a network where routing protocols introduce random delays, or a wireless channel in which protocols like Bluetooth introduce random latency delays before succesful transmission. If the delays being introduced can be modeled by a Markov chain [152], analysis techniques for jump linear Markov systems apply immediately.

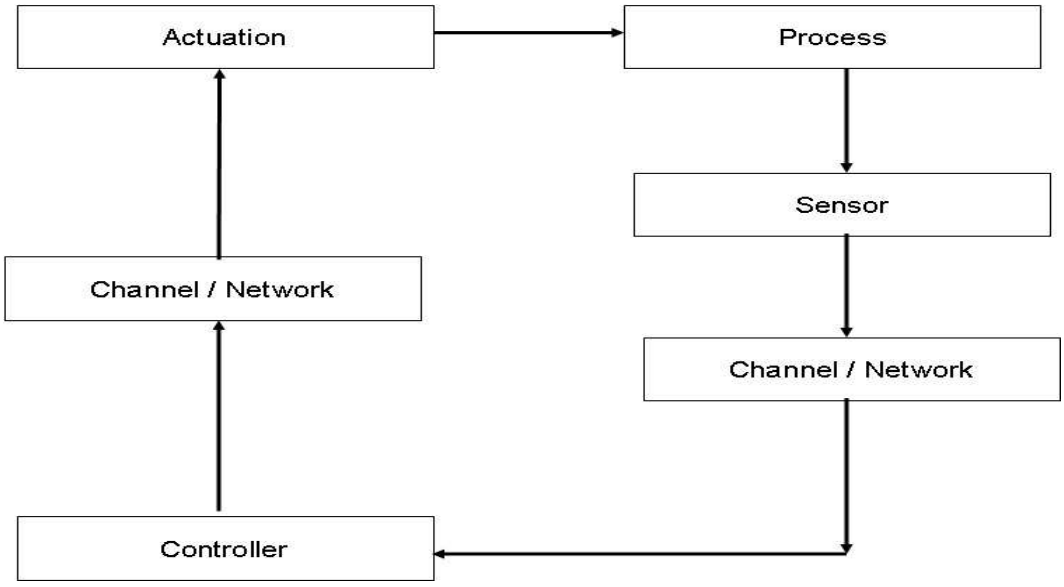


Figure 3.26: A general system in which the sensor and the controller utilize an imperfect communication channel or a network to communicate.

As stated in Section 3.1, most of the analysis techniques proposed in the literature for such systems assume the Markov state to be known. In a more practical scenario, a Markov state estimation algorithm would be used. We consider a class of Markov state estimation algorithms in which the state estimate update depends only on the latest observation value. Consider the system

described by

$$\begin{aligned} x(k+1) &= \bar{A}(r(k))x(k) + B(r(k))u(k) + w(k) \\ y(k) &= C(r(k))x(k) \\ u(k) &= F(r(k))y(k), \end{aligned}$$

where $x(k) \in \mathbf{R}^n$ is assumed to be the process state, $u(k) \in \mathbf{R}^m$ is the control input which is a linear function of the state observation variable $y(k)$, $w(k)$ is white noise with zero mean and covariance R_n and $v(k)$ is white noise with zero mean and covariance R_v. Let Q denote the transition probability matrix of the Markov chain whose states are represented by $r(k) = i \in \{1, 2, ..., s\}$. Such a system might arise, e.g., from discretization of an underlying continuous-time system, which can possibly have time-delays less than the sampling time period present in it (see [152] for details). The states of the Markov chain define the probability distribution of the various variables (e.g., the time delay) in the system. For simplicity of presentation, we will assume that there is only one variable being affected by the Markov chain. We denote the observed value of this variable at time step k by $o(k)$ and its probability density function given that the Markov state at time k is j by $f_{o(k)|r(k)=j}(t|r(k) = j)$. If the controller has the knowledge of the Markov state, it can vary the control law matrix F according to the Markov state $r(k)$. However, if the Markov state is not known, it has to depend on the estimate of the Markov state $\hat{r}(k)$. Thus, F becomes a function of $\hat{r}(k)$. Thus, we see that the system can be written as

$$x(k+1) = A(r(k), \hat{r}(k))x(k) + w(k) \tag{3.45}$$

where

$$A(r(k), \hat{r}(k)) = \bar{A}(r(k)) + B(r(k))F(\hat{r}(k))C(r(k)).$$

One central assumption we make is that the variable value being affected by the Markov chain is measurable and the value taken by the variable in every time step is available accurately to the controller. Thus, if the variable is time delay, we use measures such as time stamping of the packets; if it is the system matrix $\bar{A}$ — which can be used to model, say, the effect of varying SNR in a communication channel over which $y(k)$ is passed to the controller — then pilot tone measurements are used, and so on.

Hidden Markov Models and the Viterbi Algorithm

Consider a Markov chain with a finite number of states and given transition probability matrix $Q = [q_{ij}]$. Suppose that when a transition occurs, we cannot observe the states corresponding to the transition directly. Rather we obtain an observation related to the transition. We are given the probability $\text{Prob}(o, i)$ of the observed value of the variable being affected by the Markov state being

o when the Markov state is i. Such Markov chains are called hidden Markov models and the problem of estimating the state from the observation sequence is called the state estimation problem.

Consider the observation sequence $O_T = \{o(1), o(2), \cdots, o(T)\}$. We wish to estimate the state transition sequence $\hat{R}_T = \{\hat{r}(1), \hat{r}(2), \cdots, \hat{r}(T)\}$ that maximizes over all R_T=$\{r(1),\ r(2),\ \cdots,\ r(T)\}$ the conditional probability $\text{Prob}(R_T|O_T)$. Let the probability distribution of the states at time 0 be denoted by $\pi_{r(0)}$. It is well-established, see e.g. [17], that the optimal estimate is given by the Viterbi algorithm, which is a solution to the problem of minimizing $-\ln(\pi_{r(0)}) - \sum_{k=1}^{T} \ln(\text{Prob}(o(k), r(k)|r(k-1)))$ over all possible sequences $\{r(1), r(2), \cdots, r(T)\}$.

However, the Viterbi algorithm is non-causal in that it requires all the observations before estimating the state sequence. In practice, a causal version of the algorithm is used which is based on forward dynamic programming. If we know the smallest costs $D(k, r(k))$ from the beginning to all the states $r(k)$ on the basis of the observation sequence O_k, we compute the smallest costs $D(k+1, r(k+1))$ by using the recursion

$$D(k+1, r(k+1)) = \min[D(k, r(k)) - \ln(\text{Prob}(o(k+1), r(k+1)|r(k)))], \tag{3.46}$$

where the minimization is taken over all $r(k)$ such that $q_{r(k)r(k+1)}$ is non-zero. An advantage of this procedure is that it can be executed in real time, as soon as each new observation is obtained. Another simplification can be made if the state estimate update is made only on the basis of the current measurement. This amounts to ignoring the first term on the right hand side of equation (3.46) and minimizing the second term over all transitions from the current state estimate $\hat{r}_k$. This algorithm is refered to as the one-step Viterbi algorithm.

Stability Results

Once again, we consider the stability in the bounded second order moment sense. Thus, we call a system stable if the covariance of the system in (3.45) does not diverge. We define the conditional covariance as

$$P_{jn}(k+1) \quad = \quad E[x(k+1)x^T(k+1)|r(k+1) = j, \hat{r}(k+1) = n]$$

The expectation is taken over all the uncertainty in the system at time step k, which is due to the initial state, noise and the Markov states at previous times. We define

$$\tilde{P}_{jn}(k+1) = P_{jn}(k+1) \times \text{Prob}(r(k+1) = j, \hat{r}(k+1) = n). \tag{3.47}$$

The state covariance $P(k)$ is then given by

$$P(k) = \sum_{i=1}^{s} \sum_{j=1}^{s} \tilde{P}_{ij}(k). \tag{3.48}$$

Let R_{ij} denote the matrix $\{r_{mn|ij}\}$ where $r_{mn|ij}$ refers to the probability that $(\hat{r}(k+1) = n | \hat{r}(k) = m, r(k) = i, r(k+1) = j)$. Finally, let

$$A_{i,m} = E[A(r(k), \hat{r}(k)) \otimes A(r(k), \hat{r}(k)) | r(k) = i, \hat{r}(k) = m]. \tag{3.49}$$

Define the matrix $\text{diag}(A_{i,m})$ as a block diagonal matrix with the blocks $A_{1,1}$, $A_{1,2}$, $\cdots$, $A_{1,s}$, $A_{2,1}$, $A_{2,2}$, $\cdots$, $A_{2,s}$, $\cdots$, $A_{s,1}$, $\cdots$, $A_{s,s}$ along the diagonal and zeros elsewhere. Denote by Σ the matrix having the following structure

$$\Sigma = \begin{bmatrix} q_{11}R_{11} & q_{21}R_{21} & \cdots & q_{s1}R_{s1} \\ q_{12}R_{12} & q_{22}R_{22} & \cdots & q_{s2}R_{s2} \\ \vdots & \vdots & \vdots & \vdots \\ q_{1s}R_{1s} & q_{2s}R_{2s} & \cdots & q_{ss}R_{ss} \end{bmatrix}. \tag{3.50}$$

We have the following result.

Proposition 3.17 *Consider the system given in (3.45) and a one-step Markov state estimation algorithm. The system is stable, in the bounded covariance sense, iff the matrix* $(\Sigma \otimes I)diag(A_{im})$ *has all its eigenvalues in the unit circle.*

Proof We assume that the additive noise term in system of (3.45) is bounded in the mean square sense. Thus, it would have no effect on stability and we only need to consider an equation of the form

$$x(k+1) = A(r(k), \hat{r}(k))x(k). \tag{3.51}$$

For this system, from (3.47),

$$\tilde{P}_{jn}(k+1) = E[A(r(k), \hat{r}(k))x(k)x^T(k)A(r(k), \hat{r}(k)) | r(k+1) = j, \hat{r}(k+1) = n] \times \text{Prob}(r(k+1) = j, \hat{r}(k+1) = n).$$

Because of the definition of Markov chain and the assumption that the state estimate update would

involve only the observations obtained in time step $k+1$,

$$\begin{aligned}\text{Prob}(r(k+1)=j,\hat{r}(k+1)=n|f(x(k),r(k),\hat{r}(k)),r(k)=i,\hat{r}(k)=m)\\ =\text{Prob}(r(k+1)=j,\hat{r}(k+1)=n|r(k)=i,\hat{r}(k)=m),\end{aligned}$$

where $f(.)$ is any deterministic function of its arguments. Thus, any function involving $x(k)$, $\hat{r}(k)$ and $r(k)$ only is independent of the event $(r(k+1)=j,\hat{r}(k+1)=n)$ given the values of $r(k)$ and $\hat{r}(k)$. Now we use the fact that if C and A are independent given B,

$$E[C|A]=\sum_B \frac{\text{Prob}(A|B)\text{Prob}(B)E[C|B]}{\text{Prob}(A)}. \tag{3.52}$$

Thus, in particular, if we define

$$\begin{aligned}C &= x(k+1)x^T(k+1)\\ A &= (r(k+1)=j,\hat{r}(k+1)=n)\\ B &= (r(k)=i,\hat{r}(k)=m),\end{aligned}$$

we obtain

$$\begin{aligned}\tilde{P}_{jn}(k+1)=\sum_{i=1}^{s}\sum_{m=1}^{s}\text{Prob}(i\to j,m\to n)\times\text{Prob}(r(k)=i,\hat{r}(k)=m)\\ \times E[A(r(k),\hat{r}(k))x(k)x^T(k)A^T(r(k),\hat{r}(k))|r(k)=i,\hat{r}(k)=m].\end{aligned} \tag{3.53}$$

In the above, $\text{Prob}(i\to j,m\to n)$ represents the probability that the true Markov state goes from i in step k to j in step $k+1$ and the estimated state goes from m to n at the same time. Now we observe that

$$\begin{aligned}E[f(r(k),\hat{r}(k))g(x(k))|r(k)=i,\hat{r}(k)=m]=\\ E[f(r(k),\hat{r}(k))|r(k)=i,\hat{r}(k)=m]\times E[g(x(k))|r(k)=i,\hat{r}(k)=m],\end{aligned}$$

for any functions $f(.)$ and $g(.)$. This can be proved by considering the fact that the variable distribution depends only on the Markov state and from the equivalent condition

$$\text{Prob}(r(k),\hat{r}(k)|x(k),r(k)=i,\hat{r}(k)=m)=\text{Prob}(r(k),\hat{r}(k)|r(k)=i,\hat{r}(k)=m).$$

Thus, we can vectorize equation(3.53) to obtain

$$\text{vec}(\tilde{P}_j(k+1)) = \sum_{i,m} \text{Prob}(i \to j, m \to n) A_{i,m} \text{vec}(\tilde{P}_i(k)),$$

where $A_{i,m}$ has been defined in (3.49). This in turn yields

$$\tilde{P}(k+1) = (\Sigma \otimes I)\text{diag}(A_{i,m})\tilde{P}(k), \tag{3.54}$$

where Σ is defined in (3.50). It is apparent from the above equation that the stability of the system is given by the stability of the matrix $(\Sigma \otimes I)\text{diag}(A_{i,m})$. ∎

To compute $r_{mn|ij}$, we condition it on the value of the underlying variable varying according to the Markov chain:

$$r_{mn|ij} = \int \text{Prob}(\hat{r}(k+1) = n | r(k+1) = j, \hat{r}(k) = m, o(k+1) = t) \times$$
$$f_{o(k+1)|r(k+1)=j}(t | r(k+1) = j) dt. \tag{3.55}$$

Both the terms in the above expression are computable. The second term is known since we know the distributions of the variable in each Markov state. The first term is computable for any particular estimation algorithm satisfying the assumption stated in the theorem. For the one-step Viterbi algorithm, it computes to the probability that the cost function for state n is least among all possible states, where the cost function is given by

$$D(k+1, \hat{r}(k+1) = n | \hat{r}(k) = m, o(k+1) = t) = -\ln(q_{mn}) - \ln(f_{o(k+1)|r(k+1)=n}(t | r(k+1) = n)).$$

Examples

We consider a few examples in this subsection to illustrate the result.

Example 1

Consider the case when the estimation algorithm always gives correct results, i.e., the measurement of the variable tells us the state of the Markov chain. Then, the matrix Σ has the form

$$\left[\begin{array}{c|c|c|c} \sigma^1 & \sigma^2 & \cdots & \sigma^s \end{array} \right],$$

where the block σ^i has dimensions $s^2 \times s$ and has the structure

$$\sigma^i = \left[\begin{array}{ccc} q_{i1} & q_{i1} & \cdots \\ 0 & 0 & \cdots \\ \vdots & s \text{ rows of zeroes} & \\ \hline q_{i2} & q_{i2} & \cdots \\ 0 & 0 & \cdots \\ \vdots & s \text{ rows of zeroes} & \\ \hline \vdots & & \\ \hline q_{is} & q_{is} & \cdots \end{array}\right].$$

Thus, the final matrix $(\Sigma \otimes I)\text{diag}(A_{i,m})$ has the same structure, with each element in the above matrix replaced by a $n^2 \times n^2$ block. We note that there are a total of s^2 row blocks in the matrix, each consisting of n^2 rows. However, only the blocks 1, $s+2$, $2s+3$, ..., s^2 are non-zero. Now a $m \times m$ matrix with ith row zero has the same eigenvalues as the $(m-1) \times (m-1)$ matrix formed by removing the ith row and ith coloumn from the original matrix, except for an additional zero eigenvalue. Also, the matrix $A_{i,m}$ is the same as A_i defined in [152], when i and m are the same. Thus, the eigenvalues of the matrix $(\Sigma \otimes I)\text{diag}(A_{i,m})$ are the same as the eigenvalues of the matrix $(Q^T \otimes I)\text{diag}(A_i)$ except for some additional zero eigenvalues. Thus, our results reduce to the the results of [152] in this case, as they should.

Example 2

It is not necessary that if a process is stable when a controller based on known Markov state is used, it will be stable when the same controller is instead fed the states estimated by the one-step Viterbi algorithm. Consider the discrete time version of the system

$$\begin{aligned} \ddot{x}(t) &= u(t) \\ u(t) &= Fx(t-\tau). \end{aligned}$$

Let the time step be h=0.1. Let the Markov states be characterized by different time delays τ in passing of the sensor signal to the controller, the state 1 having a time delay uniformly distributed between 0 and $0.7h$, while the state 2 having a time delay uniformly distributed between $0.69h$ and $0.71h$. Thus, the equivalent discrete-time system is characterized by the equations

$$\begin{aligned} x(k+1) &= \phi x(k) + \gamma_0 u(k) + \gamma_1 u(k-1) \\ u(k) &= Fx(k), \end{aligned}$$

where

$$\phi = e^{Ah} \qquad A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$\gamma_0 = \int_0^{h-\tau} e^{As} ds B \qquad \gamma_1 = \int_{h-\tau}^{h} e^{As} ds B.$$

Let the control laws in the two Markov states be given by $F_1 = [-0.2 - 0.1]$ in the state 1 and by $F_2 = [0.5 - 0.2]$ in state 2. Let the transition probability matrix be

$$Q = \begin{bmatrix} 0.6 & 0.4 \\ 0.8 & 0.2 \end{bmatrix}.$$

Then, the matrix Σ is given by

$$\Sigma = \begin{bmatrix} 0.5914 & 0.5914 & 0.7886 & 0.7886 \\ 0.0086 & 0.0086 & 0.0114 & 0.0114 \\ 0 & 0 & 0 & 0 \\ 0.4 & 0.4 & 0.2 & 0.2 \end{bmatrix}.$$

When we obtain the eigenvalues of the 36×36 matrix $(\Sigma \otimes I)\text{diag}(A_{i,m})$, we obtain an eigenvalue outside the unit circle. On the other hand, the eigenvalues of the 18×18 matrix $(Q^T \otimes I)\text{diag}(A_i)$ are all inside the unit circle, with the highest absolute value being 0.9971. If we simulate the systems, we indeed find that the system is stable if Markov state is known. However, it goes unstable if the same controller is used but the one-step Viterbi algorithm is used to estimate the state. Thus, a separation property does not hold between the Markov state estimate and stability of the system.

Comment We have given necessary and sufficient conditions for stability of a jump linear Markov state when Markov state is being estimated. Stability considered is the asymptotic stability of the conditional covariances. However, this might be too strong a condition. "Almost Sure stability" might provide a better estimate of stability; however, the transient performance of the process might be unacceptable in this case. The relation between the various forms of stability is discussed in [61]. Also, note that the result can easily be extended to the case of two or more independent Markov chains modeling many separate communication links.

Optimal Controller

Consider the quadratic cost function

$$J_T = x^T(T+1)P^c(T+1)x(T+1) + E(\sum_{k=0}^{T}\left[x^T(k)Qx(k) + u^T(k)Ru(k)\right], \tag{3.56}$$

where Q, R and $P^c(T+1)$ are positive-definite matrices.

Optimal State Feedback

We begin by assuming that the full state information about $x(k)$ is available to the controller. Then, similar to [152], we have the following result.

Proposition 3.18 *Consider the problem of minimizing the cost function given by equation (3.56) for the system given by equation (3.45) with full state information about $x(k)$ available to the controller and when the one-step Viterbi algorithm is used. Assume that the Markov chain reaches a stationary state. The optimal control law is given by*

$$u(k)^* = -L(k, o(k), \hat{r}(k)) \begin{bmatrix} x(k) \\ u(k-1)^* \end{bmatrix}$$

where for $\hat{r}(k) = i \in \{1, ..., s\}$, we have

$$\begin{aligned}
L(k, o(k), i) &= (R + \tilde{S}_i^{22}(k+1))^{-1} \times \tilde{S}_i^{21}(k+1)\tilde{S}_i^{23}(k+1) \\
\tilde{S}_i(k+1) &= G^T \sum_{j=1}^{s} r_{ij} S_j(k+1) G \\
G &= \begin{bmatrix} A(o(k), r(k), \hat{r}(k)) & B(o(k), r(k), \hat{r}(k)) \\ 0 & I \end{bmatrix} \\
S_i(k) &= E\left[F_2^T(\hat{r}(k)\tilde{S}_i(k+1)F_2(\hat{r}(k)) + F_1^T(\hat{r}(k))\Pi F_1(\hat{r}(k))|\ \hat{r}(k) = i \right] \\
F_1(\hat{r}_k) &= \begin{bmatrix} I & 0 \\ \leftarrow -L(k, o(k), \hat{r}(k)) \rightarrow & \end{bmatrix} \\
F_2(\hat{r}(k)) &= \begin{bmatrix} I & 0 \\ \leftarrow -L(k, o(k), \hat{r}(k)) \rightarrow & \end{bmatrix} \\
S_i(N) &= \begin{bmatrix} P^c(T+1) & 0 \\ 0 & 0 \end{bmatrix} \\
\Pi &= \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix}.
\end{aligned}$$

The elements r_{ij} refer to the probability of the state estimate changing from i in time step k to j in the time step $k+1$. Note that the elements r_{ij} are in general a function of $o(k)$. $\tilde{S}_i^{ab}(k)$ is the block (a,b) of the symmetric matrix $\tilde{S}_i(k)$.

Proof The proof is similar to that of the problem treated in [152] and is omitted for the sake of brevity. ∎

We now give a method to calculate the probability term r_{ij} for the case when one-step causal Viterbi algorithm is being used. We need to compute

$$\text{Prob}(\hat{r}(k+1) = j | \hat{r}(k) = i, o(k) = t). \tag{3.57}$$

Note that the information vector available at the time of making the decision includes the estimated states until that time step as well as all the time delays. Let us condition on the probability of the actual Markov state at time k being l. Thus, we obtain

$$\text{Prob}(\hat{r}(k+1) = j | \hat{r}(k) = i, o(k) = t) = \sum_{l=1}^{s} P(\hat{r}(k+1) = j | \hat{r}(k) = i, o(k) = t, r(k) = l) \times \text{Prob}(r(k) = l | \hat{r}(k) = i, o(k) = t).$$

To calculate the first term on the right hand side, let us condition it on the probability of the next Markov state being m.

$$\text{Prob}(\hat{r}(k+1) = j | \hat{r}(k) = i, o(k) = t, r(k) = l) = \sum_{m=1}^{s} \text{Prob}(\hat{r}(k+1) = j | \hat{r}(k) = i, o(k) = t, r(k) = l, r(k+1) = m) \times \text{Prob}(r(k+1) = m | \hat{r}(k) = i, o(k) = t, r(k) = l).$$

Now we can evaluate all the terms. The term $\text{Prob}(\hat{r}(k+1) = j | \hat{r}(k) = i, o(k) = t, r(k) = l, r(k+1) = m)$ is simply $\text{Prob}(\hat{r}(k+1) = j | \hat{r}(k) = i, r(k) = l, r(k+1) = m)$ which was evaluated in the proof of Proposition 3.17. The term $\text{Prob}(r(k+1) = m | \hat{r}(k) = i, o(k) = t, r(k) = l)$ is simply $\text{Prob}(r(k+1) = m | r(k) = l)$ by the Markov property. To evaluate the term $\text{Prob}(r(k) = l | \hat{r}(k) = i, o(k) = t)$, note that this is the same as $\text{Prob}(r(k) = l | o(k) = t)$. To prove this, consider the equivalent condition

$$f(o(k) = t | r(k) = l, \hat{r}(k) = i) = f(o(k) = t | r(k) = l).$$

In the above equation, f refers to the probability distribution function of $o(k)$. Now $\text{Prob}(r(k) = l|o(k) = t)$ can be evaluated by Baye's rule

$$\text{Prob}(r(k) = l|o(k) = t) = \frac{f(o(k) = t|r(k) = l)\text{Prob}(r(k) = l)}{\sum_u f(o(k) = t|r(k) = u)\text{Prob}(r(k) = u)}.$$

Finally, $\text{Prob}(r(k) = u)$ can be evaluated from the stationary probabilities of the Markov chain. Thus, we can evaluate the terms r_{ij}.

It is obvious from the form of the optimal control law that no separation property holds between a controller implementing the optimal control law based on known Markov state and a Markov state estimation algorithm. In particular, even if we use a causal one-step Viterbi algorithm and feed the state derived from it into the controller which implements the optimal control law based on known Markov state, we would not obtain the lowest cost achievable with the one-step Viterbi state estimation algorithm.

Note that the form of the optimal controller derived above is similar to the controller for the case of Markov state known, as derived in [152], except for the variables on which to condition while taking the expectation. Thus, we can go ahead and derive the optimal process state estimate and show that a separation property holds between the optimal controller and the optimal process state estimate in a manner similar to that given in the above reference.

We have thus considered jump linear Markov systems in which the Markov state is not known and is being estimated for a class of estimation algorithm. Interesting properties result, e.g., a control law depending on the knowledge of the exact Markov state may no longer stabilize the system when we feed in the state estimate instead of the state itself. Future work in this stream should focus on extending the results to the estimation algorithms which take the full history into account while updating the state estimate. Such algorithms, e.g, the causal Viterbi algorithm are optimal Markov state estimators but are more complicated to analyze. Initial results are promising and point to similar theorems as given in the paper. Another possible direction for future work might be to jointly optimize the LQR problem with the estimation algorithm to see whether the Viterbi algorithm is indeed the best state estimation algorithm in this case.

Chapter 4

Distributed Estimation and Control in Presence of Stochastically Failing Channels

In this chapter, we address some problems involving distributed estimation and control when the components are communicating over communication links. These problems can also be viewed as task scheduling or resource allocation problems. To be able to explicitly include the stochastic effects of communication channels in system design, we introduce stochastic schedules to answer the question '*Who should talk when?*'. In the chapter, we also show how the tools we develop are useful for many other problems. The work presented here has partly appeared in [34, 75, 76, 77, 109, 110]

The chapter is organized as follows. We begin with a brief literature review. In Section 4.2, we formulate the problem of estimation for a given stochastic schedule for sensors. We characterize the expected performance by presenting bounds both for the case of sensors chosen in an independent fashion as well as for sensors chosen in a Markovian fashion. In Section 4.3, we adapt the performance analysis to present a method to synthesize the schedule. We present a gradient descent algorithm and demonstrate it using some examples. In Section 4.4, we discuss the effect of encoding measurements prior to transmission along the lines of the last chapter. Then, in Section 4.5, we move on to a control problem in which the topology is switching stochastically. In the appendices, we demonstrate two applications of the tools we develop in the chapter: dynamic sensor coverage and analysis of the effect of using multiple description codes to improve the estimation performance across a packet-dropping link.

Contributions

The main contributions of the work presented in the chapter are now summarized.

1. We analyze the performance of an estimator that is receiving measurements from sensors according to a stochastic schedule. We consider sensors being chosen independently from one

time step to the next as well as according to a Markov chain. The tools that we develop are applicable in many other problems.

2. We formulate and solve the problem of designing an optimal stochastic sensor schedule. Our strategy has many advantages over traditional deterministic solutions to the problem such as being able to include the effect of stochastic packet losses by the communication links, being able to consider sensor costs and so on.

3. We present a framework and initial results on a control problem in which the information topology is switching stochastically due to communication links. This can be used to characterize the performance of the system as a function of, say, the communication frequency by the nodes.

4. We consider the problem of dynamic sensor coverage in which a group of mobile sensors have to cooperatively monitor a geographical area. We provide bounds on the minimum number of sensors required and a method to generate their trajectories.

5. We analyze the estimation performance when multiple description codes, that are used in information theory as network source codes, are used for transmitting measurements of a dynamic system across a packet dropping link. We show that with no increase in the bit rate, we can obtain improved performance.

4.1 Introduction

In the previous two chapters, we have identified and addressed the two major sources of complexity in networked estimation and control design:

1. Absence of a single decision making authority with access to all the information.

2. Presence of imperfect communication links that distort any information transmitted across them.

In this chapter, we take the first steps towards combining both of these issues. We will look at some problems that have aspects both of multiple sources generating information as well as of information transmission occurring over imperfect communication channels. In Section 3.6 we already looked at one such problem in which we identified the optimal information processing algorithms for the case when many sensors transmit information to an estimator with one sensor communicating over an imperfect link.

We begin by considering the following problem. Suppose a set of sensors is jointly trying to estimate a process. Sensors take measurements at every time step and the measurements are then

exchanged among all the sensors, or equivalently, transmitted to an estimator. However, only one (or a subset) of the sensors can communicate at any time step. This is a representative resource allocation problem and this particular situation may arise in many cases:

1. There is one physical sensor but the sensor-estimator communication link stochastically drops packets. Thus, from the point of view of the estimator, there are two sensors: the first one corresponding to the physical sensor and the second one to no measurement being taken. Only one of these is active at any time step. This case has been analyzed in [174, 178]. If partial observation loss can happen, the case has been looked at in, e.g., [134].

2. There are multiple sensors, some of which communicate over (say) wireless links and hence are subject to packet losses.

3. Multiple sensors that cannot operate simultaneously are present, e.g. echo based sensors like sonars in the same frequency band.

4. In tracking and discrimination problems, a sensor (e.g., a radar) can often make different types of measurements by transmitting a suitable waveform each of which has a different power requirement. Each of the waveforms corresponds to a different sensor.

5. There might be shared communication resources (e.g., broadcast channels or a shared communication bus) that constrain the usage of many sensors at the same time. Such a situation arises, e.g., in telemetry-data aerospace systems.

Many other situations can be looked at in a similar framework. We will give two rather non-obvious examples in the appendices. There are two problems that we will consider:

1. For a given schedule of the sensor usage, what is the performance of the estimator?

2. How do we come up with an optimal sensor schedule? This is often referred to as the sensor scheduling problem.

Because of the wide range of problems this framework can address, it has received considerable attention in literature. The seminal work of Meier et al. [102] showed that for linear plants and quadratic cost functions, a separation property holds between the optimal plant control policy and the measurement control policy. The measurement control problem, which is the sensor scheduling problem, was cast as a non-linear deterministic control problem and shown to be solvable by a tree-search in general. That work proposed forward dynamic programming and a gradient method for solution. To deal with the complexity of a tree-search, greedy algorithms have been proposed many times, some examples being the works of [75, 111, 158]. Allied contributions have dealt with robust sensor scheduling as in Savkin et al. [172], a greedy algorithm with an information based

cost measure [209], a numerical method for obtaining sub-optimal schedules with error bounds as in Alriksson and Rantzer [1] and include the works of [117, 142, 161] etc. A different numerical approach to solve the problem was provided by Athans in [4] who cast the problem as a two-point boundary value problem. The non-linear matrix differential equations thus obtained were solved numerically by a min-H technique. These ideas were extended to discrete-time systems by Kerr in [113] and the two point boundary value problem was converted to an initial value problem by Kerr and Oshman in [114]. In [97] the technique was generalized to consider multiple devices being chosen at the same time.

However, all these approaches assumed *deterministic* sensor schedules. With communication links dropping packets randomly, stochastic schedules have to be considered. However, apart from some special cases in [134, 178], stochastic schedules have not been analyzed. We present such an analysis both for the case when sensors are chosen independently from one time step to the next and when they are chosen in a Markovian fashion. We then use these results to obtain a stochastic sensor scheduling algorithm that is based on the idea of letting the sensors switch randomly according to some optimal probability distribution to obtain the best *expected* steady-state performance.[1] Besides being numerically more tractable than tree-search based and other solutions proposed in the literature, it does not rely on the sensors having considerable computational power or knowledge about other sensors. There are numerous other advantages as will be pointed out later.

We then turn our attention to a control problem. Consider N agents that are trying to minimize a joint cost function. The agents can share information according to some topology. As an instance, the topology may be a completely decentralized one in which the agents do not share any information or it may be a completely centralized one in which every agent communicates with all other agents. If the agents share information over links that stochastically drop packets, it makes sense to look at stability and performance of the system for topology changes that happen in a random fashion. Moreover, since communication is usually expensive (e.g., in terms of power) such a topology change may also be implemented by the sensors themselves and it then becomes important to characterize the performance as a function of (say) the frequency of communication by each node.

As we saw in Chapter 2, the problem of distributed control even without the presence of communication channels is largely open. Thus, the effects of communication channels on distributed control have received relatively little attention in the community, apart from notable exceptions like [168, 194], that included propagation delays in their theory. Seiler and Sengupta [175] posed the problem of $\mathcal{H}_\infty$ optimal control of networked systems with stochastic packet-losses (with failures modeled as time-homogeneous Markovian processes with known transition probability matrix). While they were able to derive LMI conditions for the existence of a stabilizing controller in this

[1] As is made clear later, for computational ease, we actually minimize an upper bound on the expected steady-state performance.

case, the latter was centralized. Langbort et al. [124] explored the synthesis of distributed controllers in the presence of stochastically failing channels. Their conditions, however, were only sufficient and resulted in sub-optimal controllers. We consider a simple model of control in the presence of packet drop channels and provide some initial results. Even though the topologies switch in an i.i.d. fashion, we do not assume a fixed controller structure. Hence, the problem cannot be analyzed in the framework of jump linear Markov systems (as outlined in, e.g., [152, 175]). Instead we solve for the optimal time-varying controller.

The framework and tools that we develop in this chapter are more widely applicable. In the appendices, we present two such examples. The first appendix deals with dynamic sensor coverage which is the problem of sensor trajectory generation for optimal coverage of an area. This problem arises when there are some specified number of mobile sensors that can each sense over a limited region but together they must monitor a given area. The problem of optimal sensor location in case there are no bounds on the range over which the sensors can sense leads to the problem of Voronoi partitioning of the space and has been solved both in a centralized framework [50, 155, 156] and in a decentralized fashion [38]. The case when sensors are mobile and hence one needs to optimize their trajectories was considered in Chapter 2, Appendix B. The problem when there are range (or direction) limitations on the sensors has also been looked at in the literature. However, most of the approaches that have been proposed are very application specific [96, 144, 151]. Determining the optimal trajectory is, in general, a tree search problem and greedy approaches have often been proposed [13, 147]. We can again obtain many advantages over such algorithms by using our method.

In the second appendix, we apply our tools to the problem of network source coding for the purpose of estimation over a communication link that drops packets randomly. We show how multiple description (MD) source codes [62] that have been considered in the information theory literature [65, 69] and used in, e.g., transmission of real-time video over the Internet [70, 125] can increase the performance of an estimator interested in estimating the state of a dynamical system across a digital packet dropping link, without increasing the bit rate. This is the first time that network codes have been utilized for networked estimation and control. We will apply the tools that we developed in the chapter earlier for analyzing such systems.

4.2 Estimation in Presence of a Stochastic Sensor Schedule

Consider the problem setup shown in Figure 4.1. A process evolves according to the equation

$$x(k+1) = Ax(k) + w(k), \tag{4.1}$$

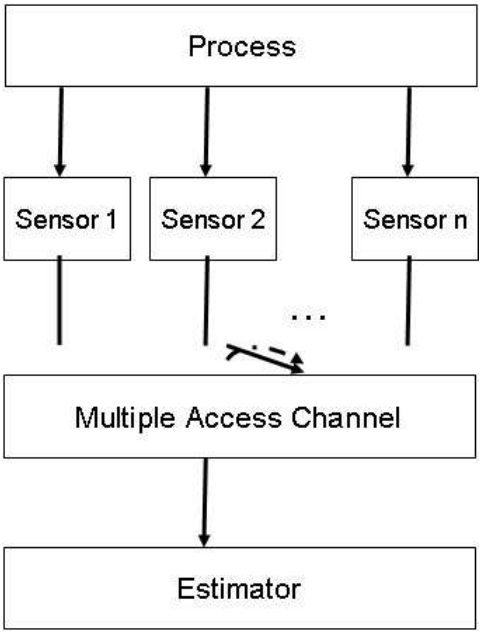


Figure 4.1: Structure of the sensor scheduling problem.

where $x(k) \in \mathbf{R}^n$ is the process state at time step k and $w(k)$ is the process noise assumed white, Gaussian and zero mean with covariance matrix R_w. The initial state $x(0)$ is assumed to be a Gaussian zero mean random variable with covariance $P(0)$. The process state is being observed by N sensors with the measurement equation for the i-th sensor (denoted by S_i) being

$$y_i(k) = C_i x(k) + v_i(k), \tag{4.2}$$

where $y_i(k) \in \mathbf{R}^{m_i}$ is the measurement. The measurement noises $v_i(k)$'s for the sensors are assumed independent of each other and of the process noise. Further the noise $v_i(k)$ is assumed to be white, Gaussian and zero mean with covariance matrix $R_{v,i}$. At every time step, one sensor takes a measurement which is then communicated to all the sensors (or a data sink such as a central estimator) in an error-free manner. Two things may be noted.

1. The assumption of one sensor per time step is without loss of generality.

2. If packets can be dropped, the drop event is equivalent to a sensor with noise covariance $R_{v,i} \to \infty$ being chosen.

Since all the measurements are being shared, every sensor has the same estimate of the process state $x(k)$, denoted by $\hat{x}(k|k-1)$ or $\hat{x}(k)$. Further the optimal estimate is given by a Kalman filter assuming a time-varying sensor. Assuming that the i-th sensor takes the measurement at time step

k, the covariance of the estimate error $P(k)$ evolves according to the Riccati recursion

$$P(k+1) = AP(k)A^T + R_w - AP(k)C_i^T \left(C_i P(k) C_i^T + R_{v,i}\right)^{-1} C_i P(k) A^T, \tag{4.3}$$

with the initial condition given by $P(0)$. It is obvious from (4.3) that error covariance is a function of the sensor schedule. For future use, we introduce the following notation. Consider a sensor that senses a process of form (4.1) according to the relation

$$y(k) = Cx(k) + v(k),$$

where the noise $v(k)$ has covariance matrix R_v. Then, we define

$$f_C(P) = APA^T + R_w - APC^T \left(CPC^T + R_v\right)^{-1} CPA^T \tag{4.4}$$

and further denote

$$f_C^k(P) = \underbrace{f_C\left(f_C\left(\cdots\left(f_C(P)\right)\right)\right)}_{f_C \text{ applied } k \text{ times}}. \tag{4.5}$$

Thus, the error covariance of the estimate at time step $k+1$ if Kalman filter is being used is given by $f_C(P)$, where P is the error covariance at time step k.

4.2.1 Preliminary Results

We will need the following properties of the operator $f_C(.)$.

Lemma 4.1 *$f_C(P)$ is concave in P provided P is positive semi-definite and R is positive definite.*

Proof We use the fact [24] that a function $f(x)$ is concave in x if and only if $f(x_0 + th)$ is concave in the scalar t for all x_0 and h. Thus, consider

$$\Sigma = A(P_0 + tZ)A^T - A(P_0 + tZ)C^T \left(R + C(P_0 + tZ)C^T\right)^{-1} C(P_0 + tZ)A^T.$$

Denoting $R + C(P_0 + tZ)C^T$ as X, we obtain

$$\frac{\partial \Sigma}{\partial t} = AP_0A^T - AZC^TX^{-1}C(P_0+tZ)A^T - A(P_0+tZ)C^TX^{-1}CZ\left[I - C^TX^{-1}C(P_0 + tZ)\right]A^T.$$

Thus, the second derivative is given by

$$\frac{\partial^2 \Sigma}{\partial t^2} = -2\Psi X^{-1}\Psi^T,$$

where

$$\Psi = A\left[(P_0 + tZ)C^T X^{-1} CZC^T - ZC^T\right].$$

Note that X is positive definite under the stated conditions on P and R, hence X^{-1} exists and is positive definite. The second derivative is negative everywhere which proves the assertion. ∎

Lemma 4.2 *1. Let X and Y be two positive semi-definite matrices. If $X \leq Y$, then $f_C(X) \leq f_C(Y)$.*

2. For any C and for any positive semi-definite matrix X, $f_C(X) \geq R_w$.

Proof Introduce

$$\phi(K, X) = R_w + (A + KC)\, X\, (A + KC)^T + KRK^T,$$

and note that

$$K = -AXC^T \left(CXC^T + R\right)^{-1} \triangleq K_X$$

minimizes $\phi(K, X)$. Moreover, $\phi(K_X, X) = f_C(X)$. Finally, note that $\phi(K, X)$ is an increasing function in the second argument. Thus, we see that

$$f_C(X) = \phi(K_X, X) \leq \phi(K_Y, X) \leq \phi(K_Y, Y) = f_C(Y).$$

This proves the first part of the lemma. For the second part, note that

$$f_C(X) = \phi(K_X, X) \geq \phi(K_X, 0) = R_w. \quad ∎$$

Lemma 4.3 (Jensen's Inequality) *If a function $g(x)$ is concave,*

$$g\left(E[x]\right) \geq E\left[g(x)\right].$$

Proof Proof is standard. See, e.g., [71]. ∎

4.2.2 Sensors Chosen Independently from one Time Step to the Next

Consider the time-varying Kalman filter recursion given in (4.3) for the system given by (4.1). Suppose that at every time step, the probability of S_i being chosen as the sensor is $p_i(k)$. To begin with, assume that the choice is done independently at each time step. Suppose that the sensor chosen at time step k has sensing matrix $C(k)$ and noise covariance matrix $R(k)$. Then the error covariance evolves as

$$P(k+1) = f_{C(k)}\left(P(k)\right),$$

with $P(0)$ as the initial condition. Note that the error covariance $P(k+1)$ is random since it depends on the particular sequence of chosen S_i's ($0 \leq i \leq k$). To characterize it, we consider its expected value. Thus, we are interested in

$$E\left[P(k+1)\right] = E\left[f_{C(k)}\left(P(k)\right)\right] \tag{4.6}$$

Explicitly evaluating this expectation appears to be intractable in general. To understand its behavior, we look for upper and lower bounds.

4.2.2.1 Upper Bound

For the upper bound, we have the following result.

Proposition 4.4 *Let there be N sensors out of which one sensor is randomly chosen per time step for taking measurements. If the i-th sensor is chosen at time step k with probability $p_i(k)$ independently at each time step, then the expected error covariance of the estimate is upper bounded by $\Delta(k+1)$ given by the recursion*

$$\Delta(k+1) = R_w + A\Delta(k)A^T - \sum_{i=1}^{N} p_i(k)\Big[A\Delta(k)C_i^T(R_{v,i} + C_i\Delta(k)C_i^T)^{-1}C_i\Delta(k)A^T\Big], \tag{4.7}$$

with the initial condition $\Delta(0) = P(0)$.

Proof First note that the quantities $P(k)$ and $C(k)$ are independent. This is so since the error covariance at time step k depends only on the sensor choices till time step $k-1$ which are independent of the choice of the sensor at time step k. Thus, we can explicitly take the expectation in (4.6) with respect to the probability distribution of $C(k)$ and write

$$E\left[P(k+1)\right] = \sum_{i=1}^{N} p_i(k) E\Big[f_{C_i}\left(P(k)\right)\Big],$$

where the expectation on the right hand side is now over $C(0), \ldots, C(k-1)$. Now we use Jensen's inequality on account of Lemma 4.1. Thus, we obtain

$$E\left[P(k+1)\right] = \sum_{i=1}^{N} p_i(k) E\Big[f_{C_i}\left(P(k)\right)\Big] \leq \sum_{i=1}^{N} p_i(k) f_{C_i}\left(E\left[P(k)\right]\right) \tag{4.8}$$

Since $f_{C_i}(.)$ is an increasing operator, we obtain the required upper bound. ∎

We need to check for the convergence of (4.8) as time progresses. The convergence of the upper bound would imply boundedness of the recursion in (4.6). We have the following result.

Proposition 4.5 *In the setting of Proposition 4.4, suppose that the sensor probabilities $p_i(k)$ tend to constants q_i as $k \to \infty$. If there exist matrices K_1, K_2, $\cdots$, K_N and a positive definite matrix P such that*

$$P > R_w + \sum_{i=1}^{N} q_i \left((A + K_i C_i) P (A + K_i C_i)^T + K_i R_{v,i} K_i^T \right),$$

then the iteration in (4.7) *converges for all initial conditions $P(0) \geq 0$ and the limit $\bar{P}$ is the unique positive semi-definite solution of the equation*

$$X = R_w + AXA^T - \sum_{i=1}^{N} q_i A \left[X C_i^T (R_{v,i} + C_i X C_i^T)^{-1} C_i X \right] A^T. \tag{4.9}$$

Proof Proof is similar to the one of Theorem 1 in [178]. We redefine the quantities

$$\begin{aligned} \mathcal{L}(Y) &= \sum_{i=1}^{N} q_i (A + K_i C_i) Y (A + K_i C_i)^T \\ \phi(K_i, P) &= R_w + \sum_{i=1}^{N} q_i \left((A + K_i C_i) P (A + K_i C_i)^T + K_i R_{v,i} K_i^T \right) \end{aligned}$$

and follow the arguments given in that proof. ∎

As an example, if all eigenvalues of A are strictly less than unity in magnitude, we can always find matrices K_i's and P satisfying the above conditions by choosing K_i's as the zero matrices and P as $2\bar{P}$ where $\bar{P}$ is the positive definite solution of the Lyapunov equation

$$\bar{P} = A\bar{P}A^T + R_w.$$

Thus, as long as A is stable, the recursion in (4.8) converges. The case when A is stable (and thus the process to be estimated does not grow unbounded) is very important in a large number of practical applications.

For the cases when A is not stable, we need to find out if (4.6) diverges. We now obtain a lower bound for the recursion. If the lower bound does not converge, it will imply the non-convergence of the expected steady state error covariance.

4.2.2.2 Lower Bound

We have the following result for obtaining a lower bound on $E[P(k)]$.

Proposition 4.6 *Suppose there are N sensors out of which one sensor is randomly chosen per time step for taking measurements and the j-th sensor is chosen with probability $p_j(k)$ at time step*

k independently at each time step. Define

$$\bar{p}_j(k-t+1,k) = p_j(k-t+1)p_j(k-t+2)\cdots p_j(k).$$

Then, the expected error covariance of the estimate at time step k is lower bounded by $X(k)$ where $X(k)$ is obtained from the equation

$$\begin{aligned} X(k) = {} & \bar{p}_j(0,k-1) f_{C_j}^k\left(P(0)\right) + \left(1-p_j(k-1)\right) R_w + \bar{p}_j(k-1,k-1)\left(1-p_j(k-2)\right) f_{C_j}\left(R_w\right) \\ & + \bar{p}_j(k-2,k-1)\left(1-p_j(k-3)\right) f_{C_j}^2\left(R_w\right) + \cdots + \bar{p}_j(1,k-1)\left(1-p_j(0)\right) f_{C_j}^{k-1}\left(R_w\right), \quad (4.10) \end{aligned}$$

where $P(0)$ is the initial error covariance. Note that one such lower bound results for each value of $j = 1, \cdots, N$.

Proof The value of $E[P(k)]$ is determined by the sensor schedule till time step $k-1$. Consider any sensor S_j. The event space for the all these choices can be partitioned into $k+1$ disjoint events E_i:

1. For the value $i = k$, one event of the form: Sensor S_j was not chosen at time step $k-1$.
2. For the values of $1 \leq i \leq k-1$, k events of the following form: The sensor S_j was chosen at all time steps $k-i$, $k-i+1$, $\cdots$, $k-1$. Moreover, at time step $k-i-1$, sensor S_j was not chosen.
3. For the value $i = 0$, one event of the form: Sensor S_j has been chosen for all time steps 0, 1, $\cdots$, $k-1$.

The expected error covariance $E[P(k)]$ is given by

$$E\left[P(k)\right] = \sum_{i=0}^{k} \text{Prob}\left(E_i\right) V\left(E_i\right),$$

where $\text{Prob}(E_i)$ refers to the probability of E_i occurring and $V\left(E_i\right)$ refers to the value of error covariance under the event E_i. We note the following facts that we use to lower bound each of the terms $V\left(E_i\right)$.

1. If Σ is the error covariance at any time step m and the sensor S_j is chosen at time step m, the error covariance at the time step $m+1$ is given by $f_{C_j}(\Sigma)$. This is by definition of the operator $f_C(.)$.
2. If any sensor other than S_j is chosen at any time step m, the error covariance at time step $m+1$ is lower bounded by R_w. This is using Lemma 4.2.

Using both these facts, if sensor S_j was chosen at time step $m+1$ but not at time m, then the error covariance at time step $m+2$ is lower bounded by $f_{C_j}(R_w)$.

Now consider the event E_i where $1 \leq i \leq k-1$. By the argument given above,

$$V(E_i) \geq f^i_{C_j}(R_w).$$

The probability of event E_i happening is just the probability that at time $k-1-i$ sensor S_j was not chosen and that it has been chosen from time $k-i$ to $k-1$. Thus,

$$\text{Prob}(E_i) = \bar{p}_j(k-i, k-1)\,(1 - p_j(k-i-1)).$$

Thus, we obtain

$$\text{Prob}(E_i)V(E_i) \geq \bar{p}_j(k-i, k-1)\,(1 - p_j(k-i-1))\, f^i_{C_j}(R_w).$$

A similar argument for the events E_0 and E_k shows that

$$\begin{aligned} \text{Prob}(E_0)V(E_0) &\geq (1 - p_j(k-1))R_w \\ \text{Prob}(E_k)V(E_i) &\geq \bar{p}_j(0, k-1) f^k_{C_j}(R_w). \end{aligned}$$

By adding together the terms $\text{Prob}(E_i)V(E_i)$, we obtain that $X[k]$ as given in (4.10) is indeed a lower bound for the expected error covariance. ∎

We can also obtain a necessary condition for $E[P(k)]$ being bounded as $k \to \infty$ by studying the behavior of $X(k)$ as time k progresses. We have the following result.

Proposition 4.7 *Consider the setting of Proposition 4.6. If the sensor probabilities $p_i(k)$'s tend to constants q_i as $k \to \infty$, then a necessary and sufficient condition for $X(k)$ given in (4.10) to stay bounded as $k \to \infty$ is*

$$q_j |\lambda_{\max}(\bar{A}_j)|^2 < 1, \tag{4.11}$$

where q_j is the probability of choosing the j-th sensor while $\lambda_{\max}(\bar{A}_j)$ refers to the eigenvalue with the maximum magnitude of the unobservable part of A when the pair (A, C_j) is put in the observable canonical form [51].

Proof We can assume without loss of generality that the pair (A, C_j) is in the observer canonical form[2]. Denote the matrices in the form

$$A = \begin{bmatrix} \bar{A}_{11} & 0 \\ \bar{A}_{21} & \bar{A}_{22} \end{bmatrix} \qquad C_j = \begin{bmatrix} \bar{C}_{j,1} & 0 \end{bmatrix}.$$

[2]If not, an invertible linear transformation can convert it to the companion form. This transformation will not affect the boundedness of estimation error.

Denote by $e(k)$ the error in estimating $x(k)$. We can identify a suitable division of $e(k)$ in accordance with the observer canonical form so that we can write it as

$$e(k) = \begin{bmatrix} e_1(k) \\ e_2(k) \end{bmatrix},$$

where $e_1(k)$ corresponds to the observable part of the system and $e_2(k)$ to the unobservable part. The covariance of the error is given as

$$P(k) = \begin{bmatrix} E\left[e_1(k)e_1^T(k)\right] & E\left[e_1(k)e_2^T(k)\right] \\ E\left[e_2(k)e_1^T(k)\right] & E\left[e_2(k)e_2^T(k)\right] \end{bmatrix} = \begin{bmatrix} P_{11}(k) & P_{12}(k) \\ P_{21}(k) & P_{22}(k) \end{bmatrix},$$

with $P(0) = R_w$. Since $e_1(k)$ corresponds to the observable part of the system, $P_{11}(k)$ is bounded. $P_{22}(k)$ evolves as

$$P_{22}(k+1) = \bar{A}_{22}P_{22}(k)\bar{A}_{22}^T + \Sigma(k),$$

where $\Sigma(k)$ is a matrix that depends on R_w, $R_{v,j}$, $P_{11}(k)$ and $P_{12}(k)$ but not on $P_{22}(k)$. Thus, $\Sigma(k)$ remains bounded as $k \to \infty$ if $P_{11}(k)$ and $P_{12}(k)$ remain bounded. $X[k]$ can be obtained through a summation of the form

$$\begin{aligned} X(k) &= \sum_{i=0}^{k-1} q_j^{i-1}(1-q_j) f_{C_j}^{i-1}(R_w) \\ &= \sum_{i=0}^{k-1} \gamma(i) P(i-1) \\ &= \begin{bmatrix} \sum_{i=0}^{k-1} \gamma(i) P_{11}(i-1) & \sum_{i=0}^{k-1} \gamma(i) P_{12}(i-1) \\ \sum_{i=0}^{k-1} \gamma(i) P_{21}(i-1) & \sum_{i=0}^{k-1} \gamma(i) P_{22}(i-1) \end{bmatrix}, \end{aligned}$$

where $\gamma(i) = q_j^{i-1}(1-q_j)$. There are four terms here whose boundedness needs to be considered. Now, as already stated, the $(1,1)$ term is bounded because of the observability assumption. The $(2,2)$ term is bounded if and only if

$$q_j |\lambda_{\max}\left(\bar{A}_{22}\right)|^2 < 1.$$

Also, if both $(1,1)$ and $(2,2)$ terms are bounded, the off-diagonal terms of $P(k)$ are bounded by the Cauchy-Schwarz inequality. But $\bar{A}_{22}$ is the unobservable part of A when it is observed using matrix C. Thus, a necessary and sufficient condition for convergence of the lower bound $X(k)$ is given by (4.11). ∎

Remarks

1. The condition (4.11) makes intuitive sense. Suppose a sensor S_j cannot observe certain modes in the system. Then, for the observation error covariance to remain bounded, we depend on the other sensors to obtain information about those modes. Thus, the more unstable those modes are (higher the value of $|\lambda_{\max}\left(\bar{A}_{22}\right)|^2$ is), the more should other sensors be used (the sensor S_j should be used less often) and lower the value of q_j should be.

2. The bounds we have obtained can be specialized to cases when one sensor pertains to a data loss situation similar to that considered in [134, 178].

4.2.3 Sensors Chosen According to a Markov Chain

If the sensors are being chosen according to a Markov chain, we can still obtain upper and lower bounds in a similar form as the ones discussed above. We present the bounds below. Let $Q = [q_{ij}]$ be the transition probability matrix of the Markov chain. Also, let $p_i(k)$ be the probability of being in Markov state i at time step k, i.e., the i-th sensor being used at time step k.

4.2.3.1 Upper Bound

The upper bound can be derived as follows.

Proposition 4.8 *Suppose a process of the form (4.1) is being observed by a sensor chosen from N sensors of the form (4.2). If the sensors are chosen according to a Markov chain, then the expected estimate error covariance $E[P(k)]$ is upper bounded by $\Delta(k)$ where*

$$\Delta(k+1) = \sum_{j=1}^{N} p_j(k)\Delta_j(k+1),$$

and $\Delta_j(k)$'s evolve according to

$$p_j(k)\Delta_j(k+1) = \sum_{i=1}^{N} f_{C_j}\left(\Delta_i(k)\right) q_{ij} p_i(k-1). \tag{4.12}$$

Proof Let $C(k) = j$ denote that the j-th sensor was chosen at time step k. Then, by partitioning the event space at time step k, we see that

$$E\left[P(k+1)\right] = \sum p_j(k) E\left[P(k+1)|C(k) = j\right].$$

Now consider the conditional expected error covariance $E\left[P(k+1)|C(k) = j\right]$. We can further

condition it on the sensor choice at time step $k-1$ to obtain

$$p_j(k)E\left[P(k+1)|C(k)=j\right] \\ = p_j(k)\sum_{i=1}^{N} E\left[P(k+1)|C(k)=j, C(k-1)=i\right]\text{Prob}(C(k-1)=i|C(k)=j).$$

But since at time step k, for all the terms in the expectation above, the j-th sensor was used,

$$E\left[P(k+1)|C(k)=j, C(k-1)=i\right] = E\left[f_{C_j}(P(k))|C(k-1)=i, C(k)=j\right].$$

Moreover, using Bayes rule,

$$p_j(k)\text{Prob}(C(k-1)=i|C(k)=j) = q_{ij}p_i(k-1).$$

Thus, we have

$$p_j(k)E\left[P(k+1)|C(k)=j\right] = \sum_{i=1}^{N} E\left[f_{C_j}(P(k))|C(k-1)=i, C(k)=j\right] q_{ij}p_i(k-1).$$

But $f_{C_j}(P_k)$ and C_k are conditionally independent given C_{k-1}. Thus,

$$p_j(k)E\left[P(k+1)|C(k)=j\right] = \sum_{i=1}^{N} E\left[f_{C_j}(P(k))|C(k-1)=i\right] q_{ij}p_i(k-1).$$

Finally, using Jensen's inequality we obtain

$$p_j(k)E\left[P(k+1)|C(k)=j\right] \leq \sum_{i=1}^{N} f_{C_j}\left(E\left[P(k)|C(k-1)=i\right]\right) q_{ij}p_i(k-1).$$

Since $f_{C_j}(.)$ is an increasing operator, we obtain the required bound. ∎

We can also look at the convergence conditions for the upper bound. We have the following result.

Proposition 4.9 *Consider the setting of Proposition 4.8. Assume that the Markov chain transition probability matrix Q is such that the states reach a stationary probability distribution with the probability of the j-th sensor being used as $\pi_j > 0$. If there exist N positive definite matrices X_1, X_2, $\cdots$, X_N and N^2 matrices K_{11}, K_{12}, $\cdots$, K_{1N}, $K_{2,1}$, $\cdots$, $K_{N,N}$ such that*

$$\pi_j X_j > \sum_{i=1}^{N}\Big((A+K_{ij}C_j)X_i(A+K_{ij}C_j)^T + R_w + K_{ij}R_{v,j}K_{ij}^T\Big)q_{ij}\pi_i,$$

then, as $k\to\infty$, the upper bound $\Delta(k)$ as defined in Proposition 4.8 converges for all initial condi-

tions $\Delta(0) > 0$. In the limit, the quantities $\bar{\Delta}_j$'s are unique positive semi-definite solutions X_j's of the equations

$$\pi_j X_j = \sum_{i=1}^{N} f_j\left(X_i\right) q_{ij} \pi_i. \tag{4.13}$$

The upper bound for the error covariance, Δ, is given by

$$\Delta = \sum_{j=1}^{N} \pi_j \bar{\Delta}_j.$$

Proof Proof is along the lines of the proof given for i.i.d. sensor choices given in Section 4.2.2.1. Redefine the quantities

$$\begin{aligned} \mathcal{L}\left(Y_j\right) &= \sum_{i=1}^{N} q_{ij}\pi_i (A + K_{ij}C_j) Y_j (A + K_{ij}C_j)^T \\ \phi_j\left(K_{ij}, Y_i\right) &= \Big((A + K_{ij}C_j) Y_i (A + K_{ij}C_j)^T + R_w + K_{ij} R_{v,j} K_{ij}^T\Big) q_{ij}\pi_i, \end{aligned}$$

and follow the arguments in that proof. ∎

4.2.3.2 Lower Bound

A lower bound for $E[P(k)]$ can be derived in a similar way as before. We give the result below.

Proposition 4.10 *Suppose a process of the form (4.1) is being observed by a sensor chosen from N sensors of the form (4.2). If the sensors are chosen according to a Markov chain, then the expected error covariance $E[P(k)]$ is lower-bounded by $Y(k)$ where*

$$Y(k) = \sum_{i=1}^{k} q_{jj}^{i-1} \left(p_j(k+1-i) - q_{jj} p_j(k-i)\right) f_{C_j}^{i}\left(R_w\right) + q_{jj}^{k-1} p_j(0) f_{C_0}^{k}\left(P(0)\right).$$

Proof Proof follows exactly that of the i.i.d. case in Section 4.2.2.2 and is omitted. ∎

A necessary and sufficient condition for convergence of the lower bound can also be derived by following the derivation in Proposition 4.7. We state the result below without proof.

Proposition 4.11 *Consider the setting of Proposition 4.10. A necessary and sufficient condition for $X(k)$ given in (4.10) to stay bounded as $k \to \infty$ is*

$$q_{jj} |\lambda_{\max}\left(\bar{A}_j\right)|^2 \leq 1, \tag{4.14}$$

where q_{jj} is the probability of choosing the j-th sensor on two consecutive time steps while $\lambda_{\max}\left(\bar{A}_j\right)$ refers to the eigenvalue with the maximum magnitude of the unobservable part of A when the pair (A, C_j) is put in the observable canonical form.

We have thus characterized the expected error covariance by giving upper and lower bounds when the sensors are chosen in an i.i.d. or a Markovian fashion. The bounds can also be used for constructing an optimal sensor schedule as discussed in the next section.

4.3 Stochastic Sensor Scheduling

So far in the chapter, we have assumed that the sensor schedule was given to us a priori. A related problem is to come up with a sensor schedule that is optimal with respect to some cost function. Thus, consider once again a system evolving as in (4.1) being observed by N sensors with the i-th sensor providing an observation $y_i(k)$ according to (4.2). Suppose that only one sensor can be used at any time and the measurements are exchanged among all the sensors (or sent to a central estimator) in a noise-free manner. The estimator calculates the minimum mean squared error estimate $\hat{x}(k|k-1)$ (referred to simply as $\hat{x}(k)$) of the process state $x(k)$. If we denote the estimate error covariance as $P(k)$, we wish to find the sensor schedule that minimizes some function of $P(k)$ over a given time horizon. We begin by presenting two simple tree search based deterministic algorithms.

4.3.1 Deterministic Scheduling Algorithms

For simplicity and without loss of generality, we consider only two sensors and define the cost function, J_T, to be the sum of the traces of the error covariance matrices over the running time of the system:

$$J = \sum_{k=0}^{T} \text{trace}(P(k)).$$

In a more general case, the covariances can be variously weighted to set up the cost function if getting a good estimate either at some time steps or for some sensors is more important than others.

It is obvious that we can represent all the possible sensor schedule choices by a tree structure, as shown in Fig. 4.2 for the case of two sensors. Each node on the tree represents a particular time step, with the root defined to be time zero. The branches from each node correspond to choosing a particular sensor to be active at the next time instant. Thus, the path from the root to any node at depth $d+1$ represents a particular sensor schedule choice for time steps 0 to d. We can associate with each node the cost function evaluated using the sensor schedule corresponding to the path from the root to that node. Obviously, finding *the* optimal sequence requires traversing all the paths from the root to the leaves in a binary tree (for the case of two sensors). If the leaves are at a depth T, a total of 2^T schedules need to be compared (N^T if N sensors are present). This procedure might place too high a demand on the computational and memory resources of the system. Moreover, in practical applications, T might not be fixed a-priori. Hence, we need some sort of on-line optimization procedure. We now present some approximations that address these

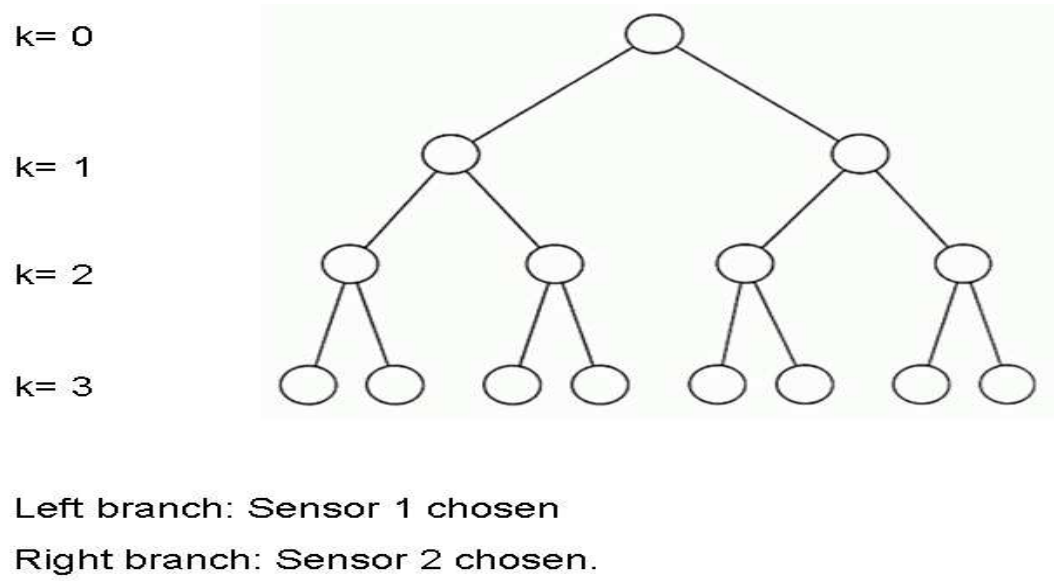


Figure 4.2: The tree structure defined by the various possible choices of sensor schedules, illustrated for the case of 2 sensors.

difficulties.

The basic idea behind the two approximations is to prune the tree to a manageable size. However, the pruning should ensure, with a high probability, that the optimal sequence is not lost. The algorithms presented involve choosing some arbitrary parameters which depend on the problem and the computation/memory resources available. Choosing these parameters conservatively will ensure that the sub-optimal solution achieved will be closer to the optimal solution but it might mean maintaining a large part of the tree intact. Thus, there is a trade-off involved. However, in the numerical examples studied, relatively liberal choices keep the tree size fairly small.

4.3.1.1 The Sliding Window Algorithm

This algorithm is similar to a pseudo real-time version of the Viterbi algorithm [63]. We define a window size d where $d < T$. The algorithm proceeds as follows:

1. Initialization: Start from root node at time $k = 0$.
2. Traversal:
   (a) Traverse all the possible paths in the tree for the next d levels from the present node.
   (b) Identify the sensor sequence S_k,S_{k+1},S_{k+2},..., S_{k+d-1} that yields the minimum cost at the end of this window of size d.
   (c) Choose the first sensor S_k from the sequence.
3. Sliding the Window:

(a) If $k = T$ then quit, else go to the next step.

(b) Designate the sensor S_k as the root.

(c) Update time $k = k + 1$.

(d) Repeat the traversal step.

The arbitrary parameter for this algorithm, mentioned earlier, is the window size d. If the window size is large enough, the sequence yielding the lowest cost will resemble the optimal sequence for the entire time horizon. Also, note that when we slide the window, we already have the error covariances for the first $d - 1$ time steps stored; hence they do not need to be recalculated. Consequently, the method is not very computationally intensive.

4.3.1.2 The Thresholding Algorithm

This algorithm is similar to that presented in [130], in the context of choosing the optimal controller from a set of many possible choices. We define a factor f where $f \geq 1$. The algorithm proceeds as follows:

1. Initialization : Start from root node with cost $J = 0$.

2. Pruning:

   (a) Extend the tree by one level (i.e. time step) through all possible paths from the current node.

   (b) Calculate the minimum cost up to that time step.

   (c) Prune away any branch that yields the cost greater than f times the minimum.

   (d) For the remaining branches, denote the cost of the nodes as the cost achieved by moving down the tree till the node.

3. Update: Consider each node in the next time step as the root node and repeat the pruning step.

4. After T time steps, declare the optimal sequence to be the one yielding the minimum cost till that time step.

The intuition behind the method is that any sequence which yields too high a cost at any intermediate time step would probably not be the one that yields the minimum cost over-all. By playing with the factor f, we obtain a trade-off between the certainty that we would not prune away the optimal sequence and the number of branches in the tree that need to be traversed.

4.3.2 A Stochastic Scheduling Algorithm

The algorithms considered above are both deterministic and are approximations to tree search algorithms. Now we present an algorithm that aims to do away with tree search altogether. The algorithm is based on the idea of letting the sensors switch randomly according to some optimal probability distribution to obtain the best *expected* steady-state performance.[3] Besides being numerically more tractable than tree-search based and similar solutions proposed in the literature, it does not rely on the sensors having considerable computational power or knowledge about other sensors. However, the chief attraction of the algorithm is that it can easily include the effect of stochastic packet drops by the communication channels over which the sensors are transmitting their measurements. Hence, it is suited naturally to the networked estimation situations that we have been considering in this dissertation.

Since the schedules we consider are stochastic, the error covariance is a random quantity. We will adopt the trace of the steady state expected error covariance as the metric to be minimized. As we saw in Section 4.2 exact evaluation of the steady state expected error covariance seems intractable. Instead, we consider an upper bound on the expected error covariance by using the results we derived above. We begin with the case when the sensors are being chosen in an independent and identically distributed (i.i.d.) manner from one time step to the next.

4.3.2.1 Sensors chosen in an i.i.d. manner

Denote the probability of sensor m being chosen at time step k by q_i. We know from Propositions 4.4 and 4.5 that the expected error covariance $E[P(k)]$ can be upper bounded by a quantity $\Delta(k)$ that evolves in a recursive fashion and further has a steady state value X under some conditions. We will adopt trace(X) as the metric to be minimized as an approximation to minimizing the expected error covariance itself. Divergence of the upper bound is a necessary condition for the divergence of the expected error covariance; hence the design can be expected to be conservative in this sense. Let there be N virtual sensors present. The design problem is

$$\begin{aligned} \min_{q_i} \ & \text{trace}(X) \\ & \text{s.t. } X = AXA^T + R_w - \sum_{i=1}^{N} q_i AXC_i^T \left(C_i X C_i^T + R_{v,i}\right)^{-1} C_i X A^T \\ & \sum q_i = 1 \qquad 0 \le q_i \le 1 \qquad X \ge 0. \end{aligned} \tag{4.15}$$

For a problem of small size, a brute force search suffices to find the optimal probabilities. However, we can also use a gradient descent algorithm to solve the problem. For ease of notation, we adopt

[3] As is made clear later, for computational ease, we actually minimize an upper bound on the expected steady-state performance.

the following notation. Define

$$g_q(X) = AXA^T + R_w - \sum_{i=1}^{N} q_i AXC_i^T \left(C_i XC_i^T + R_{v,i}\right)^{-1} C_i XA^T,$$

where q is the vector of q_i's. The cost function of our problem is trace(X), or equivalently, trace$(g_q(X))$. We will refer to any vector γ whose components γ_i's are non-negative and sum to 1 as a valid probability vector. The algorithm proceeds as follows:

1. Initialize at step $k = 1$, with an arbitrary valid probability vector $\gamma(1)$ and calculate the positive semi-definite matrix $X(1)$ that satisfies

$$X(1) = g_{\gamma(1)}\left(X(1)\right).$$

2. At every step k, do the following :

   (a) Calculate $\gamma_{\min}$ as a valid probability vector that minimizes trace of the quantity $g_\gamma(X(k))$.

   (b) Calculate

$$\bar{\gamma}(k) = \gamma(k) + \delta\left(\gamma_{\min} - \gamma(k)\right),$$

   where δ is the step size parameter between 0 and 1.

   (c) Obtain $\gamma(k+1)$ by projecting $\bar{\gamma}(k)$ on the set of valid probability vectors.

   (d) Calculate

$$X(k+1) = g_{\gamma(k+1)}\left(X(k)\right).$$

   (e) If $\gamma(k) = \gamma(k+1)$ (within a prescribed tolerance) then break else repeat the loop.

3. Output $\gamma(k+1)$ as the minimizing vector and trace $\left(X(k+1)\right)$ as the minimum cost function.

Remarks

1. $\gamma_{\min}$ is obtained through an optimization problem of the form

$$\arg\min_{\gamma} \ \text{trace}\left(AXA^T + R_w - \sum_{i=1}^{N} \gamma_i AXC_i^T \left(C_i XC_i^T + R_{v,i}\right)^{-1} C_i XA^T\right)$$
$$\sum \gamma_i = 1 \qquad 0 \le \gamma_i \le 1,$$

   where X is a given positive semi-definite matrix. This is a linear program and can be solved efficiently.

2. The projection step in the algorithm is required since $\bar{\gamma}(k)$ may have individual components that are negative or greater than 1, even though they sum up to 1. The optimal projection

would be to find out the vector that is a valid probability vector and minimizes the Euclidean distance from the original vector. In practice, however, heuristics such as setting the negative components to 0 and redistributing their weight to all the other components seem to work well.

3. There might be additional constraints placed on the probability vector. As an example, consider the situation when there are N physical sensors each of which communicate over a communication link that drops packets with a probability λ. Then, the situation is the same as if there were $N+1$ sensors, the first sensor corresponding to no measurements being taken with a fixed probability λ, while the other N sensors correspond to the physical sensors being used with probabilities $q_1(1-\lambda)$, $q_2(1-\lambda)$ and so on, where the q_i's still sum to 1. Such constraints can be easily considered in the algorithm above.

4. The algorithm assumes some shared randomness and synchronization among the sensors so that two sensors are not turned on at the same time. This can readily be achieved, e.g., through a common seed for a pseudo-random number generator available to all the sensors. Alternatively a token-based mechanism for the scheme can be implemented.

5. Also, note that the algorithm is run off-line and it has to be re-applied every time the number of sensors changes. However, if a sensor is stochastically failing with a known probability, we can model that in the algorithm.

Convergence of the Algorithm Even though simulations indicate that the algorithm usually converges to the global optimum, we have not been able to obtain analytical results except the following result for the case when the system being estimated is a scalar system. We first prove a lemma.

Lemma 4.12 *For two arbitrary positive semi-definite matrices X and Y,*

$$X \leq Y \Rightarrow g_q(X) \leq g_q(Y),$$

for any arbitrary valid probability vector q.

Proof The proof is straightforward.

$$\begin{aligned} g(X) &= \sum_i q_i \min_{K_i}((A+K_iC_i)^T X(A+K_iC_i) + K_i^T R_{v,i} K_i + R_w) \\ &\leq \sum_i q_i \min_{K_i}((A+K_iC_i)^T Y(A+K_iC_i) + K_i^T R_{v,i} K_i + R_w) \\ &= g(Y). \quad \blacksquare \end{aligned}$$

Suppose that we choose an initial guess $X(1)$ for the error covariance that is very small. We disregard the projection step and assume that at every step k, we do a greedy gradient descent. That is at every time step k, we calculate the probability vector that results in the minimum $X(k+1)$ given the current guess $X(k)$. Then, we can prove the following.

Proposition 4.13 *Suppose that the algorithm described in Section 4.3.2.1 (while disregarding the projection step) is used for calculating the optimal sensor schedule for estimating the state of a scalar process. If the algorithm converges, it converges to the globally minimum value of X.*

Proof Denote $\tilde{X}$ as the global minimum covariance and $\tilde{q}$ as the global optimum probability vector that achieves $\tilde{X}$. By choice of the initial condition, $X(1) \leq \tilde{X}$. Now two inequalities hold:

1. If we propagate $X(1)$ for one time step through the probability vector $\tilde{q}$ and obtain $\bar{X}$, then $X(2) \leq \bar{X}$. This is true because of the definition of $\tilde{q}$.

2. Because of Lemma 4.12, $\bar{X} \leq \tilde{X}$.

Combining the two inequalities, we obtain $X(2) \leq \tilde{X}$. Now we can apply the same argument repeatedly to obtain $X(k) \leq \tilde{X}$, for any time k. But if $X(k)$ converges, this equation means that we have obtained a cost less than or equal to the global optimum cost. By definition of the global optimum cost, thus, $X(k) = \tilde{X}$ on convergence. ∎

4.3.2.2 Sensors chosen according to a Markov chain

Denote the probability of sensor i being chosen at time step k by $\pi_i(k)$ and the probability of the sensor j being chosen at time step $k+1$ given that sensor i was chosen at time step k by q_{ij}. We again use the upper bound derived in Section 4.2. Similar to the i.i.d. case, if we assume N sensors to be present, we can pose the following optimization problem to solve for the elements of the transition probability matrix q_{ij}.

$$\begin{aligned} & \min_{q_{ij}} \ \operatorname{trace}(X) \\ \text{s.t.} \quad & X = \sum_{j=1}^{N} \pi_j X_j \qquad \pi_j X_j = \sum_{i=1}^{N} f_{C_j}(X_i)\, q_{ij} \pi_i \\ & \sum_j q_{ij} = 1 \qquad 0 \leq q_{ij} \leq 1 \\ & X_j \geq 0 \qquad \pi_i = \sum_{j=1}^{N} q_{ji} \pi_j. \end{aligned} \tag{4.16}$$

This can again be solved by an algorithm similar to the one proposed above for the i.i.d. case. The step of finding the minimizing q_{ij}'s remains a linear program.

4.3.3 Examples

We now apply our algorithm to a few sample problems to illustrate the scheduling algorithms. Since the main focus of the section is in presenting the stochastic sensor scheduling algorithm, we will concentrate on that algorithm. Assume a vehicle moving in 2-D space according to the standard constant acceleration model [12]. This model assumes that the vehicle has a constant acceleration equal to zero except for a small perturbation. We assume that the vehicle moves in a plane. Denoting the position of the vehicle in the two dimensions by p_x and p_y, the velocities by v_x and v_y and with a discretization step size of h, the dynamics of the vehicle are of the form (4.1) where

$$A = \begin{bmatrix} 1 & 0 & h & 0 \\ 0 & 1 & 0 & h \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} h^2/2 & 0 \\ 0 & h^2/2 \\ h & 0 \\ 0 & h \end{bmatrix} \qquad x = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}.$$

The term $w(k)$ is the perturbation term in acceleration and is modeled as a zero mean white Gaussian noise. In the numerical example, $h = 0.2$. The process noise is considered to have covariance matrix R_w given by

$$R_w = \begin{bmatrix} 1 & 0.25 \\ 0.25 & 1 \end{bmatrix}.$$

We assume two sensors with the measurements taken by the two sensors, y_1 and y_2 being described by

$$y_i(k) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x(k) + v_i(k). \tag{4.17}$$

The terms $v_i(k)$ model the measurement noise, again assumed white, zero mean and Gaussian and also independent from each other and from $w(k)$. We consider values of the sensor noise covariances as

$$R_1 = \begin{bmatrix} 2.4 & 0 \\ 0 & 0.4 \end{bmatrix} \qquad R_2 = \begin{bmatrix} 0.7 & 0 \\ 0 & 1.4 \end{bmatrix}. \tag{4.18}$$

We assume that only one sensor can take a measurement at any time step and the measurement is then transmitted in an error-free manner to the other sensor.

The plot given in Figure 4.3 illustrates that choosing any one sensor at all time steps is not optimal. The figure plots the cost measured as the sum of the traces of the error covariance matrices of the estimates at the two sensors when they adopt the strategy of choosing only sensor 1 or only sensor 2 or when they choose an arbitrarily generated schedule over 50 time steps. For comparison, the cost achievable by the optimal sensor strategy found by a sliding window approach to the tree search is also given. We see that the even an arbitrary sensor switching strategy can help to bring

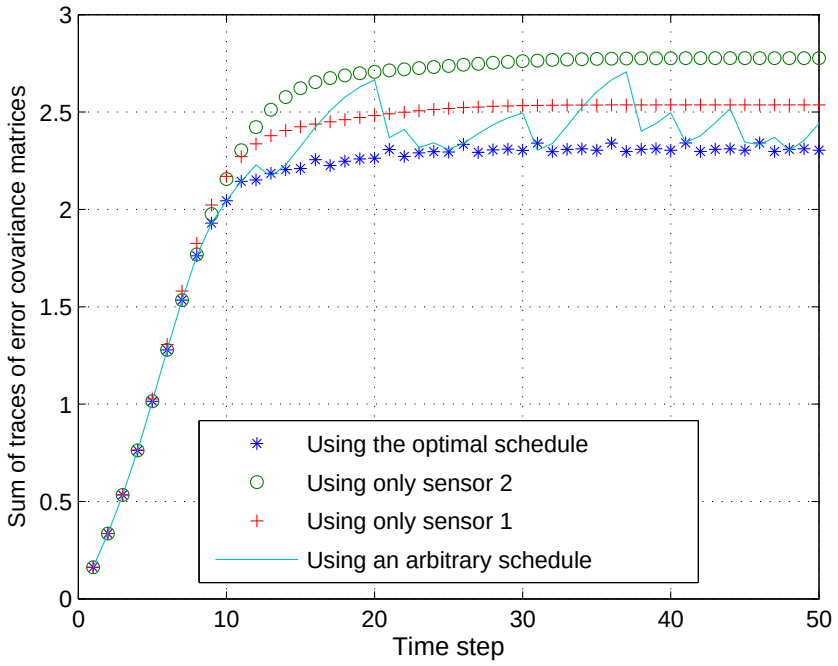


Figure 4.3: Sensor switching helps to bring the cost down.

down the cost.

Next, we apply our stochastic algorithm to find the optimal probability distribution. On optimizing the upper bound in (4.9) over q_1 and q_2, the optimal probability for sensor 1 turns out to be $q_1 = 0.395$. Indeed, if we find the optimal sequence by a complete tree search, it turns out that in the steady state, the percentage of sensor 1 in the sequence is about 37%. For this probability distribution, the steady state value of the upper bound of the sum of the traces of the expected error covariance matrices for the two sensors turns out to be 2.3884, which compares well with the value of about 2.3 obtained by the optimal strategy. Note that our algorithm results in orders of magnitude less calculation than tree search algorithms and finds a near-optimal schedule in the steady state. When 1000 test cases of random sensor schedules with $q_1 = 0.395$ were generated and used, the steady state cost averaged over time for each schedule turned out to be 2.3664. The spread of the cost is shown in Figure 4.4.

The computational savings can be very significant if we need to study the optimal sensor schedule as some sensor parameter is varied. As an example, let the measurement noise covariance of the second sensor be given by

$$R_2 = \alpha \begin{bmatrix} 0.7 & 0 \\ 0 & 1.4 \end{bmatrix}.$$

Figure 4.5 plots the optimal percentage of use of sensor 1, as the parameter α is varied. The plot

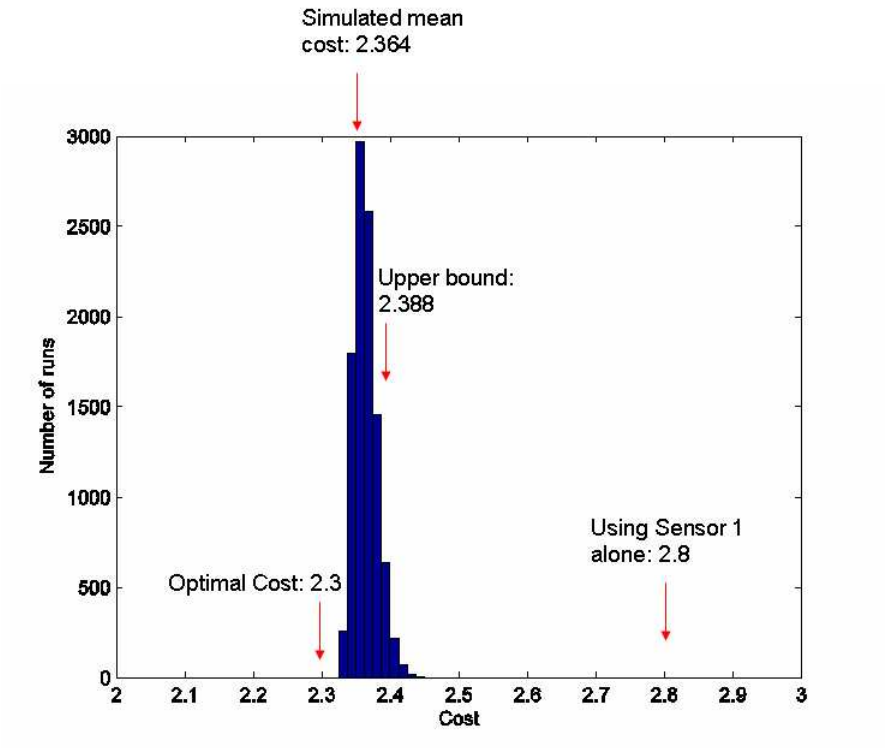


Figure 4.4: Spread of the error covariance when the optimal probability distribution is used.

shows that there is a threshold phenomenon such that increasing the noise of one sensor beyond that level results in that particular sensor never being used. The reduction in the computation needed makes our algorithm much more scalable in the number of sensors than the tree-based algorithms.

In addition, there are several unique advantages that our algorithm offers over the conventional algorithms. A very important one is the issue of sensor costs. Frequently, there are other considerations beyond estimation accuracy in using one sensor over another. As an example, it might be more costly to use a very accurate sensor at every time step. Similarly, we might want some sort of fairness such that one sensor is not used all the time because that may result in all its power being drained. Usually, it is not clear how to appropriately weight the sensor costs for a fair comparison with the estimation costs. Thus, it is not clear how to even generate a tree for the sensor schedule choices. However, it is easy to take sensor costs into account with our algorithm. As an example we consider three sensors of the form of (4.17) being present with the measurement noise covariances being given by

$$R_1 = \begin{bmatrix} 3.24 & 0 \\ 0 & 1.04 \end{bmatrix}, \qquad R_2 = \begin{bmatrix} 0.25 & 0 \\ 0 & 1.36 \end{bmatrix}, \qquad R_3 = \begin{bmatrix} 0.56 & 0 \\ 0 & 0.56 \end{bmatrix}.$$

Suppose that the three sensors are transmitting to a single data sink so that the only energy con-

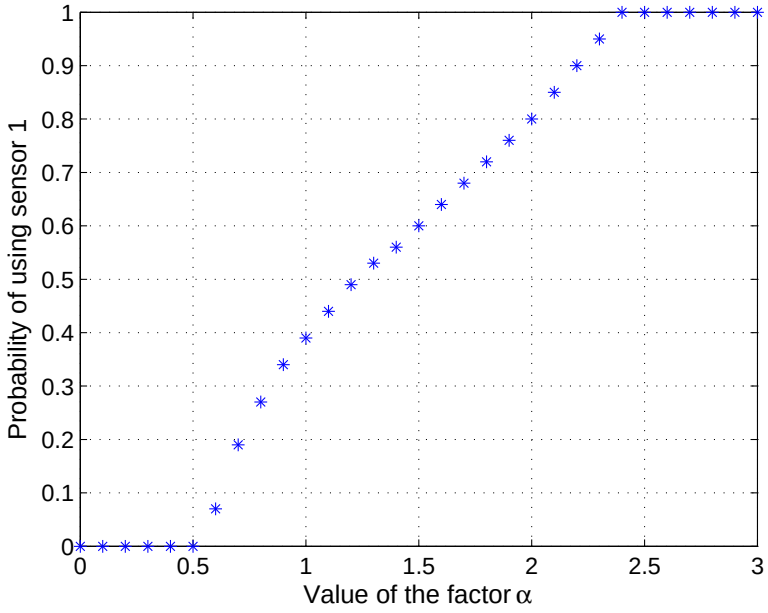


Figure 4.5: Optimal probability of use of sensor 1 as sensor 2 gets noisier.

sumption is in taking a measurement and then broadcasting it. If we try to optimize the probability distribution, we obtain that sensor 2 should be chosen with a probability of 0.2 and sensor 3 with a probability of 0.8. However, such a strategy would lead to sensor 3 draining away all its power very quickly and thus we might want to impose an additional constraint that on average, no sensor is used more than twice as much as any other sensor. It is not even apparent whether such constraints can be taken into account while using the traditional greedy algorithms or differential equation based algorithms. For a full tree-search algorithm, the computational burden would be immense. However, in our algorithm, We restrict our search to the relevant $q_1 - q_2$ space and come up with the optimal probabilities satisfying the additional constraint as sensor 1 being used with a probability of 0.2 and sensors 2 and 3 being used each with a probability of 0.4.

Another situation in which our algorithm is much more easily used is when there is some randomness imposed on the system. As an example, consider the case of two sensors with measurement noise covariances given by the values in (4.18). Suppose that the sensors are communicating with a data sink over a communication channel that randomly drops packets with probability λ. Compared to the conventional methods, it is easy to take the channel into account while using our algorithm. We set up equation (4.7) assuming that there are three sensors present. The first two sensors have covariance matrices given above and they are chosen with probabilities $q_1(1-\lambda)$ and $q_2(1-\lambda)$. The third sensor corresponds to the packet being dropped (and hence no measurement being taken) and it is chosen with a probability of $(q_1 + q_2)\,\lambda$. Then, we optimize this bound over the parameters q_1 and q_2. Figure 4.6 shows the change in the optimal probability of choosing sensor 1 as the packet

drop probability λ is varied. The plot shows that the packet drop probability plays a big role in determining the optimal sensor schedule.

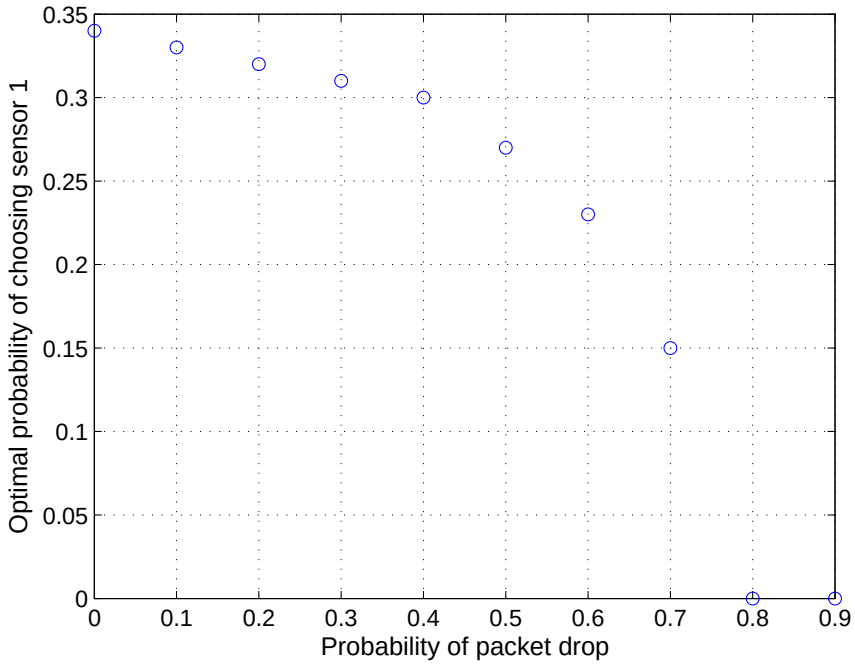


Figure 4.6: Optimal probability of use of sensor 1 varies if the channel is dropping packets.

The lower bound derived in Theorem 4.6 is useful for obtaining the region in the sensor usage probability space where the expected error covariance diverges. Consider the same system as in (4.1) being measured by two sensors of the form of (4.17). The measurement noise covariances are given by the values in (4.18). The upper bound and the lower bounds are plotted in Figure 4.7. We can see that the lower bound may not be very tight. However, the main utility of the lower bound is in predicting when the expected error covariance necessarily diverges. We consider the same example with the second sensor replaced by a sensor of the form

$$y(k) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x(k) + v(k),$$

with the sensor noise covariances given by (4.18). We see that the plant is unobservable while using the second sensor alone and hence, as the probability of using the second sensor increases, the error covariance would diverge. It can be shown that although there is a huge gap between the lower and upper bounds, both bounds diverge at $q_1 = 0.56$ which is thus the critical probability for error divergence. This value also matches the value given in Theorem 4.6 since the largest eigenvalue of the unobservable part of A is 1.5. In general, the probabilities when the bounds diverge will not

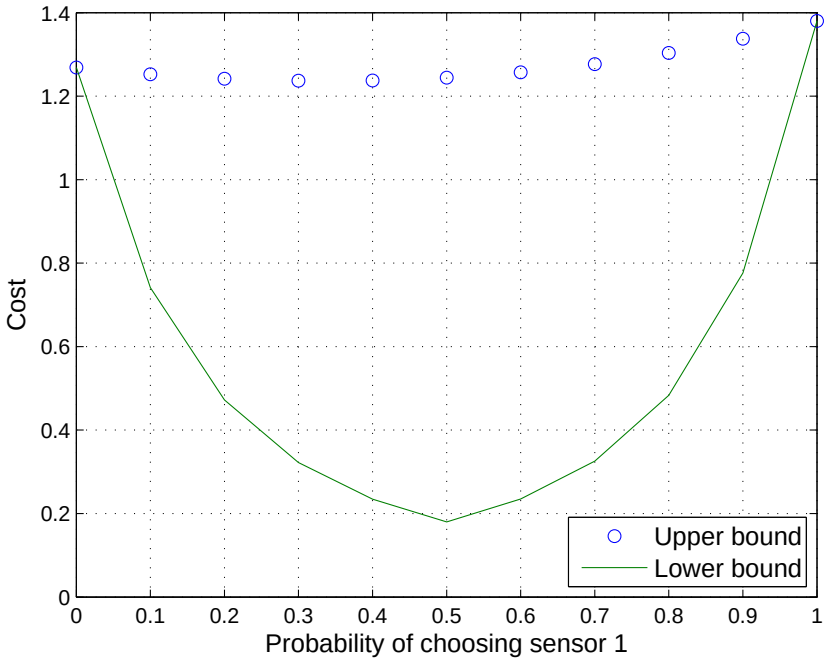


Figure 4.7: The lower bound may not be very tight.

match and they serve as lower and upper bounds on the critical probability. An important special case is when any one of the matrices C_i is invertible which renders the condition of divergence of the lower bound both necessary and sufficient for the divergence of the expected error covariance.

Our results can also be used in other problems that have a similarity to the sensor scheduling problem. While we will consider two such applications in detail in the appendices, let us consider a simple pursuit-evasion example here. Suppose that there are N pursuers on the ground that are tracking N targets. We assume that the target assignment problem has been solved. Each pursuer depends on a UAV to obtain information about its target. Suppose there are n UAVs present. Each UAV chooses a pursuer at random with equal probability and transmits the corresponding target's location. We consider the targets to execute a random walk according to the equation

$$x_i(k+1) = 1.1x_i(k) + w_i(k),$$

where $w_i(k)$ is white noise with variance unity. Figure 4.8 shows a lower bound on the number of UAVs needed as the number of pursuers is increased to keep the expected error covariance (and hence the cost involved in tracking) bounded for all pursuers. We can also study the performance of this system using the tools presented above.

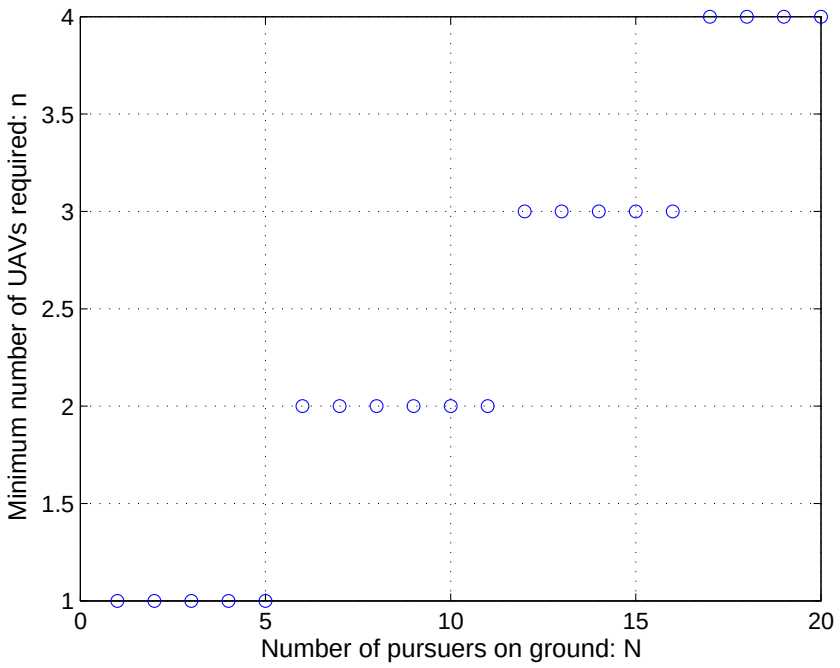


Figure 4.8: Minimum number of UAVs required as a function of the number of pursuers.

4.4 Encoding Information at the Sensor End

As we saw in Chapter 3, encoding information at the sensor end prior to transmission can yield large performance gains. For the problem considered in this chapter, so far we have assumed that the sensors were transmitting measurements without any encoding. If there is no packet loss and the reason the transmissions need to be scheduled is that only one sensor can measure per time step (e.g., sonars at the same frequency) then this strategy is optimal. However, if every sensor is observing the process at every time step even though only one sensor can transmit, it is natural to ask if the sensors can transmit more information than just the latest measurement that they have.

We can look at an algorithm whose performance upper bounds the performance achieved by any coding scheme. The best a sensor can do with any algorithm is to transmit all the measurements it has ever taken when it is its turn to transmit. Consider, as before, a process defined as in (4.1) being observed by N sensors of the form (4.2). We will use the Riccati operator defined in (4.4) and (4.5). The i-th sensor is used with probability q_i in an i.i.d. fashion. Assume, for simplicity, that there are no packet drops in the channel, although the packet drops can be taken care of by simply introducing one more sensor. We are interested in the expected error covariance $E[P(k)]$ at the estimator end.

Consider the set S of all possible combinations of $n \leq N-1$ sensors. For any element s of S, define C_s to be the corresponding sensing matrix and q_s to be the probability that one of the n

sensors is used. As an example, for the case of 3 sensors,

$$\begin{aligned} S &= \{1,2,3,12,13,23\} \\ C_s &\in \{C_1, C_2, C_3, \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}, \begin{bmatrix} C_1 \\ C_3 \end{bmatrix}, \begin{bmatrix} C_2 \\ C_3 \end{bmatrix}\} \\ q_s &\in \{q_1, q_2, q_3, q_1+q_2, q_1+q_3, q_2+q_3\}. \end{aligned}$$

We have the following result.

Proposition 4.14 *Consider the problem definition given above with each sensor transmitting all its previous measurements whenever it is its turn to transmit. A necessary and sufficient condition for $E[P(k)]$ to stay bounded as $k \to \infty$ is that the following set of equations hold:*

$$q_s \mid \bar{\lambda}(A_s) \mid^2 \leq 1 \qquad \forall s \in S,$$

where A_s is the unobservable part of A when the pair (A, C_s) is put in observer canonical form.

Proof For notational ease, we present the proof for $N = 2$ sensors. The case for general N is similar. We are interested in the behavior of $E[P(k)]$ for large k. Denote by $P^\star$ the steady state error covariance in estimating $x(k)$ given all measurements from both sensors till time step $k-1$. Since the convergence of the Ricatti recursion is exponential and we will be dealing with geometric series below, we will substitute $P^\star$ for error covariance for a large enough time k.

We can condition the event space for calculating $E[P(k)]$ into events of the form $\mathcal{E}_{mn}$ where the last time sensor 1 transmitted was at time step m and sensor 2 transmitted was at time step n. If we use the values $m = -1$ or $n = -1$ to denote that sensor 1 or sensor 2 have never transmitted, (m, n) can assume values

$$(m,n) \in \{(k,k-1),(k,k-2),\cdots,(k,-1),(k-1,k),(k-2,k),(k-3,k),\cdots,(-1,k)\}.$$

Accordingly, we can write

$$\begin{aligned} E[P(\infty)] &= \sum_{(m,n)} E[P(\infty)|\mathcal{E}_{mn}]\text{Prob}\left(\mathcal{E}_{mn}\right) \\ &= q_1q_2 f_{C_1}\left(P^\star\right) + q_1(1-q_2)q_2 f_{C_1}^2\left(P^\star\right) + q_1(1-q_2)^2 q_2 f_{C_1}^3\left(P^\star\right) + \cdots \\ &\quad + q_1q_2 f_{C_2}(P^\star) + q_2(1-q_1)q_1 f_{C_2}^2\left(P^\star\right) + q_2(1-q_1)^2 q_1 f_{C_2}^3\left(P^\star\right) + \cdots \\ &= q_1q_2 \sum_{j=0}^{\infty}(1-q_2)^j f_{C_1}^{j+1}\left(P^\star\right) + q_1q_2 \sum_{j=0}^{\infty}(1-q_1)^j f_{C_2}^{j+1}\left(P^\star\right). \end{aligned}$$

Following the arguments in the proof of Proposition 4.7 for each sum to be bounded, we obtain that

a necessary and sufficient condition for $E[P(\infty)]$ not to diverge is that the following conditions hold.

$$\begin{aligned}(1-q_2) \mid \lambda(\bar{A}_1) \mid^2 &\leq 1\\ (1-q_1) \mid \lambda(\bar{A}_2) \mid^2 &\leq 1.\end{aligned}$$

This completes the proof for $N = 2$. For general N the proof is similar. ∎

The conditions in the above proposition identify the necessary and sufficient probability region for the expected error covariance to be stable. These are, in particular, necessary for the case when only measurements are transmitted, which can be observed easily for the case $N = 2$ by comparing with the conditions in Proposition 4.7. The performance in terms of $E[P(k)]$ can also be calculated by similar conditioning.

Of course, this result is for the optimal algorithm that entails an increasing amount of data transmitted. The problem whether there exists a recursive algorithm that attains the above performance is still open. The algorithm that we identified in Section 3.6 can be adapted to this case. However, it no longer remains recursive. The reason is that to calculate the global covariance matrix P in the algorithm, the sensors need to know which sensors have been used at all times in the past. Every time a sensor is used, the situation is the same as if it had been used at all previous time steps as well. Thus, every sensor needs to store all previous global error covariance matrices and re-update them to obtain the new error covariance. This entails a large amount of memory since a sensor may not be used for a long time.

4.5 Distributed Control with Packet Losses

In this section, we take another look at the problem of distributed control that we considered in Chapter 2. Suppose there are N agents that aim at minimizing a joint cost function. In general, the optimal control law would require a fully connected topology in which every agent knows the state of all the agents. However, this may entail a lot of communication. An alternative strategy is for the agents to exchange information only at some time steps, while using this information to generate the control inputs at the times they are not able to communicate. This can also model a situation in which the agents are not able to exchange information because of packet losses by the channel.

Thus, the topology of the network switches between many states. We are interested in finding the optimal controller that minimizes a quadratic cost. We will assume that that if an agent A is unable to communicate with any other agent B, then no agent (including A) uses the current state value of agent A while calculating its control input. While the analysis we give below is generalizable to consider topologies where any subset of the nodes does not transmit, it is notationally and

pedagogically much easier to restrict ourselves to the case when the topology switches between two states: a fully centralized topology in which every agent communicates with everybody else and a completely disconnected one in which no agent is communicating. As in the sensor scheduling analysis, we assume that the topologies switch stochastically between each other, so as to be able to include the effect of switching due to packet loss.

The assumption about the state value of an agent being used in a binary manner (either all agents use it or none does) needs to be discussed. The assumption is admittedly rather limiting and the resulting controller would essentially be a centralized one. However, this is a first step towards building a theory of distributed control with communication losses. While one may view these results as the LQG counterpart of the H_∞ results presented in [175], it may be noted that the latter assumes that the controller is time-invariant, apart from switching to a different mode because of the packets being received or not at the current time step. We make no such assumption and solve for the optimal time-varying controller that takes into account the entire history of the packet drops.

4.5.1 Problem Formulation

Consider N agents present with the i-th agent having the state space description $x_i(k)$ and evolving as

$$x_i(k+1) = \bar{A}x_i(k) + \bar{B}u_i(k) + w_i(k),$$

where $w_i(k)$ is zero mean white Gaussian noise independent of all other $w_j(k)$'s. Thus, the entire system evolves as

$$x(k+1) = (I \otimes \bar{A})x(k) + (I \otimes \bar{B})u(k) + w(k),$$

where $x(k)$, $u(k)$ and $w(k)$ are system state, control input and noise vector obtained by stacking the individual $x_i(k)$'s, $u_i(k)$'s and $w_i(k)$'s respectively. We will denote

$$A = I \otimes \bar{A} \qquad\qquad B = I \otimes \bar{B}.$$

At any time step k, all the agents are able to exchange their state values with probability[4] p. Said another way, every agent knows the entire state vector $x(k)$ with a probability p at any time step k. With the remaining probability, the agents do not have any knowledge of the value of the state vector at that time step and they rely on the previous values of the state vector that were received to generate the control input at that time step. We prescribe an upper limit n such that if no state value has been received for n time steps, the agents evolve open loop, i.e., they do not apply any control $u_i(k)$. The results can be generalized to any other value of control input as well, that may

[4]The assumption about the exchange happening with the same probability at every time step is merely for simplicity. The analysis can easily be generalized to, for instance, Markov chain based models.

correspond to the agents switching to some safe maneuver and waiting for a successful transmission to restart the joint mission. The agents aim at minimizing a joint cost function

$$J_T = \sum_{k=0}^{T} E\left[x^T(k)Qx(k) + u^T(k)Ru(k)\right] + E\left[x^T(T+1)P^c(T+1)x(T+1)\right],$$

where the expectation is taken over the noises $w_i(k)$'s and the probability of communication p.

The system can thus be described as follows. A process evolves as

$$x(k+1) = Ax(k) + Bu(k) + w(k), \tag{4.19}$$

where $x(k)$ is the process state, $u(k)$ is the control input and $w(k)$ is the process noise assumed white, Gaussian, zero mean and covariance matrix R_w. The state of the process is observed by a sensor with observations of the form[5]

$$y(k) = Cx(k) + v(k),$$

where $v(k)$ is the measurement noise again assumed white, Gaussian, zero mean and covariance R_v. Further, the noises $w(k)$ and $v(k)$ are assumed independent of each other. We assume that the pair (A, B) is controllable and (A, C) is observable.

The measurements are transmitted over a packet erasure channel to all the agents. At every time step, the channel erases the measurement with a probability $(1-p)$ independently of all previous time steps. Thus, with a probability p at any time, the agents receive a packet containing the current measurement. There are three possibilities:

1. If the agents receive a packet at time step k, they calculate the control inputs $u_i(k)$'s, or equivalently the control input $u(k)$ based on all previous measurements that have been received up to and including the measurement at time k.

2. If the agents do not receive anything at time step k but they did receive a packet at some time j between $k-n$ and k, $u(k)$ is calculated based on all previous measurements received till and including time step j.

3. If the agents have not received anything since time $k-n-1$, the control input $u(k)$ is set to zero.

The agents aim at minimizing the finite-horizon cost function

$$J_T = \sum_{k=0}^{T} E\left[x^T(k)Qx(k) + u^T(k)Ru(k)\right] + E\left[x^T(T+1)P^c(T+1)x(T+1)\right].$$

[5]In Chapter 2, we had assumed C to be identity and noise $v(k)$ not to be present. This discussion is more general.

Two questions arise:

1. How to design the control inputs $\{u(k)\}$?

2. How does the performance of the system vary as a function of the parameters n and p?

4.5.2 Analysis

4.5.2.1 The Markov Chain Model

It is fairly easy to see that the situation can be modeled as a Markov chain with $n+2$ states. The first $n+1$ states correspond to the last packet being received at the actuator t time steps ago, for the values of $t = 0, 1, \cdots, n$ while the $n+2$-th state corresponds to the last packet being received t time steps ago, where $t \geq n+1$.

The Markov chain corresponds to the system evolving *backwards* in time. Thus, the initial probability of being in the i-th state is the probability that at time step T, the last packet was received i time steps ago. Since the packet drops are independently and identically distributed, for the first $n+1$ states, this probability is $p(1-p)^{i-1}$ while for the $n+2$-th state it is $1-\sum_{i=0}^{n} p(1-p)^{i-1}$. We will refer to this probability distribution set as $\{\pi^j(T)\}$, where the superscript j stands for the state of the Markov chain while the argument T refers to the fact that these probabilities correspond to the time step T of the system evolution. To calculate the probabilities at time step $T-1$, we need the transition probability matrix of the Markov chain. While the probabilities are straight-forward to compute, they are notationally cumbersome to describe for the general case. We will, instead, illustrate the probabilities for the cases $n = 0$ and 1. The general case is similar.[6]

For $n = 0$, there are two states in the chain: state 1 corresponds to packet not being lost at the current time step, while state 2 corresponds to packet being lost. If we denote q_{ij} to be the probability of being in state j at time step $k-1$ given that the system was in state i at time step k, then the transition probabilities are given by

$$q_{11} = 1-p \qquad q_{12} = p$$
$$q_{21} = 1-p \qquad q_{22} = p.$$

For $n = 1$, the situation is similar. There are three states: state 1 corresponds to the last packet that arrived being transmitted at the same time step, state 2 to the last packet being transmitted one time step ago and state 3 to it being transmitted more than 1 time step ago. We can thus

[6]It should be emphasized that even if the packets are being lost according to, say, a Gilbert-Elliot channel model, the framework given here applies. The values of the probabilities in the transition probability matrix calculated below would obviously differ.

evaluate

$$\begin{array}{lll} q_{11} = 1-p & q_{12} = p(1-p) & q_{13} = p^2 \\ q_{21} = 1 & q_{22} = 0 & q_{23} = 0 \\ q_{31} = 0 & q_{32} = 1-p & q_{33} = p. \end{array}$$

Having defined the Markov chain model, we now proceed to the analysis of the cost function.

4.5.2.2 A Closer Look at the Cost Function

We begin by extracting the terms dependent on $x(T)$ and $u(T)$. We can write them as

$$\Upsilon(T) = E\left[x^T(T)Qx(T) + u^T(T)Ru(T) + x^T(T+1)P^c(T+1)x(T+1)\right].$$

We can condition $\Upsilon(T)$ on the event that the Markov state is in the state i at time step T. Let us denote this event by $m(T) = i$. Thus,

$$\Upsilon(T) = \sum_{i=1}^{n+2} \pi^i(T)\Upsilon^i(T) \tag{4.20}$$

where

$$\Upsilon^i(T) \;=\; E\big[x^T(T)Qx(T) \;+\; u^T(T)Ru(T) \;+\; x^T(T \;+\; 1)P^{c,i}(T \;+\; 1)x(T \;+\; 1)|m(T) \;=\; i\big]$$

and we have denoted the quantity $P^c(T+1)$ entering the i-th term in the summation as $P^{c,i}(T+1)$. The state i fixes the information set available to the controller and hence the control input $u(T)$. Let us denote the control input calculated for the time step k given the received measurements till time j as $u(k,j)$. Then, for $i = 1, \cdots, n+1$,

$$u(T) = u(T, T-i+1),$$

while for $i = n+2$, $u(T) = 0$. To see what the terms $u(T, T-i+1)$ should be, let us isolate the corresponding term $\Upsilon^i(T)$ from the summation. We have

$$\begin{aligned} \Upsilon^i(T) = E\Big[x^T(T)Qx(T) + u^T(T)Ru(T) + (Ax(T) + Bu(T, T-i+1) \\ + w(T))^T P^{c,i}(T+1)(Ax(T) + Bu(T, T-i+1) + w(T))\Big], \end{aligned}$$

and finally as

$$\Upsilon^i(T) = E\Big[\big(u(T,T-i+1) + (\Delta^i)^{-1}B^T P^{c,i}(T+1)Ax(T)\big)^T \Delta^i \\ \big(u(T,T-i+1) + (\Delta^i)^{-1}B^T P^{c,i}(T+1)Ax(T)\big) + w^T(T)P^{c,i}(T+1)w(T) \\ + x^T(T)\left(Q + A^T P^{c,i}(T+1)A - P^{c,i}(T+1)B(\Delta^i)^{-1}B^T P^{c,i}(T+1)\right)x(T)\Big],$$

where we have used the fact that the noise $w(T)$ is zero mean and have denoted

$$\Delta^i = R + B^T P^{c,i}(T+1)B.$$

Thus, it is apparent that $u(T,T-i+1)$ should be chosen so as to minimize the mean squared error

$$E\Big[\big(u(T,T-i+1) + (\Delta^i)^{-1}B^T P^{c,i}(T+1)Ax(T)\big)^T \Delta^i \\ \big(u(T,T-i+1) + (\Delta^i)^{-1}B^T P^{c,i}(T+1)Ax(T)\big)\Big].$$

Thus, based on all the measurements received till the time step $T-i+1$, the agents should calculate the minimum mean squared estimate of $x(T)$ and then multiply it by the matrix $(\Delta^i)^{-1}B^T P^{c,i}(T+1)A$ to determine $u(T,T-i+1)$. Let us denote the corresponding error covariance incurred as $\Lambda^i(T)$. Note that while calculating $u(T,T-i+1)$, every agent knows all the control inputs applied by the system till time step $T-i$. Further, if the input $u(T,T-i+1)$ is used at time step T, the agents know that no packet was transferred successfully after time step $T-i+1$. Hence, they can also determine the control inputs applied from time $T-i+1$ till time $T-1$. Thus, they know all the previous control inputs while estimating $x(T)$. As a result, $\Lambda^i(T)$ is independent of all previous control inputs.

With the optimizing choice of $u(T,T-i+1)$, the term $\Upsilon^i(T)$ becomes

$$\Upsilon^i(T) = \Lambda^i(T) + E\Big[w^T(T)P^{c,i}(T+1)w(T) \\ + x^T(T)\left(Q + A^T P^{c,i}(T+1)A - A^T P^{c,i}(T+1)B(\Delta^i)^{-1}B^T P^{c,i}(T+1)A\right)x(T)\Big].$$

For ease of notation, for the values of $i = 1, \cdots, N+1$, let us define an operation $f^i(.)$ as

$$f^i(X) = Q + A^T XA - A^T XB(R + B^T XB)^{-1}B^T XA.$$

Thus,

$$\Upsilon^i(T) = \Lambda^i(T) + E\left[w^T(T)P^{c,i}(T+1)w(T) + x^T(T)f^i\left(P^{c,i}(T+1)\right)x(T)\right].$$

This form of $\Upsilon^i(T)$ holds for $i = 1, \cdots, n+1$. For $i = n+2$, $u(T) = 0$ and

$$\begin{aligned}\Upsilon^{n+2}(T) &= E\left[x^T(T)Qx(T) + (Ax(T) + w(T))^T P^{c,i}(T+1)(Ax(T) + w(T))\right] \\ &= E\left[w^T(T)P^{c,i}(T+1)w(T) + x^T(T)(Q + A^T P^{c,i}(T+1)A)x(T)\right].\end{aligned}$$

Thus, for the optimizing choice of control inputs at time step $T+1$, we can finally write

$$\begin{aligned}\Upsilon(T) &= \sum_{i=1}^{n+2} \pi^i(T)\Upsilon^i(T) \\ &= \sum_{i=1}^{n+2} \pi^i(T)E\left[w^T(T)P^{c,i}(T+1)w(T)\right] + \sum_{i=1}^{n+2} \pi^i(T)\Lambda^i(T) \\ &\quad + \sum_{i=1}^{n+2} \pi^i(T)E\left[x^T(T)f^i\left(P^{c,i}(T+1)\right)x(T)\right],\end{aligned}$$

where we have defined

$$\begin{aligned}\Lambda^{n+2}(T) &= 0 \\ f^{n+2}(X) &= Q + A^T X A.\end{aligned}$$

The cost function after choosing the control inputs at time K optimally can thus be rewritten as

$$\begin{aligned}J_T = \sum_{k=0}^{T-1} E\left[x^T(k)Qx(k) + u^T(k)Ru(k)\right] + \sum_{i=1}^{n+2} \pi^i(T)E\left[w^T(T)P^{c,i}(T+1)w(T)\right] \\ + \sum_{i=1}^{n+2} \pi^i(T)\Lambda^i(T) + \sum_{i=1}^{n+2} \pi^i(T)E\left[x^T(T)f^i\left(P^{c,i}(T+1)\right)x(T)\right].\end{aligned}$$

The second summation involves only the noise terms and thus cannot be affected by the choice of the control inputs. The third summation involves the estimation error covariance incurred while calculating $u(T)$ and, as explained earlier, this term is also independent of all previous control input choices. Thus, to optimally choose all the control inputs from time 0 to time $T-1$, the agents only need to consider the terms forming the first and the fourth summations. Let us take a closer look

at the fourth summation and denote it by $\Gamma(T)$. We have

$$\begin{aligned}
\Gamma(T) &= \sum_{i=1}^{n+2} \pi^i(T) E\left[x^T(T) f^i\left(P^{c,i}(T+1)\right) x(T)\right] \\
&= \sum_{i=1}^{n+2}\sum_{j=1}^{n+2} \pi^i(T) q_{ij} E\left[x^T(T) f^i\left(P^{c,i}(T+1)\right) x(T) | m(T-1)=j\right] \\
&= \sum_{j=1}^{n+2}\sum_{i=1}^{n+2} \pi^i(T) q_{ij} E\left[x^T(T) f^i\left(P^{c,i}(T+1)\right) x(T) | m(T-1)=j\right] \\
&= \sum_{j=1}^{n+2} E\left[x^T(T)\left(\sum_{i=1}^{n+2} \pi^i(T) q_{ij} f^i\left(P^i(T+1)\right)\right) x(T) | m(T-1)=j\right].
\end{aligned}$$

Let us define

$$\pi^j(T-1)P^{c,j}(T) = \sum_{i=1}^{n+2} \pi^i(T) q_{ij} f^i\left(P^{c,i}(T+1)\right).$$

Thus,

$$\begin{aligned}
\Gamma(T) &= \sum_{j=1}^{n+2} E\left[x^T(T)\left(\pi^j(T-1)P^{c,j}(T)\right) x(T) | m(T-1)=j\right] \\
&= \sum_{j=1}^{n+2} \pi^j(T-1) E\left[x^T(T) P^{c,j}(T) x(T) | m(T-1)=j\right].
\end{aligned}$$

Finally, the cost function can be written as

$$\begin{aligned}
J_T &= \sum_{k=0}^{T-1} E\left[x^T(k)Qx(k) + u^T(k)Ru(k)\right] + \sum_{j=1}^{n+2} \pi^j(T-1) E\left[x^T(T)P^{c,j}(T)x(T) | m(T-1)=j\right] \\
&= \sum_{j=1}^{n+2} \pi^j(T-1)\left[\sum_{k=0}^{T-1} E\left[x^T(k)Qx(k) + u^T(k)Ru(k) + x^T(T)P^{c,j}(T)x(T) | m(T-1)=j\right]\right],
\end{aligned}$$

where we have ignored the second and the third summations that play no role in further minimization. But now we can again extract the term $\Upsilon(T-1)$ in a form similar to the one in (4.20) and our argument from then on did not rely on the time index T. Thus, we can carry out a similar argument to evaluate the optimal control inputs at all time steps and the resulting cost function. We have, in effect, proven a separation principle in the problem setting we are considering.

Proposition 4.15 *Consider the problem setting described in Section 4.5.1. The optimal value of control input $u(k,j)$, i.e., the control input to be applied at time k given that the last measurement was received at time step $j \geq k-n$ is given by*

$$u(k,j) = F(k)\hat{x}(k|j) = \left(\Delta^i\right)^{-1} B^T P^{c,i}(k+1) A\hat{x}(k|j),$$

where

$$\begin{aligned} \Delta^i &= R + B^T P^{c,i}(k+1)B \\ \pi^j(k-1)P^{c,j}(k) &= \sum_{i=1}^{n+2} \pi^i(k) q_{ij} f^i\left(P^{c,i}(k+1)\right), \end{aligned}$$

$P^{c,i}(T+1) = P^c(T+1)$ *and* $\hat{x}(k|j)$ *is the minimum mean squared error estimate of the state* $x(k)$ *of the system given all the received measurements till time step* j *and control inputs till time step* $k-1$.

4.5.2.3 Stability and Performance Analysis

We can consider the optimal cost as the time horizon T becomes larger. For the infinite time horizon problem, we will consider the cost

$$J(\infty) = \lim_{T\to\infty} \frac{1}{T} J(T).$$

As we have assumed in this dissertation, the system is stable if this cost is bounded. Looking at the analysis in Section 4.5.2.2, there are two reasons that the cost may grow unbounded:

1. The terms $\Lambda^i(k)$ grow unbounded. This is a function of how fast does the estimation error grow.

2. The terms $P^{j,c}(0)$ grow unbounded. This is a function of the how fast does the system grow if uncontrolled.

We will now consider these two effects.

Let us begin with the terms $\Lambda^i(k)$. $\Lambda^i(k)$ represents the estimation error covariance incurred when the control input for time step k is calculated based on measurements received till time $k-i+1$. However, since there is always a measurement received at time $k-i+1$, this quantity can never diverge. Stated another way, even though the estimation error can grow worse as no measurements are received for a long time, the measurement received at time $k-i+1$ always keeps it finite. Thus, the effect of $\Lambda^i(k)$ shows up only in the performance analysis and not in the stability of the system.

For the terms $P^{j,c}(k)$, let us first identify the recursions according to which these terms evolve. These terms evolve *backwards* in time according to the coupled equations

$$\pi^i(k-1)P^{j,c}(k) = \sum_{i=1}^{n+2} \pi^i(k) q_{ij} f^i\left(P^{c,i}(k+1)\right), \tag{4.21}$$

where $\pi^j(k)$ is the probability of being in state j at time k, q_{ij} is the transition probability of being in state j at time step $k-1$ given that the state at time state k was i and the operators $f^i(.)$ have been defined earlier. The initial values are $P^{c,i}(T+1) = P(T+1) \quad \forall i$. The behavior of these

equations has been analyzed in Section 4.2.3.1. In particular we can prove the following result along the lines of Proposition 4.9.

Proposition 4.16 *Consider the recursion defined in (4.21). Assume that the Markov chain transition probability matrix is such that the states reach a stationary probability distribution with the probability of being in the j-th state as π^j. Further assume that all π^j's are strictly positive. If there exist $n+2$ positive definite matrices X^1, X^2, $\cdots$, X^{n+2} and $(n+2)^2$ matrices $K^{1,1}$, $K^{1,2}$, $\cdots$, $K^{1,n+2}$, $K^{2,1}$, $\cdots$, $K^{n+2,n+2}$ such that*

$$\pi^j X^j > \sum_{i=1}^{n+2}\Big((A^T + K^{ij}B^T)X^i(A^T + K^{ij}B^T)^T + Q + K^{ij}R(K^{ij})^T\Big)q_{ij}\pi^i,$$

then (4.21) converges for all initial conditions $X^i(T+1) > 0$ and the limit $\bar{X}^j$ is the unique positive semi-definite solution of the equation

$$\pi^j X^j = \sum_{i=1}^{n+2} f^j\left(X^i\right) q_{ij}\pi^i. \tag{4.22}$$

Proof Proof is along the lines of the proof given for Proposition 4.9 and is omitted. ∎

This result provides a sufficient condition for stability. For a necessary condition, we can use the stability of the lower bound treated in Proposition 4.11. Thus, we can immediately prove the following result.

Proposition 4.17 *Let the probability of choosing the $n+2$-th state at two consecutive time steps be $q_{n+2,n+2}$. Denote $\rho(A)$ as the spectral radius of matrix A. Then, a sufficient condition for the expected estimate error covariance to diverge from at least one initial value is given by*

$$q_{n+2,n+2}|\rho(A)|^2 > 1.$$

Since we can evaluate the terms $P^{j,c}(k)$ exactly through recursion and obtain bounds on the terms $\Lambda^i(k)$ from our work in Section 4.2.3, we can obtain lower and upper bounds on the cost J_T as well. The details are omitted.

4.6 Discussion

In this chapter, we considered problems that had both the features of multiple components and imperfect communication links present. We began by analyzing a situation in which sensors were transmitting information to an estimator according to a stochastic schedule. Then, we used this analysis to propose a sensor scheduling algorithm that could explicitly include the effects of communication channels. We saw that the algorithm provides a useful tool in many situations and has

unique advantages over the tree-search based algorithms proposed in the literature. However, there is more left to do. We do not yet have an understanding of how tight the bounds that we are using for our algorithm are. Moreover, we are optimizing a steady state criterion and can not yet include the transient effects. While it seems that a greedy algorithm may alleviate this problem, more work needs to be done.

We also characterized the behavior of an optimal encoding algorithm in which the sensors transmit all their measurements at every time step. However, the design of a recursive algorithm to achieve this performance is still open. We also looked at a control problem with stochastically switching topologies. While the results were interesting, they still assumed that if an agent cannot communicate, no agent can access its state value. It would be nice to remove this assumption and consider more general topologies in which an agent may be able to communicate with some, but not all, other agents at any time step. A step towards that may be to use the sub-optimal distributed control algorithm described in Chapter 2 and invoke some of the tools introduced above to consider switching topologies.

The tools that we have developed are more widely applicable than the specific problems considered so far. We demonstrate two such applications in the appendices.

Appendix A: Dynamic Sensor Coverage

In this appendix we consider one application of the tools developed in the chapter. Consider a geographical area in which various regions need to be patrolled. We study the problem of using a small number of mobile sensors to effectively cover the entire area. Using the results presented in the main part of the chapter, we propose a stochastic sensor movement strategy. We also present simple conditions under which it is not possible to maintain a bounded estimate error covariance for all the threats.

Let the geographical region that needs to be monitored be divided into a grid of N points. There are dynamical processes occurring at these points whose state we want to estimate. Denote the state at the i-th point at time k by $x_i(k)$. The process at the i-th point is driven by $w_i(k)$, assumed to be zero mean, white and Gaussian with covariance matrix $R_{w,i}$. We consider two distinct cases:

1. Coupled processes: The processes at points i and j are coupled. Thus, the process at a point i evolves as
$$x_i(k+1) = A_i x_i(k) + \sum_{j \neq i} A_{i,j} x_j(k) + w_i(k),$$
where all the matrices $A_{i,j}$ are not zero.

2. Uncoupled processes: The processes at distinct points i and j are unaffected by each other. Thus, all matrices $A_{i,j}$ are zero.

Denote the state of the entire region obtained by stacking all $x_i(k)$'s as $x(k)$. Then, $x(k)$ evolves according to the equation
$$x(k+1) = Ax(k) + w(k),$$
where $w(k)$ is the vector formed by stacking $w_i(k)$'s and is assumed to have covariance R_w. If the processes are uncoupled, A is a (block) diagonal matrix with A_i's along the diagonal. If the processes are coupled, A is, in general, a full matrix.

The region is monitored using n sensors. If the m-th sensor is at point i at time k, it generates the measurement
$$y_m(k) = x_i(k) + v_m(k), \tag{4.23}$$
where $v_m(k)$ is zero mean white Gaussian noise with covariance $R_{v,m}$, assumed independent of all other noises present. This can be rewritten as an observation of the state $x(k)$ as
$$y_m(k) = C_i x(k) + v_m(k),$$
where C_i is a row vector with zeros everywhere except the i-th element which is replaced by 1.[7] This

[7] This description of C_i assumes the states $x_i(k)$'s to be scalars. The extension to the vector case is obvious. Similarly we can consider a sensing matrix being present in (4.23).

gives rise to the concept of a virtual sensor. A *physical* sensor at point i gives rise to a *virtual* sensor being used with sensing matrix C_i. This can obviously be generalized to more than one physical sensor. As an example, if there are physical sensors at points i and j, we will say that a virtual sensor is being used that has a sensing matrix with rows C_i and C_j.

If there is more than one physical sensor present, we assume that all measurements are exchanged without delay or distortion. Thus, based on all the measurements, any physical sensor can compute an estimate of the state $x(k)$.[8] Let the estimate be denoted by $\hat{x}(k)$. Also, let $P(k)$ denote the covariance of the estimate error. It is obvious that the error covariance is a function of the sensor schedule for the virtual sensor, or the trajectory of the physical sensor. There are two basic problems that arise.

1. What conditions should the sensor trajectories satisfy to guarantee that $P(k)$ is bounded as time k increases? Also, how many sensors should be present?

2. What is the optimal trajectory that minimizes the steady-state value of $P(k)$?

As we discussed in the context of sensor scheduling in the main body of the chapter, we can represent all the possible sensor schedule choices for the virtual sensor by a tree structure. Finding *the* optimal sequence requires traversing all the paths from the root to the leaves in the tree. This procedure might place too high a demand on the computational and memory resources of the system. We are instead interested in stochastic trajectories, i.e., the sensors choose their positions at any time step at random according to a probability distribution. The probability distribution is chosen so as to minimize the expected steady state error covariance. Note that we cannot calculate the exact value of the error covariance since that will depend on the specific sensor schedule chosen.

In this appendix, we assume that the sensor trajectories are designed independent of each other. There are two particular cases of sensor motion that we will study:

1. The choice for the position of the j-th sensor at time step $k+1$ is done in an i.i.d. fashion at each time step with probability q_i of being at the i-th point.

2. The choice is done according to a Markov chain with transition probability matrix Q. This can model physical constraints on the sensor motion, e.g., the probability q_{ij} is 0 if i and j are points that are physically distant from each other.

Note that we have assumed that each sensor chooses its trajectory according to the same parameters (probabilities q_i's or the transition probability matrix Q). We will say that the coverage problem can be solved if there exists at least one choice of parameters such that beginning from any initial condition $P(0)$, the expected error covariance remains bounded as time progresses. If there exists no such choice of parameters, we say that the problem cannot be solved.

[8] Since every sensor has access to the same information set, they would have identical estimates.

In the above description, we have assumed that all the agents are interested in coming up with an estimate for the processes in the area. If the data was being transmitted to a central data processing node, we can use the same framework. In fact, in this case, we can also allow for communication channel imperfections. As an example, if the communication channel loses packets stochastically, we can model the time instants at which data loss occurs as being used up by a *fictitious* (as opposed to physical and virtual) sensor which has sensing matrix 0. The data can be lost in either an i.i.d. or a Markovian fashion.

Having defined the problem set-up, we now move on to analyze it. We will consider the problem with various assumptions which will be clear before each result. We begin by presenting a set of impossibility results, i.e., conditions on the matrix A such that the expected error covariance cannot be bounded for any probability distribution. These results also present a bound on the minimum number of sensors that need to be present. Some of the results for uncoupled processes were also presented in [187].

Motion governed by i.i.d. choices

In this section, we will consider the case when each sensor is choosing the next point to move to in an i.i.d. fashion. Let the probability of the sensor m being at point i at time k be given by q_i. We begin with the case when the dynamics of the processes at various points are uncoupled.

Uncoupled processes

As defined above, let A_i denote the system evolution matrix of the process at the i-th location. Let λ_i refer to the eigenvalue with the maximum magnitude of the matrix A_i. Without loss of generality, we can assume that

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_N|. \tag{4.24}$$

We can prove the following result.

Proposition 4.18 *Consider the sensor coverage problem when N physical points are to be patrolled by one sensor. If the processes at the different points are uncoupled and (4.24) holds, then the problem cannot be solved if the following condition holds for any $1 \leq m \leq N-1$,*

$$\frac{\binom{N-1}{m}}{|\lambda_1|^2} + \frac{\binom{N-2}{m-1}}{|\lambda_2|^2} + \cdots + \frac{\binom{N-1-m}{0}}{|\lambda_{m+1}|^2} < \binom{N-1}{m-1}. \tag{4.25}$$

Proof The m-th sufficient condition is obtained by considering all virtual sensors formed by considering sets of m points. We say that a virtual sensor is used if the physical sensor is present at any point in the set of m points that the virtual sensor represents. There are $\binom{N}{m}$ such virtual sensors. Denote the probability of choosing the j-th virtual sensor by p_j. For a virtual sensor with the set of

m points denoted by $\mathcal{M}$, p_j is of the form

$$p_j = \sum_{t \in \mathcal{M}} q_t.$$

For each virtual sensor, let the lowest i which is not included in its set of m points be i_{min}. Then, the condition for stability when that virtual sensor is used is (using the condition in Proposition 4.7)

$$p_j |\lambda_{i_{min}}|^2 < 1.$$

Simple algebra yields that λ_t occurs in $\binom{N-t}{m-t+1}$ such inequalities. Adding all the inequalities together, we obtain that at least one inequality will be violated if

$$\frac{\binom{N-1}{m}}{|\lambda_1|^2} + \frac{\binom{N-2}{m-1}}{|\lambda_2|^2} + \cdots + \frac{\binom{N-1-m}{0}}{|\lambda_{m+1}|^2} < \binom{N-1}{m-1}.$$

Considering different values of m, we obtain the result. ∎

For specific values of m, the condition in equation (4.25) looks as follows. For $m = 1$, the condition is

$$\frac{N-1}{|\lambda_1|^2} + \frac{1}{|\lambda_2|^2} < 1. \tag{4.26}$$

For $m = N - 1$, the condition is

$$\sum_{i=1}^{N} \frac{1}{|\lambda_i|^2} < N - 1. \tag{4.27}$$

Neither of the conditions is more general. As an example, for a system with $\lambda_1 = 2$, $\lambda_2 = \sqrt{3}$, $\lambda_3 = 1/\sqrt{2}$ the problem is predicted to be unsolvable by (4.26) but not by (4.27). The opposite is true for a system with $\lambda_1 = \lambda_2 = \lambda_3 = \sqrt{2}$. Hence, both the conditions are useful.

We now move on to the case when there is more than one physical sensor, i.e., $n > 1$. To begin with, consider the case of only two points to be patrolled, i.e., $N = 2$.

Proposition 4.19 *If $N = 2$ points have to be patrolled by n sensors with the assumptions stated above, the sensor coverage problem cannot be satisfied if*

$$\frac{1}{|\lambda_1|^{\frac{2}{n}}} + \frac{1}{|\lambda_2|^{\frac{2}{n}}} < 1.$$

Proof There are 2^n virtual sensors in this case, corresponding to the n physical sensors being present at either of the two points. When both the points are covered by at least one physical sensor, the entire system matrix A is observed. There are two cases when A is not observed

1. all the physical sensors are located at the first point. This event occurs with a probability $(q_1)^n$; or

2. all the physical sensors are located at the second point. This occurs with a probability $(q_2)^n$ or $(1-q_1)^n$.

Thus, the conditions for covariance of the error to diverge are for any one of the following inequalities to be true,

$$\begin{aligned}(q_1)^n|\lambda_2|^2 &> 1\\ (1-q_1)^n|\lambda_1|^2 &> 1.\end{aligned}$$

Adding the inequalities completes the proof. ∎

Combining the proof technique of Propositions 4.18 and 4.19 immediately leads to the generalization stated below.

Proposition 4.20 *The sensor coverage problem when N physical points are to be patrolled using n sensors, but otherwise the same assumptions hold as above, cannot be solved if the following sufficient condition holds for any $1 \leq m \leq N-1$,*

$$\frac{\binom{N-1}{m}}{|\lambda_1|^{\frac{2}{n}}} + \frac{\binom{N-2}{m-1}}{|\lambda_2|^{\frac{2}{n}}} + \cdots + \frac{\binom{N-1-m}{0}}{|\lambda_{m+1}|^{\frac{2}{n}}} < \binom{N-1}{m-1}. \tag{4.28}$$

Proof Proof follows that of the proposition 4.18. The m-th sufficient condition is obtained by considering all virtual sensors formed by considering sets of m points. We say that a virtual sensor is used if no physical sensor is present outside the set of m points the virtual sensor represents. There are $\binom{N}{m}$ such virtual sensors. Denote the probability of choosing the j-th virtual sensor by p_j. For a virtual sensor with the set of m points denoted by $\mathcal{M}$, p_j is of the form

$$p_j = \left(\sum_{t\in\mathcal{M}} q_t\right)^n.$$

For each virtual sensor, let the lowest i which is not included in its set of m points be i_{min}. Then, the condition for stability when that virtual sensor is used is $p_j|\lambda_{i_{min}}|^2 < 1$. Simple algebra yields that λ_t occurs in $\binom{N-t}{m-t+1}$ such inequalities. Adding all the inequalities together, we obtain that at least one inequality will be violated if (4.28) holds. Considering various values of m, we obtain the result. ∎

Note that we recover the results of previous propositions for the special cases when $N = 2$ and when $n = 1$. We now consider the case of coupled processes.

Coupled processes

Let the process at the i-th point evolve as

$$x_i(k+1) = A_i x_i(k) + \sum_{j \neq i} A_{i,j} x_j(k) + w_i(k).$$

As long as all $A_{i,j}$'s are not zero, it is possible to obtain information about $x_j(k)$ even though the sensor is at point i. Moreover, the eigenvalues of the unobservable modes when considering two physical sensors located at points i and j may have no relation to the eigenvalues when the sensors are at points i and k. These facts make the analysis a bit more involved in this case. We give a general result below and then consider some special cases. As defined earlier, let A be the system evolution matrix for the vector $x(k)$ formed by stacking all $x_i(k)$'s. Similarly, let C_i be the observation matrix relating to the observation from a sensor at point i to $x(k)$. We will consider virtual sensors formed by sets of m physical points and say that a virtual sensor is used if none of the physical sensors are located outside the specified m points. Denote the set of all such virtual sensors by $\mathcal{S}_m$. For any member M of this set, consider the sensing matrix C_M formed by stacking all the C_i's such that i belongs to the set of m points corresponding to M. Denote by α_M the eigenvalue with the maximum magnitude of the unobservable part of A when the pair (A, C_M) is put in the observer canonical form.

Proposition 4.21 *The sensor coverage problem for N physical points and n sensors with the above assumptions cannot be satisfied if for any m such that $1 \leq m \leq N-1$,*

$$\sum_{M \in \mathcal{S}_m} \frac{1}{|\alpha_M|^{\frac{2}{n}}} < \binom{N-1}{m-1}.$$

Proof Proof follows exactly along the lines of that of Proposition 4.20. The difference is that for every virtual sensor, we have to consider the eigenvalue with the maximum magnitude of the unobservable part of the system matrix A when that virtual sensor is used. In Proposition 4.20, this was related to the eigenvalues of A, because A was (block) diagonal. However, in the present case, this relation is lost and we have to calculate the values of α_M for every virtual sensor. However, other details of the proof are identical. ∎

Perhaps the easiest version of the result is obtained by considering $m = 1$. If we define β_i as the eigenvalue with the maximum magnitude of the unobservable part of A when (A, C_i) is put in the observable canonical form, the condition reduces to

$$\sum_{i=1}^{N} \frac{1}{|\beta_i|^{\frac{2}{n}}} < 1.$$

For the case when the systems at various points are uncoupled, A becomes (block) diagonal and Proposition 4.21 reduces to Proposition 4.20.

Motion governed by a Markov chain

We now consider the more general case where each sensor decides its position at time step $k+1$ according to its position at time step k by using a transition probability matrix Q. This can also model the case when data loss due to the communication channels is occurring according to a Markov chain. We still assume that the various sensors act independently. We assume that the Markov chain is positive recurrent and irreducible, thus there exists a unique stationary distribution [48]. Let π_i denote the stationary probability of being in the i-th state. The general result statement along the lines of Proposition 4.21 is now presented. Define α_M as before.

Proposition 4.22 *The sensor coverage problem for N physical points and n sensors with the above assumptions cannot be satisfied if for any m such that $1 \leq m \leq N-1$, any of the following $\binom{N}{m}$ conditions are satisfied*

$$\frac{1}{\sum_{j\in M}\pi_j}\left[\sum_{i\in M}\sum_{j\in M} q_{ij}\pi_i\right] > \frac{1}{|\alpha_M|^{\frac{2}{n}}}.$$

Proof Proof is along the lines of Proposition 4.21. The only trick is in the calculation of the probability q_{ii} for the i-th virtual sensor. For a Markov chain with transition probability matrix Q and a set of states S, the probability that the state at time $k+1$ belongs to S given that the state at time k belonged to S is given by the expression

$$\frac{1}{\sum_{j\in S}\pi_j(k)}\left[\sum_{t\in S}\sum_{j\in S} q_{tj}\pi_t(k)\right],$$

where $\pi_t(k)$ is the probability of being in state t at time k. If the Markov chain reaches a unique stationary distribution, $\pi_t(k) \to \pi_t$ for large enough k. Thus, the terms q_{ii} can be evaluated. The rest of the proof is along similar lines as of proposition 4.21 and is omitted. ∎

Remarks

1. One special case is when the sensors are chosen in an i.i.d. fashion. This case requires a transition probability matrix with the property that $q_{ij} = q_j$ for all pairs (i,j). In this case, the conditions in proposition 4.22 reduce to

$$\sum_{j\in M} q_j > \frac{1}{|\alpha_M|^{\frac{2}{n}}}.$$

Summing all $\binom{N}{m}$ inequalities obtained by the various choices of M yields the condition in proposition 4.21.

2. As in the case of sensors being chosen in an i.i.d. fashion, when the processes at various points are uncoupled, the terms α_M are expressible in terms of the eigenvalues with the maximum magnitude λ_i of the processes at the various points. Let the points be numbered such that

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_N|.$$

Let $\bar{M}$ be the set of points $i \in \{1, 2, \cdots, N\}$ that are not contained in M. Then, $\alpha_M = \lambda_i$, where $i = \min_{j \in \bar{M}}(j)$.

3. The choice $m = 1$ yields the N conditions

$$q_{ii} > \frac{1}{|\beta_i|^{\frac{2}{n}}},$$

where β_i is the eigenvalue with the maximum magnitude of the unobservable part of A when the pair (A, C_i) is put in the observable canonical form.

So far, we have dealt with obtaining conditions on the processes so that sensor coverage is not possible. We can also design the optimal sensor trajectory using the algorithms presented in Section 4.3.2. We omit the details.

Examples

We now illustrate our results with the help of some simple examples. As the first example, consider a grid of 6 points such that the value at each point represents a flow traveling from the first node towards the sixth node. Thus, the dynamic equation at points 2 through 6 is given by

$$x_i(k+1) = x_{i-1}(k) + w_i(k),$$

while for point 1 it is given by

$$x_1(k+1) = w_1(k).$$

We assume that the covariance matrix of the noise $w_i(k)$ is $R_i = 0.5$. Consider only one sensor of the form (4.23) with the measurement noise covariance matrix $R_m = 0.1$ that chooses its position independently from one time step to the next. There are 6 virtual sensors with the sensing matrix of the i-th sensor, C_i, being a row vector with all zeros except a 1 at the i-th place. The noise covariance matrix for all the sensors is R_m. The process matrix A is a 6×6 identity matrix. Let q_i denote the probability of its being at the i-th point. Naïvely, we may assume that the optimal

probability distribution would be either to spend a lot of time at the source, i.e., the first node or equally among all the nodes. However, if we optimize the probability distribution, it turns out that for the optimal distribution, $q_3 = 1$. The optimal cost is 6.28. If the sensor spends all its time at the source node, the cost is 8.42 while for a strategy of spending time with the same probability at all the points, it is 8.23. If we use a greedy strategy in which the sensor moves to minimize the cost at every time-step, it leads to the sensor spending all its time at the fourth point, leading to a cost of 6.69. Thus, our algorithm performs better than heuristic or greedy strategies.

As our second example we choose a ring network of 4 agents in which each agent is trying to calculate the average of the values of all the agents. Thus, for the i-th agent

$$x_i(k+1) = x_i(k) - h(2x_i(k) - x_{i+1}(k) - x_{i-1}(k)) + w_i(k), \tag{4.29}$$

where the addition in the agent number i is done modulo 4 and h is a positive constant. For a small enough value of h, the agents will calculate the average if no noise were present. We assume the noises $w_i(k)$ to be independent of each other and with variance $R_1 = R_4 = 1$ and $R_2 = R_3 = 0.8$. We again assume that there is only one sensor that is choosing its position in an i.i.d. fashion. We will consider the value $h = 0.2$. We use the gradient descent algorithm with initial probability distribution $q_1 = q_2 = 0.5$ and a step size of 0.01. On optimizing the distribution, the values turn out to $q_1 = q_4 = 0.3$ and $q_2 = q_3 = 0.2$ with an optimal cost (upper bound) of 5.81. Indeed if we run 10000 random runs of the system generating sensor switching with this probability, we obtain a mean steady state error covariance trace of 5.8. Hence, the upper bound is pretty tight at least in this example. If we plot a histogram of the steady state costs along different runs, we obtain the figure shown in Figure 4.9. We can see that the spread of the mean costs is not huge either. Our algorithm is also useful if there is some stochasticity already present in the system. Let the sensor transmit measurements to a central data processing station. However, the measurements are dropped with probability λ. Figure 4.10 shows the optimal probability of using the sensor at the fourth point as a function of λ. It can be seen that the value of λ has a huge effect.

We can also impose the restriction that the sensor can only move from one physical point to its neighbors. Thus, the sensor positions are chosen according to a Markov chain. Let us assume no packet loss for simplicity. Because of the symmetry of the system, we look for transition probability matrices of the form

$$\begin{bmatrix} 1-2\lambda_1 & \lambda_1 & 0 & \lambda_1 \\ \lambda_2 & 1-2\lambda_2 & \lambda_2 & 0 \\ 0 & \lambda_2 & 1-2\lambda_2 & \lambda_2 \\ \lambda_1 & 0 & \lambda_1 & 1-2\lambda_1 \end{bmatrix}.$$

Then, the optimal parameters turn out to be $\lambda_1 = \lambda_2 = 0.5$. As we vary the value of h in (4.29),

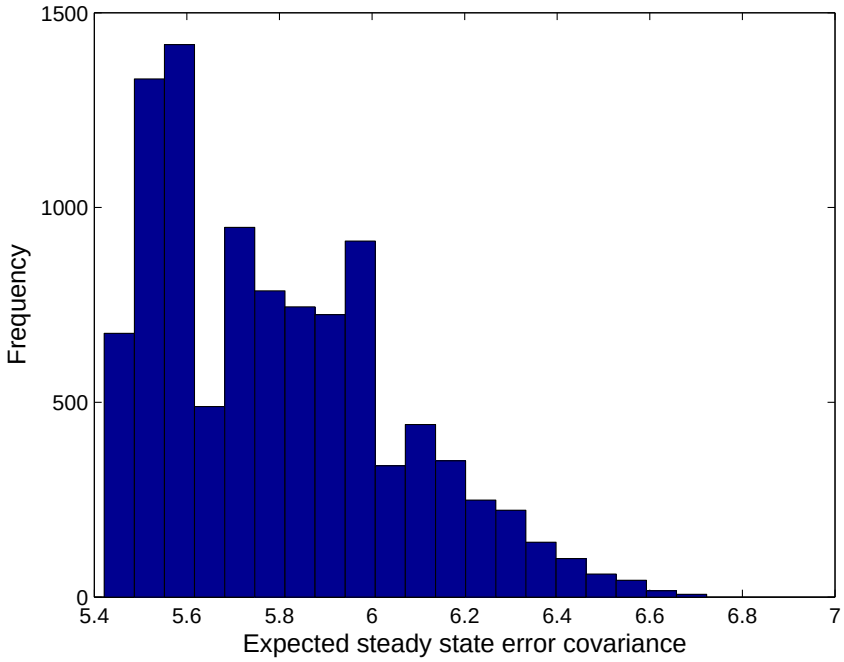


Figure 4.9: Spread of the steady state expected error covariance.

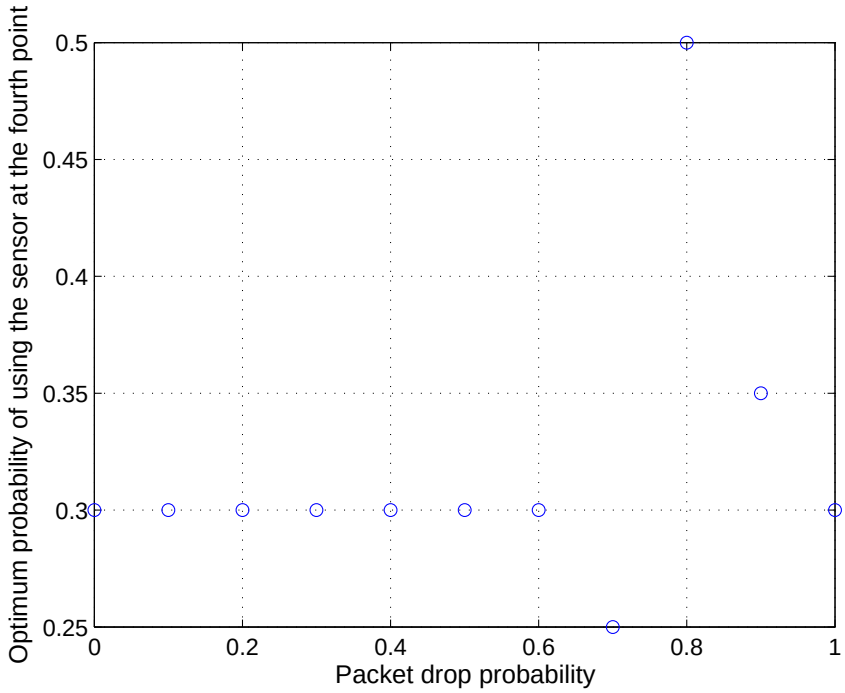


Figure 4.10: Probability of using the sensor at the fourth point as a function of the packet drop probability.

the stability properties of the system change. Thus, to keep the error covariance bounded, we need different number of sensors. Figure 4.11 shows a bound on the number of sensors required, as predicted by Proposition 4.21.

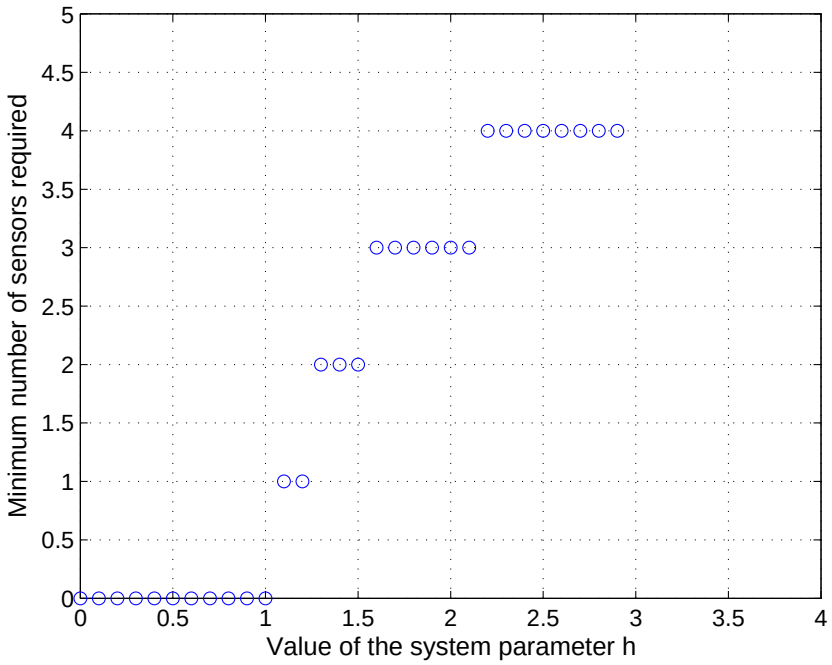


Figure 4.11: Lower bound on the number of sensors required.

Appendix B: Multiple Description Coding for Estimation over Communication Links

In this appendix, we use the tools presented in the chapter to analyze the performance when a form of network source code - Multiple Description (MD) code - is applied to the problem of estimation over a communication link. Consider the process

$$x(k+1) = Ax(k) + w(k)$$

being observed by a sensor of the form

$$y(k) = Cx(k) + v(k).$$

The state vector $x(k) \in \mathbf{R}^n$ and measurement vector $y(k) \in \mathbf{R}^m$. The noises $w(k)$ and $v(k)$ are assumed zero mean, white, Gaussian, independent of each other and with covariances R_w and R_v respectively. The measurements are put into bit packets[9] and transmitted over a communication channel. The bit packet is either received successfully at the receiver or erased. We assume that the channel does not provide preferential treatment to any packet. Thus, e.g., a multiple resolution source code is not a good choice for us. An estimator receives the measurement packets that are successfully transmitted by the channel and comes up with an estimate $\hat{x}(k|k-1)$ (referred from now on as $\hat{x}(k)$) of the state $x(k)$.

We know that if measurement packets are being dropped, the performance of the estimator worsens. One device to improve the performance is to transmit every data packet multiple times. However, this strategy leads to high bit rates being transmitted and can actually be counter-productive by increasing the congestion in the channel and hence the packet erasure probability. Instead, we use the idea of Multiple Description source coding to transmit multiple packets of information while keeping the bit rate low.

MD codes have been considered for long in the information theory literature. According to results in traditional quantization theory [73], the distortion of a given source distribution when it is represented by a quantizer with rate R typically decays as $D(R) \propto 2^{-2R}$. Multiple description codes are designed specifically to achieve good rate-distortion performance over lossy communication links. The unique feature of MD codes is that instead of using one single description to represent one source sample, MD codes use two or more descriptions that are transmitted in separate packets. So at the end of the link, the decoder has much less chance of losing all the descriptions. The distortion at the decoder depends on how many descriptions it receives and could be at various quality levels. Also, unlike the strategy of transmitting a packet multiple times, the total rate is also kept small.

[9] In Chapter 3 we had assumed that there were enough bits available in a packet so that effectively we were transmitting real numbers. In this work, we are interested in packets with finite (and limited) number of bits.

Thus, the design of a MD code is a problem of minimizing the size of the code over the redundancy between the descriptions. MD codes are non-hierarchical so that the receiving order of descriptions is not important.

For the design procedure and theoretical limits of performance of MD codes, we refer the reader to [65, 190]. In this discussion, we will concentrate mostly on the case of 2-description MD codes. Thus, for our purposes, it is sufficient to note that for a 2-description MD code it is possible to achieve the distortions

$$\begin{aligned} D_c &\approx C_0 2^{-2R(1+\alpha)} \\ D_i &\approx C_1 2^{-2R(1-\alpha)}, \end{aligned}$$

where D_c refers to the distortion if both the descriptions (or packets) of the data reach the decoder, D_i refers to the distortion if only one description reaches the decoder, C_0 and C_1 are constants, R is the rate and $\alpha \in [0, 1]$ is a parameter used to trade-off between the decay speeds of D_c and D_i. If we assume that the measurements are being quantized by a uniform quantizer prior to being MD coded, the case when both the descriptions are received achieves the same distortion as due to a uniform quantizer. For a quantizer of step size Δ, thus[10]

$$D_c \approx \frac{\Delta^2}{12}.$$

At high data rates, this yields

$$D_i \approx C_1 \left(\frac{1}{12C_0}\right)^{\frac{1-\alpha}{1+\alpha}} (\Delta)^{2\frac{1-\alpha}{1+\alpha}}.$$

We will model this quantization noise as additive white Gaussian noise with variance given by D_c for both descriptions being received and D_i for only one description being received. This approximation becomes better as the rate of the code increases. This discussion can be generalized for the case of more number of descriptions being used. In the sequel, we will assume that 2-description MD codes are being used. The analysis for higher number of descriptions is similar.

I.I.D. Packet Drops

Suppose, first, that the packets are dropped by the channel in an independent and identically distributed (i.i.d.) fashion with drop probability λ for every packet. Thus, there are three scenarios.

1. The estimator receives both the descriptions. This occurs with probability $(1 - \lambda)^2$. In this

[10]The approximation becomes tighter as the step size Δ decreases.

case the estimator has access to a measurement of the form

$$z(k) = Cx(k) + n(k), \tag{4.30}$$

where $n(k)$ has covariance $G_0 = R_v + D_c$.

2. The estimator receives only one of the descriptions. This occurs with probability $2\lambda(1-\lambda)$. In this case, the estimator has access to measurements of the form (4.30) where $n(k)$ has covariance $G_1 = R_v + D_i$.

3. The estimator does not receive any description. This occurs with probability λ^2. In this case, the estimator has access to measurements of the form (4.30) where $n(k)$ has covariance $G_2 = \sigma^2 I$ where $\sigma^2 \to \infty$.

We can obtain bounds on the expected error covariance and find conditions for stability by using the analysis in Section 4.2. Denote the Riccati update operators in the three cases mentioned above by

$$\begin{aligned} f_0(P) &= APA^T + R_w - APC^T \left(CPC^T + G_0\right)^{-1} CPA^T \\ f_1(P) &= APA^T + R_w - APC^T \left(CPC^T + G_1\right)^{-1} CPA^T \\ f_2(P) &= APA^T + R_w. \end{aligned}$$

Proposition 4.23 *Consider the setting described above with a 2-description MD code being used and the channel dropping packets in an i.i.d. fashion. Then, the expected error covariance $E[P(k)]$ at the estimator is upper bounded by $\Delta(k)$ where $\Delta(k)$ evolves as*

$$\Delta(k+1) = APA^T + R_w - 2\lambda(1-\lambda) f_1\left(\Delta(k)\right) - (1-\lambda)^2 f_0\left(\Delta(k)\right).$$

Further $E[P(k)]$ is lower bounded by $Y(k)$ where $Y(k)$ satisfies

$$Y(k+1) = \lambda^2 AY(k)A^T + R_w.$$

Proof Upper bound follows from Proposition 4.4. For the lower bound, apply (4.10) for the sensor corresponding to case 3, i.e., no packet reaching the estimator. The lower bound at time k can be written as

$$Y(k) = \lambda^{2k} f_2^k(P(0)) + (1-\lambda^2)R_w + \sum_{i=1}^{k-1} \lambda^{2i}(1-\lambda^2) f_2^i(R_w).$$

Thus

$$\begin{aligned}
\lambda^2 AY(k)A^T + R_w &= \lambda^2 A\left(\lambda^{2k} f_2^k(P(0)) + (1-\lambda^2)R_w + \sum_{i=1}^{k-1}\lambda^{2i}(1-\lambda^2)f_2^i(R_w)\right)A^T + R_w \\
&= \lambda^{2k+2}\left(Af_2^k(P(0))A^T + R_w\right) + \lambda^2(1-\lambda^2)AR_wA^T + (1-\lambda^2)R_w \\
&\quad + A\sum_{i=1}^{k-1}\lambda^{2i+2}(1-\lambda^2)f_2^i(R_w)A^T + \sum_{i=1}^{k}\lambda^{2i}(1-\lambda^2)R_w \\
&= \lambda^{2k+2}f_2^{k+1}(P(0)) + \lambda^2(1-\lambda^2)AR_wA^T + \sum_{i=2}^{k}\lambda^{2i}(1-\lambda^2)f_2^i(R_w) \\
&\quad + (1-\lambda^2)R_w + \sum_{i=1}^{k}\lambda^{2i}(1-\lambda^2)R_w - \sum_{i=2}^{k}\lambda^{2i}(1-\lambda^2)R_w \\
&= Y(k+1). \quad \blacksquare
\end{aligned}$$

We can also obtain conditions for the expected error covariance to converge. As an example a necessary condition for the expected error covariance to converge when 2-description MD coding is being used is that

$$\lambda^2\left(\rho(A)\right)^2 < 1 \Rightarrow \lambda < \frac{1}{\rho(A)},$$

where $\rho(A)$ is the spectral radius of matrix A. Comparing the condition for stability when no source coding is done (as presented, e.g., in [178]) which is

$$\lambda\left(\rho(A)\right)^2 < 1 \Rightarrow \lambda < \frac{1}{\left(\rho(A)\right)^2},$$

we see that the requirements on the maximum drop probability allowable are indeed loosened. We have, in fact, gained exactly the advantages (in terms of stability) as of transmitting the packet twice, but while keeping the bit rate low. As can be seen by comparing the upper bound relations, the performance is also improved by using MD coding.

The results above can be generalized for l-description coding where $l \geq 2$. As an example, the necessary stability condition becomes

$$\lambda^l\left(\rho(A)\right)^2 < 1 \Rightarrow \lambda < \frac{1}{\left(\rho(A)\right)^{\frac{2}{l}}}.$$

Unfortunately finding the optimal l-description MD code for an arbitrary l is still an open problem in information theory.

Markovian Packet Drops

If the packet drops are occurring according to a Markov chain, we can apply the results of the chapter to obtain similar bounds. Suppose the channel can be modeled as a 2-state Markov chain

with transition probability matrix Q given by

$$Q = \begin{bmatrix} q_{00} & q_{01} \\ q_{10} & q_{11} \end{bmatrix},$$

where 0 is the good state (corresponding to a packet being received) and 1 is the bad state (corresponding to a packet loss). For the case of 2-MD code, we are thus interested in a 4-state Markov chain where the states correspond to both packets lost, only the 1st packet lost, only the 2nd packet lost and no packet lost. The transition probability matrix of this chain is given by

$$\bar{Q} = \begin{bmatrix} q_{00}^2 & q_{00}q_{01} & q_{01}q_{10} & q_{01}q_{11} \\ q_{10}q_{00} & q_{10}q_{01} & q_{11}q_{10} & q_{11}^2 \\ q_{00}^2 & q_{00}q_{01} & q_{01}q_{10} & q_{01}q_{11} \\ q_{10}q_{00} & q_{10}q_{01} & q_{11}q_{10} & q_{11}^2 \end{bmatrix}. \tag{4.31}$$

Note that the state in which both packets are lost is equivalent to no observation coming through, while all the rest of the states correspond to the system being observed. We can calculate the probability $p_i(k)$ of being in the i-th state at time k and thus obtain results analogous to the i.i.d. case. Define the Riccati update operators in the four cases needed for the bigger Markov chain with a transition probability matrix $\bar{Q}$ by

$$\begin{aligned} f_0(P) &= APA^T + R_w - APC^T \left(CPC^T + G_0\right)^{-1} CPA^T \\ f_1(P) &= f_2(P) = APA^T + R_w - APC^T \left(CPC^T + G_1\right)^{-1} CPA^T \\ f_3(P) &= APA^T + R_w. \end{aligned}$$

The following result is easily proven.

Proposition 4.24 *Consider the same setting as in Proposition 4.23 but with the packet drops occurring according to a Markov chain with transition probability matrix Q as described above. If a 2-description MD code is being used, the expected error covariance $E[P(k)]$ is upper bounded by $\Delta(k)$ where*

$$\Delta(k) = \sum_{j=0}^{3} p_j(k)\Delta_j(k),$$

where

$$p_j(k)\Delta_j(k+1) = \sum_{i=0}^{3} f_i\left(\Delta_i(k)\right) q_{ij} p_i(k-1).$$

Similarly the expected error covariance is lower bounded by $Y(k)$ where

$$Y(k) \;=\; q_{33}^{2k} p_3(0) f_3^k(P(0)) \;+\; \sum_{i=1}^{k-1} q_{33}^{2i} \left(p_3(k+1-i) - q_{33}^2 p_3(k-i) \right) f_3^i(R_w) \;+\; \sum_{j=0}^{2} p_j(k) f_j(R_w).$$

Conditions for convergence can also be derived. As an example, a necessary condition for the expected error covariance to be bounded is

$$q_{33}^l \left(\rho(A)\right)^2 < 1 \Rightarrow q_{33} < \frac{1}{\left(\rho(A)\right)^{\frac{2}{l}}}.$$

Example

We illustrate the performance improvement obtained by using MD codes by a simple example. We consider the system

$$x(k+1) = -1.25x(k) + w(k),$$

being observed by a sensor of the form

$$y(k) = x(k) + v(k).$$

The noises $w(k)$ and $v(k)$ have zero means and variances $R_w = 1$ and $R_v = 2.5$ respectively. We design a balanced 2-description MD code such that the distortion when both descriptions are received is $D_0 \approx 8.33 \times 10^{-6}$ and when only one description is received is $D_1 \approx 1.56$. The rate demanded by the code is 12 bits per sample. For no MD code being used (also called the single-description code), the same distortion of 8.33×10^{-6} can be achieved at a rate of 10 bits per sample.

Figure 4.12 plots the simulated performance achieved for different number of descriptions used in the MD code. Each system is run 1000 times and each simulation is run for 2000 time steps. We can see that even for the same rate, the performance improves with higher number of descriptions being used. Also, there is a trend of diminishing returns when the number of descriptions is increased more and more. The stability conditions are seen to be loosened as well.

Figure 4.13 shows the same system but with the channel dropping packets in a bursty fashion according to a Markov chain. We can once again see the benefits of using a MD code. The probabilities at which the expected error covariance diverges match with the ones predicted by theory.

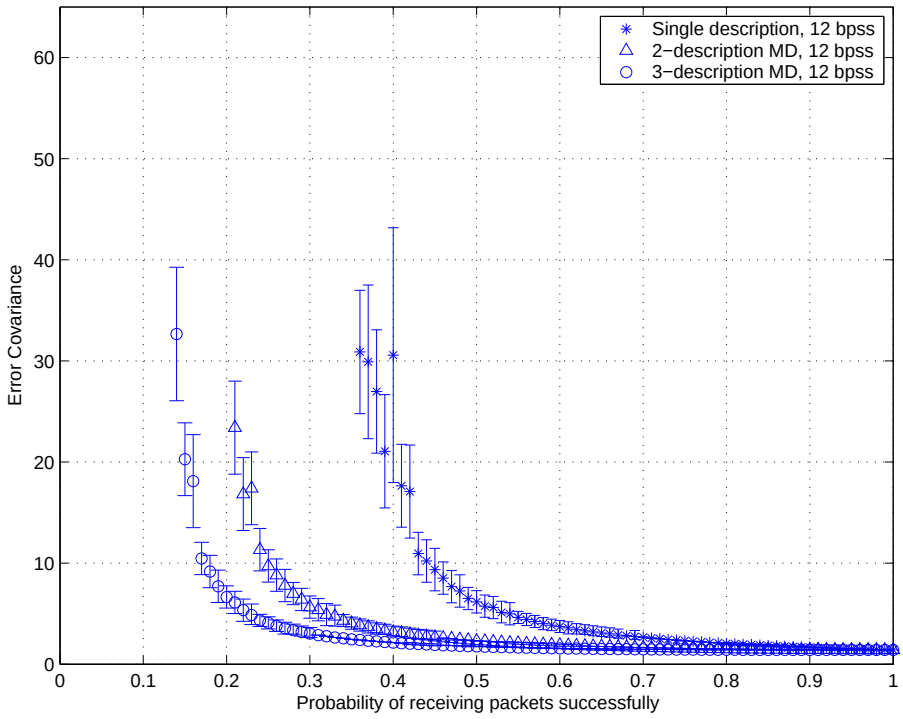


Figure 4.12: Mean values of error covariance for various number of descriptions in the MD code but using the same rate

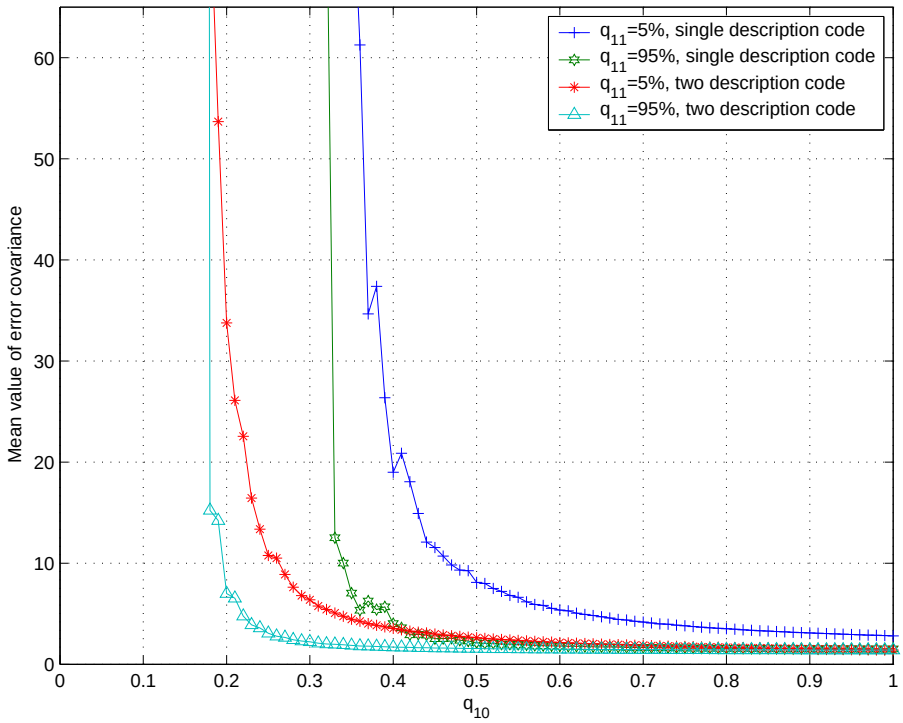


Figure 4.13: Mean values of error covariance for the bursty error case.

Chapter 5

Perspectives and Future Directions

In this dissertation, we have been interested in distributed estimation and control of cooperative networked systems. Such problems are important both because they show up in new engineering systems being envisaged today as well as because they provide an interesting setting where traditionally separate theories of control, communication and computation have to be integrated. From the viewpoint of estimation and control that we took in this dissertation, such problems are difficult because they challenge two standard assumptions:

1. There is no longer a central node that has access to all the information being generated. Agents have to make local decisions based on locally available information.

2. Information flow is no longer perfect. Any data that needs to be shared has to be transmitted over communication channels which introduce effects like random delay, stochastic information loss, data corruption and so on.

The approach that we proposed was a joint design of the information flow and the control laws to be followed by the agents. Within the constraints imposed by the communication channels, the design of the information flow provided an important degree of freedom and allowed us to benefit both in terms of stability as well as performance of the system. We formulated several problems within the framework of networked cooperative systems and obtained some very interesting and promising results. Obviously, however, there is more work that needs to be done to come up with a complete theory of estimation and control of cooperative networked systems. We have been mentioning some of the open directions of research in each chapter. In this chapter, we step back and identify the next set of major problems that needs to be attacked. The work presented in this chapter has partly appeared in [86].

The chapter is organized as follows. We begin with a brief summary of each chapter and identify a couple of major open problems in each area. In Section 5.2, we then consider some other areas in which work needs to be done. In the appendix, we provide some initial results on the issue of robustness in such networked systems which is identified as one of the major open problems.

5.1 Directions Considered in the Dissertation

We begin by summarizing the major research areas addressed in this dissertation and identify some of the open problems in each area.

5.1.1 Distributed Estimation and Control

In Chapter 2, we dealt with the problem of information being diffused through out the system. The problem setup that we considered was a formation of N dynamic agents each with a local controller. The N agents need to minimize a joint cost function; however, each agent can obtain information only about a subset of other agents. This information flow is given by a graph topology. Traditional control theory requires the controller at each agent to have access to the state (or measurement) value from every other agent. The presence of a topology does not allow this and requires the controller to satisfy structural constraints. We posed two problems in the chapter.

1. Given a topology, what is the optimal control law to be followed by the agents? (*What should the agents do?*)

2. How do we design the optimal topology? (*Whom should the agents communicate with?*)

We saw that the problem of designing an optimal structured controller is NP-hard in general, but we designed several numerical algorithms for solving the problem. The algorithms involve solving linear equations only and hence are free from convergence issues plaguing other algorithms proposed in the literature. To design the topology and the control law jointly, we provided a model for the added cost of every new communication edge and considered the problem of determining when such edges should be added. The results are very interesting since there are cases when it is more cost beneficial for agents not to cooperate. We finished by considering a problem of distributed motion control for estimation by N mobile sensors.

The major direction for future research in this area is a more thorough understanding of the role of topology in distributed control. Choosing a topology or rules according to which topologies are formed (say the communication radius) for each agent is an integral part of design. Our results are only a first step towards it. The problem is involved because of the following factors:

1. Finding an analytic expression for the optimal performance achievable with a given topology is an open problem.

2. We are often interested in additional properties that a topology needs to satisfy. Thus, for instance, there should not be a central node for robustness purposes.

3. Topology design may affect other properties of the system. Thus, if the topology involves large amounts of data being communicated through the system, congestion may result in high data

loss rates.

4. Topologies may be time-varying. Thus, for a group of mobile agents, motion control has to tie in with the topology design.

Even though we have presented some interesting results in this direction, incorporating all these effects and coming up with a systematic way to synthesize topologies is required. Thus, we pose the following open research problem in this direction.

Problem 5.1 *To design a topology for distributed control that achieves a desired control performance while considering the various effects that topologies can have on the system properties.*

5.1.2 Control in Presence of Communication Channels

In Chapter 3, we considered the problem of controlling a dynamical system when information from the sensors was transmitted to the controller over a network of communication links. To design the control law and the information flow jointly, we answered two questions in the chapter.

1. Can we identify simple algorithms to preprocess information prior to transmission to combat the distortion of data by the communication channels? (*What should the agents communicate?*)

2. What is the optimal control law to be followed by the agent given whatever information it has access to? (*What should the agents do?*)

We first proved a separation principle that allowed us to solve the two questions independently. We were able in many cases to obtain simple information encoding strategies that are recursive, yet optimal for estimation and control. Benefits both in terms of stability and performance were demonstrated. This viewpoint also allowed us to move beyond treating the network simply as a collection of links with reliability providing the bottleneck for transmission of data across it for estimation. Instead, we used the intermediate nodes to provide a similar function as repeaters in a digital communication system. While most of the work we presented was done when the communication links were modeled as packet erasure links, we also briefly considered other effects of communication links.

The idea of encoding for estimation and control is very powerful. However, the applicability of this idea depends on the ability to identify simple algorithms that are optimal (or at least close to optimal). We provided such algorithms for a few cases. However, determining the optimal algorithms for other cases is still open. As a representative problem consider the system

$$x(k+1) = Ax(k) + w(k) \tag{5.1}$$

being observed by 2 sensors of the form

$$y_i(k) = C_i x(k) + v_i(k), \qquad i = 1, 2. \tag{5.2}$$

Sensors 1 and 2 communicate their information over packet erasure communication links L_1 and L_2 respectively to an estimator. In Chapter 3, we solved for the optimal estimation algorithm when either both L_1 and L_2 were the same link L (in other words, they transmitted or dropped packets simultaneously) or if only one of the links L_1 and L_2 dropped packets. However, the optimal algorithm when both the links drop packets independently is still unknown[1]. If we are not interested in the algorithm being recursive, the optimal strategy is obviously for each sensor to transmit all its previous measurements at every time step. For this optimal strategy, we can identify the stability region as follows.

Proposition 5.1 *Consider the process (5.1) being observed by sensors of the form (5.2) which transmit all their previous measurements at each time step to a minimum mean squared error (mmse) estimator over communication links that erase packets with a probability p_1 and p_2 respectively. A necessary and sufficient condition for the expected error covariance to converge is that the following equations all be satisfied*

$$\begin{aligned} p_1 \mid \lambda_{\max}\left(\bar{A}_2\right) \mid^2 &< 1 \\ p_2 \mid \lambda_{\max}\left(\bar{A}_1\right) \mid^2 &< 1 \\ p_1 p_2 \mid \rho(A) \mid^2 &< 1, \end{aligned} \tag{5.3}$$

where $\lambda_{\max}\left(\bar{A}_j\right)$ is the eigenvalue with the maximum magnitude of the unobservable part of matrix A when the pair (A, C_j) is put in the observer canonical form and $\rho(A)$ is the spectral radius of A.

Proof Following the terminology in the previous chapter, denote by $f_{C_i}(.)$ the Riccati operator

$$f_{C_i}(X) = AXA^T + R_w - AXC_i^T \left(C_i X C_i^T + R_v\right)^{-1} C_i X A^T,$$

and

$$f_{C_i}^k(X) = \underbrace{f_{C_i}\left(f_{C_i}\left(\cdots\left(f_{C_i}(X)\right)\right)\right)}_{\text{applied } k \text{ times}}.$$

Also, let C_3 denote the sensor corresponding to no measurement being taken. Now, consider the error covariance at time step k, denoted by $P(k+1)$. We can partition the event space into $(k+2)^2$

[1] Obviously the problem is solved if the two sensors can cooperate over a perfect channel. The two sensors then effectively form one big sensor.

events of the form

$$E_{mn}: \begin{cases} \text{no transmission received on } L_1 \text{ after time } m \\ \text{no transmission received on } L_2 \text{ after time } n, \end{cases}$$

where $m \in \{-1, 0, 1, 2, \cdots, k\}$ and $n \in \{-1, 0, 1, 2, \cdots, k\}$ and the event $m = -1$ or $n = -1$ denotes that no transmission was ever received on the relevant link. Under the event E_{mn}, let us denote the conditional error covariance as $P_{mn}(k+1)$. For considering the convergence of $E[P(k+1)]$, we need to consider the limit $k \to \infty$. Thus, similar to the arguments in Section 3.5.4, we can assume for a large enough j,

$$P_{jj}(j+1) = P^\star,$$

the steady-state estimate error covariance given all measurements from both sensors at every time step. Now,

$$\begin{aligned}
P(\infty) &= \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} \text{Prob}\left(E_{mn}\right) P_{mn}(\infty) \\
&= (1-p_2)\left((1-p_1)P^\star + (1-p_1)p_1 f_{C_2}(P^\star) + (1-p_1)p_1^2 f_{C_2}^2(P^\star) + \cdots\right) \\
&\quad +(1-p_2)p_2\Big((1-p_1)f_{C_1}(P^\star) + p_1(1-p_1)f_{C_3}(P^\star) + (1-p_1)p_1^2 f_{C_3}(f_{C_2}(P^\star)) + \cdots\Big) \\
&\quad +(1-p_2)p_2^2\Big((1-p_1)f_{C_1}^2(P^\star) + p_1(1-p_1)f_{C_3}(f_{C_1}(P^\star)) + (1-p_1)p_1^2 f_{C_3}^2(P^\star) + \cdots\Big) \\
&\quad +\cdots \\
&= (1-p_2)\left((1-p_1)P^\star + (1-p_1)p_1 f_{C_2}(P^\star) + (1-p_1)p_1^2 f_{C_2}^2(P^\star) + \cdots\right) \\
&\quad +(1-p_2)p_2\Big(p_1 Q + (1-p_1)\Big(f_{C_1}(P^\star) + p_1 A P^\star A^T + p_1^2 A f_{C_2}(P^\star)A^T + \cdots\Big)\Big) \\
&\quad +(1-p_2)p_2^2\Big(p_1 Q + p_1^2 A Q A^T + (1-p_1)f_{C_1}^2(P^\star) + p_1(1-p_1)A f_{C_1}(P^\star)A^T + \cdots\Big) \\
&\quad +\cdots \\
&= p_1 p_2 Q + p_1^2 p_2^2 A Q A^T + p_1^3 p_2^3 A^2 Q (A^T)^2 + \cdots \\
&\quad +(1-p_1)(1-p_2)\left(P^\star + p_2 f_{C_1}(P^\star) + p_2^2 f_{C_1}^2(P^\star) + \cdots\right) \\
&\quad +p_1 p_2 (1-p_1)(1-p_2) A\left(P^\star + p_2 f_{C_1}(P^\star) + p_2^2 f_{C_1}^2(P^\star) + \cdots\right) A^T \\
&\quad +p_1^2 p_2^2 (1-p_1)(1-p_2) A^2\left(P^\star + p_2 f_{C_1}(P^\star) + p_2^2 f_{C_1}^2(P^\star) + \cdots\right)(A^T)^2 + \cdots \\
&\quad +(1-p_1)(1-p_2)p_1\left(f_{C_2}(P^\star) + p_1 f_{C_2}^2(P^\star) + \cdots\right) \\
&\quad +(1-p_1)(1-p_2)p_1^2 p_2 A\left(f_{C_2}(P^\star) + p_1 f_{C_2}^2(P^\star) + \cdots\right) A^T + \cdots .
\end{aligned}$$

Denote, if they exist, the quantities

$$\begin{aligned} S_1 &= p_1 p_2 Q + p_1^2 p_2^2 A Q A^T + p_1^3 p_2^3 A^2 Q (A^T)^2 + \cdots \\ S_2 &= P^\star + p_2 f_{C_1}(P^\star) + p_2^2 f_{C_1}^2(P^\star) + \cdots \\ S_3 &= f_{C_2}(P^\star) + p_1 f_{C_2}^2(P^\star) + \cdots . \end{aligned}$$

Note that since all the terms in the expression for $P(\infty)$ are positive, a necessary condition for it to converge is that each of the terms S_1, S_2 and S_3 converge. Along the lines of the proof of Proposition 4.7 we can prove that these terms will converge only if (5.3) hold.

Now, to prove that the conditions in (5.3) are sufficient for $P(\infty)$ to converge, suppose that the conditions are true. Then, S_1, S_2 and S_3 are finite. Thus,

$$\begin{aligned} P(\infty) = S_1 + (1-p_1)(1-p_2)\left(S_1 + p_1 p_2 S_1 + p_1^2 p_2^2 S_1 + \cdots\right) \\ + (1-p_1)(1-p_2)p_1\left(S_2 + p_1 p_2 S_2 + p_1^2 p_2 S_2 + \cdots\right), \end{aligned}$$

which is bounded. ∎

The performance of the optimal algorithm can also be obtained by using tools from Markov jump linear systems as in Chapter 3. This result thus provides necessary conditions for stability and lower bounds for performance for any other algorithm. However, whether there exists a recursive algorithm for which these conditions are sufficient as well is still unknown. Providing the best recursive algorithm for multi-sensor multi-channel problems such as this is an important direction of future research. There are many variations of the problem we can consider. As an example, what are optimal coding strategies for channels that have additional effects like quantization, data corruption and so on? So far, in all the algorithms that we considered, acknowledgements about whether or not the receiver received a packet were not being used at the transmitter. Is this a general property? Answering similar questions such as these would likely draw on tools from information theory as well. We can thus identify the major next challenge in this direction as

Problem 5.2 *To come up with simple encoding and decoding strategies for information flow to counter the effects of communication channels in distributed control and estimation.*

5.1.3 Distributed Estimation and Control with Imperfect Links

In Chapter 4, we considered problems which had both the aspects - multiple sources of information and imperfect communication links - present. In the sensor scheduling problem, e.g., a subset of N sensors could be used at any time step and the information was transmitted over a communication link to the estimator. To begin with, we obtained the structure of the optimal estimator. Because

of the stochastic packet erasures imposed by the link, the performance of the estimator is not well-understood. We characterized the expected error covariance of the estimator and provided conditions for it to be bounded. We then moved on to the problem of designing a sensor schedule (i.e., the problem of *Who should communicate when*?). The basic idea we used was coming up with stochastic schedules to escape the exponential complexity of a tree search as well as to be able to include the effects of communication links such as stochastic packet loss. The tools that we developed are widely applicable and we demonstrated that through a couple of examples - dynamic sensor coverage and network source coding for estimation across communication links. We also considered the problem of distributed control with switching topologies. We identified the optimal control input and provided conditions for stability of the system.

Even though the algorithm we proposed has several nice features, there is much left to be done in the area. Our algorithm is an offline algorithm. The schedule is computed prior to the system operation and the cost optimized is the steady-state cost. The next major step would be to come up with an *adaptive* algorithm. One desirbale feature of networked systems is the ability to 'plug-and-play'. As an example, in the sensor scheduling problem it would be useful not to be forced to cease the system operation and re-optimize the schedule whenever a new sensor is added or an existing sensor is taken out. In general, resource allocation algorithms that adapt to the available resources or constraints automatically are desirable. We pose the major research challenge in this area as

Problem 5.3 *Identify a computationally feasible adaptive method for resource allocation in networked systems that takes into account the presence of communication links.*

5.2 Some More Open Problems

This dissertation is only a first attempt towards solving the complicated problem of distributed estimation and control in networked systems. We made several assumptions in this dissertation that made the problem tractable and allowed us to get some initial results. We mention two such assumptions, removal of which will be required in the next step towards obtaining a more general and effective theory.

5.2.1 Asynchronous Systems

We have assumed throughout the dissertation that there is a global system clock available to all agents. In general, this will not be true. Each component will have access to a local clock which will drift with time. Since communication links introduce random delays, synchronization cannot be possible by naïvely transmitting the local times periodically and calculating offsets. Algorithms in networked systems need to be robust to some level of asynchrony. There are two main ways how one may move towards ensuring that:

1. Coming up with ways to synchronize the local clocks periodically by transmitting data. This approach, e.g., has been proposed in [68].

2. Designing algorithms that can tolerate some level of asynchrony. At present, not many such algorithms are known.

Both these approaches need to be explored. This is an important area where much more research needs to be undertaken.

Problem 5.4 *Identify algorithms that are robust to different agents having access to local clocks that may drift apart.*

5.2.2 Robustness to Agent Loss

We have, so far, assumed that no agent malfunctions during the course of operation. For any networked system to be practical, we need to ensure that the failure of one agent to perform its designated duties does not imperil the joint task. Thus, there is need for some notion of robustness to agent failure. This has, so far, been largely ignored in the control community. In any application involving huge numbers of sensors/agents, a typical component will be cheap and off-the-shelf. It is reasonable to assume that such components will have high failure rates and the algorithm should be able to deal with such failures.

One possible definition of this concept can be based on a similar concept studied in the distributed computation literature for consensus algorithms (see, e.g., [136] for a good overview). It has long been realized in that community that distributedness in algorithms does not inherently lead to robustness. In general, we need to ensure that agents receive enough information from their neighbors to be able to detect and isolate faulty agents. This point is of interest while designing multi-agent systems and evaluating their performance.

We present some initial work on defining this concept in the appendix. We can pose the problem on which more attention needs to be focussed.

Problem 5.5 *Come up with a general theory to analyze and synthesize distributed systems that are robust to (possibly temporary) agent failures.*

This problem would likely tie in with concepts from fault detection and isolation as well.

Appendix A: Robustness in Networked Systems

We first set up a basic framework for defining and analyzing distributed algorithms. We will concentrate only on discrete-time algorithms and synchronous networks. We begin by defining a physical agent.

Definition 5.1 *(A Controlled Agent:) We define an agent as a collection of 4 quantities* (X, U, X_0, f).

1. $x(k) \in X$ *is the* state*; X represents the state space.*
2. $u(k) \in U$ *is the* control input*; U is the input space.*
3. $x(0) \in X(0)$ *is the* initial condition*, where $X(0) \subset X$ is the set of allowable initial states.*
4. $f : X \times U \to X$ *is a map that defines the* dynamics *of the agent.*

In words, each agent has a state $x(k)$ at time k. Given a control input $u(k)$, the state evolves according to the dynamics f, i.e., $x(k+1) = f\left(x(k), u(k)\right)$. As an example, the agent has linear dynamics if $f\left(x(k), u(k)\right)$ is of the form $A(k)x(k) + B(k)u(k)$ where $A(k)$ and $B(k)$ are given. The state space X can in general be a continuous space (such as $\mathbf{R}^n$) or a discrete space (such as nodes of a graph). We have not yet discussed how the control input $u(k)$ is calculated.

Definition 5.2 *(Network of Controlled Agents) (following [139]): We define a network of N agents using three quantities* $(I, \mathcal{A}, \mathcal{G}_{comm})$.

1. $I = \{1, \cdots, N\}$ *is the set of unique identifiers for each of the N agents.*
2. $\mathcal{A} = \{A_i\}_{i \in I}$ *is the set of controlled agents. Each agent A_i is in turn defined as in definition 5.1. We will refer to the state of the i-th agent at time k as $x_i(k)$, the control input as $u_i(k)$ and the corresponding sets of allowed values as X_i and U_i respectively.*
3. $\mathcal{G}_{comm}$ *is the set of allowed communication graphs. At each time step k, the communication graph $\mathcal{E}_{comm}(k)$ over N nodes is an element of $\mathcal{G}_{comm}$. Every node is identified with the identifier i corresponding to a unique physical agent. The edges in the graph represent communication edges in the network. Thus, if the pair (i, j) is an edge in $\mathcal{E}_{comm}(k)$, the agents with identifier j can communicate with the agent with identifier i at time step k .*

We will assume *undirected* graphs. Agent i is a neighbor of agent j if the two can communicate. Note that the word communication is used in a loose sense here and essentially includes any means of gathering information about the state of another agent. This may be, e.g., through sensing without explicit physical communication occurring. To fully characterize a networked system, we need to also define communication and control laws according to which the agents choose the messages

transmitted to the neighbors and the control inputs for their own dynamics. However, for our present purpose the two definitions given above suffice. For more details, see [86].

There might be additional variables involved in the problem specification, which we refer to collectively as *environmental variables* and denote by the set V. For example, these can pertain to the locations of obstacles when the agents are robots moving in an area. Similarly for algorithms which assume a fixed and given communication graph, the graph is an environmental variable.

We now define a cooperative task that needs to be carried out by the agents. We will define a task in terms of a cost function.

Definition 5.3 *(Cooperative Task) A cooperative task is defined by a cost function C which is a function of the state trajectories of all the agents, the control inputs applied by them, their initial conditions and possibly some environmental variables.*

$$C : \prod_i \{x_i(k)\}_{k=0}^{\infty} \times \prod_i \{u_i(k)\}_{k=0}^{\infty} \times \prod_i x_i(0) \times V \mapsto \mathcal{R}^+.$$

The aim of any algorithm that carries out the task is to minimize the cost function. Note that for a task that is informally described in words, say 'rendezvous', there might exist many choices of possible cost functions. We will associate a separate task with each cost function.

Definition 5.4 *(Cooperative Algorithm) A cooperative algorithm is a choice of communication and control laws for every agent. It usually aims at minimizing the cost function associated with a particular cooperative task.*

Note that by defining the algorithm in this way, we are making a distinction between the cost of the underlying cooperative task the algorithm is trying to solve and the cost function that the algorithm is actually minimizing. The two costs may be different. Also, note that there can be constraints on the form of control and communication laws that an algorithm must satisfy. As an example, for robotic agents moving in physical space, it might be the case that only a specific function of the state of the neighbors can be sensed (output-measurable). Hence, the messages have to depend on that function and not on the state.

Examples

1. Average Consensus [157]: This task, in its basic form, considers N agents, each of which is provided a scalar value. The arithmetic mean of the values across the agents is m. The task is to ensure that on termination, each agent has the value m. The i-th agent has scalar dynamics of the form

$$x_i(k+1) = x_i(k) + u_i(k),$$

with $x_i(0)$ given. There are many cost functions that are possible for posing the task. One such possible cost function is

$$C = \lim_{k\to\infty} \left(\sum_{\text{all nodes } i} \left(x_i(k) - \frac{1}{N} \sum x_i(0) \right)^2 \right).$$

This is clearly a function of the state values of the agents and their initial conditions and thus fits in our framework. The network consists of N such agents. The communication graph is fixed and given. The only requirement on the edge set is that the graph be connected. Let h be a small positive number. Then, the control input applied by the i-th agent is

$$u_i(k) = -h \sum_{j:j \text{ is a neighbor of } i} (x_i(k) - x_j(k)).$$

Note that the algorithm minimizes C by minimizing the cost function

$$C_{algo} = \lim_{k\to\infty} \left(\sum_{(i,j) \text{ being neighbors}} (x_i(k) - x_j(k))^2 \right),$$

with the constraint

$$\sum_i x_i(k) = \sum_i x_i(0).$$

This separation of the *task* from the *solution* is an important feature. If all the agents are functional, it can be proven that minimizing the algorithm cost C_{algo} yields the solution that minimizes the task cost as well.

This algorithm is similar to the rendezvous algorithm without connectivity constraint proposed in [140] and is related to Vicsek's model discussed in [107, 192, 164].

2. Sensor Deployment: This problem and its solution have been widely studied [155]. We adopt the algorithm that is described in [38]. The basic problem is for N agents to position themselves in a convex region Q such that the total distance from each point in the region to the nearest agent (possibly weighted by a non-negative density function) is minimized. The agents once again have first order dynamics. They are assumed to move in the convex region Q. Thus, $X_i = Q$ for every agent i. The cost function is defined in terms of a density function $\phi(x)$ that has a non-negative value at all points x in a region. The cost function then is

$$C = \lim_{k\to\infty} \int_Q \min_i |q - x_i(k)|_2^2 \phi(q) dq.$$

In addition to the state values, the cost also depends on the region Q and the function $\phi(x)$, which are given environmental variables. The network considered in [140] is the Delaunay

graph. Construct the Voronoi partition of the region Q as generated by the positions of the N agents at time k. Then, two nodes i and j share an edge if their Voronoi cells share an edge. The control input is designed so that each agent moves towards the centroid of its Voronoi cell. The details are given in section III-B of [38].

Failure Modes and Robustness

One way to characterize an algorithm is through the value of the problem cost function C that it achieves. We will denote the performance cost achieved by the algorithm by $\mathcal{PC}$. Since the cost function C can be a function of the initial conditions $x_i(0)$ and the values of the environmental variables, so can be $\mathcal{PC}$. To characterize the algorithm itself, we can get rid of this dependence in two ways.

1. We can consider the *average* cost, $\mathcal{PC}_{avg}$ obtained by averaging $\mathcal{PC}$ across many runs as the initial conditions and values of the environmental variables are chosen randomly from a given set S using a given probability distribution function.

2. We can consider the *worst case* cost $\mathcal{PC}_{wc}$ which is obtained by computing the supremum of the $\mathcal{PC}$ as the initial conditions and values of the environmental variables are varied across a set S.

In general, the performance cost will also depend on the number of agents. To show this dependence explicitly, we will sometimes denote the performance cost by $\mathcal{PC}_{avg}(N)$ or $\mathcal{PC}_{wc}(N)$ if N agents are present.

Before defining the key property of robustness, we need to define an agent failure. During the execution of an algorithm, an agent may stop functioning in many ways. When an agent fails, it alters the control law and the communication law that it follows. We can define some failure modes as follows:

1. Failure mode 1: An agent may fail by simply ceasing to communicate with other agents. This is the most popular agent failure model considered in the literature. In the language of [136], this is similar to saying that the process suffers from a stopping failure.

2. Failure mode 2: An agent fails by setting its state value $x_i(k)$ to a constant. Thus, the control input $u_i(k)$ that ensures $x_i(k+1) = x_i(k)$ is used at every time step k. The constant state value can be any value that the state $x_i(k)$ is allowed to take, i.e., in the set X_i. Moreover, any communication from a failed agent is also affected accordingly. Thus, the messages it transmits to its neighbors also assume constant values for all time k.

3. Failure mode 3: The agent alters the control input to set its state at every time step k to an arbitrary value in the set X_i. The sequence of the values can be chosen maliciously so that

the other agents are hindered in the pursuit of the cooperative task. Moreover, the messages a failed agent transmits are also chosen arbitrarily. This is akin to the way agents fail as described in [120] and is referred to as the Byzantine failure mode in [136].

In the language of [120], the assumption that any communication from a failed agent is also affected according to the failure mode means that agents communicate "orally" and not through "signed messages". Note that this list is not exhaustive and other modes of failure can readily be thought of.

When a given number p out of a total of N agents executing a certain algorithm fail according to a certain mode, the situation is as if the p agents follow a new control and communication law while the remaining $N-p$ agents follow the original laws. We can calculate the performance of this new algorithm. We now define robustness of an algorithm with respect to a particular agent failure model.

Definition 5.5 *(Robustness of an Algorithm): Consider an algorithm being executed on a system of N agents out of which p agents fail according to a particular failure model. Denote the performance cost achieved through the remaining $N-p$ agents as $\mathcal{PC}_{wc}(N,p)$ where the supremum is also taken over all groups of p agents that can fail. An algorithm is said to be worst-case robust to a particular failure mode up to p agents if*

$$\mathcal{PC}_{wc}(N,p) = O\left(\mathcal{PC}_{wc}(N-p)\right).$$

If $\mathcal{PC}_{wc}(N,p) = \Omega\left(\mathcal{PC}_{wc}(N-p)\right)$ but $\mathcal{PC}_{wc}(N,p) \neq \Theta\left(\mathcal{PC}_{wc}(N-p)\right)$, the algorithm is said to be worst-case non-robust[2]*.*

- If instead of the worst case performance costs, we consider the average performance costs $\mathcal{PC}_{ave}(.)$ (however, while still taking the supremum over the p agents that fail), we obtain the definition of average case robustness. While the worst case robustness tells us if the algorithm will perform correctly for *any* set of initial conditions (similar to the case in robust control), average case robustness guarantees that the algorithm will perform correctly *on an average*. We can also talk about almost sure (a.s.) robustness when the algorithm is worst case robust as the initial conditions and values of the environmental variables are varied across a set S, except on a region with measure zero.

- Strictly speaking, the definitions given above pertain to the robustness over the set S over which the initial values and the environmental variables are allowed to vary.

[2]We say $f(x) = O(g(x))$ iff $\exists$ numbers x_0 and $M > 0$ such that $|f(x)| \leq M|g(x)|$ for $x > x_0$. $f(x) = \Omega(g(x))$ iff $\exists$ numbers x_0 and $M > 0$ such that $|f(x)| \geq M|g(x)|$ for $x > x_0$. Finally, iff $\exists$ x_0, $M_0 > 0$ and $M_1 > 0$ such that $M_1 g(x) \geq f(x) \geq M_0 g(x)$ for $x > x_0$ then $f(x) = \Theta(g(x))$

- The basic intuition behind the definition is that a distributed algorithm should lead to better performance as the number of agents increases. We can expect a hit in the performance if a subset of the agents fail. However, if we calculate the performance loss in two situations:
  - N agents were present to begin with but p of them failed, and
  - Only $N-p$ agents were present to begin with, (Equivalently, N agents were present and p failed but they were detected and removed)

  then the rate at which adding functional agents decreases the cost should not be adversely affected. In other words, the impact due to agents failing should not increase as more functional agents are added.

We note the following properties that follow from the definitions. We present the proofs for worst-case robustness. The proofs for average-case robustness are similar.

Proposition 5.2 *If an algorithm is non-robust for p failed agents to failure mode 2, it is non-robust to p failed agents to failure mode 3. Similarly if an algorithm is robust for p failed agents to failure mode 3, it is robust to p failed agents to failure mode 2.*

Proof Let the control inputs used in the calculation of the performance cost for failure mode 3 be given by $\{u_i(k)\}_3$ for agent i and the messages sent be given by $\{m_i(k)\}_3$. Similarly, let the control inputs used in the calculation of the performance cost for failure mode 2 be given by $\{u_i(k)\}_2$ for agent i and the messages sent by $\{m_i(k)\}_2$. Consider the choice of the control inputs. The set in which the control inputs are allowed to vary for mode 3 also contains as a particular element $\{u_i(k)\}_2$. Since, by definition, the cost in mode 3 is maximized by $\{u_i(k)\}_3$; in particular, the cost achieved by using $\{u_i(k)\}_2$ is not more than when $\{u_i(k)\}_3$ is used. But the cost achieved when $\{u_i(k)\}_2$ is used is the cost in failure mode 2. Thus,

$$\mathcal{PC}_{wc}(N,p)_{\text{ failure mode 3}} \geq \mathcal{PC}_{wc}(N,p)_{\text{ failure mode 2}}.$$

If the algorithm is non-robust to failure mode 2, there exists a constant c such that

$$\mathcal{PC}_{wc}(N,p)_{\text{ failure mode 2}} \geq c\mathcal{PC}_{wc}(N).$$

The above two equations together prove that the algorithm is non-robust to failure mode 3 as well. The second part can be proved similarly. ∎

However, a similar statement cannot be said for failure modes 1 and 2. Even if an algorithm is non-robust to failure mode 1, it can be robust to failure mode 2.

Proposition 5.3 *If an algorithm is non-robust to failure of p agents in failure mode 3, it is also non-robust to failure of t agents in failure mode 3 where $t \geq p$. Similarly if the algorithm is robust to failure of t agents in failure mode 3, it is also robust to p failures where $p \leq t$.*

Proof Consider the case when p agents fail. Consider the choice of initial conditions, control inputs and messages for the failed agents that corresponds to the worst case of the performance cost. Choose an arbitrary set S of $t-p$ functional agents. For this choice denote the control input that the agent i in the set S of $t-p$ functional agents applies by $\{u_i(k)\}$ and the messages it transmits by $\{m_i(k)\}$. Now, consider the case when t agents can fail. Choose the same initial conditions as the previous case. Let the t agents that fail be chosen such that they consist of the p agents that failed in the previous case and the $t-p$ agents in the set S. Also, let the p agents apply the same control inputs and transmit the same messages as the previous case. Let the i-th agent in set S apply control input $\{u_i(k)\}$ and transmit messages $\{m_i(k)\}$. Thus, the evolution of the system will be identical to the case when only p agents failed. Hence, $\mathcal{PC}_{wc}(N,t) \geq \mathcal{PC}_{wc}(N,p)$ and the result follows. The second part can be proved along the same lines. ∎

Proposition 5.4 *Suppose $\mathcal{PC}_{wc}(N) = \Theta(\mathcal{PC}_{ave}(N))$. Then, if the algorithm is worst-case robust to failure of p agents to a particular failure mode, it is also average-case robust to failure of p agents to that failure mode. Similarly of the algorithm is average-case non-robust to failure of p agents to a particular failure mode, it is also worst-case non-robust to failure of p agents to that mode.*

Proof Proof follows from the definitions once we note that

$$\mathcal{PC}_{wc}(N,p) \geq \mathcal{PC}_{ave}(N,p). \quad ∎$$

A similar statement can also be made about the relation between worst-case robustness and a.s. robustness.

Examples

We now illustrate the above definitions using specific algorithms. For the case where the algorithms involve agents moving in physical space, we will model the agents as point masses and ignore issues such as collision avoidance.

Average Consensus [157] Since the algorithm minimizes the given task cost function only for fully connected graphs, we will assume that to be the case as long as no agents fail. For average-case robustness, we will consider the initial conditions to be chosen uniformly over the set $[-1,1]$.

- Assume that the p agents that fail are allowed to be chosen so that the graph of the remaining $N-p$ agents is disconnected. Then, the algorithm is worst-case non-robust to failure mode 1.

If the graph remains connected, then the algorithm is worst-case, average-case and a.s. robust to failure mode 1.

Proof First consider the case when we allow the graph of the remaining agents to be potentially disconnected. Consider the case when $p = 1$ and let $N = 2m+1$. Choose the graph of N agents as a line. Let the agent i fail such that two distinct connected sub-groups of agents are formed, each with m agents. Also, suppose the initial conditions are chosen such that every agent in the first sub-group has value 1 and every agent in the second sub-group has value -1. Thus, the algorithm will converge with each agent retaining its value, as against converging to the correct mean for the $N-1$ agents, which is 0. Thus,

$$\mathcal{PC}_{wc}(N,1) \geq \sum_{i=1}^{m} (1)^2 + \sum_{i=1}^{m} (-1)^2 = N-1.$$

If N agents were present, they would have all converged to the mean as long as the graph was connected. Thus, $\mathcal{PC}_{wc}(N) = 0$. Thus, the algorithm is worst-case non-robust. If the graph remains connected, $\mathcal{PC}_{wc}(N,p) = \mathcal{PC}_{wc}(N) = 0$. Hence, the algorithm is worst-case robust. A similar argument shows that the algorithm is average-case robust. ∎

In a similar manner, it can be proven that the algorithm, if it runs on a l-connected graph, is robust to the failure of $l-1$ agents. A random $G(n,p)$ graph is almost surely (a.s.) l-connected for $p \geq p_l = (\log(n) + (l-1)\log\log(n))/n$, and a.s. not l-connected otherwise [22]. Thus, if the set over which the graph (which is an environmental variable) is allowed to vary is the set of random graphs, the algorithm is average case robust to l failures in failure mode 1 for $p \geq p_l$ and non-robust otherwise.

- The algorithm is worst-case, average-case and a.s. non-robust to failure mode 2.

Proof Consider the case when $p = 1$. Let the initial conditions be such that the non-faulty $N-1$ agents have values 0 while the faulty agent has value 1. Thus, the algorithm will converge with each agent achieving the value 1, as against converging to the correct mean for the $N-1$ agents, which is 0. Thus,

$$\mathcal{PC}_{wc}(N,1) \leq \sum_{i=1}^{N} (1-0)^2 = N.$$

Since $\mathcal{PC}_{wc}(N) = 0$, the algorithm is non-robust. A similar argument holds for average case robustness. ∎

Sensor Deployment [38] We will assume below the density function $\phi(x)$ to be a constant. For the statements below, we consider the case when the communication graph is fully connected.

- The algorithm is worst-case robust to failure mode 1 but not to mode 2.

  Proof Clearly, if p agents fail according to mode 1, the remaining $N-p$ agents will perform exactly as if there were only $N-p$ agents to begin with. Robustness to failure mode 1 is thus obvious. For failure mode 2, let the agents move in the set Q which is a line segment of unit length. Let all the agents be stationed at one of the ends and the agent closest to the other end fail. Thus, $\mathcal{PC}_{wc}(N,1) \geq \int_{x=0}^{1} x^2 dx = \frac{1}{3}$. When we have N agents present, the cost is obtained by a Voronoi partition of a unit length segment by N agents. This can be shown to be $\mathcal{PC}_{wc}(N) = \frac{1}{12N^2}$. Thus, the algorithm is non-robust. ∎

- The algorithm is a.s. robust to failure modes 1 and 2 for p agents failing, where p is any constant.

  Proof Robustness to failure mode 1 follows from the worst-case robustness. For failure mode 2, consider the case when the agents are deployed along a straight line of unit length. Consider also the case when only one agent fails. Suppose that the failing agent is at the position x and there are N_1 agents in the region $[0,x)$ and $N-N_1-1$ in the region $(x,1]$. Easy algebra shows that given x and N_1 $\mathcal{PC}(N,1) \approx \frac{x^3}{12N_1^2} + \frac{(1-x)^3}{3(2N-2N_1-2)^2}$. Thus, if the agents are deployed according to a uniform distribution, we can show that each typical event will have $\mathcal{PC}(N,1) = \frac{1}{12N^2}$. When N is large, $\mathcal{PC}_{wc}(N,1) \approx \frac{1}{12N^2}$ with high probability. Since $\mathcal{PC}_{ave}(N) = \frac{1}{12N^2}$ as well, the robustness is obvious. For general sets Q, the proof is similar. ∎

Multi-Sensor Fusion through a Central Node Let us now consider an example in which there is a central data processing node. Suppose N nodes measure the value of a random variable v with some additive measurement noise. To obtain the global estimate, every node i transmits its local estimate $\hat{x}_i$ with error covariance P_i to a central node. The central node fuses the estimates to obtain the global estimate $\hat{x}$ with the error covariance P given by $P^{-1} = \sum_i (P_i)^{-1}$ and transmits it back to every node. The cost function we consider is $\sum_i \text{trace}(P_{f,i})$ where $P_{f,i}$ is the final error covariance of the i-th node. For testing the robustness we assume a star topology with node 1 being the central node. We can prove that the algorithm is not worst-case, average-case or a.s. robust to either failure mode 1 or to mode 2.

Proof We give the proof for failure mode 1 for worst-case robustness. The proof for other cases is similar. First we note that if the error covariance for the local estimate is given by $P_i = P$, then the error covariance for the global estimate will be given by $\frac{P}{N}$ if N agents are present. Thus,

$$\mathcal{PC}_{wc}(N) = (N) \times \text{trace}\left(\frac{P}{N}\right) = \text{trace}(P).$$

Now, consider the case when the agent 1 fails. Then, $P_{f,i} = P_i$. Thus, $\mathcal{PC}_{wc}(N,1) \leq \sum_i \text{trace}\,(P_i) = (N-1)\text{trace}\,(P)$and the algorithm is non-robust. ∎

This is a sanity check since the algorithm is intuitively non-robust to failure of the central node. Our analysis confirms this fact.

Discussion: How to make Algorithms Robust

In this section, we give some ideas about how to make algorithms robust. As a case study, we will consider the classical Byzantine Generals problem in which a General needs to transmit a value v to N commanders such that when the algorithm terminates

1. All the functional (or loyal) commanders make the same decision about the value. We are not concerned with the final values of the non-loyal commanders.

2. If the General is functional, all functional commanders receive the correct value.

For ease of exposition, we will also assume that the General is functional. We will consider the cost $C = \lim_{k\to\infty} \sum_{i=1}^{N} \left(x_i(k) - v\right)^2$, where x_i is the final decision of the i-th loyal commander and N is the number of loyal commanders. We will study the robustness properties of three algorithms that solve the problem. The first algorithm is similar to the average consensus algorithm discussed above. The general is assumed to be node 1. Its state remains at a constant value v that it needs to communicate to others. Every other agent updates its state according to the relation

$$x_i(k+1) = x_i(k) - h \sum_{j \neq i} \left(x_i(k) - x_j(k)\right),$$

where h is a positive constant designed to make the algorithm converge. It can be easily shown that any initial condition for the agent states is driven to a consensus vector in which every node has the value v. Thus, the algorithm solves the problem provided all the agents are functional. However, let us consider the case when p agents fail according to failure mode 2. In that case it can be shown [100] that, in general, as long as a node has a path from the failed agent that does not include the general, it does not converge to the value v. Thus, the algorithm is seen to be both worst-case and average-case non-robust for any non-zero value of p. Since the algorithm is non-robust to failure mode 2, it is also non-robust to failure mode 3.

The second algorithm was proposed by Lamport et al [120]. They demonstrated that if one-third or more agents fail according to the failure model 3, then no algorithm that solves the above problem exists. For the case of less than one-third agents failing, they give an algorithm which successfully solves the problem. In the simplest version of the algorithm, the communication graph is assumed to be fully connected. We will also make that assumption. We illustrate Lamport's algorithm with a simple example of one General and three commanders. The algorithm proceeds as:

1. At time step 1, the General transmits its value v to all the commanders.

2. At time step 2, every commander transmits its estimate of what the General transmitted to every other commander.

3. At time step 3, every commander calculates a majority of the messages it has heard, so far, and outputs its estimate of the decision. The majority function takes in as input an $n \times 1$ vector of real numbers. If there is a number x such that x occurs more times than any other number in the vector, the function outputs x. Otherwise there are at least two numbers x and y that occur the same number of times. The function outputs either x or y randomly.

If at most one commander can fail, it can be proven that the cost C is still 0. The algorithm can be extended to N commanders and can be studied under slightly less restricted communication requirements than a fully connected graph, e.g., a 3-regular graph. We note that the algorithm fits in our framework and that it is both worst-case and average-case robust to failure mode 3.

The third algorithm involves including a fault-detection step in algorithm 1. For node i, let $\mathcal{N}_i$ be the neighbor set of i and N_i be its cardinality. In this algorithm, when agent i communicates with agent j at time k, it transmits four quantities: $x_i(k)$ (denoted by $a_i(k)$), $x_i(k-1)$ (denoted by $b_i(k)$), $\sum_{l\in\mathcal{N}_i} x_l(k-1)$ (denoted by $d_i(k)$) and N_i. Given these quantities, each node carries out the following checks:

1. It checks if $a_i(k-1) = b_i(k)$.

2. It checks if $a_i(k) = (1 - hN_i)b_i(k) + hN_i d_i(k)$.

If both these checks are successful, it carries out the same step as the average consensus algorithm, otherwise it identifies the node i as faulty and disregards it from that time on. We will consider the case of one agent failing in failure mode 2. Again we note that if the failing node disconnects the network into 2 parts, there is no hope for an algorithm to be robust. We will assume the network is at least 2-connected. The general is again node 1. Without loss of generality, let node 2 fail. The following can easily be proved.

1. If $\forall k$, node 2 transmits $a_i(k) = a$, $b_i(k) = b$, $d_i(k) = d$, and $N_i = N$ then to avoid detection, $a = b$.

2. Moreover, to avoid detection by some node j_0, $x_{j_0}(k) = N_i a - d$. Thus, unless two non-faulty nodes have the same state value at all times, at least one will be able to detect the fault in node 2.

3. To ensure that $x_{j_0}(k)$ remains constant, it must be true that $\sum_{l\in\mathcal{N}_{j_0}, l\neq i} x_l(k) = N_{j_0}(Na-d)-a$.

Note that the last two conditions define two surfaces parameterized by the values of $\{x_l(k)\}_{l\neq i}$ in the (a, d, N) space where the faulty values transmitted must lie for the failing agent to go detected. Similarly the condition that the sum of the state values of all neighbors of j_0 remains constant places an algebraic condition on the state values of all other nodes of the network and defines another surface. Now, it is certainly possible to come up with initial conditions that satisfy the above constraints. As an example consider the topology in which the edges (1,2), (1,5), (2,4), (3,4), (3,5) and (4,5) are present. Node 5 is the general with value $4m - n + 1$ while node 1 fails by transmitting values $a = m$, $d = n$ and $N_1 = 2$. The initial values of nodes 2, 3 and 4 are $3m - 1$, $2m - 3n + 1$ and $2m - n$ respectively. Then, the nodes 3 and 5 will be able to detect that node 1 is faulty but not node 4. Also, the nodes will never agree on the value of node 1. We can add any number of nodes such that they transmit only to nodes 3 and 5 with the same initial conditions as node 3. Thus, they too will never reach node 1's value and the algorithm is worst-case non-robust.

On the other hand, if we choose the initial conditions randomly from a uniform distribution over the interval $(0, 1)$, the probability that the three surfaces will intersect is very small. Thus, the probability that valid values of a, d and N_i will exist is also small. Hence, with high probability, all nodes will be able to detect that node 1 is faulty and disregard it. Once the nodes disregard it, the algorithm will run on a connected graph of $N - 1$ functional agents and hence will terminate successfully. The algorithm is thus a.s. robust.

The common feature of the two robust algorithms is that every node received enough information to be able to recover from the effects of the faulty agents. In the second algorithm, enough number of edges were present for every node to obtain multiple copies of the same value. Thus, it could apply majority rule and obtain the correct value. In the third algorithm, information was being transmitted with sufficient redundancy that each node could check if its neighbor was transmitting inconsistently and hence was faulty. This *robustness through information redundancy* is different in nature from the *robustness through cost function* that was exhibited by the sensor coverage algorithm. Whether any algorithm can be made robust through such information redundancy is an open question.

This work is but a first step towards a theory of robust distributed algorithms. More work is needed to identify the properties that the cost function must satisfy for (say a gradient descent type) algorithm to be inherently robust. It will also be nice to have an analytic tool to determine the robustness properties of algorithms and to synthesize robust algorithms systematically.

Bibliography

[1] P. Alriksson and A. Rantzer. Sub-optimal sensor scheduling with error bounds. In *Proceedings of the 16th IFAC World congress*, July 2005.

[2] B. D. O. Anderson and J. B. Moore. *Linear Optimal Control*. Prentice-Hall, 1971.

[3] P. Antsaklis and J. Baillieul (editors). Special issue on networked control systems. *IEEE Transactions on Automatic Control*, 49(9):1424–1603, September 2004.

[4] M. Athans. On the determination of optimal costly measurement strategies. *Automatica*, 18:397–412, July 1972.

[5] B. Azimi-Sadjadi. Stability of networked control systems in the presence of packet losses. In *Proceedings of 2003 IEEE Conference on Decision and Control*, Dec 2003.

[6] J. Baillieul. Feedback designs for controlling device arrays with communication channel bandwidth constraints. In *ARO workshop on smart structures*, 1999.

[7] B. Bamieh, F. Paganini, and M. A. Dahleh. Distributed control of spatially invariant systems. *IEEE Transactions on Automatic Control*, 47(7):1091–1107, July 2002.

[8] B. Bamieh and P. G. Voulgaris. Optimal distributed control with distributed delayed measurement. In *Proceedings of the IFAC World Congress*, 2002.

[9] Y. Bar-Shalom. On the track-to-track correlation problem. *IEEE Trans. Automat. Contr.*, AC-26:571–572, 1981.

[10] Y. Bar-Shalom and L. Campo. The effect of the common process noise on the two-sensor fused-track covariance. *IEEE Transactions on Aerospace and Electronic Systems*, AES-22:803–805, 1986.

[11] Y. Bar-Shalom and T. E. Fortman. *Tracking and Data Association*. Academic Press, San Diego, CA, 1988.

[12] Y. Bar-Shalom and X. R. Li. *Estimation and Tracking: Principles, Techniques and software*. Artech House, 1993.

[13] O. A. Basir and H. C. Shen. Informational m,aneuvering in dynamic environment. In *IEEE Int. Conf. on Systems, Man and Cybernetics*, volume 2, pages 999–1004, 1995.

[14] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikipoulos. The explicit Linear Quadratic Regulator for constrained systems. *Automatica*, 38(1):3–20, January 2002.

[15] T. M. Berg and H. F. Durrant-Whyte. Distributed and decentralized estimation. In *Proc. of the Singapore International Conference on Intelligent Control and Instrumentation*, volume 2, pages 1118–1123, 1992.

[16] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation*. Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[17] D.P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2nd edition, 2000.

[18] N. Biggs. *Algebraic Graph Theory*. Cambridge Tracks in Mathematics. Cambridge University Press, 1974.

[19] W.P. Blair and D.D. Sworder. Feedback control of a class of linear discrete systems with jump parameters and quadratic cost criteria. *International Journal of Control*, 21(5):833–841, 1975.

[20] V. Blondel, M. Gevers, and A. Lindquist. Survey on the state of systems and control. *European Journal of Control*, 1:5–23, 1995.

[21] V. Blondel and J. N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, 2000.

[22] B. Bollobas. *Random Graphs*. Academic Press, 1985.

[23] V. S. Borkar, S. K. Mitter, and S. Tatikonda. Markov control problems with communication constraints. *Communication in Information and Systems*, 1(1):16–33, 2001.

[24] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2003.

[25] R. W. Brockett and D. Liberzon. Quantized feedback stabilization of linear systems. *IEEE Transactions on Automatic Control*, 45(7):1279–89, 2000.

[26] J. D. Bruce. Optimum quantization. Technical Report 429, Research Laboratory of electronics, Massachusetts Institute of Technology, Cambridge, 1965.

[27] N. A. Carlson. Federated square root filter for decentralized parallel processes. *IEEE Transactions on Aerospace and Electronic Systems*, AES-26:517–525, 1990.

[28] K. C. Chang, R. K. Saha, and Y. Bar-Shalom. On optimal track-to-track fusion. *IEEE Transactions on Aerospace and Electronic Systems*, AES-33:1271–1276, 1997.

[29] H. J. Chizeck, A. S. Willsky, and D. Castanon. Discrete-time Markovian-Jump Linear Quadratic optimal control. *International Journal of Control*, 43(1):213–231, 1986.

[30] C. Y. Chong. Hierarchical estimation. In *Proceedings 2nd MIT/ONR C^3 Workshop*, 1979.

[31] C. Y. Chong, S. Mori, W. H. Barker, and K. C. Chang. Architectures and algorithms for track association and fusion. *IEEE Aerospace and Electronic Systems Magazine*, 15:5 – 13, 2000.

[32] C. Y. Chong, S. Mori, and K. C. Chang. Information fusion in distributed sensor networks. In *Proc. of the American Control Conference*, pages 830–835, 1985.

[33] M. Chu, S. Mitter, and F. Zhao. Distributed multiple target tracking and data association in ad hoc sensor networks. In *Proceedings of the Sixth International Conference of Information Fusion*, volume 1, pages 447 – 454, 2003.

[34] T. Chung, V. Gupta, B. Hassibi, J. Burdick, and R. M. Murray. Scheduling for distributed sensor networks with single sensor measurement per time step. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 04)*, volume 1, pages 187 – 192, 2004.

[35] T. H. Chung, V. Gupta, J. W. Burdick, and R. M. Murray. On a decentralized active sensing strategy using mobile sensor platforms in a network. In *Proceedings of the IEEE Conference on Decision and Control (CDC '04)*, volume 2, pages 1914 – 1919, December 2004.

[36] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley and Sons, New York, 1998.

[37] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 1990.

[38] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, April 2004.

[39] T. M. Cover and J. A. Thomas. *Information Theory*. Wiley and Sons, New York, 1991.

[40] L. Cremean, B. W. Dunbar, D. van Gogh, J. Hickey, E. Klavins, J. Meltzer, and R. M. Murray. The Caltech multi-vehicle wireless testbed. In *Proc of the 2002 Conference on Decision and Control*, 2002.

[41] A. F. Dana, V. Gupta, J. P. Hespanha, B. Hassibi, and R. M. Murray. Estimation over communication networks: Performance bounds and achievability results. In *American Control Conference (ACC 07)*, 2007. Submitted.

[42] R. D'Andrea and G. E. Dullerud. Distributed control design for spatially interconnected systems. *IEEE Transactions on Automatic Control*, 48(9):1478–1495, 2003.

[43] R. D'Andrea and R. Murray. The roboflag competition. In *Proc. of the American Control Conference*, pages pp. 650–655, June 2003.

[44] S. Darbha, S. Pargaonkar, and S. P. Bhattacharya. A linear programming approach to the synthesis of fixed structure controllers. In *Proceedings of the 2004 American Control Conference*, volume 5, pages 3942–3949, 2004.

[45] G. A. de Castro. *Convex Methods for the design of Structured Controllers*. PhD thesis, University of California, Los Angeles, 2003.

[46] D. F. Delchamps. Stabilizing a linear system with quantized state feedback. *IEEE Transactions on Automatic Control*, 35:916–924, 1990.

[47] K. Dohmen. Inclusion-exclusion and network reliability. *The Electronic Journal of Combinatorics*, 5(1):Research paper 36, 1998.

[48] J. L. Doob. *Stochastic Processes*. New York: John Wiley and Sons, 1953.

[49] O. E. Drummond. Tracklets and a hybrid fusion with process noise. *Proceedings of the SPIE, Signal and Data Processing of Small Targets*, 3163, 1997.

[50] Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: applications and algorithms. *SIAM Review*, 41(4):637–676, 1999.

[51] G. E. Dullerud and F. Paganini. *A course in robust control theory: A convex approach*. Springer - New York, 2000.

[52] W. B. Dunbar and R. M. Murray. Distributed receding horizon control with application to multi-vehicle formation stabilization. *Automatica*, 2(4):549–558, 2006.

[53] R. Bellman (editor). Special issue on large systems. *IEEE Transactions on Automatic Control*, 19(5):465–465, October 1974.

[54] N. Elia and S. K. Mitter. Stabilization of linear systems with limited information. *IEEE Transactions on Automatic Control*, 46(9):1384–1400, 2001.

[55] J. E. Elliot, L. Aggoun, and J. B. Moore. *Hidden Markov Models: Estimation and Control*. Springer-Verlag, 1995.

[56] E. O. Elliott. Estimates of error rates for codes on burst-noise channels. *Bell Systems Technical Journal*, 42:1977–1997, September 1963.

[57] F. Fagnani and S. Zampieri. Stability analysis and synthesis for scalar linear systems with a quantized feedback. *IEEE Transactions on Automatic Control*, 48(9):1569–1584, 2003.

[58] C. Fan, J. L. Speyer, and Jaensch. Centralized and decentralized solutions of the Linear-Exponential-Gaussian problem. *IEEE Transactions on Automatic Control*, 39(10):1986–2003, October 1994.

[59] J. A. Fax. *Optimal and Cooperative Control of Vehicle Formations*. PhD thesis, California Institute of Technology, 2001.

[60] J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465–1476, September 2004.

[61] X. Feng, K.A. Loparo, Y. Ji, and H.J. Chizeck. Stochastic stability properties of Jump Linear systems. *IEEE Transactions on Automatic Control*, 37(1):38–53, 1992.

[62] M. Fleming, Q. Zhao, and M. Effros. Network vector quantization. *IEEE Transactions on Information Theory*, 50(8):1584–1604, 2004.

[63] Jr. G. D. Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 6:268–278, January 1973.

[64] Z. Gajic and M. T. J. Qureshi. *Lyapunov Matrix Equation in System Stability and Control*. Academic Press, Inc, 1995.

[65] A. A. El Gamal and T. Cover. Achievable rates for multiple descriptions. *IEEE Transactions on Information Theory*, 28(6):851–857, 1982.

[66] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1991.

[67] E. N. Gilbert. Capacity of burst-noise channels. *Bell Systems Technical Journal*, 39:1253–1265, Spetember 1960.

[68] Arvind Giridhar and P. R. Kumar. Distributed clock synchronization over wireless networks: Algorithms and analysis. In *45th IEEE Conference on Decision and Control (CDC)*, 2006. Submitted.

[69] V. Goyal and J. Kovacevic. Generalized multiple decription coding with correlated transforms. *IEEE Transactions on Information Theory*, 47(6):2199–2224, 2001.

[70] V. Goyal, J. Kovacevic, R. Arean, and M. Vetterli. Multiple description transform coding of images. In *Proceedings of the IEEE International Conference on Image Processing*, volume 1, pages 674–678, October 1998.

[71] I. S. Gradshteyn and I. M. Ryzhik. *Tables of integrals, Series, and Products*. Academic Press, 2000.

[72] S. Graham, G. Baliga, and P. R. Kumar. Issues in the convergence of control with communication and computing: Proliferation, architecture, design, services, and middleware. In *Proceedings of the 43rd IEEE Conference on Decision and Control (CDC 04)*, pages 1466–1471, December 2004.

[73] R. M. Gray. *Source Coding Theory*. Kluwer Academic Publishers, 1990.

[74] P. Grieder, F. Borrelli, F. Torrisi, and M. Morari. Computation of the constrained infinite time Linear Quadratic Regulator. In *Proceedings of the American Control Conference*, volume 6, pages 4711–4716, June 2003.

[75] V. Gupta, T. Chung, B. Hassibi, and R. M. Murray. Sensor scheduling algorithms requiring limited computation. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages iii – 825–8, May 2004.

[76] V. Gupta, T. Chung, B. Hassibi, and R. M. Murray. On a stochastic algorithm for sensor scheduling. In *Proceedings of the 16th IFAC World Congress (IFAC 05)*, July 2005.

[77] V. Gupta, T. H. Chung, B. Hassibi, and R. M. Murray. On a stochastic sensor selection algorithm with applications in sensor scheduling and sensor coverage. *Automatica*, 42(2):251–260, February 2006.

[78] V. Gupta, A. F. Dana, J. P. Hespanha, and R. M. Murray. Data transmission over networks for estimation. In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems, (MTNS 06)*, 2006.

[79] V. Gupta, A. F. Dana, R. M. Murray, and B. Hassibi. On the effect of quantization on performance at high rates. In *Proceedings of the American Control Conference (ACC 06)*, pages 1364–1369, June 2006.

[80] V. Gupta, B. Hassibi, and R. M. Murray. On the control of jump linear markov systems with markov state estimation. In *Proceedings of the American Control Conference (ACC 03)*, volume 4, pages 2893 – 2898, June 2003.

[81] V. Gupta, B. Hassibi, and R. M. Murray. Stability analysis of stochastically varying formations of dynamic agents. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, volume 1, pages 504–509, December 2003.

[82] V. Gupta, B. Hassibi, and R. M. Murray. On the synthesis of control laws for a network of autonomous agents. In *Proceedings of the American Control Conference*, volume 6, pages 4927–4932, June 2004.

[83] V. Gupta, B. Hassibi, and R. M. Murray. On sensor fusion in the presence of communication channels. In *IEEE Conference on Decision and Control*, pages 3547–3552, 2005.

[84] V. Gupta, B. Hassibi, and R. M. Murray. On sensor fusion in the presence of packet-dropping communication channels. In *Proceedings of the IEEE Conference on Decision and Control (CDC 05)*, pages 3547–3552, 2005.

[85] V. Gupta, B. Hassibi, and R. M. Murray. A sub-optimal algorithm to synthesize control laws for a network of dynamic agents. *International Journal of Control*, 78(3):1302–1313, November 2005.

[86] V. Gupta, C. Langbort, and R. M. Murray. When are distributed algorithms robust? In *IEEE Conference on Decision and Control*, 2006. Accepted.

[87] V. Gupta, R.M. Murray, and B. Hassibi. On the control of jump linear markov systems with markov state estimation. In *Proceedings of the American Control Conference*, 2003.

[88] V. Gupta, D. Spanos, B. Hassibi, and R. M. Murray. Optimal lqg control across packet-dropping links. In *Proceedings of the American Control Conference (ACC 05)*, pages 360 – 365, 2005.

[89] V. Gupta, D. Spanos, B. Hassibi, and R. M. Murray. Optimal lqg control across packet-dropping links. *Systems and Control Letters*, July 2006. Accepted.

[90] C. N. Hadjicostis and R. Touri. Feedback control utilizing packet dropping network links. In *Proceedings of the IEEE Conference on Decision and Control*, 2002.

[91] F. Harary. *Graph Theory*. Addison-Wesley, 1994.

[92] H. R. Hashemipour, S. Roy, and A. J. Laub. Decentralized structures for parallel Kalman filtering. *IEEE Transactions on Automatic Control*, AC-33(1):88–94, 1988.

[93] M. F. Hassan, G. Salut, M. G. Singh, and A. Titli. A decentralized computational algorithm for the global Kalman filter. *IEEE Transactions on Automatic Control*, AC-23:262–268, 1978.

[94] A. Hassibi, S. P. Boyd, and J. P. How. Control of asynchronous dynamical systems with rate constraints on events. In *Proceedings of the IEEE Conference on Decision and Control*, pages 1345–1351, Dec 1999.

[95] B. Hassibi, A. H. Sayed, and T. Kailath. *Indefinite-Quadratic Estimation and Control*. Studies in Applied and Numerical Mathematics, 1999.

[96] T. C. H. Heng, Y. Kuno, and Y. Shirai. Active sensor fusion for collision avoidance. In *Proc of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, volume 3, pages 1244–1249, 1997.

[97] K. Herring and J. Melsa. Optimum measurements for estimation. *IEEE Transactions on Automatic Control*, 19(3):264–266, June 1974.

[98] J. Hespanha, A. Ortega, and L. Vasudevan. Towards the control of linear systems with minimum bit-rate. In *Proceedings of the 15th International Symposium on Mathematical Theory of Networked Systems*, 2002.

[99] Y. C. Ho and K. C. Chu. Team decision theory and information structures in optimal control problems - part i. *IEEE Transactions on Automatic Control*, AC-17(1):15–22, January 1972.

[100] P. Hovareshti and J. S. Baras. Decentralized control of networks of dynamic agents with invariant nodes: A probabilistic view. Technical Report ISR TR 2004-39, Institute of Systems Research, University of MAryland, College Park, 2004.

[101] M. E. Liggins II, C. Y. Chong, I. Kadar, M. G. Alford, V. Vannicola, and S. Thomopoulos. Distributed fusion architechtures and algorithms for target tracking. *Proceedings of the IEEE*, 85:95–107, 1997.

[102] L. Meier III, J. Peschon, and R. Dressler. Optimal control of measurement subsystems. *IEEE Transactions on Automatic Control*, 12:528–536, October 1967.

[103] O.C. Imer, S. Yuksel, and T. Basar. Optimal control of LTI systems over communication networks. *Automatica*, July 2006. To Appear.

[104] H. Ishii and B. A. Francis. Quadratic stabilization of sampled-data systems with quantization. *Automatica*, 39:1793–1800, 2003.

[105] P. Ishwar, R. Puri, K. Ramchandran, and S. S. Pradhan. On rate-constrained distributed estimation in unreliable sensor networks. *IEEE Journal on Selected Areas in Communications: Special issue on self-organizing distributed collaborative sensor networks*, 23(4):765–775, April 2005.

[106] M. O. Jackson. A survey of models of network formation: Stability and efficiency. In G. Demange and M. Wooders, editors, *Group Formation in Economics: Networks, Clubs and Coalitions*. Cambridge University Press, 2004.

[107] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[108] Y. Ji and H.J. Chizeck. Controllability, stabilizability and continuous-time Markovian Jump Linear Quadratic control. *IEEE T. Automatic Control*, 35(7):777–788, 1990.

[109] Z. Jin, V. Gupta, B. Hassibi, and R. M. Murray. State estimation utilization multiple description coding over lossy networks. In *Proceedings of the IEEE Conference on Decision and Control (CDC 05)*, pages 872–878, 2005.

[110] Z. Jin, V. Gupta, and R. M. Murray. State estimation over packet dropping networks using multiple description coding. *Automatica*, 42(9):1441–1452, September 2006.

[111] S. Kagami and M. Ishikawa. A sensor selection method considering communication delays. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, volume 1, pages 206–211, 2004.

[112] T. Kailath, A. H. Sayed, and B. Hassibi. *Linear Estimation*. Prentice Hall, 2000.

[113] T. H. Kerr. Modeling and evaluating an empirical INS difference monitoring procedure used to sequence SSBN navaid fixes. *Journal of the Institute of Navigation*, 28(4):263–285, winter 1981-82.

[114] T. H. Kerr and Y. Oshman. Further comments on 'optimal sensor selection strategy for discrete-time estimators'[and reply]. *IEEE Transactions on Aerospace and Electronic Systems*, 31:1159 – 1167, July 1995.

[115] T. Keviczky. *Decentralized Receding Horizon Control of Large Scale Dynamically Decoupled Systems*. PhD thesis, University of Minnesota, September 2005.

[116] D. L. Kleinman, T. Fortmann, and M. Athans. On the design of linear systems with piecewise-constant feedback gains. *IEEE Transactions on Automatic Control*, AC-13(4):354–361, August 1968.

[117] V. Krishnamurthy. Algorithms for optimal scheduling and management of hidden Markov model sensors. *IEEE Transactions on Signal Processing*, 60(6):1382–1397, June 2002.

[118] H. Kwakernaak and R. Sivan. *Linear Optimal Control Systems*. John Wiley and Sons, 1972.

[119] Guest Editor L. Bushnell. Special issue on networks and control. *IEEE Control Systems Magazine*, 21(1), Feb 2001.

[120] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):392–401, 1982.

[121] P. Lancaster. *Theory of Matrices*. Academic Press, 1969.

[122] C. Langbort, R. S. Chandra, and R. D'Andrea. Distributed control design for systems interconnected over an arbitrary graph. *IEEE Transactions on Automatic Control*, 49(9):1502–1519, September 2004.

[123] C. Langbort and V. Gupta. Minimal interconnection topology in distributed control design. In *Proceedings of the American Control Conference (ACC 06)*, pages 845–850, June 2006.

[124] C. Langbort, V. Gupta, and R. M. Murray. *Distributed Control over Failing Channels*, pages 325–342. LNCIS 331. Springer, 2006. Presented at the Workshop on Networked Embedded Sensing and Control, October 2005.

[125] W. S. Lee, M. R. Pickering, M. R. Frater, and J. F. Arnold. A robust codec for transmission of very low bit-rate video over channels with bursty errors. *IEEE Transactions on Circuits, Systems and Video Technology*, 10:1403–1412, December 2000.

[126] M. D. Lemmon and Q. Ling. Control system performance under dynamic quantization: The scalar case. In *Proceedings of the IEEE Conference on Decision and Control*, 2004.

[127] W. S. Levine, T. L. Johnson, and M. Athans. Optimal limited state variable feedback controllers for linear systems. *IEEE Transactions on Automatic Control*, AC-16:785–793, 1971.

[128] B. C. Levy, D. A. Castanon, G. C. Verghese, and A. S. Willsky. A scattering framework for decentralized estimation problem. *Automatica*, 19:373–384, 1983.

[129] D. Liberzon. On stabilization of linear systems with limited information. *IEEE Transactions on Automatic Control*, 48(2):304–307, 2003.

[130] Bo Lincoln and Bo Bernhardsson. LQR optimization of linear system switching. *IEEE Transactions on Automatic Control*, 47:1701–1705, October 2002.

[131] Q. Ling and M. D. Lemmon. Power spectral analysis of netwroked control systems with data droputs. *IEEE Transactions on Automatic control*, 49(6):955–960, June 2004.

[132] Q. Ling and M.D. Lemmon. Optimal dropout compensation in networked control systems. In *Proc. of the IEEE Conference on Decision and Control*, 2003.

[133] Q. Ling and M.D. Lemmon. Soft real-time scheduling of networked systems with dropouts governed by a Markov chain. In *Proceedings of the American Conference on Control*, 2003.

[134] X. Liu and A. Goldsmith. Kalman filtering with partial observation losses. In *Proceedings of the 43rd IEEE Conference on Decision and Control (CDC)*, volume 4, pages 4180–4186, December 2004.

[135] R. Luck and A. Ray. An observer-based compensator for distributed delays. *Automatica*, 26(5):903–908, 1990.

[136] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.

[137] D. Marco and D. L. Neuhoff. The validity of the additive noise model for uniform scalar quantizers. *IEEE Transactions on Information Theory*, 51:1739–1755, May 2005.

[138] J. Marschak and R. Radner. *Economic Theory of Teams*. Cowles Foundation Monograph 22. Yale University Press, 1972.

[139] S. Martinez, F. Bullo, J. Cortes, and E. Frazzoli. On synchronous robotic networks - Part I: Models, tasks and complexity notions. *IEEE Transactions on Automatic Control*, 2006. Submitted.

[140] S. Martinez, F. Bullo, J. Cortes, and E. Frazzoli. On synchronous robotic networks - PartII: Time complexity of rendezvous and deployment algorithms. *IEEE Transactions on Automatic Control*, 2006. Submitted.

[141] L. Mihaylova, T. Lefebvre, H. Bruyninckx, K. Gadeyne, and J. De Schutter. Active sensing for robotics - a survey. In *Proceedings of the 5th International Conference on Numerical Methods and Applications*, pages 316–324, August 2002.

[142] B. M. Miller and W. J. Runggaldier. Optimization of observations: a stochastic control approach. *SIAM Journal of Control and Optimization*, 35(5):1030–1052, 1997.

[143] M. Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, pages 321–350, 2002.

[144] H. Mori. Active sensing in vision-based stereotyped motion. In *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems '90*, volume 1, pages 167–174, 1990.

[145] S. Mori, W. H. Barker, C. Y. Chong, and K. C. Chang. Track association and track fusion with nondeterministic target dynamics. *IEEE Transactions on Aerospace and Electronic Systems*, AES-38:659–668, 2002.

[146] T. Mukai and I. Ishikawa. An active sensing method using estimated errors for multisensor fusion systems. In *Proceedings of the 1994 IEEE International Conference on Multisensor Fusion, Integration Intelligent Systems*, pages 615–622, 1994.

[147] T. Mukai and I. Ishikawa. An active sensing method using estimated errors for multisensor fusion systems. *IEEE Transactions on Industrial Electronics*, 43:380–386, 1996.

[148] R. Myerson. Graphs and cooperation in games. *Mathematics of Operations Research*, 2:225–229, 1977.

[149] G. N. Nair, S. Dey, and R. J. Evans. Infimum data rates for stabilising Markov jump linear systems. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 1176–81, 2003.

[150] G. N. Nair and R. J. Evans. Stabilizability of stochastic linear systems with finite feedback data rates. *SIAM Journal on Control and Optimization*, 43(2):413–436, July 2004.

[151] T. Nakamoto, H. Ishida, and T. Moriizumi. Active odor sensing system. In *Proc. IEEE Int. Symposium Industrial Electronics, 1997*, volume 1, pages SS128–SS133, 1997.

[152] J. Nilsson. *Real-Time Control Systems with Delays*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, 1998.

[153] J. Nilsson and B. Bernhardsson. LQG control over a Markov communication network. In *Proceedings of the 36th IEEE Conference on Decision and Control*, volume 5, pages 4586–4591, 1997.

[154] R. Niu, P. Willett, and Y. Bar-Shalom. Tracking considerations in selection of RADAR waveform for range and range-rate measurements. *IEEE Transactions on Aerospace and Electronic Systems*, 38(2):467–487, 2002.

[155] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tesselations: Concept and Applications of Voronoi Diagrams*. Wiley series in Probability and Statistics. John Wiley and sons, second edition, 2000.

[156] A. Okabe and A. Suzuki. Locational optimization problems solved through Voronoi diagrams. *European Journal of Operational Research*, 98(3):445–456, 1997.

[157] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49:1520–1533, September 2004.

[158] Y. Oshman. Optimal sensor selection strategy for discrete-time state estimators. *IEEE Transactions on Aerospace and Electronic Systems*, 30:307–314, April 1994.

[159] C. H. Papadimitriou and J. N. Tsitsiklis. Intractable problems in control theory. *SIAM Journal on Control and Optimization*, 24:639–654, 1986.

[160] I. R. Petersen and A. V. Savkin. Multi-rate stabilization of multivariable discrete-time linear systems via a limited capacity communication channel. In *Proceedings of the 40th IEEE Conference on Decision and Control*, pages 304–309, 2001.

[161] C. Rago, P. Willet, and Y. Bar-Shalom. Censoring sensor: a low communication rate scheme for distributed detection. *IEEE Transactions on Aerospace and Electronic Systems*, 28(2):554–568, 1996.

[162] K. Ramachandra. *Kalman filtering techniques for radar tracking*. Marcel Dekker Inc., New York, 2000.

[163] B. S. Rao and H. F. Durrant-Whyte. Fully decentralized algorithm for multi-sensor Kalman filtering. *IEE proceedings*, 138, 1991.

[164] W. Ren. *Consensus Seeking, Formation Keeping, and Trajectory Tracking in Multiple Vehicle Cooperative Control*. PhD thesis, Brigham Young University, August 2004.

[165] W. Ren and R. W. Beard. Consensus seeking in multi-agent systems using dynamically changing interaction topologies. *IEEE Transactions on Automatic Control*, 5(5):655–661, May 2005.

[166] C. Robinson and P. R. Kumar. Control over networks of unreliable links: Location of controllers, control laws and bounds on performance. In *45th IEEE Conference on Decision and Control*, 2006. submitted.

[167] J. A. Roecker and C. D. McGillem. Comparison of two-sensor tracking methods based on state vector fusion and measurement fusion. *IEEE Transactions on Aerospace and Electronic Systems*, AES-24:447–449, 1988.

[168] M. Rotkowitz and S. Lall. Decentralized control subject to communication and propagation delays. In *Proceedings of the 43-rd IEEE Conference on Decision and Control (CDC)*, 2004.

[169] M. Rotkowitz and S. Lall. A characterization of convex problems in decentralized control. *IEEE Transactions on Automatic Control*, 51(2):274–286, February 2006.

[170] A. Sahai. *Anytime Information Theory*. PhD thesis, MIT, Cambridge, MA, 2001.

[171] N. Sandell and M. Athans. Solution of some nonclassical LQG stochastic decision problems. *IEEE Transactions on Automatic Control*, AC-19:108–116, April 1974.

[172] A. Savkin, R. Evans, and E. Skafidas. The Problem of Optimal Robust Sensor Scheduling. In *Proceedings of the 39th Conference on Decision and Control*, volume 1, Sydney, Australia, December 2000. CDC.

[173] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. S. Sastry. Foundations of control and estimation ovber lossy networks. *Proceedings of the IEEE*, 2006. To Appear.

[174] P. Seiler. *Coordinated Control of unmanned aerial vehicles*. PhD thesis, University of California, Berkeley, 2001.

[175] P. Seiler and R. Sengupta. An H_∞ approach to networked control. *IEEE Transactions on Automatic Control*, 50(3):356–364, 2005.

[176] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, July 1948.

[177] D. K. Sharma. Design of absolutely optimal quantizers for a wide class of distortion measures. *IEEE Transactions on Information Theory*, IT-24(6):693–702, November 1978.

[178] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. Jordan, and S. Sastry. Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control*, 49(9):1453–1464, Sep 2004.

[179] M. Slikker and A. van den Nouweland. *Social and Economic Networks in Cooperative Game Theory*. Kluwer Academic Publishers, 2001.

[180] R. S. Smith and F. Y. Hadaegh. Parallel estimation and control architectures for deep-space formation flying spacecraft. In *IEEE Aerospace Conference, 2006*, pages 1–12, March 2006.

[181] T. Soderstrom. On some algorithms for design of optimal constrained regulators. *IEEE Transactions on Automatic Control*, AC-23:1100–1101, 1978.

[182] D.P. Spanos, R. Olfati-Saber, and R.M. Murray. Distributed sensor fusion using dynamic consensus. In *Proceedings of the 2005 IFAC World Congress*, 2005.

[183] J. L. Speyer. Computation and transmission requirements for a decentralized linear-quadratic Gaussian control problem. *IEEE Transactions on Automatic Control*, AC-24:266–269, 1979.

[184] V. L. Syrmos, C. T. Abdullah, P. Dorato, and K. Grigoriadis. Static output feedback - a survey. *Automatica*, 33(2):125–137, 1997.

[185] S. Tatikonda. *Control under Communication Constraints*. PhD thesis, MIT, Cambridge, MA, 2000.

[186] S. Tatikonda. Some scaling properties of large scale distributed control systems. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, volume 3, pages 3142–3147, 2003.

[187] A. Tiwari, M. Jun, D. E. Jeffcoat, and R. M. Murray. The dynamic sensor coverage problem. In *Proceedings of the IFAC World Congress*, 2005.

[188] N. C. Tsai and A. Ray. Stochastic optimal control under randomly varying distributed delays. *International Journal of Control*, 68(5):1179–1202, Nov 1997.

[189] K. Umeda, J. Ota, and H. Kimura. Fusion of multiple ultrasonic sensor data and imagery data for measuring moving obstacle's motion. In *Proceedings of the International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 742–748. IEEE/SICE/RSJ, December 1996.

[190] V. A. Vaishampayan. Design of multiple description scalar quantizers. *IEEE Transactions on Information Theory*, 39(3):821–834, 1993.

[191] E. Verriest and M. Egerstedt. Control with delayed and limited information: A first look. In *Proceedings of the IEEE Conference on Decision and Control*, 2002.

[192] T. Vicsek, A. Czirok, E. Ben-Jacob, I. Cohen, and O. Shochet. Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, 75(6-7):1226–1229, 1995.

[193] R. Vidal, O. Shakernia, H. Kim, D. Shim, and S. Sastry. Probabilistic pursuit-evasion games: Theory, implementation and experimental evaluation. *IEEE Transactions on Robotics and Automation*, 18(5):662–669, October 2002.

[194] P. Voulgaris. Control of nested systems. In *Proceedings of the American Control Conference (ACC)*, 2000.

[195] P. G. Voulgaris. Optimal control of systems with delayed observation sharing patterns via input-output methods. *Systems and Control Letters*, 50:51–64, 2003.

[196] H. S. Wang and N. Moayeri. Finite-state markov channel - a useful model for radio communication channels. *IEEE Transactions on Vehicular Technology*, 44(1):163–171, February 1995.

[197] S. Wang and E. J. Davison. On the stabilization of decentralized control systems. *IEEE Transactions on Automatic Control*, 18(5):473–478, October 173.

[198] S. Waydo, Z. Jin, E. Wildanger, M. Lammers, H. Scholze, P. Foley, R. M. Murray, and D. Held. MVWT-II: The second generation caltech multi-vehicle wireless testbed. In *American Control Conference*, 2004.

[199] C. J. Wenk and C. H. Knapp. Parameter optimization in linear systema with arbitrarily constrained controller structure. *IEEE Transactions on Automatic Control*, AC-25(3):496–500, 1980.

[200] D. Willner, C. B. Chang, and K. P. Dunn. Kalman filter algorithms for a multisensor system. In *Proceedings of the 15th Conference on Decision and Control*, pages 570–574, 1976.

[201] A. S. Willsky, M. G. Bello, D. A. Castanon, B. C. Levy, and G. C. Verghese. Combining and updating of local estimates and regional maps along sets of one-dimensional tracks. *IEEE Transactions on Automatic Control*, AC-27:799–813, 1982.

[202] H. S. Witsenhausen. A counterexample in stochastic optimum control. *SIAM Journal of Control*, 6(1):131–147, 1968.

[203] H. S. Witsenhausen. Separation of estimation and control for discrete time systems. *Proceedings of the IEEE*, 59(11):1557–1566, 1971.

[204] W. S. Wong and R. W. Brockett. Systems with finite communication bandwidth-part I: State estimation problems. *IEEE Transactions on Automatic Control*, 42(9):1294–1298, 1997.

[205] W. S. Wong and R. W. Brockett. Systems with finite communication bandwidth-part II: Stabilization with limited information feedback. *IEEE Transactions on Automatic Control*, 44(5), 1999.

[206] P. Voulgaris X. Qi, M. Salapaka and M. Khammash. Structured optimal and robust control with multiple criteria: A convex solution. *IEEE Transactions on Automatic Control*, 49(10):1623–1640, October 2004.

[207] Q. Zhang and S. A. Kassam. Finite-state Markov model for Rayleigh fading channels. *IEEE Transactions on Communications*, 47(11):1688–1692, 1999.

[208] W. Zhang, M. S. Branicky, and S. M. Philips. Stability of networked control systems. *IEEE Control System Magazine*, 21(1):84–89, Feb 2001.

[209] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing Magazine*, 19(2):61–72, March 2002.