Towards a Cognitive Network Management and Control System

by

Arman Rezaee

B.S., Arizona State University (2009)
S.M., Massachusetts Institute of Technology (2011)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2020

Signature redacted

Author ....
Department of Electrical Engineering and Computer Science
January 25, 2020

Signature redacted

Certified by ..
Vincent W.S. Chan
Joan and Irwin Jacobs Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Signature redacted

Accepted by ....
Leslie Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Towards a Cognitive Network Management and Control System

by

Arman Rezaee

Submitted to the Department of Electrical Engineering and Computer Science
on January 25, 2020, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Future networks have to accommodate an increase of 3-4 orders of magnitude in data rates with heterogeneous session sizes and strict time deadline requirements. The dynamic nature of scheduling of large transactions and the need for rapid actions by the network management and control system, require timely and judicious collection of network state information. Within this context we will focus on the problem of shortest path routing, and identify pragmatic schemes that allow a central controller to collect relevant delay statistics from various links and nodes within the network.

We present *Significant Sampling* as an adaptive monitoring technique to collect and disseminate network state information when it can be of significant value to the optimal operation of the network, and in particular when it can help in identifying the shortest routes. We start by developing an analytical framework that can identify the optimal time for the collection of such information in a small but realistic setting, when the underlying delay model is a continuous-time diffusion process (e.g. Wiener process or Ornstein-Uhlenbeck process) and its parameters are known by the controller. We show that this technique balances the need for updated state information against the collection and dissemination costs and provides an algorithm that yields near optimum performance.

We then extend the results by introducing a reinforcement learning framework that learns the aforementioned optimal policy from its own interactions with the network, and without any prior assumptions regarding the underlying delay model. In addition to achieving a performance comparable to the analytically derived policies, the deep reinforcement learning solution is more flexible and general and can accommodate a diverse set of network environments. This is particularly important because it can provide good solutions for complex network environments where analytically tractable solutions are not feasible.

We conclude our work by noting that sensible network controllers should continue to deliver a good performance between distinct instances of state collection and thus any meaningful solution should strive to meet application demands despite the unavoidable uncertainty about the instantaneous state of the network. To that end, we introduce a novel diversity routing scheme that can accommodate requirements regarding delay variations despite a controller's relative uncertainty about the instantaneous state of the network. More specifically we utilize *mean-variance analysis* as the basis for traffic distribution and route selection, and show that this technique can improve the users' quality of service by taking into account the correlated nature of delay across different paths. We conclude this work by commenting on the potential application of this method to general transportation networks.

Thesis Supervisor: Vincent W.S. Chan
Title: Joan and Irwin Jacobs Professor of Electrical Engineering

Acknowledgments

I owe my deepest gratitude to my advisor Professor Vincent Chan. It is hard to put into words what he has done for me during my time as a graduate student. From a technical perspective, he has taught me to look beyond the mathematical expressions and try to understand and identify the fundamental aspects of a problem. Of course, his knowledge and wisdom goes far beyond the technical insights and I have learned enumerable life-lessons from him. I will continue to look up to him for years to come. I also wish to thank my committee members Professor Robert Gallager and Alan Kirby, who have guided me with patience and vital feedback. It would be an understatement to say that Bob was instrumental in my decision to pursue my doctoral studies. Not only has he taught me to think deeply about problems, but more importantly his compassion has inspired me to serve as a mentor and ally to others. Alan has brought engineering clarity to my research and has reminded me that posing the right questions is far more important than I had imagined. I am grateful for his time and dedication throughout the process.

During the last few years, I have had the opportunity to work alongside an amazing group of students. I would like to start by thanking my fellow groupmates: Manishika Agaskar, Matthew Carey, Shane Fink, Henna Huang, Weerachai Neeranartvong, Andrew Puryear, Yulin Shao, Anny Zhang, and Lei Zhang. Thank you for listening to my far-fetched ideas and creating a friendly environment that I looked forward to day after day.

Of course, I cannot forget the life-long friends that I have made, many of whom started their doctoral programs with me: Flavio Calmon, Weifei Zeng, Ahmad Beirami, Salman Salamatian, Jason Cloud, Anuran Makur, Ganesh Ajjanagadde, Mohamed AlHajri, David Qiu, Hajir Roozbehani, Soheil Feizi, Shirley Shi, Ali Makhdoumi, Azarakhsh Malekian, Georgious Angelopoulos, Ali ParandehGheibi, Kerim Fouli, and Vitaly Abdrashitov. We went through so many adventures with one another, from taking classes and getting through the quals to learning to ski on the bunny slopes. I could not have asked for a better crew to join during this epic adventure.

Of course, I have been blessed with friends who have enriched my life far beyond the lab: Marzieh, Hamed, Alborz, Nina, Ghazal, Payman, Amir, Mahsa, Ali, Mina, Amir-Bahador, Naghmeh, Damoon, Elham, Afrooz, Saman, Elica, Reza, Iman, and Hamid. You have made my time in Boston so memorable and I am forever grateful for that. Needless to say, Omid Abari has been a constant source of laughter and comradery and I'm indebted to his friendship.

I would like to thank my amazing family that has cheered me on for as long as I can remember. Their sacrifices and loving encouragements have been my guiding light throughout this journey. Iraj, Giti, Mehrshid, Arash, Ahdieh, Hedieh, Armin, Aram, Rozita, Rojan, and Rastan. I couldn't have done this with out you.

Finally and most importantly, I would like to thank my wife, Donna, for her unwavering love and support. I cannot put into words the countless ways in which you have brightened my days. You have been there for me through the many ups and downs, and I couldn't have done this without your support. It is to you that I dedicate this thesis.

To the loving memory of my parents.

Previously Published Material

The material Chapter 2 is in part based on [1]: A. Rezaee, V. Chan. "Cognitive Network Management and Control with Significantly Reduced State Sensing". GLOBECOM, 2018.

The material Chapter 2 is in part based on [2]: A. Rezaee, V. Chan. "Cognitive Network Management and Control with Significantly Reduced State Sensing". IEEE Transactions on Cognitive Communications and Networking, 2019.

The material in Chapter 3 is in part based on [3]: Y. Shao, A. Rezaee, S. C. Liew, and V. Chan. "Significant Sampling for Shortest Path Routing: A Deep Reinforcement Learning Solution". GLOBECOM, 2019.

The material in Chapter 3 is in part based on a submission to IEEE Journal on Selected Areas in Communications – Special Issue on Advances in Artificial Intelligence and Machine Learning for Networking: Y. Shao, A. Rezaee, S. C. Liew, and V. Chan. "Significant Sampling for Shortest Path Routing: A Deep Reinforcement Learning Solution". JSAC, 2020.

The material in Chapter 4 is in part based on [4]: A. Rezaee and V. Chan. "Diversity Routing to Improve Delay-Jitter Tradeoff in Uncertain Network Environments". ICC, 2019.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We are in the midst of a major technological storm that will change the landscape of networking for years to come. The introduction and adoption of high-definition (HD) video and a myriad of new applications that depend on it are the primary drivers of this transformation. According to Cisco, IP video traffic will be 82 percent of all IP traffic by 2021, up from 73 percent in 2016 [9, 10]. The same reports forecast live video to grow 15-fold while virtual reality (VR) and augmented reality (AR) traffic will increase 20-fold in the same period. As it stands, the majority of the global video content is intended for human consumption. But the advent of cheap and versatile Big Data applications such as facial recognition softwares, healthcare monitoring systems, and autonomous driving will tip the scale in favor of video production for machine consumption. In fact, machine-to-machine communications required for real-time applications will soon become the dominant type of data transfers over wide-area networks.

The proliferation of these applications presents a chicken and egg problem for network engineers: on the one hand, these applications require high bandwidth availability, low latency, as well as extreme reliability but at the same time the bursty and dynamic nature of their traffic introduces unpredictable delay and amplifies the jitter. Furthermore, the continuous variation and abrupt changes introduced by exogenous traffic will create temporary bottlenecks in the network which can drastically deteriorate the user's Quality of Service (QoS) and Quality of Experience (QoE).

The International Telecommunication Union [11] divides these applications into three broad categories, namely enhanced mobile broadband (eMBB), ultra-reliable and low-latency communications (URLLC), and massive machine type communications (mMTC) as depicted in Figure 1-1. They state the minimum requirements for latency as $4ms$ for eMBB and $1ms$ for URLLC. More specifically the URLLC applications expect a packet to arrive at the destination within 1 ms of transmission with packet loss rate of at most 10^{-5}. It should come as no surprise that such stringent requirements can only be achieved if the network control plane itself is very agile, responsive, and reliable. In fact [11] sets the minimum latency requirement for the control plane to be between 10-20 ms.

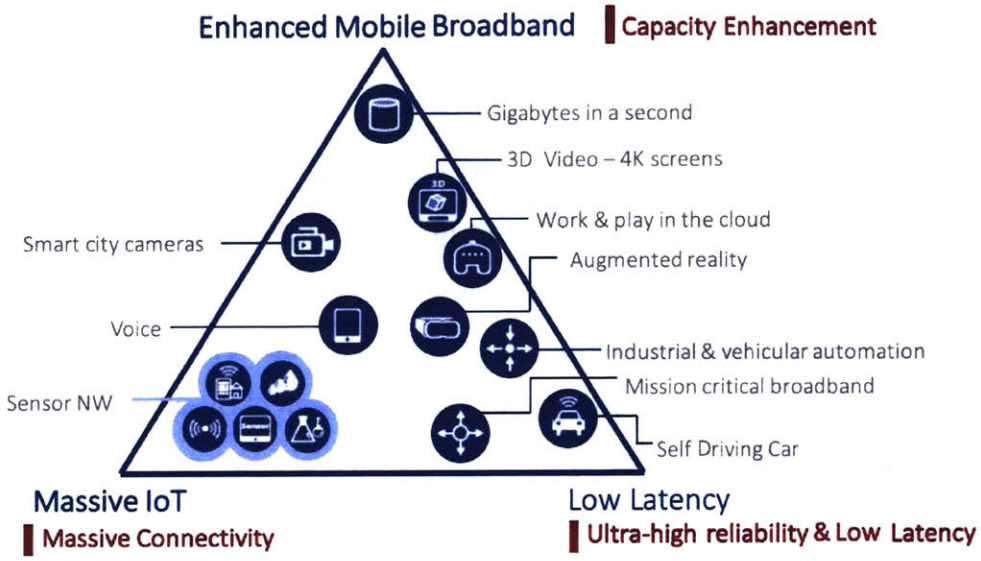


Figure 1-1: Broad categories of the 5G communication systems. Figure obtained from [5] which was adopted from [6].

Consequently, the networking landscape created by this heterogeneous set of applications and products is no longer static, or even quasi-static. The input traffic is continuously varying and can experience abrupt changes. The bursty and dynamic nature of the traffic generated by these applications requires quick (100 ms - 1 s) network adaptation to maintain quality of service and experience [12]. Unfortunately, current Network Management and Control (NMC) systems are much too slow and their operational paradigm does not scale well with network size and traffic intensity.

Before delving into the details let us sketch the bird's eye view of the roles and responsibilities of a properly designed NMC system as it interacts with the application and various network protocols and resources. Figure 1-2 depicts a hierarchical illustration of such a system. The NMC system is supposed to monitor the state of the network at all layers, reconfigure network resources when necessary, and provide data and instructions to applications upon request.

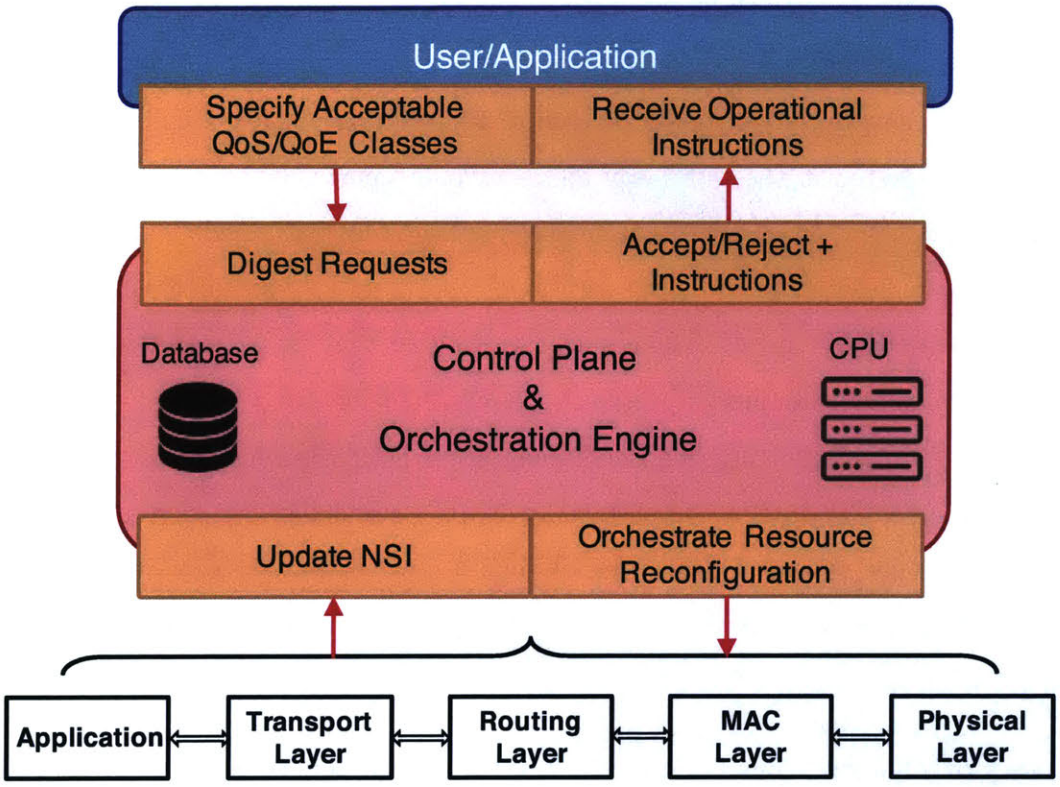


Figure 1-2: Roles and responsibilities of the Network Management and Control system. The NMC should: 1) Collect NSI from all layers and components of the communication network. 2) Interface with users and their applications, which includes: a) Digest application requests. b) Determine feasibility of application requests. c) Instruct the application of operational requirements. 3) Orchestrate reconfiguration of network resources.

More specifically, when an application requires network resources, it will contact the NMC system and specify its requirements, including delay, bandwidth, and security/reliability requirements. It is preferable for the application to specify a few possible variations of its desirable requirements, each corresponding to a different

QoS and/or QoE level. The NMC system will in turn evaluate the feasibility of the requests, and respond by specifying the resources that should be used to achieve the highest possible QoS and/or QoE levels. If the network, in its current state, is unable to satisfy the application's demand, the NMC system would either reconfigure the network to meet the requirements or reject the request.

Beyond the ability to reconfigure network resources, the NMC system should be able to force the network to drop non-essential traffic and at times partition the network to stop cascading effects such as malware propagation. As a result, the NMC system may have a very dynamic behavior, for example during nominal operation everyone is treated in the same way and everyone reports the same set of state information, upon receiving a critical message, some nodes may need to back off while others may be instructed to report additional information. Agaskar has investigated various methods for incorporating such capabilities into "smart city" applications with critical time deadlines [13].

Given that the architectural specifications are so easily stated, we may wonder why the development of such a comprehensive NMC system has not already occurred. There are many reasons for the lack of such a system, but we can point to three fundamental issues that have greatly contributed to the fragmented state of current partial solutions:

1. Lack of a homogeneous network substrate with well-defined responses to network events.

2. Lack of a universally acceptable theory for the design, analysis, and operation of networks.

3. Lack of data regarding consumer/application preferences to guide the development of such a theory.

The first issue points to the reality that what we abstractly refer to as a network is an amalgamation of many varied technologies with extremely different properties.

As an example, consider a simple search query on your cell phone. The first leg of this communication path is often a wireless medium (Wi-Fi or cellular), after which the message is transported on copper wires and/or optical links to the nearest data-center. Once the search results are obtained, the resulting content will retrace its path back to our phones. Note that the wireless medium behaves very differently from the copper wire and/or the optical links. Any viable NMC system will have to identify the correct level of abstraction for each component to avoid excessive complexity.

The second and third issues are somewhat intertwined. To see why, note that each of the newly created applications has a somewhat unique operational paradigm and thus places a different amount of importance on various network characteristics and qualities. This in turn makes it difficult to define a universally acceptable objective for the design, analysis, and operation of networks. This has resulted in the generally accepted view amongst network architects that states "build a better network and the killer app will be made for it".

Despite the persistence of these difficulties over many decades, we believe that the proliferation of two new trends in the networking world can enable us to address some of these challenges:

1. The wide adoption of Software Defined Networking (SDN).

2. The development of cheap and effective machine-learning tools.

The emergence and prevalence of SDNs in commercial networks allows us to control various network elements in a centralized fashion. This in turn gives network architects the ability to design various network protocols with a certain level of flexibility which would not be possible otherwise. Interestingly, the difficulties associated with proper configuration of SDNs is also due to their flexibility. Typical SDNs have hundreds of parameters which are coupled in rather non-intuitive ways. Fortunately, machine-learning tools are the perfect solution for such situations. Today's deep-learning models can keep track of hundreds of thousands of different parameters with reasonably small computational overload, and are able to adapt to various network conditions with very limited human intervention.

The combination of these two techniques will enable us to engage in logically centralized monitoring and control of various network elements. Given the importance of scalability, we should always strive to find ways to create abstractions of lower layer information that can be easily accessed through a universal interface.

Our goal in this thesis is to express the simplest possible abstraction of a centralized NMC system that seeks to collect and disseminate network state information with the aim of optimizing the network performance. Noting the universality of delay, we have selected it as the abstraction that can be associated with every link across the network. The reason for this universality is the general preference of all users and applications to communicate with one another in near real-time. Of course, modeling delay is itself a difficult task as we will discuss in the following chapters, but it is so central to the proper operation of the network that we shall bite the bullet and deal with its inherent complexity. As a result, we will focus on optimal collection of delay information as it relates to shortest path routing within networks. The following section discusses some of the latest approaches in identifying the value of information in shortest path routing.

1.1 Related Work

Over the years researchers have introduced a whole host of ideas with the hope of quantifying the best practices for collection of network state information. In fact there has been a resurgence of active researchers from industry and academia that recognize the importance and relevance of this topic to the efficient operation of a wide variety of applications ranging from communication systems and networks, to general transportation networks. To avoid a lengthy survey of all prior works, we shall limit our discussion to a few examples to illustrate the general landscape of the area, and various modeling approaches that have been proposed. Each example will be accompanied by a short discussion of its strengths and shortcomings. The examples cover the following categories:

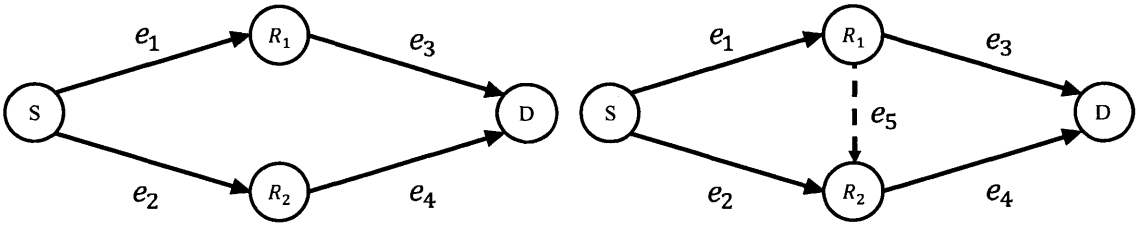
- Analysis of network performance through game theoretic approaches
- Controlling network performance through information measurements
- Fault diagnosis in optical networking
- Linear programming and the geometry of shortest path routing

1.1.1 Example of a Game Theoretic Approach

Game theory has been used in many settings to identify the strategic interaction of "rational" agents, where the assumption of rationality describes the general preference of an agent for an action/state that maximizes their utility.

Game theoretic formulations of the shortest path routing problem are often concerned with agents who are able to choose their desired paths to a given destination based on the information available to them at the time. It is within this context that one can evaluate the value of information with respect to the choices made by agents. This paradigm is often concerned with transportation networks such as interstate highway systems (roads in general) whereby a driver can independently decide on the best way to reach its destination given the available information at the time

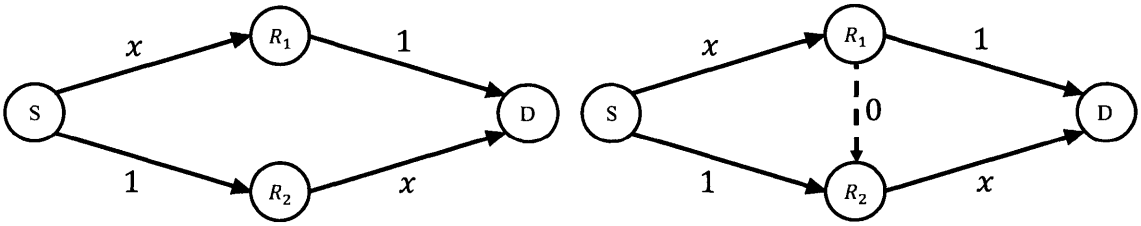
of the decision. Such systems are susceptible to certain inefficiencies which present themselves as paradoxes. We shall reference a well-studied one, known as the Braess' Paradox, which describes the counter-intuitive fact that at times adding a road to a congested network could increase the overall journey time, while removing certain roads could reduce the overall journey time. This paradox is named after the German mathematician Dietrich Braess that originally postulated it. We refer interested readers to [14] for additional information on the original formulation. We shall use a numerical example from [15] to describe the phenomenon.



(a) Original network with delay characteristic: $\{d_1(x), d_2(x), d_3(x), d_4(x)\} = \{x, 1, 1, x\}$.

(b) Addition of a new link to the original network. $d_5(x) = 0$.

Figure 1-3: Example of a simple network that showcases the Braess' Paradox.



(a) Original network with delay characteristic: $\{d_1(x), d_2(x), d_3(x), d_4(x)\} = \{x, 1, 1, x\}$.

(b) Addition of a new link to the original network. $d_5(x) = 0$.

Figure 1-4: Delay characteristics for the networks of Fig. 1-3.

Let us start by studying the networks in Figure 1-3, both of which are supposed to transport 1 unit of flow between the source and the destination. Edges in the network are labeled as e_i and there is a delay of $d_i(x)$ associated with transporting x units of flow on edge e_i. Note that the only difference between Figure 1-3a and Figure 1-3b is the presence of edge e_5 in the latter, but as we will show (and contrary to intuition) this addition can have an adverse effect on delay performance of the system.

Starting with the network in Figure 1-3a, we can see that there are exactly two paths that connect this source-destination pair, denoted by e_1e_3 and e_2e_4. As a result, the total flow may be divided when x units of it traverse e_1e_3 and the remaining $1-x$ units go through e_2e_4. The portion travelling through e_1e_3 will experience a delay of $(x+1)$, and the rest will experience a delay of $(1+(1-x))$ on e_2e_4. Note that if either path takes less time than the other, then a small fraction of rational drivers will take that path and we will not have an equilibrium. Thus an equilibrium state corresponds to the traffic allocation that provide equal delay on both paths, i.e. when $(x+1)=(1+(1-x))$ which occurs when $x=\frac{1}{2}$. This analysis presents a satisfying equilibrium solution as it postulates that the traffic will be split equally between the two paths resulting in equal delay of $\frac{3}{2}$ on both.

We shall now focus on Figure 1-3b which adds a new edge, e_5, to the network. Note that this edge exhibits 0 delay. For simplicity let us assume that the network was in equilibrium just prior to this addition and thus the traffic were equally split between e_1e_3 and e_2e_4. Upon the addition of this path, rational drivers on e_1 will realize that they can reach the destination in a shorter time if they traverse e_5e_4 instead of e_3. At the very beginning, this new path (i.e. $e_1e_5e_4$) exhibits an end-to-end delay of 1 (because $x=1/2$), but the delay will rise as more and more cars are diverted to this new path. Note that the delay on e_5e_4 is in competition with that of e_3 and will rise until they are both equal. As a result, in equilibrium the entire traffic on e_1 will be diverted to e_5e_4. Meanwhile, the traffic at the source would prefer to traverse e_1e_5 instead of e_2 as it has a lower delay, and thus the entire traffic will be diverted to e_1. As a result, in equilibrium the entire traffic will traverse $e_1e_5e_4$, resulting in an end-to-end delay of 2. Somewhat counter to what we may have expected, the introduction of the additional edge increased the delay experienced by users from $\frac{3}{2}$ units to 2. This paradoxical behavior is often referred to as the Baress Paradox.

Acemoglu *et al.* [15] have recently introduced the notion of *Informational Braess' Paradox*. This new model examines similar congestion games whereby users have access to different information sets about the set of available edges and/or the delays associated with them. In particular, they study the implications of availability of

additional information about routes and confirm that providing additional information to users may not reduce the overall journey time. More specifically they show that "[Informational Braess' Paradox] does not occur if and only if the network is *Series of Linearly Independent* (SLI). Where [such a] network is obtained by joining a collection of linearly independent networks in series and linearly independent networks are those in which each route includes at least one edge that is not part of any other route" [15, p. 2]. In other words, whether a network has this property or not is simply a property of the graph structure and is not related to edge weights. Hence, the "if and only if" statement should be interpreted as: if a network satisfies the SLI property, then the Braess' paradox does not occur regardless of how the weights are assigned but if the network does not have this property, then there exits some weight assignment that would exhibit the paradox. Clearly, one can use the impact of various information sets on the optimal operation of the network and then strategize for the collection and dissemination of network state information.

Treatment of transportation networks with game theoretic tools is rather natural and owes its usefulness to the fact that individual drivers make routing decisions in an uncoordinated fashion based on their individual preferences. This is in contrast to the way communication systems handle routing, which has traditionally been done by the system and not by the user. To that end, the Braess' paradox may seem superficial and contrived to a communication systems' engineer but at its heart it tells us that routing protocols should avoid short-sighted approaches that blindly increase traffic on routes which currently exhibit lower delay. Instead, we should focus on approaches that consider the system-wide aggregate delay and design protocols that can deliver a reasonable aggregate delay across all source-destination pairs.

We should note that new transportation paradigms that utilize autonomous cars may eventually exhibit characteristics that are more similar to communication systems than before. Despite their nascent stage of development, various companies are aiming to develop and operate large fleets of autonomous cars whose operations are coordinated by a centralized entity. As these visions take shape it would be crucial to once again focus on system-level approaches that benefit the whole system.

1.1.2 Example of an Information Measurements Approach

As we discussed before, NMC systems are primarily engaged in the collection and dissemination of information with the goal of improving the operational state of the network. As a result, many researchers have borrowed various concepts and abstractions from information theory that can help in identifying the most valuable information as it relates to the optimal operation of a network. In this section we will discuss the work of Michael Rinehart [16] with respect to "the value of information in shortest path [routing]".

We should start by noting that from an information theoretic perspective, information is a quantity that allows us to resolve our uncertainty about the realization of a random variable. In this nomenclature, we often think of a source that generates a sequence of symbols according to some pre-defined distribution. The total amount of information in a given source, X, is defined as its "entropy", and is denoted by $H[X]$. Now suppose that we have two distinct sources, X and Y, whose entropies are denoted by $H[X]$ and $H[Y]$ respectively. We can define the "mutual information" between X and Y as the average reduction in entropy of one source when we know the other. Mutual information is a measure of the mutual dependence of two random variables and is often denoted as $I(X;Y) = I(Y;X)$.

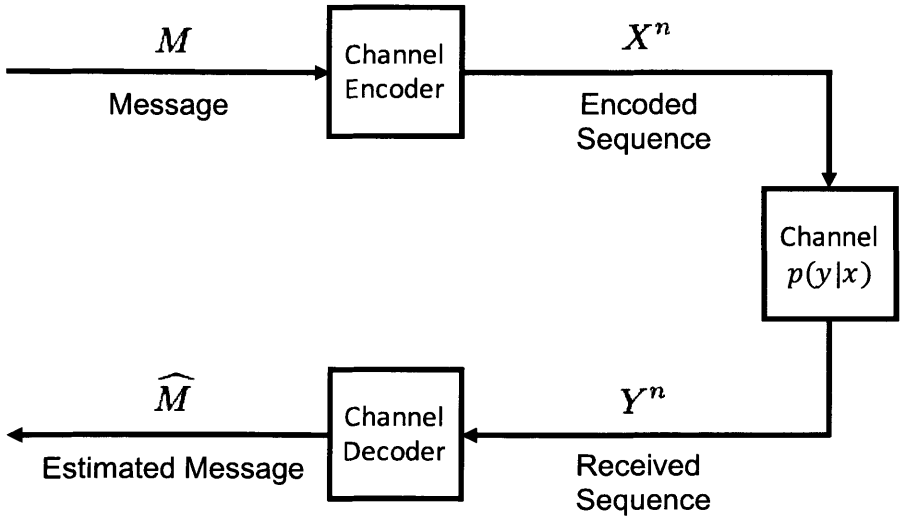


Figure 1-5: Abstraction of a communication system.

In the classical abstraction of a communication channel, depicted in Fig. 1-5, we seek to reliably transmit a message across a noisy channel, where the noisy nature of the channel is captured by the conditional probability distribution $p(y|x)$. It is a well-established fact that the rate of information transmission, and thus the fidelity of the channel is precisely captured by the mutual information between the input of the channel and its output (i.e. $I(X;Y)$). More importantly, in traditional communication systems we are often concerned with the receiver's estimation error which can be derived from the aforementioned mutual information. Furthermore, *capacity* of the channel is defined as the maximal mutual information between X and Y.

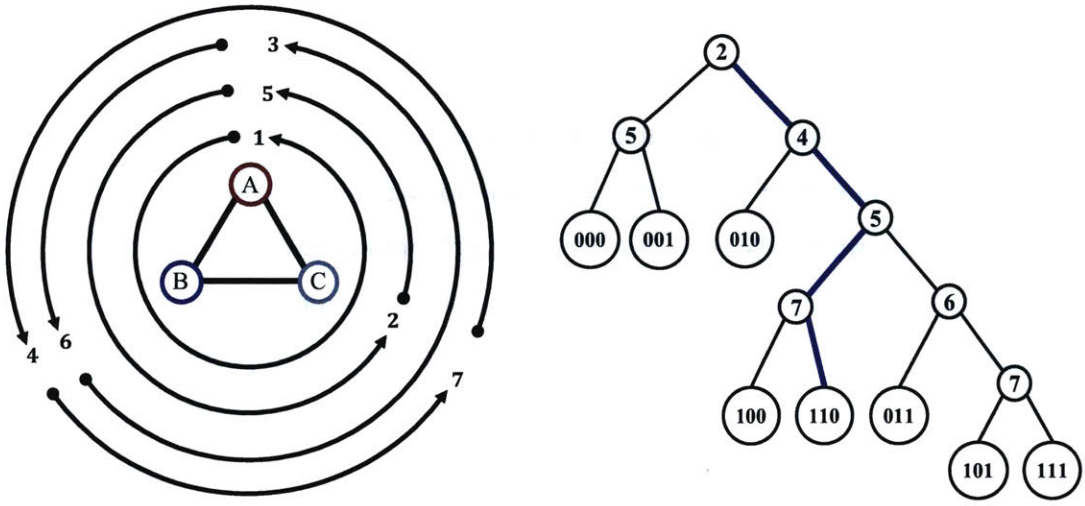
In [16], the author studies the issue of shortest path routing on a graph with random edge weights. Clearly, shortest path routing can be cast as a problem in which agents utilize the available information regarding edge weights to reach their desired destination in the shortest amount of time. The biggest difference between this context and the typical communication channel is that traditional metrics such as estimation error are no longer suitable because agents are not engaged in a decoding task. As the author correctly suggests, the "information quality [and value] is determined by the average length of the paths the agent chooses, not how often the agent decodes the optimal path." Hence, the author proposes an alternative measure of information (instead of mutual information) that is more tractable and easier to compute. Defining the average length of the paths taken by an agent as the performance metric the author derives analytical bounds on the average performance of an agent subject to a bound on the amount of information available to it. The biggest shortcoming of such approaches is that the results are often stated as a set of upper and lower bounds. While these bounds are helpful in charting the space of possibilities, they rarely lead to algorithms that would be operationally significant.

We end this example by noting that cross-pollination of ideas across diverse scientific fields has been the source of many discoveries, but we should also emphasize that information theory (or "Mathematical Theory of Communication" as it was originally called) was formulated by Shannon to shed light on and answer very specific questions, and extending them to other topics should be done with a lot of care.

1.1.3 Fault Diagnosis in Optical Networks

In the spirit of collecting relevant network state information to identify failed components, Yonggang Wen [7] studied a scheme for efficient discovery of link outages in all-optical networks. Within this context, each optical link is constrained to be in one of two states: ON or OFF, and the NMC system is expected to identify the state of every link within the network through a sequence of optical probes. The best way to visualize this scheme is to represent the network as an undirected graph where the state of each edge is represented as an independent and identically distributed Bernoulli random variable (corresponding to the distribution of ON/OFF states). The overall network state as well as the identity of the failed links can then be determined through a sequence of end-to-end probing signals, each of which corresponds to transmitting an optical signal along a particular lightpath. If the probing signal arrives at its intended destination it will indicate that all edges along that lightpath were operational (i.e. ON), otherwise we know that at least one of the edges along that lightpath was impaired (i.e. OFF).

Figure 1-6 is an illustrative example from [7], that shows the operational details of such a probing scheme. In particular, Figure 1-6a, depicts a simple three-node ring topology and enumerates the corresponding set of permissible optical probes. Clearly, identifying the state of the network can be cast as a sequential decision problem where the result of each optical probe will be used to inform the NMC's decision about the the next probe. Figure 1-6b depicts the binary decision tree corresponding to one such probing scheme. Without delving into the theoretical details of the work, we note that the author demonstrates that optimal fault diagnosis in optical networks can be formulated as a source coding problem. More importantly he has shown that minimizing the number of probes required for this diagnosis is closely related to run-length coding schemes. This unique formulation allows us to identify the ON/OFF state of all links in an optical network with the fewest possible number of probes. Hence, if the cost of information collection is proportional to the number of optical probes, then this technique minimizes the corresponding average cost of operation.



(a) Permissible Probes. (b) Probing Scheme.

Figure 1-6: a) The set of permissible probes over a three-node topology. Each probe is indexed with a number near the arrow. b) Probing scheme (decision tree) for the three-node ring topology. The inner nodes of the tree correspond to specific probes and the leaves correspond to the inferred network state. We should note that 0 denotes operational state and 1 denotes a failure in each optical probe, and corresponds to branching to the left or right on the decision tree respectively.
Source: Reproduced from [7, p. 67].

The biggest shortcoming of this approach comes from its binary assumption on the state of each optical link. An ON/OFF model may be suitable for environments that experience drastic impairments such as when a fiber is cut or when an optical component fails completely; but the day-to-day operation of optical networks is often affected by more subtle and gradual changes to the environment. These changes may result from a physical phenomenon such as temperature change, or a deviation in SNR, to user induced changes such as a surge in traffic. To avoid such pitfalls we will try to model the network in such a way as to allow for a rich set of possible states.

In addition, we should point out that optical networks are often designed so that light paths are in use virtually at all times, and thus the routine transmissions across the links should provide sufficient information to upper layers of the network regarding the ON/OFF state of individual links.

1.2 Geometry of stochastic shortest path problem

To get a geometric understanding of the shortest path routing problem we shall focus on its linear programming (LP) formulation. The formulation is standard and can be found in many introductory textbooks on linear programming. We shall consider a directed graph denoted by (V, E), where V denotes a set of vertices and E is a set of directed edges between vertices. In addition to the vertices and edges, we associate a cost[1] $c_{ij} \geq 0$ with edge $(i, j) \in E$. We are often concerned with identifying the shortest path that connects an origin node O, to a destination node D. This problem gives rise to the following LP formulation:

$$\begin{aligned} \underset{x_{ij}}{\text{minimize}} \quad & \sum_{ij \in E} c_{ij} x_{ij} \\ \text{subject to} \quad & \mathbf{x} \geq 0 \\ & \sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} -1, & \text{if } i = O \\ 1, & \text{if } i = D \\ 0, & \text{else} \end{cases} \end{aligned}$$

where $\sum_j x_{ij} - \sum_j x_{ji}$ is simply the total flow at node i, and thus the last set of constraints denote the conservation of flow at every node along the way except for the origin and destination nodes. Note that x_{ij} acts as an indicator random variable, it is 1 if edge (i, j) is part of the shortest path and 0 otherwise. In other words, the optimal solution, $\mathbf{x}^*$ to the previous LP is simply a binary string of length $|E|$.

We are ultimately interested in identifying the optimal solution to the aforementioned problem when we are faced with a set of stochastically varying c_{ij}'s. It should be clear that when c_{ij} is varying with time (*i.e.* when we have $c_{ij}(t)$), we have to continually re-solve the problem in order to identify the optimal path $\mathbf{x}^*(t)$ at various time instances. Note that as the system evolves, the length of the shortest path may get longer or shorter. But in either case, if we use the old solution $\mathbf{x}^*(t')$ instead of the latest solution $\mathbf{x}^*(t)$ we will be incurring an excess cost of $\left(\mathbf{x}^*(t')\mathbf{c}^{\mathbf{T}}(t) - \mathbf{x}^*(t)\mathbf{c}^{\mathbf{T}}(t)\right) = \left(\mathbf{x}^*(t') - \mathbf{x}^*(t)\right)\mathbf{c}^{\mathbf{T}}(t)$.

[1] In the context of shortest path routing, edge costs are constrained to non-negative numbers to avoid all issues associated with circular flows with negative costs. On the other hand, we will present a few examples of a general LP formulation that include negative costs as it allows us to easily demonstrate some of the relevant graphical aspects.

In order to keep this excess cost to a minimum, we shall expend our monitoring resources so that we can identify and adapt to the changing edge weights. Before delving into additional details, let us study how the geometry of a non-stochastic linear program is related to the cost vector. The following example, inspired by example 1.8 of [17] showcases the relationship between the cost vector $\mathbf{c}$ and the solution to the LP.

Example 1. Let us consider the following linear program:

$$\begin{aligned} \underset{x_1, x_2}{\text{minimize}} \quad & c_1 x_1 + c_2 x_2 \\ \text{subject to} \quad & x_1 - x_2 \leq 1, \\ & x_2 \leq 2, \\ & x_1, x_2 \geq 0 \end{aligned}$$

The feasible region of an LP is defined as the set of points that satisfy every constraints in the LP formulation, and thus the optimal solution to the LP (if it exists) has to be in this region. Noting that the objective function is only a function of two variables, (x_1, x_2), we can easily visualize the feasible region for this optimization on the 2D plane. Figure 1-7, depicts this feasible region which is at the intersection of 4 half-planes, namely: $x_1 \geq 0$, $x_2 \geq 0$, $x_2 \leq 2$, and $x_1 - x_2 \leq 1$. Various cost vectors $\mathbf{c} = (c_1, c_2)$ are also drawn in the figure.

Graphically, the solution of an LP problem can be identified by drawing a plane orthogonal to the cost vector and moving it in the direction of -$\mathbf{c}$ until the intersection of the orthogonal plane and that of the feasible region has reached the farthest point on the boundary of the feasible region[2]. For example, for $\mathbf{c} = (1, 1)$, and $\mathbf{c} = (-1, 2)$ the optimal solution would be at $\mathbf{x}^* = (0, 0)$, and $\mathbf{x}^* = (1, 0)$ respectively. Meanwhile for $\mathbf{c} = (1, 0)$ every point on the vertical axis that is also in the feasible region is an optimal solution, i.e. $\mathbf{x}^* = (0, x_2) \ \ \forall x_2 \in [0, 2]$. The question at hand is what

[2]For this example we have focused on linear programs where a feasible region exists and is bounded. It can be shown that in such instances the optimal solution is always on the boundary of the feasible region. If the feasible region is unbounded we may not have an optimal solution.

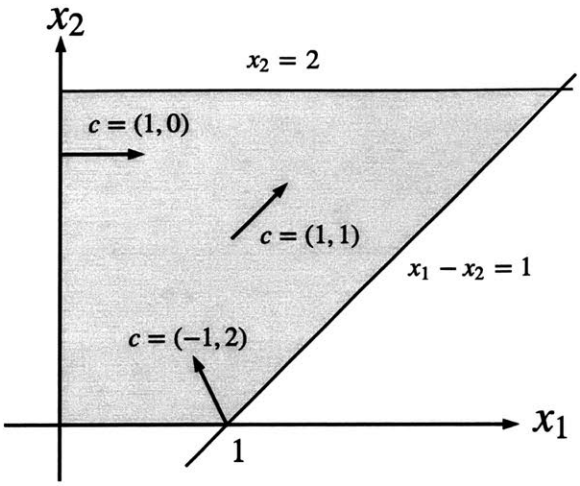


Figure 1-7: Feasible region depicted in gray which is bounded by 4 linear constraints.

happens when the cost vector changes by a small amount. Intuitively, we would expect small changes in $\mathbf{c}$ to **not** result in a change to the optimal solution. Posing the question differently, we can ask how much can the cost vector change before the optimal solution changes. One way to answer this question is to specify a range R_i for the i^{th} dimension of the cost vector c_i, so that if we **only** change one coefficient of the cost vector (while keeping others fixed) the solution would **remain valid** (there may be other solutions). For instance in our previous example, for $\mathbf{c} = (1, 1)$ we have $R_1 = [0, \infty)$ and $R_2 = [0, \infty)$; while for $\mathbf{c} = (-1, 2)$ we have $R_1 = [-2, 0]$ and $R_2 = [1, \infty)$. These ranges can be thought of as valid cost intervals for a specific solution $\mathbf{x}^*$. Hence, for an n-dimensional cost vector we can specify a matrix R as a function of $\mathbf{c}$ whose column are the respective ranges:

$$R(\mathbf{c}) = \begin{bmatrix} | & | & & | \\ R_1 & R_2 & \dots & R_n \\ | & | & & | \end{bmatrix}$$

The previous discussion may seem rather fruitful and promising but it conceals a major shortcoming. Note that the aforementioned analysis is only valid when we modify a single dimension of the cost vector while keeping others fixed. This is in

contrast to the situation in real networks, where all links undergo various changes simultaneously, which renders the applicability of the previous discussion moot. To the best of our knowledge there is no mathematically precise theory that can express the effects of simultaneous changes to various dimensions of the cost vector on the optimality of a given solution, and thus it remains an area of active research.

Before ending our discussion on the geometry of shortest path routing we would like to review some of the well understood aspects of optimizing a linear objective function with variable coefficients, a topic that has been studied under the banner of "robust optimization". The reason for this extended discussion is that we believe future advances in the theory and practice of robust optimization and linear programming will prove immensely useful for the design and analysis of communication systems and readers of this document may wish to familiarize themselves with the basic concepts of this area. To this end, the most notable insights with respect to optimizing a linear objective function with variable coefficients, is rooted in differentiating between *basic* and *non-basic* variables, which are defined as [17]:

Definition 1. Let us denote the optimal solution of a linear program as an n-dimensional vector $\mathbf{x}^*$. The i^{th} component of the vector, denoted by x_i^* is said to be a basic variable if it is **not** equal to zero, and is a non-basic variable otherwise.

As an example, let us consider the optimization problem depicted in Fig. 1-7. For $\mathbf{c} = (1, 1)$ both components of the optimal solution x_1^* and x_2^* are zero and thus non-basic, while for $\mathbf{c} = (-1, 2)$, $x_1^* = 1$ and is basic and $x_2^* = 0$ and thus non-basic.

Note that the value of the objective function (cost function) is minimized at the computed optimal solution $\mathbf{x}^*$, furthermore, and this value is simply the sum of the product of basic variables and their corresponding costs. For instance in the scenario depicted in Fig. 1-7, when $\mathbf{c} = (-1, 2)$, the objective function will evaluate to $c_1 \cdot x_1^* = -1 \cdot 1 = -1$. It should now be clear that non-basic variables of the optimal solution do not contribute to the cost/objective function while basic variables do, and thus changing the coefficients (i.e. the corresponding cost) of basic and non-basic variables will have radically different impacts on the optimal solutions to a linear program. The following subsection will elaborate on this topic.

Changing the coefficients of basic vs. non-basic variables

Given our previous discussion about the value of the objective function at the optimal solution, it is not surprising that increasing the coefficients of non-basic variables will not affect the optimal solution. Intuitively, if c_1 is the coefficient of a non-basic variable x_1, such that the initial minimization resulted in choosing $x_1^* = 0$, then we can see that increasing the cost of that "commodity" (increasing c_1) will not entice us to increase our demand as represented by x_1^*. On the other hand, decreasing c_1 will at some point cause us to increase x_1.

Varying the coefficients of basic variables creates a more complex situation because their corresponding solution are non-zero. Let us consider $\mathbf{c} = (1, -1)$, for our initial optimization and note that the optimal solution is at $\mathbf{x}^* = (0, 2)$; which makes x_2 into a basic variable. First and foremost, notice that decreasing or increasing c_2 will change the value of the objective function regardless of whether or not it impacts the optimal solution. Furthermore, as c_2 decreases, we will get additional reduction in the objective function even with the old solution x_2^*, but if c_2 decreases enough we may even choose to increase x_2 to take additional advantage of it. Note that in general the optimization may be subject to additional constraints such as $-x_1 + x_2 \leq 1$, and thus any change to x_2 may influence other dimensions as well. A similar phenomenon occurs if c_2 increases, whereby after a certain threshold, the increase in c_2 will cause us to reduce x_2. The following subsection explains the role of basic and non-basic variables in the context of shortest path routing.

Basic vs. non-basic variables in LP formulation of shortest path routing

The optimal solution, $\mathbf{x}^*$, of an LP formulation of shortest path routing is a binary sequence where a 1 in the i^{th} position indicates the presence of that edge on the shortest path, and a 0 indicates the absence of the aforementioned edge in the shortest path. Clearly, basic variables correspond to edges of the shortest path, and non-basic variables are those not included in the shortest path. Naturally, increasing the weight of non-basic variables will not affect the solution $\mathbf{x}^*$, which intuitively says that if the price of traversing an expensive road is increased, we will continue **not** to use it.

Changing the cost of basic variables is more complex but not as complex as a general LP. The simplification comes from the fact that $\mathbf{x}^*$ is a binary sequence and thus the solution space is constrained. Thus, even though reducing the cost of a basic variable may entice us to increase the corresponding basic variable, we have to stick with a value of 1. Thus, in the LP formulation of the shortest path, reducing the cost of basic variables will not impact the optimal solution; even though it impacts the value of the objective function. Intuitively, the identity of the shortest path does not change if we reduce the cost of its edges. Now, the question becomes how much could we increase the cost of a basic variable before $\mathbf{x}^*$ changes. Intuitively, the cost of this edge can increase, until the total cost of this path reaches the cost of the 2^{nd} shortest path connecting the origin-destination pair. Let us use $c]^n_{A\to B}$ to denote the cost of the n^{th} shortest path connecting node A to node B. Then, for a basic variable i that represents the edge connecting node A to node B, we have $R_i = \left[0, c]^2_{O\to D} - c]^1_{O\to A} - c]^1_{B\to D}\right]$, where O and D are origin and destination nodes.

What about the lower-bound on the range of the cost of non-basic variables? Note that reducing the cost of a non-basic variable impacts the identity of the shortest path, if and only if the new shortest path includes that non-basic variable. In other words, a new shortest path will materialize if and only if it includes the formerly non-basic edge whose cost has been reduced. Since by definition, the cost of the new shortest path has to be lower than the old shortest path, the lower bound on the cost of a non-basic edge between A and B is equal to $c]^1_{O\to D} - c]^1_{O\to A} - c]^1_{B\to D}$. Thus,

$$R_i = \begin{cases} \left[0, c]^2_{O\to D} - c]^1_{O\to A} - c]^1_{B\to D}\right] & \text{if } i \text{ is a basic variable connecting A to B} \\ \left[c]^1_{O\to D} - c]^1_{O\to A} - c]^1_{B\to D}, \infty\right) & \text{if } i \text{ is a non-basic variable connecting A to B} \end{cases}$$

This concludes our discussion of the geometry of the stochastic shortest path problem. We should note that despite the interesting insights of our previous discussion, the LP formulation does not provide a foundation for the sensitivity analysis required to handle simultaneous changes in the cost of various basic and non-basic variables.

1.3 Problem Statement

In this thesis we will focus on the problem of shortest path routing, and identify pragmatic schemes that allow a central controller to collect delay statistics from various links and nodes in an optimal fashion. The controller should be smart enough to request frequent updates when the information can be helpful in the operation of the network and reduce it when they are of less use. This approach is in contrast to how delay statistics are disseminated in current networks, which occurs at regular intervals. We should note that monitoring delay statistics within a network necessitates an underlying algorithm to determine network connectivity, and we shall assume that such underlying processes and algorithms are already in place.

To this end, the thesis will include an analytical framework that can identify the optimal time for the collection of such information in a simple two path network. We will then extend the results by introducing a reinforcement learning framework that learns the aforementioned optimal policy from its own interactions with the network.

Of course, properly designed networks should be able to operate in a reasonable way between updating times and thus any meaningful solution should strive to meet application demands despite the unavoidable uncertainty about the instantaneous state of the network. Hence, we will complement our work by introducing a diversity routing scheme that can cope with stringent requirements on delay variation despite the controller's relative uncertainty about the instantaneous state of the network.

1.4 Thesis Organization

This thesis is organized as a series of self-contained chapters. The reason for this structure is that each chapter pursues a unique objective and interested readers can focus on a specific topic without the need for excessive reference to previous chapters. With that in mind, Chapter 2 discusses the issue of shortest path routing in uncertain environments and introduces the notion of Significant Sampling as a way to optimize the amount of information collected from the network. Chapter 3 provides a deep reinforcement learning solution to the shortest path routing problem, and presents a model-free framework for Significant Sampling. Finally, Chapter 4 models the uncertainty in the state of the network and presents a unique diversity routing approach that achieves favorable tradeoffs between the expected delay and instantaneous delay averaged across paths.

Chapter 2

Significant Sampling for Shortest Path Routing in Uncertain Network Environments

2.1 Introduction

As we discussed in Chapter 1, there are a myriad of interactive, data-hungry, and demanding applications that are currently under development which will collectively induce new traffic patterns in data networks that are no longer static, or even quasi-static. One may expect the input traffic to any given link to be continuously varying and experience abrupt changes. As a result, any befitting Network Management and Control (NMC) system has to closely monitor the state of all links to be able to provide the desired quality of service to all users. Unfortunately, current NMC systems are unable to handle the enormous amount of network state information that has to be collected, analyzed and distributed to properly manage the network.

As an example, we can reference a case study by Zhang *et al.* [18], that estimated the average control traffic required for optimal scheduling of flows across a 60-node optical-WAN representing the backbone of the continental US. The average total control traffic is estimated to be as high as 300 Gbps. The collection of this much data is itself burdensome, but more importantly the computational resources required to analyze it would be impractical. With that in mind, we will introduce a cognitive approach that considers the cost associated with collection and dissemination of state information while it seeks to improve the network performance.

To illustrate the efficacy of our cognitive management approach, we will use an example that primarily focuses on challenges involved with 'shortest path' routing in dynamic networks. In this context, an NMC system has to monitor the state of queues throughout the network to decide which path(s) should be used to connect different origin-destination (OD) pairs. In practice, various shortest path algorithms (Bellman-Ford, Dijkstra, etc.) are used to identify the optimal shortest path. The correctness of these algorithms requires the assumption that after some period of time the system will settle into its steady state. We refer interested readers to [19] for a thorough explanation of these issues. Given the dynamic nature of future networks, the steady state assumption is particularly not valid, nonetheless the basic functional unit of these algorithms can be adapted to address the new challenges.

Shortest path routing algorithms assign a length to each link of the network; this length is often a proxy for traffic congestion on the link but can also incorporate other factors. Depending on the specific implementation, the length may reflect the number of packets waiting in the queue, loading of the output line, or the average packet delay on the link during a pre-specified amount of time. Neighboring nodes exchange their estimated shortest distances to all other nodes periodically and new information will be disseminated throughout the network after a few rounds of exchanges.

Suppose the routing table for each node contains the available paths to each destination and the latest estimates of their lengths, as shown in Table 2.1. Given this table, a node can identify the shortest path to all destinations. Furthermore, it can use the relative difference between the length of the shortest path and that of other paths to determine the likelihood that the shortest path changes in the near future. Hence, the frequency by which we update entries of the routing table can effectively be determined from this likelihood as well as other pairwise considerations.

Table 2.1: Partial routing table maintained by node A

Destination	Path	Length
Y	$B \rightarrow C \rightarrow D \rightarrow Y$	2
	$E \rightarrow F \rightarrow G \rightarrow Y$	3
	$H \rightarrow I \rightarrow Y$	8
Z	$J \rightarrow K \rightarrow Z$	4
	$L \rightarrow M \rightarrow Z$	7
	$N \rightarrow O \rightarrow P \rightarrow Z$	9

Thus, as opposed to updating the whole table at a minimum rate required for all dynamic situations, each entry of the routing table will be updated only when it can be of significant value to the optimal operation of the network. It is in this sense that we refer to our technique as ***significant sampling***. To illustrate this principle, we will focus primarily on a simplified model where an OD pair is connected via exactly two independent paths with time varying lengths. Within this framework, we develop an adaptive monitoring system that determines the value of updating the length of a given path as well as the cost associated with the updating process. This allows the NMC to optimally allocate its resources to collect and disseminate information regarding the state of network elements according to their importance.

The remainder of this chapter is organized as follows: Section 2.2 introduces the general stochastic model and establishes the framework through which the monitoring process is optimized. Section 2.3.1 assumes a Wiener process as the end-to-end delay process and evaluates the benefits and shortcomings of this model. Section 2.3.2 applies the framework to the Ornstein-Uhlenbeck process which circumvents a major shortcoming in the Wiener process. Section 2.4 extends the results to general networks. A summary and concluding remarks are provided in Section 2.5.

2.2 Problem Setup – General Model

Suppose that an OD pair is connected via two independent paths P_1 and P_2 as illustrated in Figure 2-1, and denote the stochastically evolving weight of P_i by $X_i(t)$. [1,2]

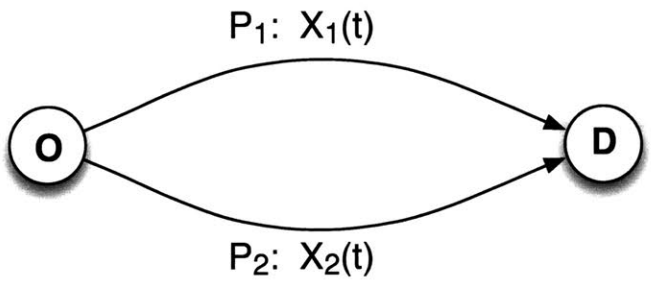


Figure 2-1: An OD pair with two independent paths.

Let us use $X(t)$ to denote the stochastic process that results from subtracting the weights of the two paths, i.e., $X(t) = X_2(t) - X_1(t)$. Clearly, continuous optimal routing can be achieved if we know whether or not $X_1(t) < X_2(t)$. This is equivalent to knowing the sign of $X(t)$ for all t. Since continuous monitoring of $X(t)$ is far from cost effective, our aim is to identify a strategy that specifies the best future updating times based on the last observed value of $X(t)$.

More concretely, consider a sample function of $X(t)$ as shown in Figure 2-2. Given $X(t_0)$, (i.e. the state of both elements at time t_0), we would like to identify the next epoch t_1 for updating our routing tables. When the value of the function at t_1, i.e. $X(t_1)$, is realized, we will use it to determine the following updating time t_2 and so forth. Notice that T_i's and $X(t_i)$'s are the fundamental random variables and can be defined in a recursive fashion. In general, the next updating epoch can be computed as $t_i = t_{i-1} + T_i$ where the update interval T_i is chosen as some function, to be determined, of the state $X(t_{i-1})$ at the beginning of the interval.

[1] $X_i(t)$ can be the rate of transmission (messages/s) times the expected delay/message on P_i, which gives it units of delay/s.

[2] It is a tacit assumption of our treatment in this chapter that a given link is carrying many sessions from a potentially large number of origin-destination pairs, and that the rerouting decisions regarding a specific session will not have a significant impact on the delay/congestion of the link.

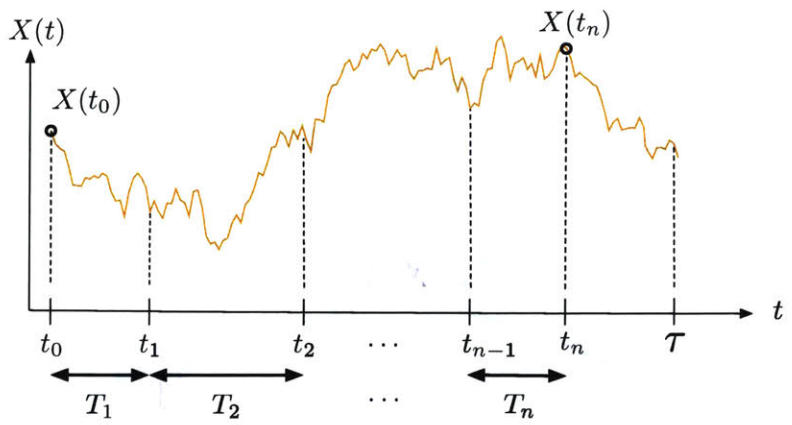


Figure 2-2: Illustration of n samples taken during a period τ.

Between two updating epochs t_{i-1} and t_i, the process $X(t)$ will evolve according to an underlying stochastic model. Recall that P_1 is the optimal route if $X(t) > 0$ and P_2 is the optimal route if $X(t) < 0$. So the communicating OD pair will experience an excess cost if the process $X(t)$ changes sign between two sampling epochs and the transmission route is not adapted. Let us use $C_{[t_{i-1},t_i]}$ to denote the cost of such errors during $[t_{i-1}, t_i]$. Without loss of generality, assume $X(t_{i-1}) > 0$ and suppose that the OD pair uses P_1 as the route during $[t_{i-1}, t_i]$. Then the cost of error during this period is simply the integral of $X(t)$ after the process experienced a sign change. In other words,

$$C_{[t_{i-1},t_i]} = -\int_{t_{i-1}}^{t_i} X^-(t)\, dx$$

where $X^-(t)$ is the negative part of the function defined as $X^-(t) = (X(t) - |X(t)|)/2$. Figure 2-3 illustrates this process through a specific example. Note that $X(t_{i-1}) > 0$, indicating that at t_{i-1}, P_1 is the shortest path. Furthermore, note that $X(t)$ becomes negative shortly after t_{i-1} at which point P_1 is no longer the shortest path. If P_1 is used during $[t_{i-1}, t_i]$, we will accrue an excess routing cost of $C_{[t_{i-1},t_i]}$ which corresponds to the red area depicted in the figure. When the stochastic nature of the underlying process is known, a distribution is induced over possible sample paths of $X(t)$, and we can use this distribution to compute the expected cost of error during this period, denoted by $\mathbb{E}\left[C_{[t_{i-1},t_i]}\right]$.

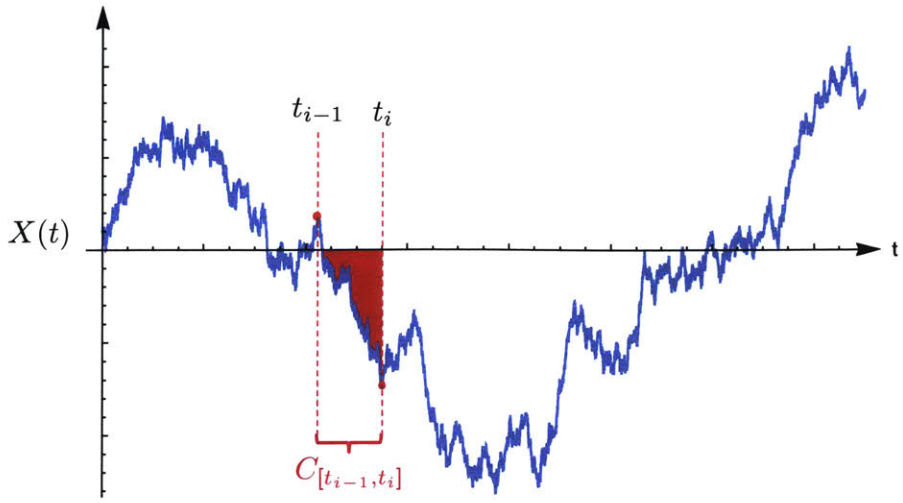


Figure 2-3: Visual depiction of $C_{[t_{i-1},t_i]}$, between two consecutive sampling epochs.

We shall now introduce the notion of *updating cost* to capture the efforts required to collect and disseminate state information throughout the network so that all routing tables are up-to-date. Google Maps offers a great example of various costs associated with such efforts, as the system gathers congestion information from individual drivers on the road. The GPS-enabled device will incur a data transmission cost by using the wireless services, and will drain its battery as a result of the required computation and communication. Furthermore, there is a cost associated with reporting the latest congestion information to all other drivers. In general, the updating cost may vary over time and can differ for various network elements, but for simplicity we will use a single figure of merit, η, to account for the cost. This is a *catchall* quantity that should represent the costs associated with collection and dissemination of the routing information across the whole network.

If we update the routing tables by sampling the process n times during $[0, \tau]$, as shown in Figure 2-2, then the expected total cost C_{Total} resulting from sampling and unintended errors is

$$\mathbb{E}\Big[C_{Total}\Big] \;=\; \sum_{i=1}^{n} \left(\eta + \mathbb{E}\left[C_{[t_{i-1},t_i]}\right]\right) + \mathbb{E}\left[C_{[t_n,\tau]}\right]$$

For a given time horizon $[0, \tau]$, an optimal *offline* sampling strategy will identify the optimal number of samples and their corresponding epochs to minimize the expected total cost. Alternatively, we may choose to minimize the cost rate (cost per unit time) for an infinite time horizon. These two approaches can be summarized as

$$\underset{n,\{t_1<\cdots<t_n\leq\tau\}}{\operatorname{argmin}} \mathbb{E}\Big[C_{Total}\Big]$$

$$\text{or} \qquad \underset{n,\{t_1<\cdots<t_n\leq\tau\}}{\operatorname{argmin}} \lim_{\tau\to\infty} \frac{\mathbb{E}\Big[C_{Total}\Big]}{\tau}$$

The aforementioned formulations are not as useful in practice because routing and updating decisions should be made in real-time. In other words, an NMC is interested in making routing and sampling decisions for the immediate future and cannot afford to plan too far into the future. Let us use $C_{[t_{i-1},t_i]}(x)$ to denote the cost of error during $[t_{i-1}, t_i]$ given that $X(t_{i-1}) = x$. Then the most informative formulation computes the optimal updating period T_i^* as

$$T_i^*(x) = \underset{T_i>0}{\operatorname{argmin}} \frac{\eta + \mathbb{E}\left[C_{[t_{i-1},t_i]}(x)\right]}{T_i}$$

If $X(t)$ is Markovian, then the distribution of its trajectory is only a function of its last observed value. Hence, for Markov processes $\mathbb{E}\big[C_{[t_i,t_i+\tau]}(x)\big] = \mathbb{E}\big[C_{[0,\tau]}(x)\big]$. For such processes we can treat every sampled value as if it had occurred at time zero. Using $C_T(x)$ as a shorthand for $C_{[0,T]}(x)$, we can rewrite our optimization as

$$T_1^*(x) \;=\; \underset{T>0}{\operatorname{argmin}} \frac{\eta + \mathbb{E}\big[C_T(x)\big]}{T} \tag{2.1}$$

This concludes our discussion of the general setting of the problem and the relevant optimization formulations. The following section focuses on appropriate models for transient behavior of congestion and/or delay.

2.3 Delay Models

Queues constitute one of the basic building blocks of a communication network and have been extensively studied for decades [20, 21]. While queuing theory has provided tremendous insight into the operation of data networks, it has struggled in providing tractable expressions that deal with transient behavior of queues (which is of immense importance to us!). For example, let us consider an $M/M/1$ queue with arrival rate λ, and service rate μ, whose state transition diagram as a Markov process is shown in Figure 2-4.

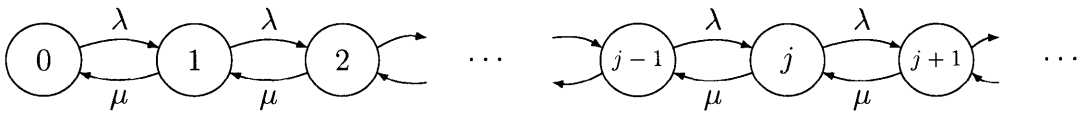


Figure 2-4: State transition diagram of the birth-death process of an $M/M/1$ queue.

Assuming that the queue is in state i at time 0, then the probability that it will be in state k at time t is given by

$$P_k(t) = \mathrm{e}^{-(\lambda+\mu)t}\left[\rho^{(k-i)/2} I_{k-i}(at) + \rho^{(k-i-1)/2} I_{k+i+1}(at) + (1-\rho)\rho^k \sum_{j=k+i+2}^{\infty} \rho^{-j/2} I_j(at)\right]$$

were $\rho = \lambda/\mu, a = 2\sqrt{\lambda\mu}$, and I_k is the modified Bessel function of the first kind defined as,

$$I_k(x) \triangleq \sum_{m=0}^{\infty} \frac{(x/2)^{k+2m}}{(k+m)!m!} \qquad k \geq -1$$

Referring to this expression, Kleinrock notes: "This last expression is most disheartening. What it has to say is that an appropriate model for the simplest interesting queueing system [$M/M/1$ queue] leads to an ugly expression for the time-dependent behavior of its state probabilities" [20, p. 78].

Hence, we advocate an approach whereby operational insights can be provided through judiciously chosen alternative models which lend themselves to easier analysis and can be insightful as engineering guidelines for the design of future networks.

Desirable transient models of delay should possess the following characteristics:

- Stability - the stochastic model of delay should be *stable* so that linear combination of two or more such processes will remain in the same family of distributions.
- Stochasticity - capture the rise in uncertainty about state of the network as more time passes from last observation.
- Simplicity - provide a formulation that is amenable to analysis and/or numerical computation, especially as we scale the network.

In selecting a model with these attributes, it is often beneficial to approximate the delay process with an appropriate diffusion process. The inherent structure of diffusion processes makes it easier to avoid some of the combinatorial challenges involved in the original problem. The following two sections describe two such models of delay. Section 2.3.1 uses a Wiener process as the underlying model for delay. As we will see shortly, the Wiener process which is the limit of M/M/1 queues when the number of arrivals and departures is very large, has one fewer variable in its description which will reduce some of the complexity that was present in the M/M/1 model and allows us to get analytical solutions that are easier to interpret. Section 2.3.2 considers the Ornstein-Uhlenbeck process as the underlying model of delay, and addresses a shortcoming of the Wiener process with respect to its stationarity.

Before getting into the details of each delay model, we should once again emphasize that these models are only appropriate when (re)routing decisions do not have a significant impact on the delay and congestion of individual links; which often occurs when the rerouted traffic is a minor part of the total traffic on a link. Otherwise, we would have to take into account the increase/decrease of delay and congestion that results from specific routing decisions.

2.3.1 Wiener Process

Numerous successful attempts have been made at modeling queuing delay as a Wiener process. Most notably, it has been shown that the normalized queue length in heavy traffic (i.e., as $\rho \to 1$) can be approximated by a one-dimensional reflected Wiener process, also known as the reflected Brownian motion [21, 22, 23]. Modeling the waiting time of customers as a Wiener process satisfies our modeling criteria because 1) Wiener process is stable, hence the waiting time of customers in cascaded series of independent links would itself be a Wiener process, 2) uncertainty in the realization of a sample path of a Wiener process grows with time. We shall define and note a few properties of the Wiener process and refer interested readers to [24] for a thorough investigation of general properties of the Wiener process.

Definition 1. A real valued stochastic process $\{W(t) : t \geq 0\}$ is a Wiener process with a start at $x \in \mathbb{R}$ if the following hold:

- $W(0) = x$
- the process has independent increments.
- for all $t \geq 0$ and $h > 0$, the increments $W(t+h) - W(t)$ are normally distributed with zero mean and variance h.
- the function $W(t)$ is continuous almost surely.

Lemma 1. A Wiener process $\{W(t) : t \geq 0\}$ with a start at 0, has following the pdf and cdf functions

$$f_{W(t)}(x) = \frac{1}{\sqrt{2\pi t}} \exp\left(\frac{-x^2}{2t}\right)$$

$$F_{W(t)}(x) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x}{\sqrt{2t}}\right)\right)$$

where $\operatorname{erf}(\cdot)$ is the error function defined as $\operatorname{erf}(\beta) = \frac{2}{\sqrt{\pi}} \int_0^\beta \mathrm{e}^{-z^2}\, dz$.

Proof. The results are direct consequences of the definition of a Wiener process. □

Recall that we used $X(t)$ to denote the stochastic process that results from subtracting the weights of the two paths, i.e. $X(t) = X_2(t) - X_1(t)$. If each $X_i(t)$ is approximated as an independent Wiener process, we can see that $X(t)$ is another Wiener process with a start at $X(0) = X_2(0) - X_1(0)$ and a variance equal to the sum of the variances of the two processes. Without loss of generality, suppose that $X(t)$ has a variance of $\alpha^2 t$ for some $\alpha \in \mathbb{R}$, and assume that $X(0) = x > 0$. This is equivalent to assuming $X(t) = x + \alpha W(t)$. As before, we will assume that the OD pair uses P_1 as the shortest route until the next update time at t_1. Noting that Wiener process is Markovian and using $C_T(x)$ as a shorthand for $C_{[0,T]}(x)$, we can see that routing through P_1 for T seconds will incur the following cost of error

$$\begin{aligned} C_T(x) &= -\int_0^T X^-(t)\,dx = -\int_0^T \min\{X(t), 0\}\,dt \\ &= -\int_0^T \min\{x + \alpha W(t), 0\}\,dt \end{aligned}$$

Appendix 2.A derives the following closed form expression for the expected value of this quantity,

$$\mathbb{E}\Big[C_T(x)\Big] = \frac{\sqrt{T}\,(2T\alpha^2 + x^2)}{3\alpha\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\alpha^2 T}\right) - \operatorname{erfc}\left(\frac{x}{\alpha\sqrt{2T}}\right)\left[\frac{xT}{2} + \frac{x^3}{6\alpha^2}\right]$$

which can be used with Eq. (2.1), as restated below, to get the optimal sampling period

$$T_1^*(x) = \operatorname*{argmin}_{T>0} \frac{\eta + \mathbb{E}\big[C_T(x)\big]}{T}$$

Interestingly, the shortest possible sampling period can be computed in closed form as well by noting that $\mathbb{E}\big[C_T(x)\big]$ is strictly decreasing in x, and thus for any $T > 0$ the expected cost of error is minimized at $x = 0$. Solving the first order optimality condition gives us

$$\min_x T_1^*(x) = T_1^*(0) = \left(\frac{18\pi\eta^2}{\alpha^2}\right)^{\frac{1}{3}} \tag{2.2}$$

This is an important quantity as it constitutes a maximum updating frequency for all "fixed-period" (*i.e.* uniform) updating strategies. In other words, if the NMC updates occur any faster, the cost of updates will be larger than the potential gains from identifying the optimal route. Furthermore, Eq. (2.2) shows that increasing the updating cost η by a factor of γ reduces the frequency of updates (i.e. $1/T_1^*$) by a factor of $\gamma^{2/3}$; while increasing parameter α has the inverse of that effect.

Noting the complexity of the general optimization, let us use a simple first order method to approximate the optimal updating period T_1^*. Furthermore, to simplify our notation, we will assume that $\alpha = 1$ in the remainder of this section.

$$\mathbb{E}\big[C_T(x)|\alpha=1\big] = \frac{\sqrt{T}\,(2T+x^2)\exp\left(-\frac{x^2}{2T}\right)}{3\sqrt{2\pi}} - \operatorname{erfc}\left(\frac{x}{\sqrt{2T}}\right)\left[\frac{xT}{2}+\frac{x^3}{6}\right]$$

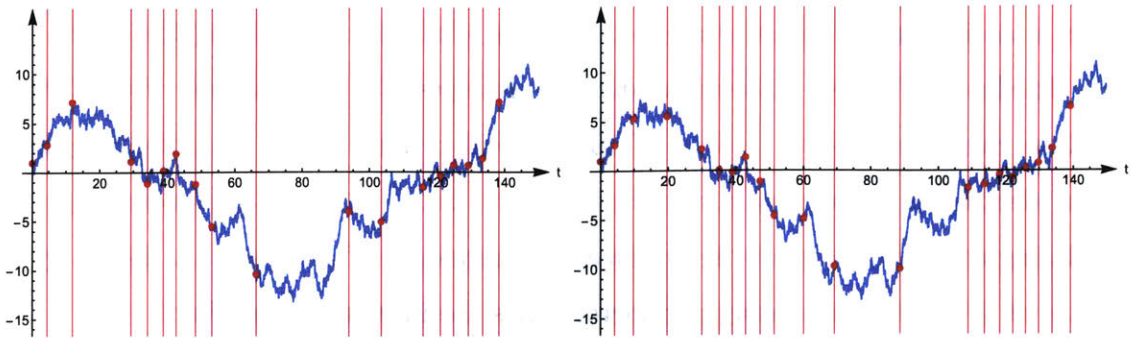
hence,

$$\frac{\partial}{\partial T}\left(\frac{\eta+\mathbb{E}\big[C_T(x)|\alpha=1\big]}{T}\right) = \frac{1}{T^2}\left(-\eta+\frac{\sqrt{T}\,(T-x^2)\,e^{-\frac{x^2}{2T}}}{3\sqrt{2\pi}}+\frac{x^3}{6}\operatorname{erfc}\left(\frac{x}{\sqrt{2T}}\right)\right)=0 \tag{2.3}$$

which can be solved numerically to obtain the optimal T_1^* for any given x. Appendix 2.B shows that solving Eq. (2.3) is approximately equal to solving the following equation for T

$$x^2 = T\ln\left(\frac{T^3}{18\pi\eta^2}\right) \tag{2.4}$$

Unfortunately T cannot be written in terms of elementary functions of x and the above expression is the best (approximate) representation of the relation between the observed value x and the next sampling time T. Figure 2-5 depicts a sample path of $X(t) = 1 + W(t)$ (i.e. $X(0) = 1$ and $\alpha = 1$), and compares the solutions obtained by numerically solving Eq. (2.3) to the approximate solution obtained through Eq.(2.4). Red vertical lines are drawn to specify the updating epochs as computed by each equation for a sampling cost of $\eta = 1$.



(a) Exact numerical solution using Eq. (2.3) (b) Approximate solution using Eq. (2.4)

Figure 2-5: Sample path of a Wiener process, and associated samples for $\eta = 1$.

Notice that both solutions are very close to each other, and in both cases we sample the process more frequently when it is closer to zero. This is because when $X(t) \approx 0$ the delay on both paths are very similar and any small perturbation can change the identity of the shortest path. Much more insight can be gained from Figure 2-5, but for brevity and to avoid repetition we will postpone this discussion to Section 2.3.2.

2.3.2 Ornstein-Uhlenbeck Process

The Wiener process, $W(t)$, discussed in Section 2.3.1 lacks stationarity and wanders to infinity as evident in the simulation of 500 sample paths of the Wiener process shown in Figure 2-6.

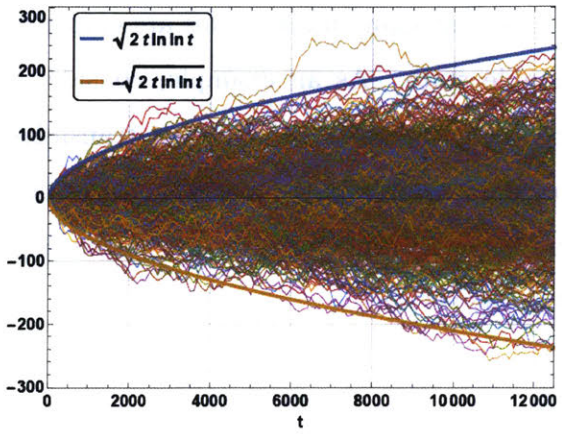


Figure 2-6: 500 sample functions of the standard Wiener process.

In fact, law of iterated logarithms can be used to show that

$$\begin{aligned} \limsup_{t\to\infty} \frac{W(t)}{\sqrt{2t\ln\ln(t)}} &= 1 \\ \liminf_{t\to\infty} \frac{W(t)}{\sqrt{2t\ln\ln(t)}} &= -1 \end{aligned}$$

While the algorithm described in the previous section can be readily deployed into real systems, the absence of stationarity makes it impossible to reason about long-term average behavior of our algorithm. To address this deficiency, we will leverage the heavy-traffic results of Halfin and Whitt [23], which show that when service times are exponentially distributed, the sequence of appropriately normalized queue lengths will converge to the Ornstein-Uhlenbeck (OU) process. It is well known that the OU process is the only non-trivial process that is simultaneously Gaussian, Markov, and stationary; all of which are ideal for our purposes. Ornstein-Uhlenbeck process is the continuous time analogue of the discrete time *auto-regressive AR(1)* process and satisfies the following stochastic differential equation:

$$dX(t) = \theta(\mu - X(t))\,dt + \sigma\,dW(t)$$

where $\sigma > 0$ is the standard deviation of the process and $W(t)$ denotes a standard Wiener process. Parameter μ is the long-term mean of the process, and $\theta > 0$ signifies the mean-reversion speed. Parameter θ may seem obscure at first glance, and the reader may wonder which network characteristic is captured by this parameter. To answer this question, note that many mechanisms are employed to steer the network towards a desirable stable state. As an example, consider the role of congestion control in TCP, which regulates the rate of packet transmission to achieve a high level of link utilization while maintaining fairness and low delay. The magnitude of θ roughly captures the strength and speed of such mechanisms and the behavior of exogenous traffic within the network.

We should note that linear combination of independent OU processes with the same θ results in another OU process with the same θ, while variance and long-term mean parameters would be added in a linear fashion.

Let us suppose that weights of paths P_1 and P_2 can be approximated by independent OU processes with the same θ; then $X(t) = X_2(t) - X_1(t)$ is another OU process. A full treatment of the general OU process is possible but to simplify our notation we shall focus on the case where the difference process, $X(t)$, has a long-term mean of $\mu = 0$. This corresponds to the case where the long-term mean delays of both paths are similar/equal. Without loss of generality suppose $X(0) = x > 0$ and assume that the OD pair uses P_1 as the optimal route until the first update epoch at T. Then the solution to the aforementioned stochastic differential equation can be written recursively as

$$X(t) = x\mathrm{e}^{-\theta t} + \sigma \int_0^t \mathrm{e}^{-\theta(t-s)}\, dW_s$$

which can be written (conditioned on $X(0) = x$) as a time-scaled Wiener process as

$$X(t) = x\mathrm{e}^{-\theta t} + \frac{\sigma \mathrm{e}^{-\theta t}}{\sqrt{2\theta}} W\left(\mathrm{e}^{2\theta t} - 1\right)$$

Given this formulation, the cost of error associated with misidentifying the shortest path during this period can be computed as

$$\begin{aligned} C_T(x) &= -\int_0^T X^-(t)\,dx = -\int_0^T \min\{X(t),0\}\,dt \\ &= \frac{x\left(\mathrm{e}^{-\theta T}-1\right)}{\theta} - \int_0^T \min\left\{\frac{\sigma \mathrm{e}^{-\theta t}}{\sqrt{2\theta}} W\left(\mathrm{e}^{2\theta t}-1\right), -x\mathrm{e}^{-\theta t}\right\} dt \end{aligned}$$

Appendix 2.C derives the following closed form expression for the expected value of this quantity,

$$\begin{aligned} \mathbb{E}\big[C_T(x)\big] &= \frac{x\left(\mathrm{e}^{-\theta T}-1\right)}{\theta} \\ &+ \frac{\sigma}{4\theta\sqrt{\theta\pi}} \int_0^{\mathrm{e}^{2\theta T}-1} \frac{\sqrt{y}}{(1+y)^{\frac{3}{2}}} \exp\left(-\frac{\theta x^2}{y\sigma^2}\right) dy \\ &+ \frac{x}{4\theta} \int_0^{\mathrm{e}^{2\theta T}-1} \frac{1}{(1+y)^{\frac{3}{2}}} \operatorname{erfc}\left(-\frac{\sqrt{\theta}x}{\sqrt{y}\sigma}\right) dy \end{aligned}$$

It comes as no surprise that $\mathbb{E}[C_T(x)]$ is the central quantity that dictates the behavior of the system, yet it presents a formidable challenge to intuition. Fortunately, $\mathbb{E}[C_T(x)]$ lends itself to a piecewise linear approximation that sheds light on its behavior. Before delving into the piecewise linear approximation, we shall investigate the shortest sampling period associated with this process, which occurs when $x = 0$. Let us start by evaluating $\mathbb{E}[C_T(x)]$ when $x = 0$,

$$\mathbb{E}\big[C_T(0)\big] = \frac{\sigma\left(\ln\left(\mathrm{e}^{\theta T}+\sqrt{\mathrm{e}^{2\theta T}-1}\right) - \sqrt{1-\mathrm{e}^{-2\theta T}}\right)}{2\theta\sqrt{\theta\pi}}$$

despite the relative simplicity of this expression, we cannot find an analytic solution to the following objective function,

$$T_1^*(0) = \underset{T>0}{\operatorname{argmin}} \frac{\eta + \mathbb{E}\big[C_T(0)\big]}{T} \tag{2.5}$$

to obtain the first order optimality condition, let us differentiate the aforementioned

expression and expand it as a Taylor series,

$$\frac{\partial}{\partial T}\left(\frac{\eta+\mathbb{E}\big[C_T(0)\big]}{T}\right) = -\frac{\eta}{T^2} + \frac{\sigma}{3\sqrt{2\pi}\sqrt{T}} + \sqrt{\mathcal{O}(T)}$$

setting the sum of the first two terms equal to zero gives us,

$$T_1^*(0) = T^* = \left(\frac{18\pi\eta^2}{\sigma^2}\right)^{\frac{1}{3}} \tag{2.6}$$

incidentally, this expression is identical to Eq. (2.2) which captured the shortest sampling period of a Wiener process. This should not be surprising because at $x = 0$, the OU process is not experiencing any mean reversion and is effectively indistinguishable from a Wiener process.

The aforementioned result can be used to create a piecewise linear approximation to $\mathbb{E}[C_T(0)]$. We can avoid the need for an excessive number of linear segments by noting that we are only interested in the behavior of the $\mathbb{E}[C_T(0)]$ when T is close to $T_1^*(0)$, and thus the simplest such approximation can be stated as[3]

$$\mathbb{E}\big[C_T(0)\big] \approx \begin{cases} 0 & \text{for } T \in \big[0, T^*\big] \\ \\ m\big(T - T^*\big) & \text{for } T > T^* \end{cases}$$

where $m = \left.\frac{\partial \mathbb{E}[C_T(0)]}{\partial T}\right|_{T=T^*} = \frac{\sigma\sqrt{1-\mathrm{e}^{-2\theta T^*}}}{2\sqrt{\theta\pi}}$ and $T^* = T_1^*(0)$.

Recall that our goal has been to solve the following general optimization problem

$$T_1^*(x) \;=\; \underset{T>0}{\operatorname{argmin}} \frac{\eta+\mathbb{E}\big[C_T(x)\big]}{T} \tag{2.7}$$

we can start by generalizing the linear approximation to $\mathbb{E}\big[C_{T^*}(0)\big]$ to non-zero values of x.

[3]We will focus on a 2-segment piecewise linear approximation in this section. A thorough development of a 3-segment piecewise linear approximation is provided in Appendix 2.D.

Note that

$$\mathbb{E}\big[C_{T^*}(x)\big] \approx \mathbb{E}\big[C_{T^*}(0)\big] + x\left(\left.\frac{\partial \mathbb{E}\big[C_{T^*}(x)\big]}{\partial x}\right|_{x=0}\right)$$

where

$$\left.\frac{\partial \mathbb{E}\big[C_{T^*}(x)\big]}{\partial x}\right|_{x=0} = \frac{\mathrm{e}^{-\theta T^*} - 1}{2\theta}$$

Note that $\mathbb{E}[C_T(x)]$ is a non-negative quantity, and is decreasing in x. As a result, we only need to consider the effects of x on the non-zero portion of the approximation, and find its corresponding interception point with the horizontal axis. Putting it all together we have:

$$\mathbb{E}\big[C_T(x)\big] \approx \begin{cases} 0 & \text{for } T \in [0, f(x)] \\ m\big(T - f(x)\big) & \text{for } T > f(x) \end{cases}$$

where $f(x) = T^* + \frac{1-\mathrm{e}^{-\theta T^*}}{\sqrt{1-\mathrm{e}^{-2\theta T^*}}} \frac{\sqrt{\pi}}{\sigma\sqrt{\theta}} x$

This approximation shows that the initial routing decision will remain correct for approximately $f(x)$ seconds; consequently the expected cost of error during this period is approximately 0. As we pass the threshold of $f(x)$ seconds, it becomes likely that the originally selected path is no longer optimal and routing erroneously through it will incur a cost of m units per second. Continuing with our approximation,

$$\frac{\eta + \mathbb{E}\big[C_T(x)\big]}{T} \approx \begin{cases} \dfrac{\eta}{T} & \text{for } T \in [0, f(x)] \\ m + \dfrac{\eta - m f(x)}{T} & \text{for } T > f(x) \end{cases}$$

The approximate cost rate function shows that if the update epoch T occurs at or before $f(x)$, we will only incur the updating cost η, amounting to a cost rate of η/T. However, if the updating epoch occurs after $f(x)$, then the cost includes the updating

cost η, as well as the cost of error which accrues at a rate of m units per second. Mathematically speaking, when $T \in [0, f(x)]$, the objective function is decreasing with T and reaches its minimum at $T = f(x)$. On the other hand, when $T > f(x)$, an increase or decrease in the objective function depends on the sign of $\eta - mf(x)$. If this expression is greater than zero, it will reach its infimum at infinity; otherwise the minimum will occur at $T = f(x)$. Putting it all together we have

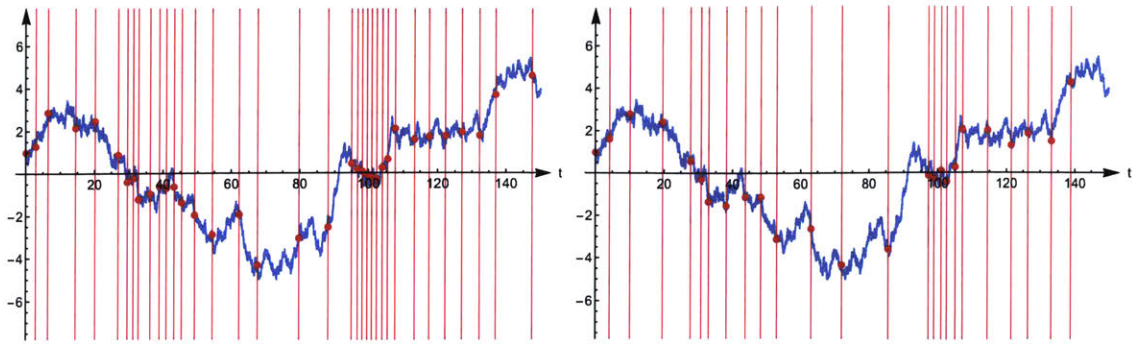
$$\underset{T>0}{\operatorname{argmin}} \frac{\eta + \mathbb{E}[C_T(x)]}{T} \approx \begin{cases} f(x) & \text{if } \eta < mf(x) \\ \infty & \text{else} \end{cases}$$

This gives us a clear description of the approximate algorithm: if the cost of updating routing information is small enough, i.e., if $\eta < mf(x)$, we should update the routing tables by sampling the process at $T = f(x)$. Otherwise, the cost of updating is too large and we should continue our previous routing decisions without any new samples.

As a result, if we constrain ourselves to cases where the updating cost $\eta < mf(x)$, we get a simple expression for the time until the next updating epoch, T, as a function of the last observed value of the process, x, namely

$$T_1^*(x) = T^* + \frac{1 - \mathrm{e}^{-\theta T^*}}{\sqrt{1 - \mathrm{e}^{-2\theta T^*}}} \frac{\sqrt{\pi}}{\sigma\sqrt{\theta}} x \tag{2.8}$$

Figures 2-7 depicts a sample path of an OU process with $\sigma = 0.5$, $\theta = 0.025$ and an initial value of $X(0) = 1$, and compares the solutions obtained by numerically solving Eq. (2.7) to the approximate solution obtained through Eq. (2.8). Red vertical lines are drawn to specify the updating epochs as suggested by our algorithm and the red dots denote the sampled values of the function. It should be noted that our approximate algorithm closely follows the exact numerical solution and is a good candidate for implementation purposes because of its significantly reduced complexity. Notice that when the process is far from zero, indicating that the delay of one of the paths is significantly less than the other, we do not need to sample their weights frequently.



(a) Numerical solution according to Eq. (2.7) (b) Approximate solution according to Eq. (2.8)

Figure 2-7: Sample path of an OU process with $\sigma = 0.5, \theta = 0.025, x_0 = 1, \eta = 0.1$.

This matches our intuition because in this scenario it is unlikely for the shorter/better path to get worse than the second path in a short period of time. On the other hand, when the process is close to zero, indicating that the weight of both paths are very similar, we require more frequent samples to track whether or not the process has changed its sign. The varying frequency of updates is revealed via the changing density of the vertical red lines in the figure.

We should additionally note that the shortest updating period occurs when the two paths have identical weights, i.e., $x = 0$. This corresponds to an updating period of $T_1^*(0) = (18\pi\eta^2/\sigma^2)^{1/3}$. Comparing our adaptive updating method to a uniform one that samples at this rate, we see a gain of

$$G = \frac{\mathbb{E}\left[T_1^*(x)\right]}{T_1^*(0)} = \frac{\int_0^\infty T_1^*(x) f_{|X(t)|}(x)\, dx}{T_1^*(0)}$$

where $f_{|X(t)|}(x)$ denotes the pdf of the (one-sided) OU process. Recalling that the OU process is Gaussian we have

$$f_{|X(t)|}(x) = 2\sqrt{\frac{\theta}{\pi\sigma^2}} \exp\left(-\frac{\theta x^2}{\sigma^2}\right) \quad \text{for } x \geq 0$$

After some algebra we get

$$G = 1 + \frac{1}{T^*} \frac{1 - \mathrm{e}^{-\theta T^*}}{\sqrt{1 - \mathrm{e}^{-2\theta T^*}}} \frac{1}{\theta\sqrt{\theta}} \tag{2.9}$$

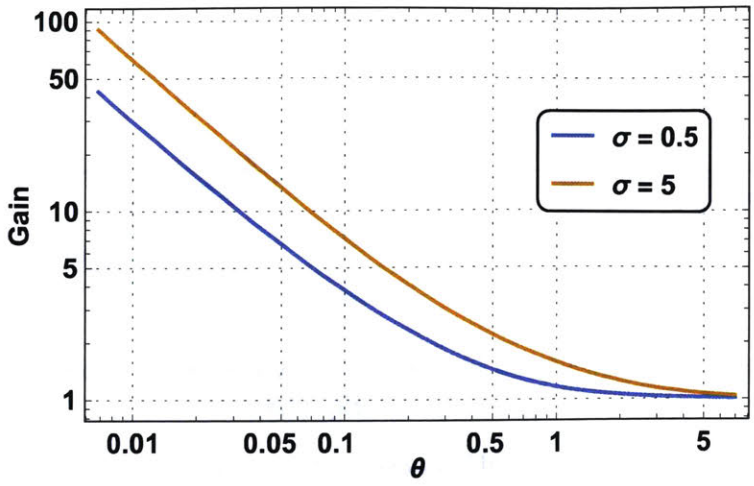


Figure 2-8: Gain of our adaptive sampling strategy ($\eta = 1$).

Which can be approximated as

$$G \approx 1 + \frac{1}{\theta}\left(\frac{\sigma}{\eta}\right)^{1/3}\left(\frac{1}{144\pi}\right)^{1/6}$$

where the approximation is accurate when θT^* is small. Note that the gain is inversely proportional to θ, and as such it improves unboundedly as θ goes to zero. This is due to the fact that θ represents the strength of mean-reversion for this process (and is inversely proportional to the coherence time of the process). As a result, when θ is small the process is not strongly attracted to its long-term mean, which significantly reduces the chances of crossing the horizontal axis. For processes with small θ, our adaptive algorithm will automatically adopt a lower sampling rate resulting in a significant reduction in sampling and updating costs. This makes sense since processes with long coherence time are very predictable and do not require much sampling. The gain will also increase, though at a lower rate, when σ increases, which amplifies the wandering behavior of the process. Figure 2-8 depicts the gain as computed by Eq. (2.9), and shows one example where the gain is nearly as high as 100. We should emphasize that such large gains reflect the fact that with small values of θ, we have effectively removed the restoring property of the underlying process and hence it closely resembles the Wiener process, and as previously discussed, in such scenarios Significant Sampling will outperform uniform sampling by a large factor.

2.4 Generalizations to Large Networks

Our analysis has so far focused on a single OD pair with two independent paths. In this section we will show that the basic operation can be extended to general networks. Recall that the shortest path can be identified through a sequence of pairwise comparisons of available paths. Furthermore, the length of each path can be updated at a time determined by our algorithm. To see the operation of such a system, let us augment the routing table of 2.1 with an additional column to track the variance of delay on each path, as shown in Table 2.2.

Table 2.2: Partial routing table maintained by node A

Destination	Path	Length	Variance
Y	$B \to C \to D \to Y$	2	4
	$E \to F \to G \to Y$	3	5
	$H \to I \to Y$	8	1
Z	$J \to K \to Z$	4	2
	$L \to M \to Z$	7	4
	$N \to O \to P \to Z$	9	7

Given the routing table in Table 2.2, node A can query nodes B and E, which are the first hops on the two shortest paths to node Y, about their respective distances to node Y according to our sampling formula

$$T_1(x) = T^* + \frac{1 - \mathrm{e}^{-\theta T^*}}{\sqrt{1 - \mathrm{e}^{-2\theta T^*}}} \frac{\sqrt{\pi}}{\sigma\sqrt{\theta}} x, \qquad T^* = \left(\frac{18\pi\eta^2}{\sigma^2}\right)^{\frac{1}{3}}$$

where $x = 3 - 2 = 1$, $\sigma = \sqrt{4+5} = 3$, and θ is a global parameter that depends on network protocols and network size. The same procedure can be used to compute the updating time for all other paths. When a link is shared between multiple OD pairs, the algorithm simply picks the smallest sampling time from those computed for all OD pairs. Not only is the proposed algorithm simple to understand and implement, it also addresses an often-overlooked byproduct of traditional routing protocols. It was shown in [25] that routing updates can inadvertently become synchronized, causing

instability as well as untimely and unmanageable bursts of traffic. To address such issues, a whole host of ad-hoc randomization procedures have been incorporated into commercial routers. In contrast, our algorithm requires each node to independently measure the variance of delay on each of its outgoing links. Given the unique geographical location of each node within the network and expected difference in their measurements, it is unlikely for the routing updates to synchronize, thus avoiding the need for explicit randomization procedures.

2.5 Conclusion

In this chapter, we introduced a new algorithm that allows us to capture the tradeoff between monitoring and optimal operation of a network. We introduced the concept of sampling/updating cost to capture the cost associated with the collection and dissemination of routing information within a network. We further studied two stochastic models of delay, namely the Wiener process and the Ornstein-Uhlenbeck process and showed that we can dynamically adjust the sampling times of each link based on their instantaneous significance to network management. The gain (as reduction in number of samples) over the traditional uniform sampling was 100 in one example. We concluded our remarks by extending our notions to general networks and suggested that a network can be operated in a decentralized fashion at high performance using ***significant sampling*** to report and update its network states, allowing the NMC system to be scalable. The essential spirit of the cognitive NMC is that it collects network states ONLY when they matter to the network performance.

Appendix

2.A $\mathbb{E}\left[C_T(x)\right]$ for a Wiener process

Recall that

$$\begin{aligned} C_T(x) &= -\int_0^T X^-(t)\,dx = -\int_0^T \min\{X(t),0\}\,dt \\ &= -\int_0^T \min\{x+\alpha W(t),0\}\,dt \\ &= -\int_0^T x+\min\{\alpha W(t),-x\}\,dt \\ &= -\int_0^T x+\alpha\min\left\{W(t),\frac{-x}{\alpha}\right\}\,dt \\ &= -xT-\alpha\int_0^T \min\left\{W(t),\frac{-x}{\alpha}\right\}\,dt \end{aligned}$$

with an expected value of

$$\mathbb{E}\left[C_T(x)\right] = -xT-\alpha\int_0^T \mathbb{E}\left[\min\left\{W(t),\frac{-x}{\alpha}\right\}\right]\,dt$$

Since $\frac{x}{\alpha}$ is a constant and is independent of $W(t)$, we can compute the distribution of the minimum as

$$\Pr\left\{\min\left\{W(t),\frac{-x}{\alpha}\right\}\le y\right\} = \begin{cases} F_{W(t)}(y) & \text{for } y<\frac{-x}{\alpha} \\ 1 & \text{for } y\ge\frac{-x}{\alpha} \end{cases}$$

Hence,

$$\begin{aligned}
\mathbb{E}\left[\min\left\{W(t), \frac{-x}{\alpha}\right\}\right] &= \int_{-\infty}^{\frac{-x}{\alpha}} y f_{W(t)}(y)\, dy + \frac{-x}{\alpha} Pr\left\{W(t) > \frac{-x}{\alpha}\right\} \\
&= \int_{-\infty}^{\frac{-x}{\alpha}} \frac{y}{\sqrt{2\pi t}} \exp\left(\frac{-y^2}{2t}\right) dy - \frac{x}{\alpha}\left[1 - F_{W(t)}\left(\frac{-x}{\alpha}\right)\right] \\
&= -\frac{\sqrt{t}}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2t\alpha^2}\right) - \frac{x}{2\alpha}\left[1 + \operatorname{erf}\left(\frac{x}{\alpha\sqrt{2t}}\right)\right]
\end{aligned}$$

Putting it all together we have

$$\begin{aligned}
\mathbb{E}\Big[C_T(x)\Big] &= -xT + \frac{\alpha}{\sqrt{2\pi}} \int_0^T \sqrt{t} \exp\left(-\frac{x^2}{2t\alpha^2}\right) dt + \frac{x}{2} \int_0^T 1 + \operatorname{erf}\left(\frac{x}{\alpha\sqrt{2t}}\right) dt \\
&= \frac{\sqrt{T}\left(2T\alpha^2 + x^2\right)}{3\alpha\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\alpha^2 T}\right) - \operatorname{erfc}\left(\frac{x}{\alpha\sqrt{2T}}\right)\left[\frac{xT}{2} + \frac{x^3}{6\alpha^2}\right]
\end{aligned}$$

2.B Approximation to $T_1^*(x)$ for a Wiener process

We shall start by proving the following lemma:

Lemma 2. For any $\beta > 0$, we have the following bound on the complementary error function

$$\frac{\left(1 - \frac{1}{2\beta^2}\right) e^{-\beta^2}}{\sqrt{\pi}\beta} < \operatorname{erfc}(\beta) < \frac{e^{-\beta^2}}{\sqrt{\pi}\beta}$$

Proof. The following proof is based on a similar result obtained in [26, p. 83]. For $\beta > 0$, a useful approximation to the complementary error function can be obtained using integration by parts:

$$\begin{aligned}
\operatorname{erfc}(\beta) &= \frac{2}{\sqrt{\pi}} \int_\beta^\infty \exp\left(-z^2\right) dz \\
&= \frac{2}{\sqrt{\pi}} \int_\beta^\infty \frac{1}{z} \left(z \exp\left(-z^2\right)\right) dz \\
&= \frac{2}{\sqrt{\pi}} \left(\frac{-1}{2z} \exp\left(-z^2\right) \Big|_\beta^\infty - \int_\beta^\infty \frac{1}{2z^2} \exp\left(-z^2\right) dz \right) \\
&= \frac{1}{\sqrt{\pi}\beta} \exp\left(-\beta^2\right) - \frac{1}{\sqrt{\pi}} \int_\beta^\infty \frac{1}{z^2} \exp\left(-z^2\right) dz \qquad \text{for } \beta > 0
\end{aligned}$$

Note that:

$$0 < \int_\beta^\infty \frac{1}{z^2} \exp\left(-z^2\right) dz < \frac{1}{\beta^3} \int_\beta^\infty z \exp\left(-z^2\right) dz = \frac{1}{2\beta^3} \exp\left(-\beta^2\right)$$

which gives us the following bounds,

$$\frac{\left(1 - \frac{1}{2\beta^2}\right) e^{-\beta^2}}{\sqrt{\pi}\beta} < \operatorname{erfc}(\beta) < \frac{e^{-\beta^2}}{\sqrt{\pi}\beta}$$

□

Let us recall the functional part of Eq. (2.3) as shown below,

$$-\eta + \frac{\sqrt{T}\left(T - x^2\right) e^{-\frac{x^2}{2T}}}{3\sqrt{2\pi}} + \frac{x^3}{6} \operatorname{erfc}\left(\frac{x}{\sqrt{2T}}\right) = 0$$

which can be rearranged as

$$2\sqrt{T}\left(T-x^2\right)e^{-\frac{x^2}{2T}}+\sqrt{2\pi}x^3\text{erfc}\left(\frac{x}{\sqrt{2T}}\right)=6\sqrt{2\pi}\eta \tag{2.10}$$

letting $\beta=\frac{x}{\sqrt{2T}}$ and applying the result of Lemma 2 we get,

$$\frac{\sqrt{2T}\left(1-\frac{T}{x^2}\right)e^{-\frac{x^2}{2T}}}{x\sqrt{\pi}}<\text{erfc}\left(\frac{x}{\sqrt{2T}}\right)<\frac{\sqrt{2T}e^{-\frac{x^2}{2T}}}{x\sqrt{\pi}}$$

rearranging the terms of the inequality gives us

$$2\sqrt{T}\left(1-\frac{T}{x^2}\right)e^{-\frac{x^2}{2T}}<\sqrt{2\pi}x\,\text{erfc}\left(\frac{x}{\sqrt{2T}}\right)<2\sqrt{T}e^{-\frac{x^2}{2T}}$$

It is clear that as $\frac{T}{x^2}\to 0$, the expression converges to the stated upper bound. Let us use the upper bound as an approximation such that:

$$\sqrt{2\pi}x\,\text{erfc}\left(\frac{x}{\sqrt{2T}}\right)\approx 2\sqrt{T}\exp\left(-\frac{x^2}{2T}\right)$$

substituting this approximation in Eq. (2.10),

$$2\sqrt{T}\left(T-x^2\right)e^{-\frac{x^2}{2T}}+x^2 2\sqrt{T}\exp\left(-\frac{x^2}{2T}\right) \;=\; 6\sqrt{2\pi}\eta$$

where terms with x^2 coefficients will cancel to give us,

$$T^{\frac{3}{2}}e^{-\frac{x^2}{2T}} \;=\; 3\sqrt{2\pi}\eta$$

Unfortunately T cannot be written in terms of elementary functions of x, and by rearranging the terms we obtain the following results:

$$x^2 \;=\; T\ln\left(\frac{T^3}{18\pi\eta^2}\right)$$

2.C $\mathbb{E}\big[C_T(x)\big]$ for a zero-mean OU process

Recall that

$$\begin{aligned}
C_T(x) &= -\int_0^T X^-(t)\,dx = -\int_0^T \min\{X(t),0\}\,dt \\
&= -\int_0^T \min\left\{xe^{-\theta t} + \frac{\sigma e^{-\theta t}}{\sqrt{2\theta}} W\left(e^{2\theta t}-1\right), 0\right\} dt \\
&= -\int_0^T xe^{-\theta t} + \min\left\{\frac{\sigma e^{-\theta t}}{\sqrt{2\theta}} W\left(e^{2\theta t}-1\right), -xe^{-\theta t}\right\} dt \\
&= -\int_0^T xe^{-\theta t}\,dt - \int_0^T \min\left\{\frac{\sigma e^{-\theta t}}{\sqrt{2\theta}} W\left(e^{2\theta t}-1\right), -xe^{-\theta t}\right\} dt \\
&= \frac{x\left(e^{-\theta T}-1\right)}{\theta} - \int_0^T \min\left\{\frac{\sigma e^{-\theta t}}{\sqrt{2\theta}} W\left(e^{2\theta t}-1\right), -xe^{-\theta t}\right\} dt
\end{aligned}$$

and

$$\begin{aligned}
&- \int_0^T \min\left\{\frac{\sigma e^{-\theta t}}{\sqrt{2\theta}} W\left(e^{2\theta t}-1\right), -xe^{-\theta t}\right\} dt \\
&= -\int_0^T \frac{\sigma e^{-\theta t}}{\sqrt{2\theta}} \min\left\{W\left(e^{2\theta t}-1\right), \frac{\sqrt{2\theta}}{\sigma} e^{\theta t}\left(-xe^{-\theta t}\right)\right\} dt \qquad (2.11) \\
&= -\int_0^T \frac{\sigma e^{-\theta t}}{\sqrt{2\theta}} \min\left\{W\left(e^{2\theta t}-1\right), \frac{-x\sqrt{2\theta}}{\sigma}\right\} dt \\
&= \frac{-\sigma}{\sqrt{2\theta}} \int_0^T e^{-\theta t} \min\left\{W\left(e^{2\theta t}-1\right), \frac{-x\sqrt{2\theta}}{\sigma}\right\} dt \\
&= \frac{-\sigma}{(2\theta)^{\frac{3}{2}}} \int_0^{e^{2\theta T}-1} \frac{1}{(1+y)^{\frac{3}{2}}} \min\left\{W(y), \frac{-x\sqrt{2\theta}}{\sigma}\right\} dy \qquad (2.12) \\
&= \frac{-\sigma}{(2\theta)^{\frac{3}{2}}} \int_0^{e^{2\theta T}-1} \frac{\min\{W(y), h\}}{(1+y)^{\frac{3}{2}}}\, dy \qquad (2.13)
\end{aligned}$$

where (2.11) is possible because $\theta > 0$, and $\sigma > 0$, and we know that $\min(A,B) = \frac{1}{\alpha}\min(\alpha A, \alpha B)$ for $\alpha > 0$ (in our scenario, $\alpha = \frac{\sqrt{2\theta}}{\sigma} e^{\theta t}$). Equation (2.12) is the result of a simple change of variable for $y = e^{2\theta t} - 1$, and (2.13) simplifies the notation by defining $h = -x\sqrt{2\theta}/\sigma$.

Hence,

$$\mathbb{E}\big[C_T(x)\big] = \frac{x\left(\mathrm{e}^{-\theta T}-1\right)}{\theta} - \frac{\sigma}{(2\theta)^{\frac{3}{2}}}\int_0^{\mathrm{e}^{2\theta T}-1}\frac{\mathbb{E}\left[\min\left\{W(y),h\right\}\right]}{(1+y)^{\frac{3}{2}}}\,dy$$

Using the same process used in Appendix 2.A to compute the expectation, we obtain

$$\begin{aligned}\mathbb{E}\left[\min\left\{W(y),h\right\}\right] &= -\frac{\sqrt{y}}{\sqrt{2\pi}}\exp\left(-\frac{h^2}{2y}\right) + h\frac{1}{2}\left[1-\operatorname{erf}\left(\frac{h}{\sqrt{2y}}\right)\right] \\ &= -\frac{\sqrt{y}}{\sqrt{2\pi}}\exp\left(-\frac{h^2}{2y}\right) + \frac{h}{2}\operatorname{erfc}\left(\frac{h}{\sqrt{2y}}\right)\end{aligned}$$

which simplifies the expected cost of error to

$$\begin{aligned}\mathbb{E}\big[C_T(x)\big] &= \frac{x\left(\mathrm{e}^{-\theta T}-1\right)}{\theta} \\ &+ \frac{\sigma}{\sqrt{2\pi}(2\theta)^{\frac{3}{2}}}\int_0^{\mathrm{e}^{2\theta T}-1}\frac{\sqrt{y}}{(1+y)^{\frac{3}{2}}}\exp\left(-\frac{h^2}{2y}\right)\,dy \\ &- \frac{\sigma}{2(2\theta)^{\frac{3}{2}}}\int_0^{\mathrm{e}^{2\theta T}-1}\frac{h}{(1+y)^{\frac{3}{2}}}\operatorname{erfc}\left(\frac{h}{\sqrt{2y}}\right)\,dy \\ &= \frac{x\left(\mathrm{e}^{-\theta T}-1\right)}{\theta} \\ &+ \frac{\sigma}{4\theta\sqrt{\theta\pi}}\int_0^{\mathrm{e}^{2\theta T}-1}\frac{\sqrt{y}}{(1+y)^{\frac{3}{2}}}\exp\left(-\frac{\theta x^2}{y\sigma^2}\right)\,dy \\ &+ \frac{x}{4\theta}\int_0^{\mathrm{e}^{2\theta T}-1}\frac{1}{(1+y)^{\frac{3}{2}}}\operatorname{erfc}\left(-\frac{\sqrt{\theta}x}{\sqrt{y}\sigma}\right)\,dy\end{aligned}$$

2.D Piecewise Linear Approximation to $\mathbb{E}\big[C_T(0)\big]$ for an OU process

Recall that

$$\begin{aligned}\mathbb{E}\big[C_T(x)\big] &= \frac{x\left(\mathrm{e}^{-\theta T}-1\right)}{\theta}+\frac{\sigma}{4\theta\sqrt{\theta\pi}}\int_0^{\mathrm{e}^{2\theta T}-1}\frac{\sqrt{y}}{(1+y)^{\frac{3}{2}}}\exp\left(-\frac{\theta x^2}{y\sigma^2}\right)dy \\ &+ \frac{x}{4\theta}\int_0^{\mathrm{e}^{2\theta T}-1}\frac{1}{(1+y)^{\frac{3}{2}}}\operatorname{erfc}\left(-\frac{\sqrt{\theta}x}{\sqrt{y}\sigma}\right)dy\end{aligned}$$

and thus,

$$\mathbb{E}\big[C_T(0)\big]=\frac{\sigma\left(\ln\left(\sqrt{\mathrm{e}^{2\theta T}-1}+\mathrm{e}^{\theta T}\right)-\sqrt{1-\mathrm{e}^{-2\theta T}}\right)}{2\theta\sqrt{\theta\pi}}$$

We shall approximate $\mathbb{E}\big[C_T(0)\big]$ via 3 linear segments. The first segment should approximate the function when $T\approx 0$, the third section should approximate the function when $T\to\infty$, and the middle section should connect the aforementioned two segments in a reasonable fashion. Let us start by computing the derivative of $\mathbb{E}\big[C_T(0)\big]$ with respect to T,

$$\begin{aligned}\frac{\partial\mathbb{E}[C_T(0)]}{\partial T} &= \frac{\partial}{\partial T}\left(\frac{\sigma\left(\ln\left(\sqrt{\mathrm{e}^{2\theta T}-1}+\mathrm{e}^{\theta T}\right)-\sqrt{1-\mathrm{e}^{-2\theta T}}\right)}{2\theta\sqrt{\theta\pi}}\right) \\ &= \frac{\sigma}{2\theta\sqrt{\theta\pi}}\frac{\partial}{\partial T}\left(\ln\left(\sqrt{\mathrm{e}^{2\theta T}-1}+\mathrm{e}^{\theta T}\right)-\sqrt{1-\mathrm{e}^{-2\theta T}}\right) \\ &= \frac{\sigma}{2\theta\sqrt{\theta\pi}}\left(\frac{\frac{2\theta\mathrm{e}^{2\theta T}}{2\sqrt{\mathrm{e}^{2\theta T}-1}}+\theta\mathrm{e}^{\theta T}}{\sqrt{\mathrm{e}^{2\theta T}-1}+\mathrm{e}^{\theta T}}-\frac{2\theta\mathrm{e}^{-2\theta T}}{2\sqrt{1-\mathrm{e}^{-2\theta T}}}\right) \\ &= \frac{\sigma}{2\sqrt{\theta\pi}}\left(\frac{\frac{\mathrm{e}^{2\theta T}}{\sqrt{\mathrm{e}^{2\theta T}-1}}+\mathrm{e}^{\theta T}}{\sqrt{\mathrm{e}^{2\theta T}-1}+\mathrm{e}^{\theta T}}-\frac{\mathrm{e}^{-2\theta T}}{\sqrt{1-\mathrm{e}^{-2\theta T}}}\right) \\ &= \frac{\sigma}{2\sqrt{\theta\pi}}\left(\frac{\frac{\mathrm{e}^{2\theta T}}{\sqrt{\mathrm{e}^{2\theta T}-1}}+\mathrm{e}^{\theta T}}{\sqrt{\mathrm{e}^{2\theta T}-1}+\mathrm{e}^{\theta T}}\frac{\sqrt{\mathrm{e}^{2\theta T}-1}-\mathrm{e}^{\theta T}}{\sqrt{\mathrm{e}^{2\theta T}-1}-\mathrm{e}^{\theta T}}-\frac{\mathrm{e}^{-2\theta T}}{\sqrt{1-\mathrm{e}^{-2\theta T}}}\right) \\ &= \frac{\sigma\sqrt{1-e^{-2\theta T}}}{2\sqrt{\pi\theta}}\end{aligned}$$

Notice that

$$\begin{aligned}\left.\frac{\partial \mathbb{E}[C_T(0)]}{\partial T}\right|_{T=0} &= 0 \\ \lim_{T\to\infty} \frac{\partial \mathbb{E}[C_T(0)]}{\partial T} &= \lim_{T\to\infty} \frac{\sigma\sqrt{1-e^{-2\theta T}}}{2\sqrt{\pi\theta}} = \frac{\sigma}{2\sqrt{\pi\theta}}\end{aligned}$$

Hence, the first segment of the approximation will be a horizontal line (i.e. with slope of 0) and the third segment will have a slope equal to $\frac{\sigma}{2\sqrt{\pi\theta}}$. The slope of the third segment signifies that the long-term *growth rate* of $\mathbb{E}[C_T(0)]$ is equal to $\frac{\sigma}{2\sqrt{\pi\theta}}$. In fact, since the process is stationary, this limit does not depend on the last observed value, and is also valid for $\mathbb{E}[C_T(x)]$. Stated another way, if we do not sample the process, in the long run, it will incur $\frac{\sigma}{2\sqrt{\pi\theta}}$ units of additional cost per unit time as a result of misidentifying the shortest path. Consequently, the segment of our approximation corresponding to the long-term behavior of $\mathbb{E}[C_T(0)]$ should have a slope of $\frac{\sigma}{2\sqrt{\pi\theta}}$. Computing the algebraic expression for the third segment of the approximation can be done by,

$$\begin{aligned}\lim_{T\to\infty} \mathbb{E}\big[C_T(0)\big] &= \lim_{T\to\infty} \frac{\sigma\left(\ln\left(\sqrt{\mathrm{e}^{2\theta T}-1}+\mathrm{e}^{\theta T}\right)-\sqrt{1-\mathrm{e}^{-2\theta T}}\right)}{2\theta\sqrt{\theta\pi}} \\ &= \frac{\sigma\left(\ln\left(2\mathrm{e}^{\theta T}\right)-1\right)}{2\theta\sqrt{\theta\pi}} \\ &= \frac{\sigma}{2\sqrt{\theta\pi}}\left(T-\frac{1-\ln(2)}{\theta}\right)\end{aligned}$$

There are many ways to choose the middle segment. One reasonable way is to choose it so that its intersection with the first segment coincides with the solution to Eq. (2.6), i.e. it will intersect the first segment at $T_1^*(0) = T^* = \left(\frac{18\pi c^2}{\sigma^2}\right)^{\frac{1}{3}}$. Furthermore, we can choose the second segment such that it will have a slope equal to the slope of $\mathbb{E}\big[C_T(0)\big]$ at T^*. Using m_1 to denote this slope, we have $m_1 = \left.\frac{\partial \mathbb{E}[C_T(0)]}{\partial T}\right|_{T=T^*} = \frac{\sigma\sqrt{1-\mathrm{e}^{-2\theta T^*}}}{2\sqrt{\theta\pi}}$ and $T^* = T_1^*(0)$.

Putting all of this together, and letting $L(0,T)$ denote our 3-segment piecewise linear approximation to $\mathbb{E}[C_T(0)]$, we get

$$L(0,T) = \begin{cases} 0 & \text{for } T \in [0, T^*] \\ m_1 \left(T - T^*\right) & \text{for } T \in [T^*, T^{**}] \\ m_2 \left(T - \dfrac{1 - \ln(2)}{\theta}\right) & \text{for } T > T^{**} \end{cases}$$

where

$$\begin{aligned} m_1 &= \frac{\sigma\sqrt{1 - 2\mathrm{e}^{-2\theta T^*}}}{2\sqrt{\theta\pi}} \\ m_2 &= \frac{\sigma}{2\sqrt{\theta\pi}} \\ T^{**} &= \frac{-T^*\sqrt{1 - \mathrm{e}^{-2\theta T^*}} + \frac{1-\ln(2)}{\theta}}{1 - \sqrt{1 - \mathrm{e}^{-2\theta T^*}}} \end{aligned}$$

Figure 2.D.1 depicts $\mathbb{E}[C_T(0)]$ as well as our 3-segment piecewise linear approximation, $L(0,T)$, for $\sigma = 0.5$, and $\theta = 0.02$.

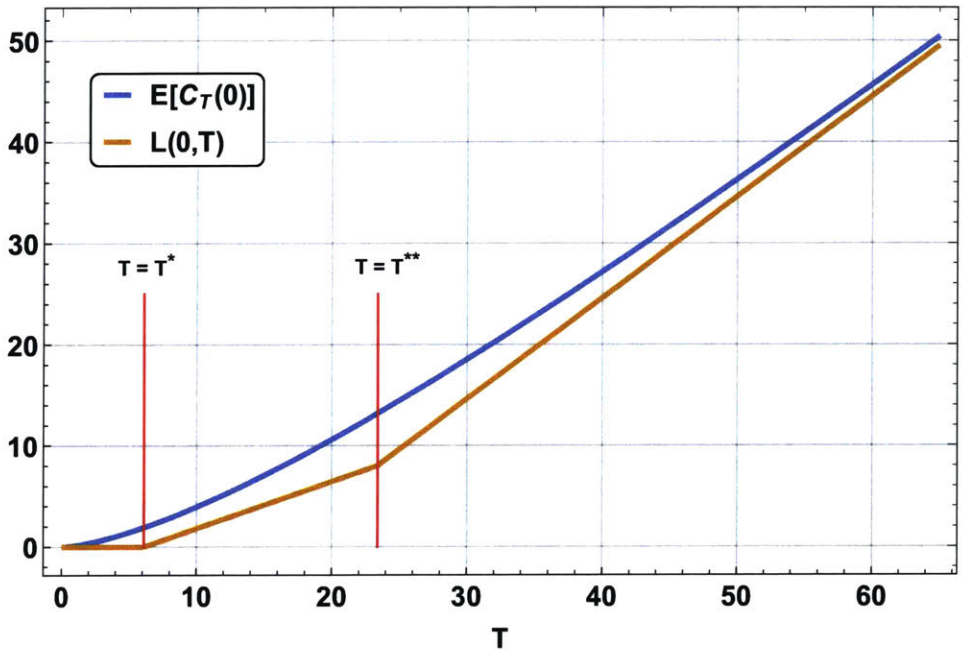


Figure 2.D.1: A 3-segment piecewise linear approximation to $\mathbb{E}[C_T(0)]$.

Chapter 3

A Deep Reinforcement Learning Solution to Significant Sampling

3.1 Introduction

In Chapter 2 we introduced a centralized routing protocol in which an NMC system monitors the network state (e.g., link/queuing delays) to decide which paths should be used to connect different origin-destination (OD) pairs. The monitoring process was interpreted as a sampling process involving three steps:

(a) **Updating**: The nodes in the network report their state (i.e. their most recent queuing delay over a period) to the NMC system.

(b) **Decision**: The NMC system identifies the shortest path between each OD pair, and computes the next report time.

(c) **Dissemination**: The NMC system disseminates the updated routing information and the next report time to all the nodes in the network.

We showed that a critical issue for this routing protocol is to determine the sampling frequency. Uniform sampling at high frequency can be burdensome and costly due to the excessive use of network transport and computational resources, while sparse sampling can lead to misconfiguration and suboptimal routing decisions. We emphasized that judicious sampling of network states should be an essential feature of any pragmatic NMC system, and introduced the notion of significant sampling, where the next report time is computed by the NMC from its cognitive understanding of the network states and short-term behavior of exogenous offered traffic. In a nutshell, the NMC will decide the next sampling time based on the likelihood that the optimal routes should be recomputed.

In an effort to get an analytical solution to the significant sampling problem, we proposed the Ornstein-Uhlenbeck (OU) process as the underlying traffic/delay model and derived the optimal sampling policy when the parameters of the OU process are known to the NMC system. Despite its advantageous analytical nature, the work in Chapter 2 has a few shortcomings that we wish to address in this chapter. First, the derived policy ϕ' suffers from an issue that we have dubbed as the "never-sample" problem. This problem arises when the cost of sampling is too high, in which case the policy ϕ' can instruct the NMC system to "never sample", a policy that can be

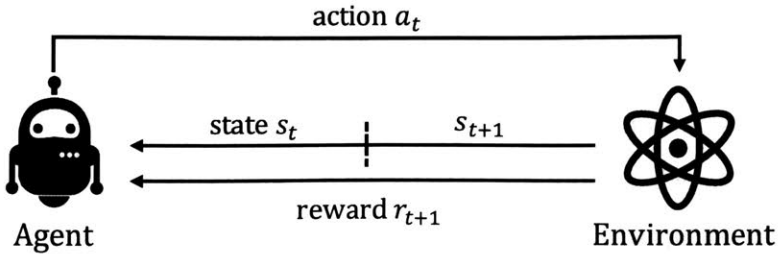


Figure 3-1: The interactions between agent and environment in reinforcement learning [8].

detrimental in the long run. Second, considering the bursty and unpredictable nature of traffic in dynamic networks, the assumed OU delay model is narrow in scope and may not apply to many networks. Last but not least, even if the OU process is a good approximation for a specific network, estimating the parameters of the OU process can be difficult and estimation errors can significantly impact the performance of the system. Furthermore, a realistic and robust sampling policy should be able handle and smoothly transition through various operating regimes, an issue that was not considered in Chapter 2.

In this chapter, we present an automated and model-free NMC system using deep reinforcement learning (DRL) techniques. As a refresher, we note that a salient feature of RL is "learning from interactions". As depicted in Fig. 3-1, the agent interacts with the environment in a sequence of time steps indexed by t. Given the observed state of the environment, s_t, the agent takes action a_t, which steers the environment to state s_{t+1} in the next time step. The environment then feedbacks a reward r_{t+1}, from which the agent can measure the quality of action a_t. A mapping between s_t and a_t is referred to as a policy function. The aim of the agent is generally to learn the optimal policy that maximizes the cumulative future rewards. The latest trend in RL research is to integrate the recent advances of deep learning [27] into the RL framework [28, 29]. RL that makes use of deep neural networks to approximate the optimal policy function – directly or indirectly – is referred to as deep reinforcement learning (DRL) [30]. DRL allows RL algorithms to be applied when the number of possible state-action pairs is enormous which inhibits traditional function approximators from accurately approximating the policy function.

The main contributions of this chapter can be summarized as follows:

- We prove that the sampling policy ϕ' derived in Chapter 2 suffers from the never-sample problem when sampling cost is high. We characterize the never-sample region for the setup in Chapter 2.

- We put forth a DRL solution to the significant sampling problem. Our DRL solution has the advantage that it makes no assumption about the underlying traffic model, but learns to sample the network in such a way as to achieve optimal performance. We demonstrate the optimality of the DRL solution by using it to learn the policy ϕ'. Experimental results confirm that the learned policy matches the analytical results of the previous chapter.

- We design a multi-step look-ahead policy, based on our DRL solution, to address the never-sample problem of policy ϕ'. This new policy considers the long-term impact of a decision, and minimizes the cost rate over an infinite time horizon. Experimental results show that our new policy outperforms policy ϕ' by 39%, in terms of the average cost rate.

- Given the model-free nature of RL, we demonstrate that our DRL solution is robust to various network conditions. In this context, we extend the DRL framework beyond that of Chapter 2 and provide solutions for general cases beyond the two-path OU process.

The rest of the this chapter is organized as follows: Section 3.2 reintroduces some of the concepts and findings of Chapter 2 with the benefit of additional nomenclature and parameters that are needed in the context of DRL. Section 3.3 points out and elaborates on two limitations of our prior solutions. Section 3.4 introduces our DRL solution to the significant sampling problem and describes our new multi-step look-ahead policy. Experimental results are presented in Section 3.5. A summary and concluding remarks are provided in Section 3.6.

3.2 System Model

3.2.1 Significant Sampling

Let us consider an OD pair in a dynamic network, and assume that the OD pair is connected via N paths $\{P_n : n \in [N]\}$, where $[N] = \{1, 2, \ldots, N\}$. The stochastically evolving weights of the N paths are denoted by $\{X_n(t) : n \in [N]\}$[1]. In shortest path routing, an NMC system will sample the weight of the N paths by orchestrating three distinct actions:

(a) The nodes in each path report their weights to the NMC system.

(b) The NMC system identifies the path with the smallest weight and computes the next sampling time.

(c) The NMC disseminates the routing information as well as the next reporting/sampling time to each node.

As an example, suppose that the nodes on the N paths report[2] their weights to the NMC at a sampling epoch t_{i-1}. Given observation $\{X_n(t) : n \in [N], t \leq t_{i-1}\}$, the NMC will decide to route through $P_{n'}$, where $n' = \text{argmin}_n X_n(t_{i-1})$. This routing information, along with the next sampling epoch, will be disseminated to all the nodes on each path. The same route will be used until the next sampling epoch t_i, at which point the NMC will use $\{X_n(t) : n \in [N], t \leq t_i\}$ to make new routing decisions. A fixed cost η is associated with one sampling operation, which captures the efforts required to collect and disseminate the NSI throughout the network[3].

[1] Usually, weight refers to delay (queuing delay in particular).

[2] At each sampling epoch, we assume that the nodes will report the trajectory of their evolving weights since the last sampling report to the NMC. For example, at t_i, the full realizations of $\{X_n(t) : n \in [N], t_{i-1} \leq t \leq t_i\}$ will be provided to the NMC, where t_{i-1} is the last sampling time.

[3] To be more specific, the cost of one sampling operation is composed of communication cost and cost of action. The communication cost is introduced in steps a) and c), and can be considered as constant. For example, the communication cost in the step a) can be fixed to one IP packet/node (considering a typical IP packet containing 1K Bytes, a node can send 500 real numbers to the NMC per update if we use 2 bytes to represent a real number). The cost of action is introduced by the computation in step b), computational cost of updating routing-tables, and other side-effects. We use η as a catch-all quantity to capture the overall cost of one sampling operation.

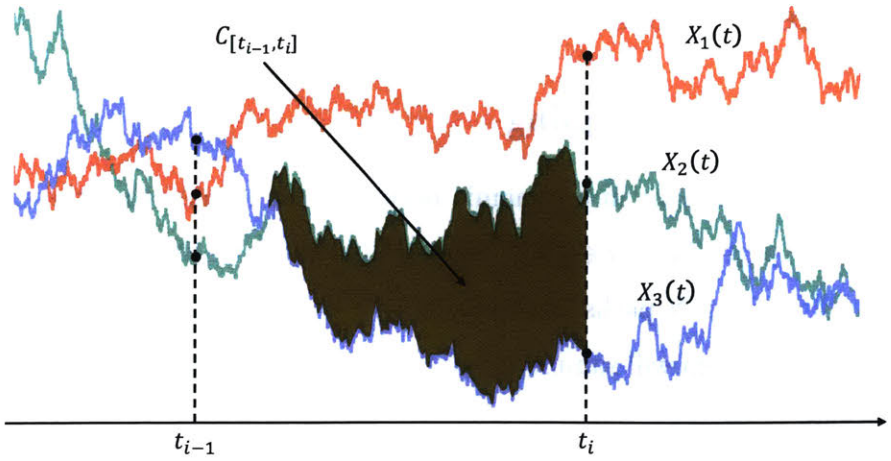


Figure 3-2: Cost of error, $C_{[t_{i-1},t_i]}$, between consecutive sampling epochs t_{i-1} and t_i.

As illustrated in Fig. 3-2, if the shortest path changes between two sampling epochs t_{i-1} and t_i, we will incur an excess cost $C_{[t_{i-1},t_i]}$ given by

$$C_{[t_{i-1},t_i]} = \int_{t_{i-1}}^{t_i} X_{n'}(t) - \min_n X_n(t)\, dt$$

where $n' = \text{argmin}_n X_n(t_{i-1})$ is the shortest path at epoch t_{i-1}, and $\min_n X_n(t)$ denotes the weight of the "true" shortest path at time t. Not surprisingly, continuous sampling of the stochastic process can reduce the cost of error to zero, but will result in an extremely high sampling cost. On the other hand, sparse sampling incurs a low sampling cost, at the expense of higher cost of error. The optimal tradeoff between sampling cost and cost of error can be described through a policy that specifies the best future sampling times based on the previous observations of the weight processes. Specifically, we write the policy at epoch t_{i-1} as a deterministic function that maps the history of weight processes $s(t_{i-1}) = \{X_n(t) : n \in [N], t \leq t_{i-1}\}$ to the next sampling interval T_i:

$$T_i = \phi(s(t_{i-1}))$$

If weight processes are Markovian (i.e., all the information is captured by the most recent observation, and prior history of the process is inconsequential for optimal decision making), we can design $s(t_{i-1})$ to be $s(t_{i-1}) = \{X_n(t_{i-1}) : n \in [N]\}$.

The optimal policy ϕ^*, which specifies the optimal T_i^* at epoch t_{i-1}, minimizes the cost rate (i.e., cost per unit time) over an infinite time horizon, giving

$$\phi^* = \underset{\phi}{\operatorname{argmin}} \lim_{L\to\infty} \frac{\sum_{l=i}^{L} \left(\eta + C_{[t_{l-1}, t_{l-1}+\phi(s(t_{l-1}))]}\right)}{\sum_{l=i}^{L} \phi(s(t_{l-1}))} \tag{3.1}$$

We refer to the problem of discovering the optimal policy ϕ^* as the "significant sampling" problem for shortest path routing.

3.2.2 Prior Solution

Chap. 2 derived analytical solutions to significant sampling under three assumptions:

1. $N = 2$, i.e., the OD pair is connected via only two paths with weights $X_1(t)$ and $X_2(t)$. Since there are only two paths, identifying the shorter path is equivalent to identifying the sign of the difference process $X(t) = X_1(t) - X_2(t) : X_1(t) \lesseqgtr X_2(t)$ if and only if $X(t) \lesseqgtr 0$.

2. Weight processes $X_1(t)$ and $X_2(t)$ are independent Ornstein-Uhlenbeck (OU) processes with the same mean value, and same θ parameter. This means $X(t)$ is a zero-mean OU process. An OU process $\mathrm{OU}(t)$ parameterized by $\{\mu_0, \theta_0, \sigma_0\}$ is governed by the following stochastic differential equation:

$$d\mathrm{OU}(t) = \theta_0(\mu_0 - \mathrm{OU}(t))\, dt + \sigma_0\, dW(t)$$

   where μ_0 is the long-term mean of the process, $\theta_0 > 0$ is the mean-reversion speed, and σ_0 is the volatility. This expression consists of two parts. The first part, $\theta_0(\mu_0 - \mathrm{OU}(t))\, dt$, captures the mean reverting behavior, which is akin to a force that pulls the process towards its long-term mean. The second part, $\sigma_0\, dW(t)$, is a standard Wiener process scaled by volatility factor σ_0. This second part acts as additive noise and counteracts the mean reversion. Hence, the OU process can be roughly described as noisy oscillations around μ_0.

3. Direct derivation of policy ϕ^* is quite tricky, as both numerator and denominator in (3.1) can increase unboundedly. In Chapter 2, we approximated the optimal policy ϕ^* by

$$\phi' = \underset{\phi}{\operatorname{argmin}} \frac{\eta + \mathbb{E}\left[C_{[t_{i-1}, t_{i-1}+\phi(s(t_{i-1}))}\right]}{\phi(s(t_{i-1}))} \tag{3.2}$$

Compared with ϕ^*, policy ϕ' is myopic and minimizes the cost rate in $[t_{i-1}, t_i]$, and takes no account of the actions after t_i. We shall refer to the policy ϕ' as the "one-step look-ahead policy".

Given the above assumptions, we had focused on sampling the difference process $X(t)$, an OU process parameterized by $\{\mu = 0, \theta, \sigma\}$[4]. Since OU process is Markovian, we can simplify $s(t_{i-1})$ to $s(t_{i-1}) = X(t_{i-1})$. The policy ϕ' can then be written as

$$\phi'(X(t_{i-1}) = x) = T_i'\Big|_{X(t_{i-1})=x} = \underset{T_i>0}{\operatorname{argmin}} \frac{\eta + \mathbb{E}\left[C_{[t_{i-1}, t_{i-1}+T_i]} | X(t_{i-1}) = x\right]}{T_i} \tag{3.3}$$

where we had further derived

$$\begin{aligned} \mathbb{E}\left[C_{[t_{i-1}, t_{i-1}+T_i]}\Big| X(t_{i-1}) = x\right] &= \frac{x\left(\mathrm{e}^{-\theta T_i} - 1\right)}{\theta} \\ &+ \frac{\sigma}{4\theta\sqrt{\theta\pi}} \int_0^{\mathrm{e}^{2\theta T_i}-1} \frac{\sqrt{y}}{(1+y)^{\frac{3}{2}}} \exp\left(-\frac{\theta x^2}{y\sigma^2}\right) dy \\ &+ \frac{x}{4\theta} \int_0^{\mathrm{e}^{2\theta T_i}-1} \frac{1}{(1+y)^{\frac{3}{2}}} \operatorname{erfc}\left(-\frac{\sqrt{\theta}x}{\sqrt{y}\sigma}\right) dy \end{aligned} \tag{3.4}$$

The optimal sampling interval $T_i'|_{X(t_{i-1})=x}$ under this policy can then be computed numerically by solving the non-convex optimization in (3.3). We have reproduced the results of Fig. 2-7.a in Fig. 3-3 below which depicts a realization of an OU process with parameters $\{\mu = 0, \theta = 0.025, \sigma = 0.5\}$ as well as the resulting sampling policy ϕ' when the sampling cost is $\eta = 0.1$.

[4]If both $X_1(t)$ and $X_2(t)$ are OU processes parameterized by $\{\mu_0, \theta_0, \sigma_0\}$, then $X(t)$ is an OU process parameterized by $\{\mu = 0, \theta = \theta_0, \sigma = \sqrt{2}\sigma_0\}$.

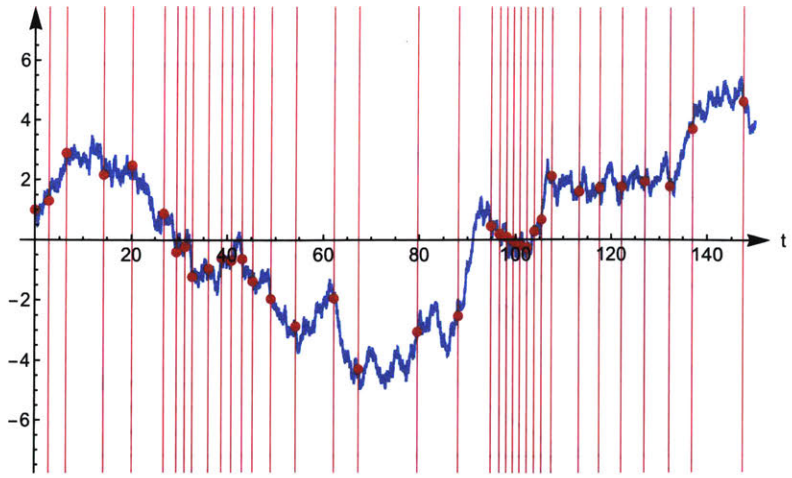


Figure 3-3: Applying the sampling policy ϕ' on a realization of an OU process with parameters $\{\mu = 0, \theta = 0.025, \sigma = 0.5\}$. The sampling cost $\eta = 0.1$.

3.3 Limitations of Prior Solution

This section elaborates on two limitations of the aforementioned analytical solution.

Hard to generalize – The assumptions stated in section 3.2.2 are meant to simplify the significant sampling problem so that analytical solutions can be derived. However, the simplified problem still leads to complicated expressions that are hard to interpret (e.g., Eq. (3.4)). Furthermore, extensions to more general cases (e.g., the N-path case, or non-OU processes) can be even more complex.

The never-sample problem – Policy ϕ' performs well when the sampling cost is small. However, when the sampling cost is large, policy ϕ' instructs the NMC system to "never sample". The details of this never-sample problem can be understood through the following Proposition 1.

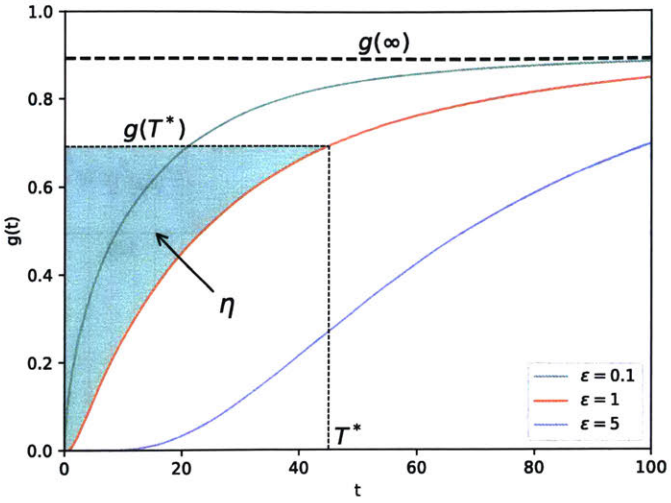


Figure 3-4: A visual explanation of policy ϕ'. Where $X(t)$ is a zero-mean OU process parameterized by $\{\mu = 0, \theta, \sigma\}$; $X(t') = \varepsilon$ is the sampled value at epoch t'; and $g(t)$ in Eq. (3.6) is the instantaneous cost rate for $t > t'$ conditioned on $X(t') = \varepsilon$.

Proposition 1. Suppose $X(t)$ is a zero-mean OU process parameterized by $\{\mu = 0, \theta, \sigma\}$. At epoch t_{i-1}, we sample $X(t)$ and obtain $X(t_{i-1})$. Then, there exists a real number $\varepsilon \geq 0$ such that $\forall\ |X(t_{i-1})| \leq \varepsilon$, policy ϕ' will instruct the NMC to never sample the process if

$$\eta > \int_0^\infty \sqrt{\frac{\sigma^2}{4\pi\theta}} - g(t)\, dt \tag{3.5}$$

where

$$g(t) = \sqrt{\frac{\sigma^2(1-e^{-2\theta t})}{4\pi\theta}} e^{-\frac{\theta\varepsilon^2}{\sigma^2(e^{2\theta t}-1)}} - \frac{\varepsilon e^{-\theta t}}{2} \operatorname{erfc}\left(\sqrt{\frac{\theta\varepsilon^2}{\sigma^2(e^{2\theta t}-1)}}\right) \tag{3.6}$$

Proof. Assuming at some epoch t', we sample $X(t)$ and obtain $X(t') = \varepsilon$. The decision rule of policy ϕ' for the next sampling interval T^* can be described visually by Fig. 3-4 (see Appendices 3.A, 3.B, and 3.C for detailed derivations and analyses).

As shown in Fig. 3-4, the shaded area increases with T; the optimal sampling interval T^* indicated by policy ϕ' is the T that makes the shaded area equal to the sampling cost η. As a result, the policy ϕ' can instruct us to "never sample" if η is greater than the shaded area even when $T \to \infty$. This gives us the lower bound for η in (3.5). When (3.5) is satisfied, if the absolute value of any sample $X(t_{i-1})$ is less than or equal to ε, policy ϕ' will instruct the NMC to never sample, and the cost rate will converge to $g(\infty) = \sqrt{\frac{\sigma^2}{4\pi\theta}}$ as time passes. □

We should point out that the "never sample" policy is not always suboptimal if our objective is to minimize the cost rate over an infinite time horizon, as in (3.1). A simple example is when $\eta \to \infty$, for which the optimal policy ϕ^* is to never sample. However, Proposition 2 below, further shows that policy ϕ' is guaranteed to be suboptimal if η is smaller than an upper bound.

Proposition 2. Policy ϕ' is suboptimal in terms of minimizing the cost rate over an infinite time horizon if

$$\int_0^\infty \sqrt{\frac{\sigma^2}{4\pi\theta}} - g(t)\, dt < \eta < \sqrt{\frac{\sigma^2}{4\pi\theta}}\frac{1}{\theta}. \tag{3.7}$$

In particular, sampling the weight process once at the equilibrium state can outperform ϕ'.

Proof. See Appendix A. □

This chapter is in particular concerned with overcoming these two limitations. In particular, we put forth a deep reinforcement learning (DRL) solution to solve the significant sampling problem. A DRL solution has the advantage that it makes no assumption about the weight processes (often referred to as the "model-free" property of DRL), and is thus more robust to various network conditions that give rise to different stochastic variations of $X_n(t)$. This enables us to remove the assumptions stated in section 3.2.2, and provide solutions to more general settings that go beyond the 2-path limitation and traffic modeling assumption (OU process). Moreover, we leverage this DRL framework to design a better policy that minimizes the cost rate over a long time horizon. This new policy, by planning farther beyond the immediate future, can effectively address the never-sample problem faced by policy ϕ'.

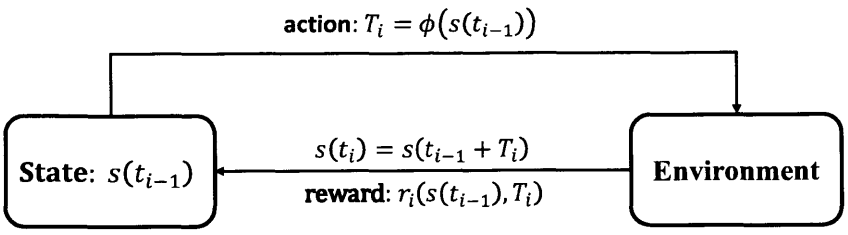


Figure 3-5: Significant sampling as a reinforcement learning problem. An agent interacts with the environment to learn the optimal sampling policy.

3.4 A DRL Solution to Significant Sampling

3.4.1 Significant Sampling as an RL Problem

Let us start by transforming the original significant sampling problem to a reinforcement learning problem. The basic idea is that we can view the NMC system as an agent, and the underlying weight processes $\{X_n(t)\}$ as the environment. As shown in Fig. 3-5, the agent will maintain a ledger that contains the state of the environment, i.e., $s(t_{i-1}) = \{X_n(t) : n \in [N], t \leq t_{i-1}\}$. Given state $s(t_{i-1})$, the agent's policy ϕ will output an action $T_i = \phi(s(t_{i-1}))$, which effectively specifies the next epoch for sampling the environment and thus the time for updating the ledger. At $t_i = t_{i-1} + T_i$, the agent will sample the environment, update the ledger with $s(t_i) = \{X_n(t) : n \in [N], t \leq t_i\}$, and the cycle continues.

The agent's ability to improve its actions depends on a feedback system that can assign values to various actions so that the agent can distinguish good actions from bad ones. This feedback is often expressed in terms of a *reward* function which indicates the reward associated with taking a particular action when the environment is in state $s(t_{i-1})$. For significant sampling, the environment will provide $\{X_n(t) : n \in [N], t_{i-1} \leq t \leq t_i\}$ as feedback at the end of a time step, from which the agent computes a reward r_i (to be defined later) to evaluate the action T_i in this time step. We shall refer to a complete cycle from t_{i-1} to t_i as one "time step". The learning process will traverse many time steps, and each time step provides the agent with a new experience denoted by the quaternion $\{s(t_{i-1}), T_i, r_i, s(t_i)\}$.

Action-value function

In standard RL algorithms, the agent's objective at epoch t_{i-1} is to learn a policy that maximizes the future *return* R_i, a common definition of which is the sum of discounted future rewards

$$R_i = \sum_{l=i}^{\infty} \gamma^{l-i} r_l \tag{3.8}$$

where $\gamma \in [0, 1]$ is a discounting factor. An action-value function (also known as the Q function) describes the maximum achievable return if the agent takes action T_i in state $s(t_{i-1})$, and then follows the optimal policy from then on. That is,

$$Q^*(s(t_{i-1}), T_i) \;=\; \max_{\phi} \mathbb{E}_{\text{env}} \left[R_i | s(t_{i-1}), T_i, \phi\right] \;=\; \mathbb{E}_{\text{env}} \left[r_i + \gamma \max_{\phi} Q^*(s(t_i), \phi(s(t_i)))\right] \tag{3.9}$$

where $Q^*(s(t_{i-1}), T_i)$ is often referred to as the Q value of action T_i in state $s(t_{i-1})$. This recursive form is known as the Bellman equation [8]. The action-value function is important because it is a direct reflection of the optimal policy. That is, the action with the maximal Q value should be the action chosen by the optimal policy. On the other hand, if we know the action-value function, the optimal policy can be easily extracted by acting greedily (i.e., choose the action with the maximal Q value).

Learn policy ϕ'

As an example, we can make use of the DRL framework to learn the policy ϕ'. To this end, the reward and return can be defined as

$$r_i = -\frac{\eta + C_{[t_{i-1}, t_i]}}{T_i}, \qquad R_i = r_i. \tag{3.10}$$

The action-value function is then given by

$$Q^*(s(t_{i-1}), T_i) = \max_{\phi} \mathbb{E}_{\text{env}} \left[r_i \mid \phi\right], \tag{3.11}$$

As shown in Eq. (3.11), the policy ϕ' is myopic and only maximizes the one-step reward. Inspired by (3.8), we can design a farsighted significant sampling policy that considers future rewards, well beyond the immediate one-step future.

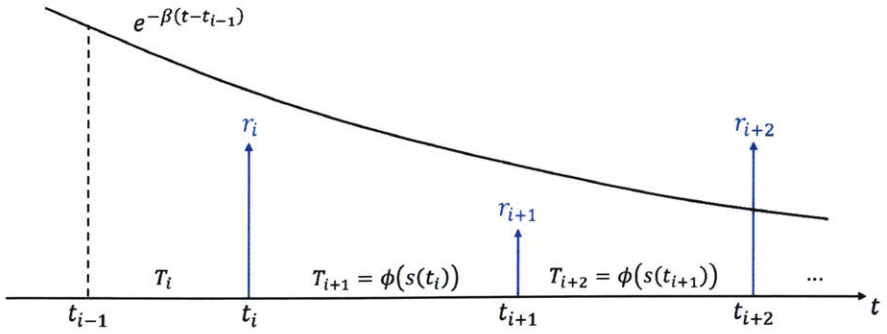


Figure 3-6: Illustration of exponentially discounting reward for significant sampling, in which the dynamics of the environment are continuous.

3.4.2 Action-value Function Design

In standard RL problems, the agent interacts with the environment in uniform time steps, and rewards are received at the end of time steps. The algorithm can then use the (discrete) γ-discounting to define the return as in (3.8). On the other hand, the durations of time-steps in significant sampling vary from time step to time step, and the dynamics of the environment are continuous. To address this issue, we propose a more appropriate exponentially discounting method for the computation of the return, instead of using the discrete γ-discounting. As illustrated in Fig. 3-6, the agent takes action T_i in state $s(t_{i-1})$ and receives the feedback $\{X_n(t) : n \in [N], t_{i-1} \leq t \leq t_i\}$ from the environment at t_i. Let us define

$$\varphi_i(t) = X_{n'_{i-1}}(t) - \min_n X_n(t), \qquad t_{i-1} \leq t \leq t_i,$$

where n'_{i-1} is the index of the shortest path observed at epoch t_{i-1}. The exponentially discounted reward within $[t_{i-1}, t_i]$ can then be defined as

$$r_i = -\eta e^{-\beta T_i} - \int_{t_{i-1}}^{t_i} e^{-\beta(t-t_{i-1})} \varphi_i(t)\, dt \tag{3.12}$$

where $\beta > 0$ is a exponential discount factor. Note that $\varphi_i(t)dt$ is the cost incurred subsequent to t_{i-1} and is therefore discounted according to the time elapsed since t_{i-1}. Likewise, the next sampling cost η is incurred at time $t_i = t_{i-1} + T_i$ and is therefore discounted corresponding to T_i, the time elapsed since t_{i-1}.

Similarly, at t_i, the agent takes action $T_{i+1} = \phi(s(t_i))$ and receives the feedback $\{X_n(t) : n \in [N], t_i \leq t \leq t_{i+1}\}$ from the environment at t_{i+1}. The exponentially discounted reward within $[t_i, t_{i+1}]$ is then given by

$$r_{i+1} = -\eta e^{-\beta T_{i+1}} - \int_{t_i}^{t_{i+1}} e^{-\beta(t-t_i)} \varphi_{i+1}(t)\, dt$$

assuming discounting begins at t_i. Future rewards within each time step can be computed in a similar fashion. Overall, the return due to the action at t_{i-1} can be defined as

$$R_i = r_i + e^{-\beta T_i} r_{i+1} + e^{-\beta(T_i+T_{i+1})} r_{i+2} + \cdots = r_i + \sum_{l=i}^{\infty} e^{-\beta \sum_{l'=i}^{l} T_{l'}} r_{l+1} \tag{3.13}$$

Note that in the above, the discounting of the constituent rewards in R_i begins at time t_{i-1}, hence the additional discounting factors for r_{i+1}, r_{i+2}, ... Essentially, return R_i is the "present value" of all future rewards, where a reward received t time units later is worth only $e^{-\beta t}$ times what it would be worth if it were received immediately. The discounting is introduced to prevent the return from diverging to infinity, as in (3.8).

The action-value function can then be rewritten in the following recursive form:

$$\begin{aligned} Q^*(s(t_{i-1}), T_i) &= \max_{\phi} \mathbb{E}_{\text{env}} \left[R_i | s(t_{i-1}), T_i, \phi \right] \\ &= \mathbb{E}_{\text{env}} \left[r_i + e^{-\beta T_i} \max_{\phi} Q^*(s(t_i), \phi(s(t_i))) \right] \end{aligned} \tag{3.14}$$

Given the Bellman-style equation in (3.14), we can leverage DRL algorithms to learn the action-value function, and extract the corresponding optimal policy in a greedy manner. We shall refer to this new policy, which maximizes the return in (3.13), as the "multi-step look-ahead policy".

Compared with the optimal policy ϕ^* in (3.1), the multi-step look-ahead policy maximizes the sum of the exponentially discounted rewards the agent receives over an infinite time horizon. This is a better approximation to ϕ^* than the one-step look-ahead policy ϕ', especially when the discount exponent β is made to be small.

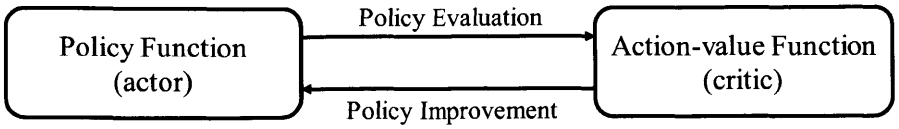


Figure 3-7: RL algorithms as a generalized policy iteration.

3.4.3 An Actor-critic Solution

In RL, the policy function is often referred to as the "actor", as it controls the behavior of the agent; the action-value function is often referred to as the "critic", as it measures the quality of an action. An actor-critic solution for RL problems can be described as a generalized policy iteration (GPI) [8] between the actor and the critic, as illustrated in Fig. 3-7.

The policy function (actor) and action-value function (critic) will be initialized randomly, and their evolution will progress through a series of policy evaluations and policy improvements. Policy evaluation makes the action-value function more consistent with the current policy in that it reflects better the returns produced by the current policy (using Bellman equation). Policy improvement, on the other hand, makes the policy greedier with respect to the current action-value function (using policy gradient [31]). These two processes alternate and progress until the actor converges to the optimal policy, and the critic converges to the optimal Q function.

Deep Deterministic Policy Gradient (DDPG) [32] is an actor-critic approach for MDP with continuous action space. This paper adapts DDPG to solve the RL problem associated with significant sampling. Specifically, we use two deep neural networks (DNNs), an actor network and a critic network, to approximate the policy function and the action-value function, respectively. Fig. 3-8 depicts the logical interaction of the actor and critic networks [33]. The actor, parameterized by λ_a, outputs an action T_i when the network state $s(t_{i-1})$ is used as its input, i.e., $T_i = \Psi_a(s(t_{i-1}); \lambda_a)$. The critic network, parameterized by λ_c, outputs a Q-value estimation $Q(s(t_{i-1}), T_i)$ when the state-action pair, $(s(t_{i-1}), T_i)$, is used as its input, i.e., $Q(s(t_{i-1}), T_i) = \Psi_c(s(t_{i-1}), T_i; \lambda_c)$.

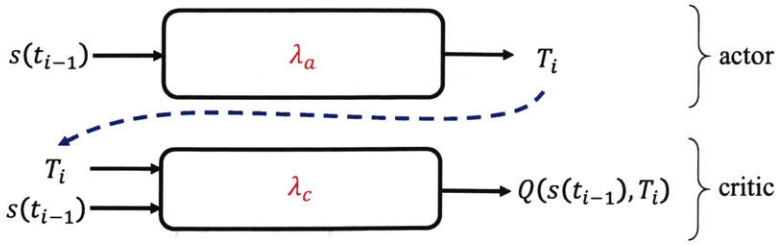


Figure 3-8: The neural networks in actor-critic RL. The actor approximates the policy function, and the critic approximates the optimal action-value function.

These two DNNs are initialized randomly. As learning continues, we use the accumulated experiences to train both DNNs, as follows [33]. Suppose we have an experience $\{s(t_{i-1}), T_i, r_i, s(t_i)\}$.

- The critic is updated using the Bellman-style equation in (3.14). Specifically, given the state-action pair $\{s(t_{i-1}), T_i\}$, we update parameters λ_c in the direction that minimizes the mean square error (MSE) loss between the target Q value $r_i + e^{-\beta T_i}\Psi_c(s(t_i), \Psi_a(s(t_i); \lambda_a); \lambda_c)$ and the Q value $\Psi_c(s(t_{i-1}), T_i; \lambda_c)$ estimated by the current critic network:

$$-\nabla_{\lambda_c}\mathcal{L}_c = -\nabla_{\lambda_c}\left[r_i + e^{-\beta T_i}\Psi_c(s(t_i), \Psi_a(s(t_i); \lambda_a); \lambda_c) - \Psi_c(s(t_{i-1}), T_i; \lambda_c)\right]^2 \tag{3.15}$$

  Updating λ_c in the negative gradient direction given by (3.15) makes the output of the critic network approach the target Q value. In turn the critic network approximates the Q function more closely, resulting in more accurate policy evaluations.

  In practice, directly using (3.15) to update the critic usually leads to divergence of the Q function approximation. This is because the target Q value and the estimated Q value in (3.15) are highly correlated (as they are computed using the same critic network – note: the second term in the target Q is computed using the current critic network as well). To tackle this instability, people often use separate actor and critic networks (as opposed to using the current actor and critic networks in the second component of the target Q above) to compute the target Q value (see (3.17) below). This is known as "fixed Q-target" [28].

- The actor is trained using policy gradient. That is, for state $s(t_{i-1})$, we update parameters λ_a in the direction that maximizes the action-value function (i.e., the output of the critic):

$$\nabla_{\lambda_a} Q^{\lambda_c} = \nabla_{\lambda_a} \Psi_c(s(t_{i-1}), \Psi_a(s(t_{i-1}); \lambda_a); \lambda_c). \tag{3.16}$$

  Updating λ_a in the gradient direction given by (3.16) makes the actor greedier with respect to the current critic. This is a process of policy improvement.

Following the practice of DQL [28] and DDPG [33], we also employ the ideas of experience replay in the implementation. The goal of experience replay is to remove the temporal correlations between successive experiences, such that the experiences used for training are less dependent. To this end, instead of learning experiences online, we put the accumulated experiences in a FIFO buffer of size M (experiences will be placed into and removed from this FIFO buffer continuously), and will randomly sample experiences from the FIFO to train the DNNs. The following section will provide a detailed exposition of the algorithm as well as the relevant parameters.

3.4.4 Algorithm

The pseudocode for the actor-critic solution to significant sampling is given in Algo. 1.

Initialization

As shown, we start by initializing a FIFO buffer of size M (for experience replay); the actor and critic networks with parameters λ_a and λ_c; and the target actor and critic networks with parameters $\lambda_{a'}$ and $\lambda_{c'}$ (for calculating target Q value). In particular, the parameters of target networks are initialized to $\lambda_{a'} = \lambda_a$ and $\lambda_{c'} = \lambda_c$. The initial sampling time is set to t_0, where $s(t_0)$ is given.

Algo. 1: Significant Sampling with Deep Reinforcement Learning.

Initialization:

Initialize a FIFO of size M for experience replay.
Initialize the actor and critic networks randomly with parameters λ_a and λ_c.
Initialize the target actor and critic networks with parameters $\lambda_{a'}$ and $\lambda_{c'}$.
Let $\lambda_{a'} = \lambda_a$ and $\lambda_{c'} = \lambda_c$.
Set mini-batch size to K, evaluation cycle to B.
Set time step $i = 1$, training step $j = 1$.
Set resampling window to t_w, the number of artificial experiences to V.
Set parameter G.
In epoch t_0, all the nodes report their weights, obtain $s(t_{i-1})$.

while 1 **do**

New experience:

Feed $s(t_{i-1})$ into the actor network for action $T_i = \Psi_a\left(s(t_{i-1}); \lambda_a\right)$.
In epoch $t_i = t_{i-1} + T_i$:
All the nodes report their weights in $t \in [t_{i-1}, t_i]$.
Compute reward r_i by Eq. (3.12), and store experience $\{s(t_{i-1}), T_i, r_i, s(t_i)\}$ into FIFO.
$i = i + 1$

Artificial experiences:

$v = 1$.
while $v \leq V$ **do**
Select a random starting epoch t'_{v-1} in $[t_i - t_w, t_i]$,
Feed state $s(t'_{v-1})$ into the actor network for $T'_v = \Psi_a\left(s(t'_{v-1}); \lambda_a\right)$.
if $t'_{v-1} + T'_v < t_i$ **then**
Compute reward r'_i by Eq. (3.12).
Store experience $\{s(t'_{i-1}), T'_i, r'_i, s(t'_i)\}$ into FIFO.
$v = v + 1$.
end
end

DNN training:

if $i * (V + 1) > M/5$ **then**
Randomly sample G mini-batches from FIFO, a mini-batch has K experiences.
for each mini-batch **do**
Update parameters λ^c (critic) in the direction given by Eq. (3.17).
Update parameters λ^a (actor) in the direction given by Eq. (3.18).
Update the target networks by Eq (3.19).
end
if $mod(j, B) == 0$ **then**
Evaluate the performance of current actor (policy) network on the evaluation trajectory.
end
$j = j + G$.
end
end

Experience collection

The next phase is "experience collection". At the beginning of a time step (i.e., at t_{i-1}), the agent feeds the current observation $s(t_{i-1})$ into the actor network, which outputs action T_i. Note that this action is bad early on, but will improve as learning progresses (to guarantee exploration, Gaussian noise is added to the raw output of the actor [33]). At the end of the time step (i.e., at t_i), the agent computes a reward r_i from the environment's feedback (see (3.12)). Overall, one time step provides the agent with one experience $\{s(t_{i-1}), T_i, r_i, s(t_i)\}$. All experiences will be stored in the FIFO for experience replay.

A challenge in learning the optimal policy for significant sampling is the inefficiency with which experiences are collected. Note that, one time step lasts for T_i time units, yet it gives the agent only one experience, hence is quite inefficient. This raises the following question: besides the collected experience, is there a way for the agent to generate more experiences artificially? We answer this question affirmatively by noting that in each time step, the continuously evolving weights $\{X_n(t) : n \in [N]\}$ in interval $[t_{i-1}, t_i]$ are collected and reported to the NMC. As a result, at epoch t_i, all past weights $\{X_n(t) : n \in [N], t \leq t_i\}$ are in fact known to the agent.

In this context, besides this new experience, the agent will resample the past and generate *artificial experiences*. Specifically, each time a new experience is collected, the agent will randomly select V starting epochs in period $[t_i - t_w, t_i]$, where t_w is a resampling window[5]. For each starting epoch $\{t'_{v-1} : v \in [V]\}$, the agent will feed $s(t'_{v-1})$ into the current actor network for action T'_v. It can then compute reward r'_v which constitutes an artificial experience $\{s(t'_{v-1}), T'_v, r'_v, s(t'_v)\}$. This enables the agent to collects $V + 1$ experiences during each time step, V of which are artificial.

[5]We have set t_w to a large value (e.g., 2000 time units, much larger than the duration of one time step) because initially we will be considering stationary environments. For non-stationary environments, the choice of t_w depends on the speed at which the environment varies, because new environment requires the agent to "forget" about the past and "relearn" from the latest observations. More discussions on this topic can be found in Section 3.5.

DNN training

When the number of experiences is larger than $M/5$ (M is the buffer size), the agent starts to train the DNNs by mini-batch gradient descent [8]. We will randomly sample G mini-batches from the FIFO, with each mini-batch containing K experiences. For each mini-batch, we have $\{s(t_{k-1}), T_k, r_k, s(t_k)\}, k \in \{i_1, i_2, \ldots, i_K\}$,

- Update parameters λ_c in the direction that minimizes the mean square error loss

$$-\nabla_{\lambda_c} \frac{1}{K} \sum_{k \in \{i_1,\ldots,i_K\}} \left[r_k + e^{-\beta T_k} \Psi_{c'}(s(t_k), \Psi_{a'}(s(t_k); \lambda_{a'}); \lambda_{c'}) - \Psi_c(s(t_{k-1}), T_k; \lambda_c) \right]^2 \tag{3.17}$$

  Unlike (3.15), the target Q value in (3.17) is computed using a target actor network and a target critic network.

- Given state $s(t_{i-1})$, we update parameters λ_a in the direction that maximizes the output of the critic (i.e., the Q value) by sampled policy gradient, giving

$$\begin{aligned} \nabla_{\lambda_a} Q^{\lambda_c} &\approx \frac{1}{K} \sum_{k \in \{i_1,\ldots,i_K\}} \nabla_{\lambda_a} \Psi_c \left(s(t_{k-1}), T; \lambda_c \right) \Big|_{T = \Psi_a(s(t_{k-1}); \lambda_a)} \\ &= \frac{1}{K} \sum_{k \in \{i_1,\ldots,i_K\}} \nabla_T \Psi_c \left(s(t_{k-1}), T; \lambda_c \right) \nabla_{\lambda_a} \Psi_a \left(s(t_{k-1}); \lambda_a \right) \end{aligned} \tag{3.18}$$

- Each time the actor and critic networks are updated, the parameters of target networks will be updated using soft replacement [33]:

$$\lambda_{a'} = \rho \lambda_a + (1 - \rho) \lambda_{a'}, \qquad \lambda_{c'} = \rho \lambda_c + (1 - \rho) \lambda_{c'} \tag{3.19}$$

Performance monitoring

Finally, to monitor and illustrate the continuous improvements of the learned policy, we evaluate the performance of the policy/actor network every B training steps. The evaluation is performed on an independent trajectory of $X_n(t)$ (this evaluation set is completely independent from the sample path of the process that were observed during the training process).

3.5 Experimental Results

This section presents experimental results of the actor-critic DRL solution to significant sampling. Our main goals for this section are threefold: 1) to confirm that our DRL approach can learn the one-step policy ϕ' by adopting a corresponding one-step action-value function; 2) to show that, by adopting a multi-step look-ahead action-value function, our DRL approach can learn the multi-step policy that addresses the "never-sample" problem associated with the policy ϕ', and this policy is more optimal than the policy ϕ' in terms of minimizing long-term cost rate; 3) to show that our DRL approach can provide effective solutions for more general environment with more than two paths, and with weight-evolution processes beyond the OU process.

3.5.1 Learn the One-step Look-ahead Policy ϕ'

The importance of the analytical solutions in (3.3), in the context of this experiment, is that they allow us to benchmark the performance of our DRL solution against the theoretical result. A comparable performance will provide us with the foundation to extend our objective and setup beyond what was presented in Chapter 2 with a reasonable degree of confidence in its ability to perform close to optimal.

Let us begin by assuming the same system setup as in Chapter 2. Specifically, the OD pair is connected via two paths, and the evolving weights of both paths are represented by independent OU processes with the same mean value. Thus, the difference process $X(t)$ is a zero-mean OU process, and we can focus on sampling $X(t)$. The analytically derived policy ϕ' in (3.3) will serve as the benchmark. To learn the one-step look-ahead policy, we can set the action-value function to be (3.11). That is, the objective of the agent is simply to maximize the one-step reward rate. The MSE loss in (3.17) can then be simplified to

$$-\nabla_{\lambda_c}\mathcal{L}_c = -\nabla_{\lambda_c}\frac{1}{K}\sum_{k\in\{i_1,\ldots,i_K\}}\left[\frac{r_k}{T_k} - \Psi_c(s(t_{k-1}), T_k; \lambda_c)\right]^2 \tag{3.20}$$

Compared with (3.17), the target value in (3.20) is simply r_k/T_k, and hence the target networks can be removed in this experiment.

Table 3.1: Hyper-parameter settings for the one-step look-ahead policy.

Description	Symbol	Value
FIFO size	M	32768
Mini-batch size	K	128
Resampling window	t_w	2000
# artificial experiences	V	31
# mini-batches per training	G	5
Maximum sampling interval	$T_{\max}$	40

Since the OU process is Markovian, we can set $s(t_{i-1}) = X(t_{i-1})$, i.e., the state of the environment is a real number. The actor and critic networks are designed to be fully-connected (feedforward) neural networks. The actor network consists of five fully connected layers, and the numbers of neurons in the five layers are $\{1, 32, 64, 32, 1\}$. The three hidden layers use rectifier non-linearity as activation functions, and the output layer uses hyperbolic tangent (tanh) non-linearity as the activation function. The output of the actor network u takes value in $(-1, 1)$ and is mapped to the sampling interval $T_i = T_{\max} * (u+1)/2$, where $T_{\max}$ is the maximum sampling interval. The architecture of the critic network is the same as the actor network except that a) the input layer has two neurons instead of one; b) the output layer uses no activation function, and directly produces the Q value given a state-action pair (See Fig. 3-8 for additional reference).

The hyper-parameter settings are listed in Table 3.1. We used Adam optimizer for training the neural network, and set the learning rates for the actor and critic networks to 0.0005 and 0.001, respectively.

We generate two independent trajectories of $X(t)$ for training and evaluation purposes respectively. As with previous examples, the sampling cost is set to $\eta = 0.1$ and the OU parameters for $X(t)$ are chosen to be $\{\mu = 0, \theta = 0.025, \sigma = 0.5\}$. The duration of the training trajectory is 2×10^5 time units, and the duration of the evaluation trajectory is 2×10^4 time units.

Over the course of training, for every $B = 100$ training steps, we assess the performance of the actor network on the evaluation trajectory. Specifically, let us denote by ϕ_j the sampling policy learned by the actor network after j training steps. We will apply the policy ϕ_j on the whole evaluation trajectory, and compute the cost rate by

$$\overline{c}_j = \frac{\sum_{l=1}^{L_j}(\eta + C_{[t_{l-1},t_l]})}{\sum_{l=1}^{L_j} T_l}, \tag{3.21}$$

where L_j is the number of samples of policy ϕ_j when applied on the whole evaluation trajectory, and the sampling epochs on the evaluation trajectory are $\{t_i : i \in [L_j]\}$.

Fig. 3-9(a) presents the cost rate as a function of training steps in the RL process. As a benchmark, we also plot the minimum achievable cost rate (0.0492) which is obtained by applying the analytically derived one-step look-ahead policy given by (3.3) on the evaluation trajectory. We can see that the cost rate achieved by the learned policy decreases as training continues and eventually approaches the optimal cost. The learned policies at different times of the learning process are plotted in Fig. 3-9(b-e). The x-axis is the observed value of the weight process $X(t)$, and the y-axis is the corresponding action T_i given by the learned policy at that training step. The benchmark, again, is the analytically derived one-step look-ahead policy. As shown, the actor network will learn the optimal policy after enough training steps.

3.5.2 Learn the Multi-step Look-ahead Policy

Subsection 3.5.1 confirms that the actor-critic DRL approach can learn the one-step look-ahead policy derived in Chapter 2. This is an important achievement in and of itself because the DRL system is model-free. Specifically, unlike the analysis giving rise to the analytical solution, the agent in DRL is unaware that the underlying delay model is an OU process and yet "learns" to obtain a near-optimal sampling policy.

This subsection evaluates the multi-step look-ahead policy based on the actor-critic framework. We have repeated the experiment in subsection 3.5.1 assuming a small sampling cost followed by assuming a large sampling cost.

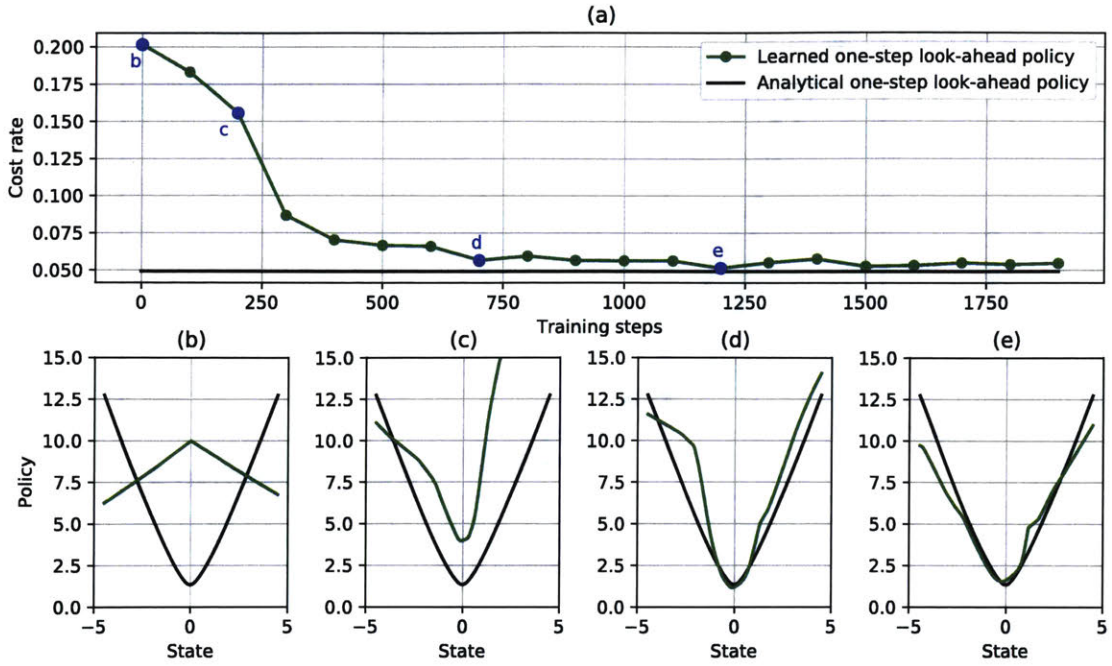


Figure 3-9: The cost rate achieved on the evaluation trajectory, and the policy improvements in the RL process: (a) cost rate achieved on the evaluation trajectory versus the number of training steps; (b) the learned policy of actor at point b; (c) the learned policy of actor at point c; (d) the learned policy of actor at point c; (e) the learned policy of actor at point e.

Fig. 3-10 shows the performance of the one-step and multi-step look-ahead policy when $\eta = 0.1$. The performance curves for one-step look-ahead policy, both analytical and learned, are reproduced from Fig. 3-9 (the difference is that the y-axis is in log scale). As shown, with $\eta = 0.1$, the multi-step look-ahead policy achieves only minor gain over the one-step look-ahead policy. This indicates that the one-step look-ahead policy is a good policy when η is small in terms of minimizing the long-term cost rate. This is not the case, however, for more general η.

In the second experiment, we use a larger sampling cost $\eta = 15$. Notice that $\eta = 15$ falls into the region $12.2 < \eta < 35.68$ given by (3.7) if we set $\varepsilon = 0.1$. Thus, if the absolute value of one sample is less than or equal to 0.1, the one-step look-ahead policy will instruct us to never sample, and the cost rate will eventually converge to $g(\infty) = \sqrt{\frac{\sigma^2}{4\pi\theta}}$.

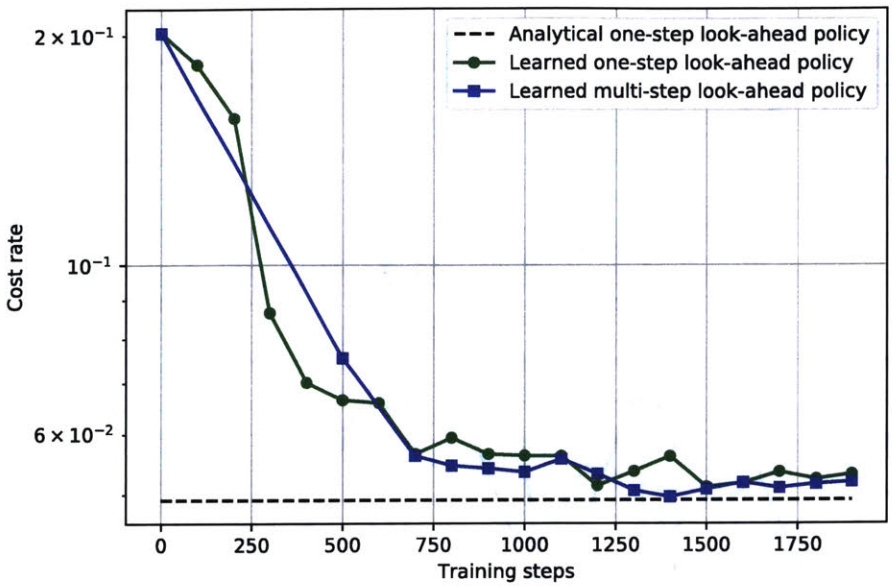


Figure 3-10: The cost rate achieved on the evaluation trajectory when the sampling cost $\eta = 0.1$.

Fig. 3-11 shows the cost rates achieved by the learned one-step and multi-step look-ahead policies. The analytical cost rate for one-step look-ahead policy is $g(\infty) \approx 0.89$ (i.e., the dashed straight line in Fig. 3-11). We note that the learned one-step look-ahead policy is better than the analytical results. This is because an "artificial" maximum sampling interval $T_{\max}$ is set in the implementation (recall that the output of the actor DNN is in $[0, T_{\max}]$, where $T_{\max} = 120$ in this experiment). Thus, if the absolute value of one sample is less or equal to 0.1, the action chosen by the one-step look-ahead policy is 120 rather than infinity. This is confirmed in Figure 3-12, where the learned policies are plotted. As shown, for the one-step look-ahead policy, the learned policy is consistent with our prediction. When a sampled value is close to 0, the action given by the learned one-step look-ahead policy reaches the maximal sampling interval. On the other hand, for the multi-step look-ahead policy, the learned policy matches with our intuition of a good policy, i.e., when $X(t)$ is close to zero, sampling should be more frequent; when $X(t)$ is far away from zero, sampling should be more sparse. The never-sample problem is well addressed by our multi-step look-ahead policy.

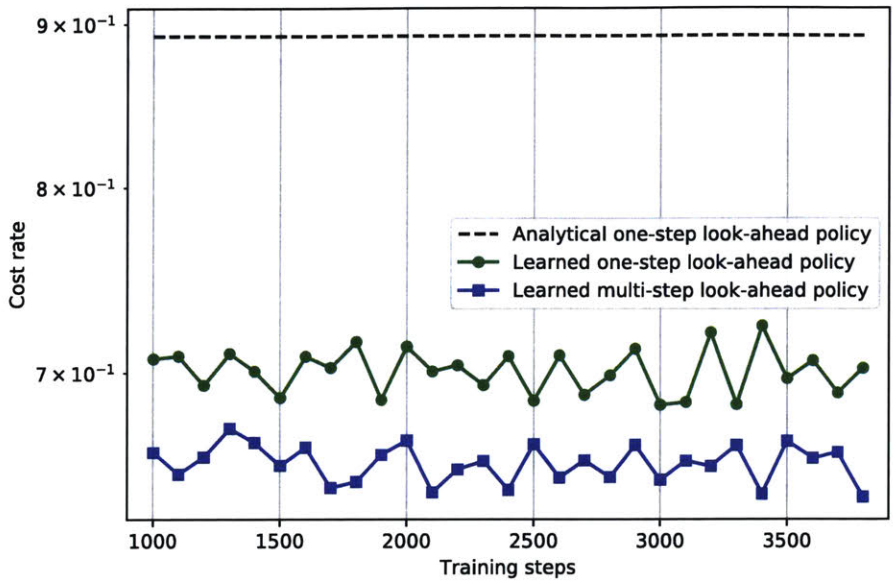


Figure 3-11: The cost rate achieved on the evaluation trajectory when the sampling cost $\eta = 15$.

Overall, the learned multi-step policy achieves the best long-term cost rate. Considering the average cost rate achieved on the evaluation trajectory, our multi-step policy outperforms the analytical one-step policy, and the learned one-step policy by 39% and 8%.

3.5.3 Extensions to More General Environments

Our previous discussions focused on the same system setup as in Chapter 2, i.e., the OD pair is connected via only two paths, and the weight processes of both paths are OU processes. Thanks to the model-free property of RL, an advantage of our DRL approach is that we can provide solutions for more general system setups with only slight modifications to the algorithm. This subsection will study three generalizations to the previously discussed system setup, namely:

- N paths with OU weight processes.
- Two paths with weight processes with memory.
- Two paths with non-stationary weight processes.

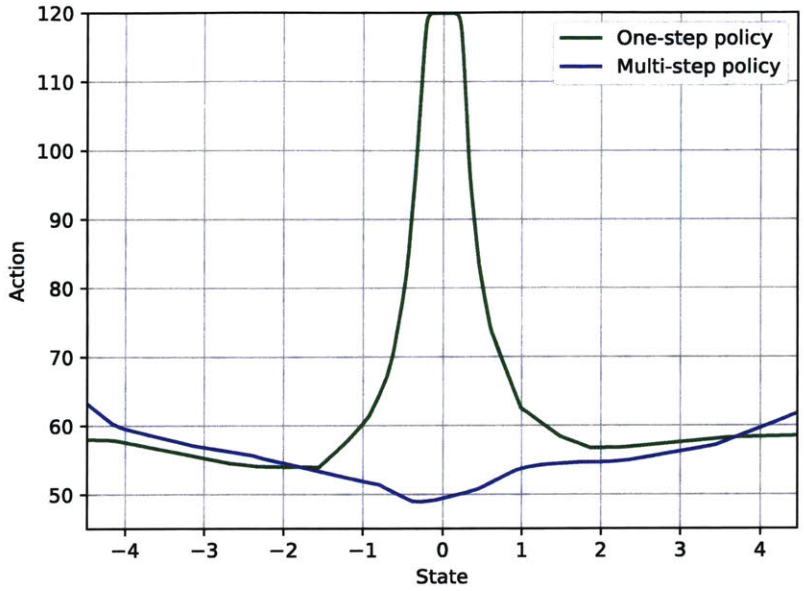


Figure 3-12: The one-step and multi-step look-ahead policies when $\eta = 15$.

N paths with OU weight processes

This experiment removes the $N = 2$ assumption. We assume there are N paths and the evolving weight of each path is an OU process. In particular, we choose $N = 3$ in the experiment.

The state of the environment is defined to be $s(t_{i-1}) = \{X_n(t_{i-1}) : n \in [N]\}$. That is, the state input of the actor and critic networks is a vector of length N. We can then increase the number of neurons in the input layer of the actor and critic networks to N and $N + 1$, respectively. The hyper-parameters are the same as that in Table 3.1. The three OU processes are generated using the same parameter set $\{\mu = 0, \theta = 0.025, \sigma = 0.3\}$, and the sampling cost $\eta = 0.1$. We use the DRL framework to learn the multi-step look-ahead policy in the implementation.

Fig. 3-13 shows the cost rate achieved by the multi-step look-ahead policy on the evaluation trajectory. To evaluate the learning results, we apply the learned policy (after 25000 training steps) on a separate realization of $X_n(t)$, and plot the sampling results in Fig. 3-14. The learned policy matches our intuition of a good policy: it samples more frequently when the values of $\{X_n(t) : n \in [3]\}$ are close, and more sparsely when one of the three weights is much smaller than the other two paths.

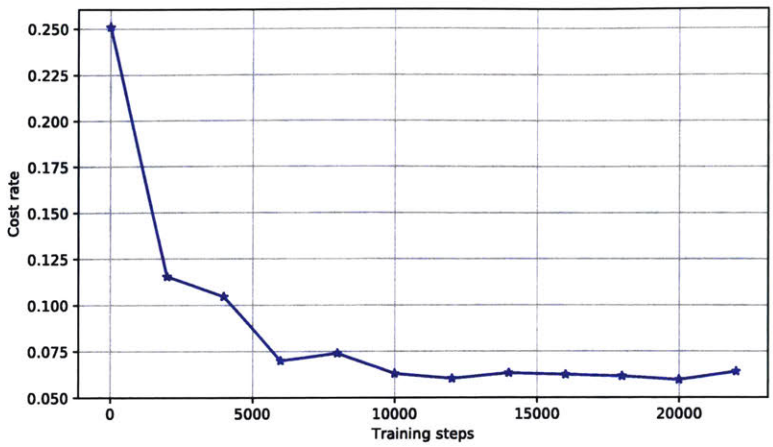


Figure 3-13: The cost rate achieved on the evaluation trajectory in the RL process. There are three paths, and we use the DRL framework to learn the multi-step look-ahead policy.

Two paths with weight processes with memory

In previous sections, we showed that the DRL framework can solve the significant sampling problem when the underlying weight process is taken to be an OU process. A favorable property of the OU process is that it is Markovian. Evolution of such an environment is completely captured through the latest sample, and hence we can define the state of the environment as $s(t_{i-1}) = \{X_n(t_{i-1}) : n \in [N]\}$.

Unlike the OU process, real-world weight processes may exhibit complex inter-dependencies over time and have memory. This suggests that optimal sampling of such processes depends not only on the latest sample, but also on the history of the process. To address this issue, one may define the state of the environment as $s(t_{i-1}) = \{X_n(t) : n \in [N], t \leq t_{i-1}\}$, that is, we feed the full history of the weight process into the DNNs for decision making. However, such an approach will require an incredibly large and potentially infinite state space due to the continuous nature of $X_n(t)$, and is thus impractical. Hence, we seek to reduce the state space required for the optimal sampling of general processes.

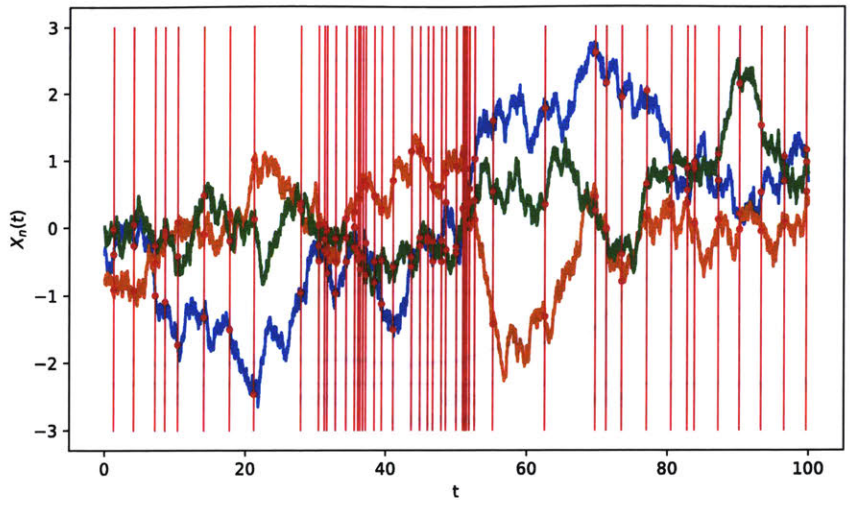


Figure 3-14: The result of applying the learned policy (after 25000 training steps) on a realization of $\{X_n(t) : n = [3]\}$.

This section combines the following two schemes to achieve this goal:

a) We characterize each weight process, $X_n(t)$, in the frequency domain, and use the ledger to represent $X_n(t)$ as a real vector obtained by uniformly sampling the process (as per Nyquist sampling theorem [34]). This allows us to reduce/compress the information contained in the continuous process/signal.

b) We assume the decision depends only on a finite period of recent history, and truncate early uniformly sampled points before this period.

Denote by $\boldsymbol{X}(t_{i-1}) = \{X_n(t_{i-1}) : n \in [N]\}$ the N samples at a sampling epoch t_{i-1}. The state of the environment can then be defined as,

$$s(t_{i-1}) = \left\{ \boldsymbol{X}\left(t_{i-1} - \frac{L-1}{f_{\max}}\right), \boldsymbol{X}\left(t_{i-1} - \frac{L-2}{f_{\max}}\right), \ldots, \boldsymbol{X}\left(t_{i-1} - \frac{1}{f_{\max}}\right), \boldsymbol{X}(t_{i-1}) \right\}, \tag{3.22}$$

where $f_{\max}$ is the uniform sampling rate, and the history before $t_{i-1} - (L-1)/f_{\max}$ is considered to be irrelevant to the decision making agent. In a nutshell, state $s(t_{i-1})$ is composed of samples at the L most recent sampling epochs, if we sample the history of $X_n(t)$ uniformly at rate $f_{\max}$.

The uniform sampling rate $f_{\max}$ is designed to be $f_{\max} = \max\{f_1, f_2, \ldots, f_N\}$, where f_n is the uniform sampling rate for the n-th path. To determine f_n, we first transform $X_n(t)$ to the frequency domain. If $X_n(t)$ is band-limited, f_n can be set to the Nyquist rate of $X_n(t)$ (twice its bandwidth). If $X_n(t)$ is not band-limited as are most real-world signals/weight processes, we will pass $X_n(t)$ through a low-pass filter and truncate high frequency components in the stopband of the filter (empirically, the spectrum of most practical process centers around DC, and progressively gets smaller and smaller as frequency increases). Then, the filtered $X_n(t)$ is a band-limited signal, and we set f_n to be its Nyquist rate.

To confirm the proposed schemes are effective, we conduct the following experiment, in which an artificial weight process with memory is used as input. In this example, we assume the OD pair is connected via two paths, and we focus on sampling the difference process $X(t)$. In particular, we artificially generate $X(t)$ by adding a periodicity (i.e., a rectangular wave) to an OU process,

$$X(t) = \mathrm{OU}(t) + \sqrt{\frac{2\sigma^2}{\theta}} \sum_i \mathrm{Rect}(t - iD_R), \tag{3.23}$$

where $\mathrm{OU}(t)$ is an OU process parameterized by $\{\mu = 0, \theta = 0.025, \sigma = 0.5\}$; the rectangular wave $\mathrm{Rect}(t) = \mathrm{sgn}(t), -D_R/2 \leq t < D_R/2$ (D_R is the duration of one period, we set $D_R = 20$ in the experiment). The rectangular wave is scaled by the two standard deviations of the OU process. A realization of $X(t)$ is shown in Fig. 3-16. Compared with OU process, this new process undergoes sudden changes, and hence is challenging for the DRL framework to learn a good sampling policy[6].

To learn a good policy for the weight process in (3.23), the first step is to determine an uniform sampling rate as in (3.22). Following scheme a), we set the uniform sampling rate $f_{\max} = 1$ sample per unit time (sampling at this rate, the dominant spectrum of $X(t)$ has already been captured). As a result, at an epoch t_{i-1}, we will uniformly sample the history of $X(t)$, and set the state of the environment at t_{i-1}

[6]We successfully experimented with various processes with memory, including ARMA, ARIMA, filtered OU, etc. but the process in Eq. (3.23) is a good candidate to showcase the ability of our algorithm, as the sudden changes of the rectangular wave present a challenging environment.

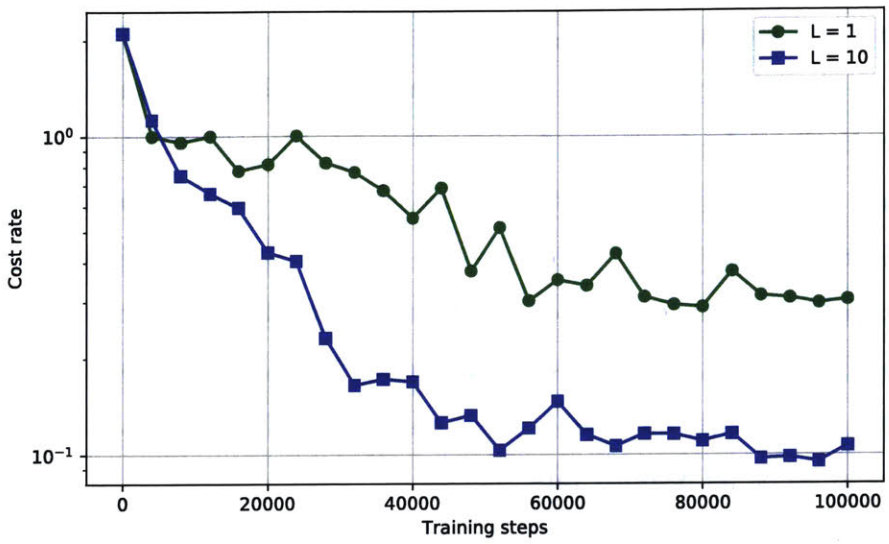
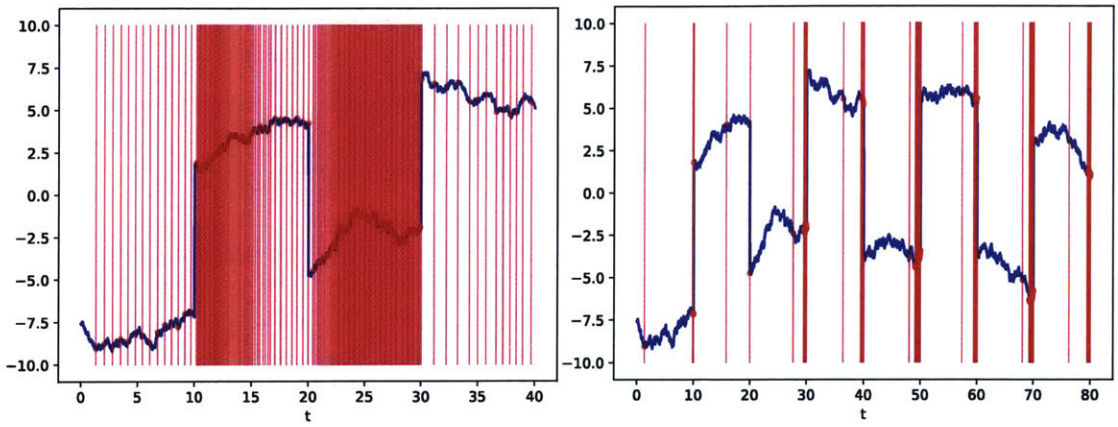


Figure 3-15: The cost rate achieved on the evaluation trajectory given different input length L.

to be a collection of L most recent uniform samples, i.e., $s(t_{i-1}) = \{X(t_{i-1} - l), l = L-1, L-2, \ldots, 0\}$. We will monitor the performance of the learned policy for different values of L.

In the experiment, the sampling cost is set to $\eta = 0.1$, and we let the DRL framework learn the multi-step look-ahead policy. All experimental setups are the same as Section 3.4 except for the DNN architecture. The shape of the actor is now $\{L, 64, 128, 64, 32, 1\}$, where the L input neurons are used to represent state $s(t_{i-1})$. The shape of the critic is $\{L + 1, 64, 128, 64, 32, 1\}$.

First, we set $L = 1$. This means the actor network needs to determine the optimal sampling interval T_i from state $s(t_{i-1}) = X(t_{i-1})$. We then assess the learned policy on an evaluation trajectory every $B = 4000$ training steps, and plot the achieved cost rate in Fig. 3-15. Then, we increase L. The final performance of the learned policy improves as L increases until $L = 10$. No further improvements were observed for $L > 10$. Note that when $L = 10$, the history captured in state $s(t_{i-1})$ is half the period of the rectangular wave. Overall, choosing $L = 10$ helps improve the average cost rate (at convergence) by a factor of 3.2 in comparison to when $L = 1$.



(a) $L = 1$ (b) $L = 10$

Figure 3-16: Evaluation of the learned policy when $L = 1$ and $L = 10$.

To evaluate the results, we apply the learned policy (after 80000 training steps) on a realization of $X(t)$, and plot the sampling results in Fig. 3-16. Note that the learned policy performs really poorly when $L = 1$, as captured by the large number of unnecessary samples depicted in the figure. In this scenario, the actor network is unable to make a good decision given only the latest observation of $X(t)$. On the other hand, when $L = 10$, the learned policy samples the weight process mostly at the half-cycle of the rectangular wave where the sign of the rectangular wave changes.

Two paths with non-stationary weight processes

In previous sections, we extended our results by successfully applying the DRL framework to a network with more than two paths, as well as a network whose underlying weight process was non-Markovian and exhibited memory. In this section we will show that the DRL framework can also be used in networks that exhibit a non-stationary weight process. This is a particularly important capability because realistic and robust systems should be able to identify and adapt to various operating regimes. More explicitly, we will show that our DRL solution can automatically adjust the sampling policy and adapt to new environments in a reasonable amount of time.[7]

[7]Let us define the phrase "transitional period" to denote the amount of time it takes for a given algorithm to adapt to a new environment. There are many ways to make this definition rigorous, but for simplicity we shall use it to mean the amount of time it takes for the achieved cost to reach within 10% of the optimal cost.

Table 3.2: Hyper-parameter settings for the non-stationary environment.

Description	Symbol	Value
FIFO size	M	10250
Mini-batch size	K	128
Resampling window	t_w	1000
# artificial experiences	V	1024
# mini-batches per training	G	20

There are two important indicators when we consider the performance of a controller in a non-stationary environment. First, we should ensure that the system can operate optimally in a wide range of environments and thus, can accommodate various network conditions. Second, we prefer a controller that can quickly transform its policy based on its perception of the environment. Unfortunately, these two goals are often at odds with one another. Said another way, an agent's ability to identify the optimal action depends on its ability to maintain and analyze past behaviors, on the other hand, adapting to a new environment requires the agent to "forget" the past states and "relearn" from the latest observations, and thus there is a clear tradeoff between optimality and adaptability. The size of the FIFO M as well as the resampling window t_w are the critical factors with respect to this tradeoff.

Given Algorithm 1, we evaluate the agent's ability in adapting to a new environment by observing the behavior of the agent when the underlying weight process experiences a sudden change. To simulate this sudden change, the training data was generated by concatenating two independent sample paths each of which corresponds to an OU process with a unique parameter set. The first portion of the training data has a duration of $D_1 = 3000$ and is a realization of an OU process with parameter set $\{\mu = 1, \sigma = 0.5, \theta = 0.025\}$. The second portion of the training data has a duration $D_2 = 2000$ and is a realization of an independent OU process with parameter set $\{\mu = 0, \sigma = 1, \theta = 0.025\}$. The hyper-parameter settings for this experiment is given in Table 3.2, and a sampling cost of $\eta = 0.1$ was assumed.

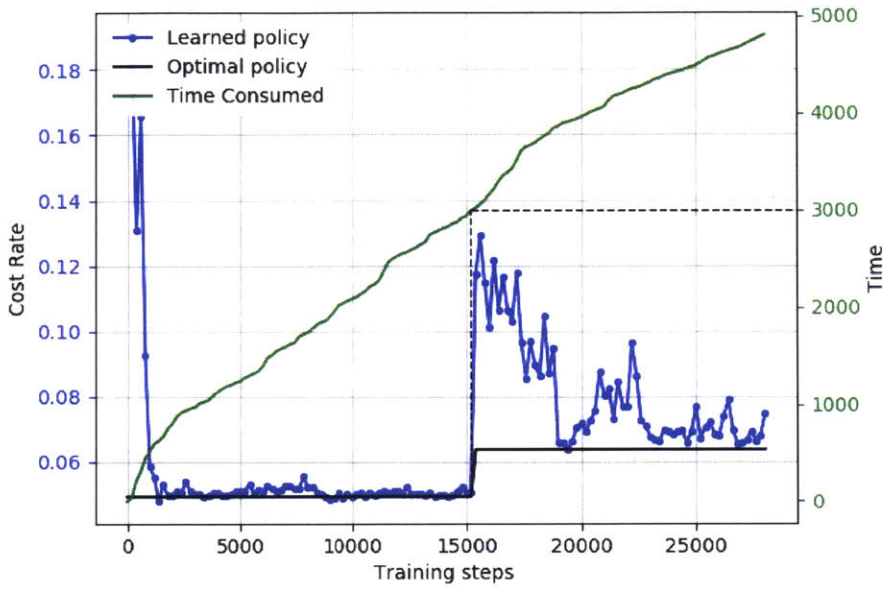


Figure 3-17: The cost rate on the evaluation trajectory of a non-stationary process. The black line corresponds to the cost of an optimal policy.

To evaluate the performance of the actor network fairly, we have employed different evaluation data for different periods of the learning process. Specifically, in the first $D_1 = 3000$ time units, we use an evaluation trajectory generated by the first parameter set, and in the remaining $D_2 = 2000$ time units, we use another evaluation trajectory generated by the second parameter set. Both evaluation trajectories have a duration of 20000 time units. Over the course of training, for every $B = 200$ training steps, we assess the actor network on the evaluation trajectory.

Fig. 3-17 presents the cost rate achieved on the evaluation trajectory. Let us first focus on the required time curve (the second y-axis on the right). The agent starts to collect and resample experiences from $t = 0$. The first period is from $t = 0$ to $t = 3000$ (i.e., training steps 0 to $\sim$ 15000), where the agent learns to adapt to the OU process generated by the first parameter set. The second period is from $t = 3000$ to $t = 5000$ (i.e., training steps $\sim$ 15000 to the end), where the agent learns to adapt to the OU process generated by the second parameter set.

In the first period, the actor network takes approximately 1200 training steps to approach the optimal cost (the cost yielded by the optimal policy which is depicted as a black line in the figure). As training progresses, the realized cost oscillates near the optimal cost. Then, after 3000 time units (corresponding to approximately 15000 training steps), the underlying process changes abruptly. In particular, the σ of the process is doubled, hence a more frequent sampling policy is needed. Clearly, the policy learned during the first period performs poorly when applied to the second process, as shown by the sudden increase in the cost of the learned policy. Moreover, the poor performance persists for nearly 1000 time units. This is because in interval $t \in [3000, 3000 + t_w] = [3000, 4000]$, the agent will still resample experiences from the first period. These bad experiences will contribute negatively to the training of DNNs and hence prolong the transitional period. Overall, various experiments have shown that the transitional period is nearly equal to t_w. After $t = 3000 + t_w$, the system has nearly reached the optimal cost and will once again oscillates near this value. A simple solution to alleviate the oscillations is to increase the resampling window, or increase the buffer size. However, both solutions can further prolong the transitional period. As discussed earlier there is a fundamental tradeoff between optimality and quick adaptability.

In conclusion, we can see that the DRL framework enables the system to adapt to distinct operational regimes in a reasonable amount of time.

3.6 Discussion and Conclusion

We started this thesis by noting that any high-performance, resilient, and scalable networking platform should be able to monitor the state of various elements within the network and reconfigure them as needed. We also emphasized that traditional methods for implementing such a capability would be particularly costly especially for highly dynamic networks. We then suggested the notion of significant sampling, which allows a centralized network management and control system to sample the state of a given element if and only if this information is of significant value to the operation of the network. In reality we should adopt a wide interpretation of the sampling operation to include a process of collecting network state information (NSI), identifying the shortest path(s) for different origin-destination (OD) pairs, and disseminating routing information to the networks.

A robust and adaptive sampling policy specifies the next sampling time based on the cognitive understanding of the network states at this moment. Within the narrower context of shortest path routing, the goal of significant sampling is to discover the optimal sampling policy that effectively balances the cost of sampling and the cost of error associated with mis-identifying the shortest path until the next sampling time. Of course in real applications, monitoring and identifying the shortest path through significant sampling should run on top of other algorithms that determine network connectivity, and we have assumed that the underlying processes and algorithms for that are already in place.

In Chapter 2, we focused on obtaining analytical results that could confirm the benefits of such an approach. In order to get a tractable analytical solution, we made a few (potentially unrealistic) simplifying assumptions and studied what may be considered the simplest possible network consisting of an origin-destination pair connected via two independent paths. We ultimately decided that the delay on each path can be modeled as an Ornstein-Uhlenbeck process; The unique non-trivial stochastic process that is simultaneously Gaussian, Markov, and stationary. We should also note that the presented model is inherently focused on situations were we are changing

routes that carry reasonably small traffic. This is important because if/when traffic is rerouted it should be a minor part of the total traffic on the link, so as to make the OU-process a good model for the system. Despite the aforementioned assumptions, shortcomings, and constraints, we showed that significant sampling, if implemented properly, is able to provide enormous value to the performance of management systems, and significantly reduce the overhead associated with the monitoring process.

The ideas in this chapter have been focused on the possibility of creating a more autonomous network management and control system that can accomplish the underlying goals of significant sampling in a wide variety of network conditions that may no longer satisfy the aforementioned assumptions. Of course, we would like to verify that the behavior of this system, when faced with the environment described in Chapter 2, is similar to the corresponding analytical results. Hence, the importance of the result in Chapter 2 is in its ability to provide us with a benchmark to test our new system.

We accomplished this task by presenting a deep reinforcement learning (DRL) solution to the significant sampling problem. Modeling the problem as a Markov Decision Process (MDP), we treated the NMC system as an agent that samples the state of various network elements in order to identify the shortest path(s) and make optimal decisions on the sampling frequency. The agent periodically receives a reward commensurate with the quality of its actions. The decision on when to sample will progressively improve as the agent learns the relationship between the sampling frequency and the reward function.

Benchmarked against the policy ϕ' derive in Chapter 2, our DRL solution showed its ability to learn the target policy without a need for an explicit knowledge of the traffic model or its parameters. In addition, we designed a new multi-step look-ahead policy to address the never-sample problem faced by policy ϕ'. Experimental results showed that average cost rate of policy ϕ' is 1.39 times of the DRL policy. Another advantage of our DRL solution is that it is robust to various network conditions and stochastic variations of traffic, thanks to the model-free property of RL. We also presented three extensions to the original system setup. In one extension, we

considered the case of an N-path OU process. In the other extension, we considered a much more complex traffic model that is a mixture of OU process and periodic rectangular wave. Our final extension considered a non-stationary environment where the agent has to relearn and adapt to new environments. We confirmed that our DRL approach can provide good solutions to all extensions.

Before concluding our remarks, we would like to discuss a few issues regarding the use of neural-networks, and deep leaning models (including DRL) for real life applications. Deep learning has experienced an explosive growth and adoption in the last decade, and we expect to see many more remarkable achievements in the years to come. As you may imagine and despite the current hype, deep learning is not a panacea and we should be careful as we incorporate it into real world applications; especially those that are critical to the well-being of the general public. One of the biggest shortcomings in the current stage of deep learning research is its black-box nature and lack of interpretability that has beleaguered many practitioners. This has created a rather unsettling situation for many of us who are interested in understanding the inner-workings of a system or the fundamental aspects of various scientific or mathematical phenomena. New initiatives such as Explainable AI, and Interpretable AI are hoping to address some of these issues but we seem to be far from a universally acceptable solution.

This lack of interpretability has also impacted the natural way researchers develop and deploy new systems. A typical development cycle for any deep-learning application involves an extended period whereby the output/performance of the system is far from expectations, yet the neural network cannot be logically decomposed into smaller components in such a way as to aid in the discovery of the problematic component. Of similar or even higher importance is when we identify a problematic/unexpected behavior in a deployed system. In such scenarios, many other systems may have been designed to work with this system and depend on it for their operation, and once again lack of interpretability will result in a slow and arduous diagnosis process.

As another specific challenge in this area, we should mention the tuning of the hyper-parameters which is more of an art that a science. There are many instances where a system fails to converge to a stable policy simply because the hyper-parameters are off by a small amount. Many a trial and errors may be required to identify the right set of hyper-parameters and achieve satisfactory performance.

Another shortcoming of deep learning solutions is that they are very sensitive to small variations of the input. In other words, a small variation in the input may lead to drastically different outputs. This is particularly problematic in adversarial settings where a small and potentially imperceptible amount of noise can have catastrophic results. In fact, we experienced some version of this phenomenon in our work as well. There were instances were an atypical input (i.e. an input that should be statistically rare) would be observed by the system and would negatively impact the sampling policy.

In conclusion, we wish to convey that deep-learning models should be used with abundance of caution and only in situations were the consequences of mis-behavior are minimal. One way to ensure safe deployment of deep learning systems would be to have a simple fail-safe mechanism whose behavior is fully understandable to human operators. For example, we may wish to ensure that the delay on a given link does exceed a certain threshold, a task that was originally assigned to a DRL agent. We can put a fail-safe mechanism that monitors this link and if the threshold is crossed, it will take over the operation of the link and potentially only allow high priority traffic to go through. Such techniques will act as a guard rail to ensure that the behavior of the DRL agent is within a reasonable range and major surprises can be avoided.

Appendix

3.A The One-step Look-ahead Policy

This Appendix proves Proposition 1 and 2.

Suppose $X(t)$ is a zero-mean OU process parameterized by $\{\mu = 0, \theta, \sigma\}$. Without loss of generality, we sample $X(t)$ at time $t = 0$, and obtain $X(0) = \varepsilon$. Then, following the one-step look-ahead policy ϕ' in (3.3), the optimal sampling interval T^* can be obtained from

$$T^* = \arg\min_{T>0} \frac{\eta + \mathbb{E}[C_{[0,T]}|X(0) = \varepsilon]}{T} \triangleq \arg\min_{T>0} h(T)$$

where $C_{[0,T]}$ is the accumulated cost of error in $[0, T]$. Set $\frac{dh(T)}{dT} = 0$, we have

$$\begin{aligned} \frac{dh(T)}{dT} &= \frac{d\,\mathbb{E}[C_{[0,T]}|X(0) = \varepsilon]}{dT} T - (\eta + \mathbb{E}[C_{[0,T]}|X(0) = \varepsilon]) = 0 \\ \eta &= \frac{d\,\mathbb{E}[C_{[0,T]}|X(0) = \varepsilon]}{dT} T - \mathbb{E}[C_{[0,T]}|X(0) = \varepsilon] \end{aligned} \tag{3.24}$$

$$\tag{3.25}$$

To compute $\mathbb{E}[C_{[0,T]}|X(0) = \varepsilon]$, we can write $C_{[0,T]}$ as $C_{[0,T]} = \int_0^T \varphi(t)dt$, in which $\varphi(t)$ is the instantaneous cost of error at epoch $t \in [0, T]$, and is given by $\varphi(t) = -X(t)\mathbb{1}_{(-\infty,0)}(X(t))$ (that is, when $X(t) \leq 0$, $\varphi(t) = -X(t)$; else, $\varphi(t) = 0$).

Thus,

$$\mathbb{E}[C_{[0,T]}|X(0)=\varepsilon] = \int_0^T \mathbb{E}[\varphi(t)|X(0)=\varepsilon]\,dt \triangleq \int_0^T g(t)\,dt$$

and (3.24) can be refined as

$$\eta = \frac{d\int_0^T g(t)\,dt}{dT}T - \int_0^T g(t)\,dt = Tg(T) - \int_0^T g(t)\,dt \tag{3.26}$$

We then compute $g(t)$. Given $X(0)=\varepsilon$, we have

$$X(t) = \varepsilon e^{-\theta t} + \frac{\sigma}{\sqrt{2\theta}}e^{-\theta t}W(e^{2\theta t}-1), \quad t>0, \tag{3.27}$$

where $W(e^{2\theta t}-1)$ is a time-scaled Winner process, $W(e^{2\theta t}-1) \sim \mathcal{N}(0,\, e^{2\theta t}-1)$. This means $X(t) \sim \mathcal{N}(\mu_t,\, \sigma_t^2)$ at time $t>0$, where $\mu_t = \varepsilon e^{-\theta t}$ and $\sigma_t^2 = \frac{\sigma^2}{2\theta}(1-e^{-2\theta t})$. Function $g(t)$ can then be computed as

$$\begin{aligned} g(t) &= \int_{-\infty}^{0} -x f_{X(t)}(X(t)=x)dx = \frac{\sigma_t}{\sqrt{2\pi}}e^{-\frac{\mu_t^2}{2\sigma_t^2}} - \frac{\mu_t}{2}\operatorname{erfc}\left(\frac{\mu_t}{\sqrt{2\sigma_t^2}}\right) \\ &= \sqrt{\frac{\sigma^2(1-e^{-2\theta t})}{4\pi\theta}}e^{-\frac{\theta\varepsilon^2}{\sigma^2(e^{2\theta t}-1)}} - \frac{\varepsilon e^{-\theta t}}{2}\operatorname{erfc}\left(\sqrt{\frac{\theta\varepsilon^2}{\sigma^2(e^{2\theta t}-1)}}\right). \end{aligned} \tag{3.28}$$

In fact, $g(t)$ is the expected instantaneous cost of error at t given $X(0)=\varepsilon$ which describes the transient behavior of $X(t)$ after sampling. Two key observations from (3.28) are

- $g(t)$ is an increasing function of t, and a decreasing function of ε;
- $g(0)=0$ and $g(\infty) = \sqrt{\frac{\sigma^2}{4\pi\theta}}$ are constants irrelevant to ε.

3.B Intuitive Explanations of the One-step Look-ahead Policy

Given (3.26) and (3.28), the one-step look-ahead policy ϕ' can be illustrated by Fig. 3-4, where we plot $g(t)$ when $\varepsilon = 0.1$, 1, and 5, respectively. Let us focus on $\varepsilon = 1$ case, the optimal sampling interval T^* given by the one-step look-ahead policy is the T that makes the shaded area equal to the sampling cost η. This implies, the one-step look-ahead policy can instruct us to "never sample" if

$$\eta > \int_0^\infty g(\infty) - g(t)\, dt. \tag{3.29}$$

In this case, the shaded area will always be less than η as $t \to \infty$, and the system will never sample again. Moreover, since $g(t)$ is a decreasing function of ε, the policy basically tells us to never sample again if the absolute value of one sample is less or equal to ε. As time goes, the cost rate of the one-step look-ahead policy converges to $g(\infty) = \sqrt{\frac{\sigma^2}{4\pi\theta}}$.

Eq. (3.29) gives us the lower bound for η. The next subsection gives an upper bound for η to guarantee that "never sample" is suboptimal. We will show that, if we sample once at the equilibrium state long time later than $t = 0$, and then never sample again, the achieved cost rate can beat the one-step look-ahead policy.

3.C Sample at the Equilibrium State

At the equilibrium state, an OU process behaves as

$$X(t) = \frac{\sigma}{\sqrt{2\theta}} e^{-\theta t} W(e^{2\theta t})$$

Assuming we sample at $t = \tau$ in the equilibrium state, and obtain $X(\tau) = x_\tau$, then $x_\tau \sim \mathcal{N}(0, \sigma_\tau^2)$, where $\sigma_\tau^2 = \frac{\sigma^2}{2\theta}$. Given observation $X(\tau) = x_\tau$, the value of $X(t)$ at epoch $\tau + t$ $(t > 0)$ follows (3.27) and is a Gaussian random variable. We have

- $X(\tau + t) \sim \mathcal{N}(\mu_{\tau+t}, \sigma_{\tau+t}^2)$, $t > 0$, where $\mu_{\tau+t} = x_\tau e^{-\theta t}$ and $\sigma_{\tau+t}^2 = \frac{\sigma^2}{2\theta}(1 - e^{-2\theta t})$.
- The instantaneous cost of error $\varphi(\tau + t) = -X(\tau + t)\mathbb{1}_{(-\infty,0)}(X(\tau + t))$.

Then, for $x_\tau \geq 0$, we have

$$\begin{aligned} \mathbb{E}[\varphi(\tau + t)|X(\tau) = x_\tau] &= \int_{-\infty}^{0} -x f_{X(\tau+t)}\left(X(\tau + t) = x\right) dx \\ &= \frac{\sigma_{\tau+t}}{\sqrt{2\pi}} e^{-\frac{\mu_{\tau+t}^2}{2\sigma_{\tau+t}^2}} - \frac{\mu_{\tau+t}}{2} \operatorname{erfc}\left(\frac{\mu_{\tau+t}}{\sqrt{2}\sigma_{\tau+t}}\right) \end{aligned}$$

Averaged over all possible x_τ, $x_\tau \sim \mathcal{N}(0, \sigma_\tau^2)$,

$$\mathbb{E}[\varphi(\tau + t)] = \int_{-\infty}^{\infty} \mathbb{E}[\varphi(\tau + t)|X(\tau) = x_\tau] f_{X(\tau)}\left(X(\tau) = x_\tau\right) dx_\tau = \sqrt{\frac{\sigma^2}{4\pi\theta}}(1 - e^{-\theta t})$$

Thus, if we sample at $t = \tau$, and the never sample again, the overall cost from $t = \tau$ to $t = \infty$ will be $\eta + \int_0^\infty \mathbb{E}[\varphi(\tau + t)]dt$. Let $\eta + \int_0^\infty \mathbb{E}[\varphi(\tau + t)]dt < \int_0^\infty g(\infty)dt$, we have

$$\eta < \int_0^\infty [g(\infty) - \mathbb{E}[\varphi(\tau + t)]]dt = \sqrt{\frac{\sigma^2}{4\pi\theta}} \int_0^\infty e^{-\theta t} dt = \sqrt{\frac{\sigma^2}{4\pi\theta}} \frac{1}{\theta} \tag{3.30}$$

Combing (3.29) and (3.30) gives us Proposition 2.

Chapter 4

Diversity Routing and its Impact on Delay Variation

4.1 Introduction

As we discussed earlier, the networking world is preparing for an unprecedented growth in adoption of time-sensitive and high bandwidth applications in the upcoming decade. The interactive nature of many of these applications requires low latency but more importantly they are extremely sensitive to "variation of delay". The variation of delay is often referred to as *jitter* but as accurately described in [35], "this term causes confusion as different practitioners use it in different ways". To reduce the potential for this confusion we shall also avoid this nomenclature when possible and refer to this phenomenon as delay variation. In traditional packet switched networks, delay variation is often the result of the varying queueing delays at routers along a given path. Let us ignore the mathematical complexity of queuing delay for a moment and consider a source node that injects a set of packets into the network. Each packet will potentially experience a different amount of delay as it traverses various links and buffers until it arrives at the destination. As a result, it is often more appropriate to think of the delay that will be experienced by a given packet as a random variable. We will define delay variation as the standard deviation of this random variable in our analysis and exposition. The aforementioned standard deviation measures how much the delay of a given packet deviates from the "average" packet delay.

Applications such as high frequency trading, and tele-surgery are extremely sensitive to delay variations. For example, high frequency traders would like to guarantee that their orders reach various exchanges at the same time (<1 ms), otherwise the execution of the first order at a given exchange may reveal their intent to other investors who can manipulate market prices by front-running the rest of their orders. Similarly, surgeons who want to conduct a remote operation on a patient (*i.e.* tele-surgery) expect a network that can deliver responsive and consistent haptic feedback. The proliferation of these applications presents a chicken and egg problem for network engineers: on the one hand, these applications require low latency as well as small delay variation but at the same time the bursty and dynamic nature of their traffic introduces unpredictable delay and amplifies the delay variations.

In addition to the specific examples of previous paragraphs, we should note that delay variation is a symptomatic characteristic of dynamic network environments and arises naturally as a result of dynamic resource consumption and exhaustion within networks. In such networks, users and applications can introduce bursty and unpredictable new flows into a given packet stream. Introduction of potentially massive new flows into a given stream will cause additional competition for available network resources causing additional buffering at queues and increasing the delay along those paths. Similarly, when an application terminates a large flow, it will reduce congestion and buffering at various queues, hence decreasing the delay along those paths. Such rapid changes in the end-to-end delay of a path connecting an Origin-Destination (OD) pair is common place in todays networks, and applications experience its effects as additional delay variation.

The continuous variation and abrupt changes introduced by exogenous traffic will create temporary bottlenecks in the network which manifest themselves as variations in delay. Traditionally, packet buffers were deployed to combat the negative effects associated with delay variation. When the instantaneous packet arrival rate at a queue exceeds its output rate, packets are stored in the buffer until they can be transmitted through outgoing ports. Not surprisingly, increasing the buffer size is not an attractive solution as it is costly and will increase the potential for excessive delay. Other solutions include over-provisioning the network or providing dedicated paths/circuits to such applications, both of which are not economical.

We may wonder how variations of delay impact the Quality of Service (QoS) to the end user? To answer this question we should note that users and applications prefer to operate in a static environment, because it allows them to tune their internal parameters for optimal operation in that specific environment. As an example, consider a video teleconferencing application. Such an application would seek to provide the highest possible video quality to users. Given a fixed bandwidth and a prespecified delay, the application can decide on a specific video coding scheme (codec) that closely matches the desired QoS metrics. Now, if the available bandwidth or the delay performance of the network changes, the application has to determine its latest state

and adapt the video codec to the new environment. It should be noted that when the state of the network changes too rapidly (as measured by its standard deviation), it becomes almost impossible to accurately identify and track the current state of the network. Furthermore, any mistake that results in a wrong choice of codecs will either degrade the QoS or waste valuable network resources. Other applications will suffer in a similar fashion when unpredictable network dynamics induce additional delay variations or other erratic behavior.

One may think that variations of delay can be accounted for by a Network Management and Control (NMC) system that continuously monitors the state of the network to guarantee the desired Quality of Service (QoS). However, as discussed in the previous two chapters, tracking the state of dynamic networks by the required level of precision is a rather costly undertaking. Hence, any meaningful solution should strive to meet application demands despite the unavoidable uncertainty about the instantaneous state of the network.

This brings us to the ultimate question: *can we accommodate these new applications with their stringent latency and delay variation requirements despite our relative uncertainty about the state of the network and without massive over-provisioning?* In this chapter we will demonstrate that in certain cases we can answer this question with a yes. The solution involves an innovative technique to distribute the traffic flow over multiple paths in such a way that guarantees lower aggregate delay variations despite the delay variations on individual links. This may seem counterintuitive at first, but as demonstrated in the following sections if we account for the correlated nature of delay across various paths we can reduce the apparent variations in delay.

This novel solution is inspired by Harry Markowitz's Nobel prize winning work on *portfolio selection* [36]. His work has been instrumental in construction of investment portfolios that exhibit a pre-determined risk-return behavior. In that context, he expressed the interest of the investors as hoping to achieve the lowest risk (lowest standard deviation) for a desired expected return on investment. In other words, his formulation tries to identify asset allocations that exhibit minimum variance for a desired average/expected return. This objective can be formalized mathematically

through the well-studied quadratic programming problem, where a quadratic objective function is optimized subject to linear constraints. We do not seek credit for any of the mathematical formulations and/or developments of this subject which have been exhaustively studied in economics literature. On the other hand, we are unaware of other works that apply these ideas to communication networks and specifically questions of delay. We refer interested readers to [37] and [38] for a short history on the development of Modern Portfolio Theory, as well as the mathematical derivations and consequences of the theory. The namesake "diversity routing" has been chosen to draw parallels to diversification of financial investments.

The rest of this chapter is organized as follows: Section 4.2 introduces the general model under which diversity routing is considered. Section 4.3 casts the optimal allocation of traffic as a convex quadratic optimization problem and describes the solution space. Section 4.5 discusses the theoretical limits of diversification. Section 4.6 incorporates additional cost criteria into the optimization framework. Section 4.7 extends the results to general transportation networks. Discussion of our contributions as well as future works is given in Section 4.8. Concluding remarks are provided in Section 4.9.

4.2 General Model

Consider a network management and control system that monitors the state of the network at all layers, reconfigures network resources when necessary, and provides data and instructions to applications upon request. More specifically, when an application requires network resources, it will contact the NMC system and specify its requirements, including delay, maximum allowable delay variations, and bandwidth. It is preferable for the application to specify a few possible classes of its desirable requirements, each corresponding to a different QoS and/or QoE level. The NMC system will in turn evaluate the feasibility of the requests, and respond by specifying the routes that should be used to achieve the highest possible QoS and/or QoE levels. If the network, in its current state, is unable to satisfy the application's demand, the NMC system would either reconfigure the network to meet the requirements or reject the request. Figure 4-1 illustrates such an interaction.

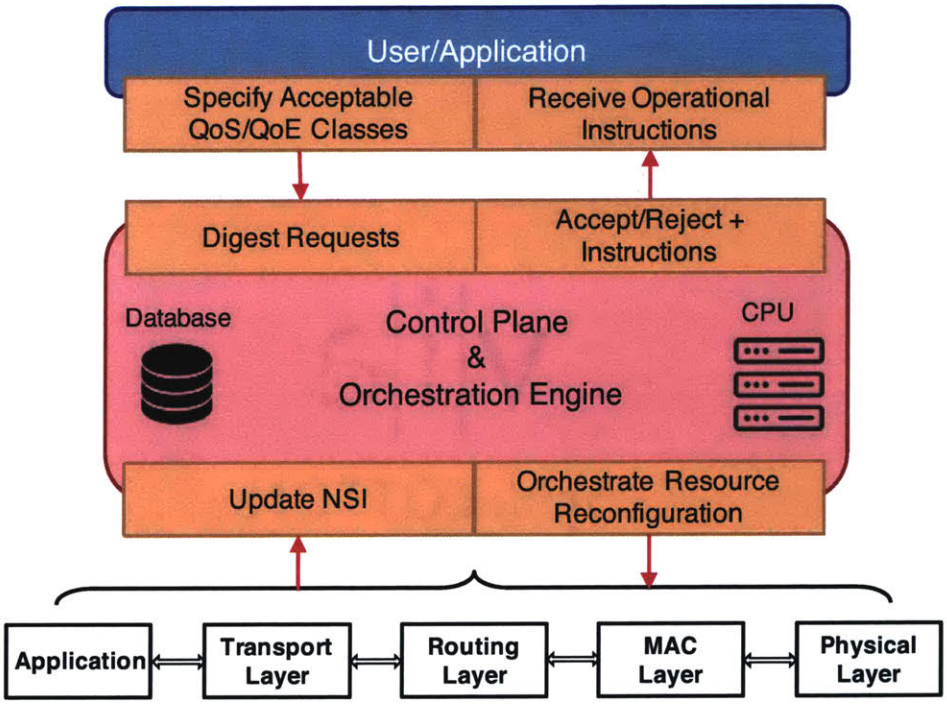


Figure 4-1: Roles and responsibilities of the NMC system: 1) Determine feasibility of application requests, 2) Instruct the application of operational requirements, 3) Orchestrate reconfiguration of network resources.

We should acknowledge that in current systems, the network is often unaware of the user's specific quality of service requirements (except for some limited SDN services). Our proposal requires a deliberate negotiation between the user and the NMC system to convey such requirements in order to achieve better efficiency and performance. We would like to be clear that our decision to take this (unconventional) approach is a conscious trade-off. The following discussion exemplifies this process in the context of routing with requirements on delay variations.

Let us suppose that an origin-destination (OD) pair is connected via n paths $P_1, \ldots, P_n$ as shown in Figure 4-2. Based on the information available to the NMC system, packets transmitted on path P_i will experience a delay d_i, where d_i is a random variable with known mean and variance denoted by $\mu_i = \mathbb{E}[d_i], \sigma_i^2 = \mathrm{Var}[d_i]$. Recall that delay variation is defined as the standard deviation of delay, and thus σ_i denotes the delay variation on path P_i.

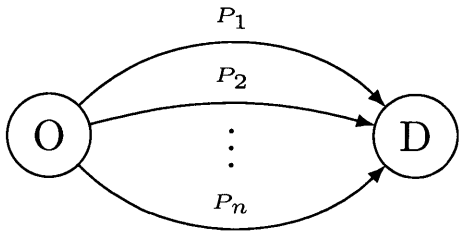


Figure 4-2: Depiction of an origin-destination pair connected via n paths.

With this description, each path connecting the OD pair can be visualized as a point on the Cartesian plane if we use the expected value and standard deviation of delay as its corresponding coordinates. Figure 4-3 illustrates the relationship between a few arbitrarily chosen paths and their respective mean delay and delay variations. Clearly, applications that utilize this network for data transport are affected by the delay performance of its individual paths. But, can the network as a whole provide delay characteristics which outperform the convex hull created by individual path characteristics? The following section demonstrates how diversity routing enables applications to meet delay requirements that in a certain sense exceed the performance of individual paths as well as the convex hull of their performance.

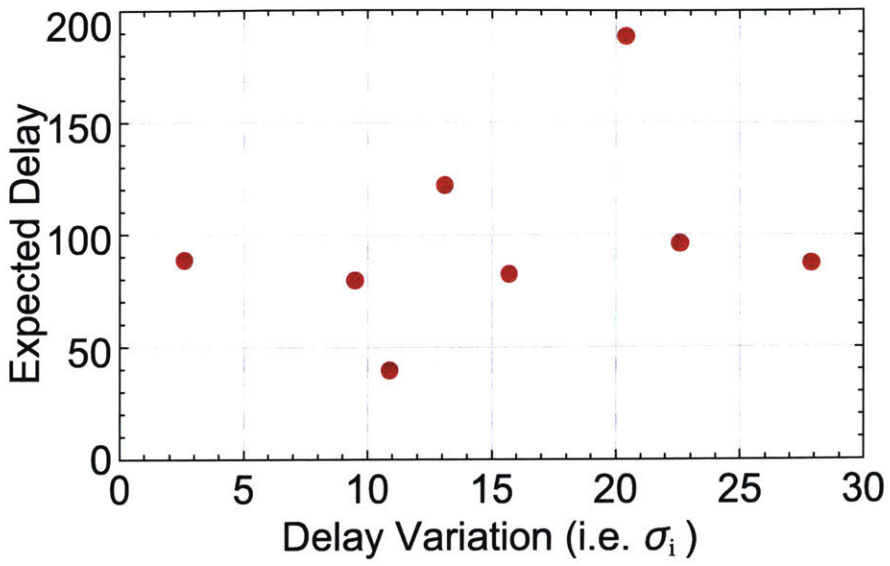


Figure 4-3: Representing the delay of each link on the Cartesian plane, using its expected delay and delay variation as the coordinates.

Note that in general, the delay incurred on these paths are not independent.[1] In what follows, we use $\sigma_{i,j}$ to denote the covariance between delays on paths P_i and P_j, *i.e.* $\sigma_{i,j} = \text{Cov}(d_i, d_j)$. Hence we can denote this covariance matrix as,

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_1^2 & \sigma_{1,2} & \cdots & \sigma_{1,n} \\ \sigma_{2,1} & \sigma_2^2 & \cdots & \sigma_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n,1} & \sigma_{n,2} & \cdots & \sigma_n^2 \end{pmatrix}$$

Before concluding this section, we should clarify that delay variation can be caused in two distinct ways and the receiver is often unable to resolve which type of delay variation it is experiencing:

- **Type 1**: Variations due to the naturally fluctuating queuing delays at routers.
- **Type 2**: Variations that are due to the sender's (random or deterministic) choice of paths which exhibit different expected delays.

[1] This fact is contrary to the assumption used in many queuing theory text books, known as the Kleinrock Independence Approximation [20].

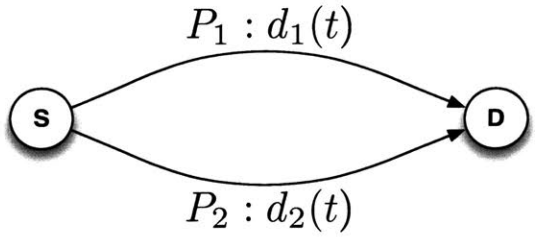


Figure 4-4: Depiction of an origin-destination pair connected via two paths.

To see the difference, let us consider the following example.

Example 1. Suppose that an origin-destination pair is connected via two paths as shown in Fig. 4-4. Suppose that path P_1 has a fixed delay of $d_1 = 100$ (ms) and path P_2 has a fixed delay of $d_2 = 300$ (ms), and packets on neither path will experience any changes in their queueing or transmission delays. In other words,

$$\mathbb{E}[d_1] = 100, \qquad \mathbb{E}[d_1] = 300, \qquad \sigma_1 = \sigma_2 = 0$$

Suppose that 100 packets are equally split between the two paths. Clearly, 50 of the packets will experience a delay of 100 (ms) and the other 50 will experience a delay of 300 (ms). In reality, the receiver may be unaware or unable to distinguish between the two sets of packets, and may perceive this observation as equivalent to delay variations of the first type. In this case, the receiver may interpret and compute the standard deviation of the observed delays (which is equal to 100), and declare it as the delay variations. Of course, such a statement would seem naive and misleading to an omniscient observer.

In some applications, Type 2 variations of delay can be easily eliminated if the expected delay on both paths are known to the transmitter. In such cases, the source will identify the path with the shorter expected delay (i.e. P_1) and transmit its corresponding packets after a fixed period of time reflecting the difference in expected delay of both paths (i.e. 200 (ms)). This proactive approach results in near simultaneous arrival of packets at the receiver and delay variation would seem to have disappeared.

Such practices are commonplace in the financial sector, where a client such as Goldman Sachs may demand that the service provider (say AT&T) guarantees extremely low Type 1 delay variation (< 1 ms) on paths that connect its trading desk to various stock exchanges. Suppose that a trading desk in Boston wishes to simultaneously execute its orders at the New York Stock Exchange and the San Francisco Stock Exchange, which are on average 15 ms and 40 ms away respectively. In this case, the expected delays are taken out by sending the orders ahead of the desired trade execution time by the aforementioned one-way delay so the orders hit various exchanges at nearly the same time (within 1 ms).

That being said, our upcoming analysis will address some of the difficulties experienced as a result of the presence of Type 1 delay variations and hence, in some examples we assume that Type 2 variations are already eliminated.

Last but not least, we should note that modeling the delay on a given link as a simple random variable with a fixed mean and standard deviation carries the underlying assumption that traffic assignments will not significantly change the traffic flow on that link. This is a reasonable assumption when the amount of traffic assigned to a given link by a specific application/session is a small portion of the total traffic on that link. [2]

[2]In the financial literature, there is a similar concept known as the *market impact* which describes the effects of a market participant on the price of an asset. Such effects can be modeled and incorporated into the overall optimization but they come at great expense to simplicity and insight.

4.3 Optimal Traffic Allocation

4.3.1 Formulation

Consider a routing algorithm that assigns a fraction f_i of the total flow to path P_i. Let us use $\mathbf{F}$, and $\boldsymbol{\mu}$ to denote the vector of fractions, and vector of mean delays respectively,

$$\mathbf{F} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}, \qquad \boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{pmatrix}$$

Note that each vector $\mathbf{F}$ corresponds to a unique *Traffic Allocation*, and as described in the previous section if the transmitting source is not proactive, packet delays perceived at the receiver will include both Type 1, and Type 2 variations. Of course, reasonable implementations of what we describe in the following paragraphs should be such that the source is proactive and transmission of packets on paths with different expected delays are properly handled as to align their expected arrival at the destination, and thus avoid Type 2 variations.[3] On the other hand, the mathematical treatments that are to follow will not address this aspect for two reasons: 1) The additional variables that are needed to capture this implementation issue will unnecessarily complicate our expressions and may obscure the central issues, and more importantly 2) In other networks, we may be concerned with a transmission cost on each link that is by no means related to delay and thus near simultaneous arrival of packets may be of little value. An example of such a network is provided at the end of Section 4.7. With that in mind, we will identify ways to think about and model this complex stochastic process as to get some meaningful insights and results.

[3]Needless to say, when an application at the source is tasked with keeping packets in its internal buffers in order to align and match the average delay of all paths, we may have to consider the costs associated with the local storage and buffering of these packets and furthermore, independently examine the costs of such actions to the application vs. the benefits of these techniques to the network. This particular issue requires a separate investigation and is outside the scope of this thesis.

A reasonable way is to suppose that each packet transmitted on a path experiences a delay, which leads us to think about the sequence of delays on a given path as a discrete random process. Similarly we can think about the sequence of delays over multiple paths as a discrete vector process. Suppose that we combine the portions of the data stream on these paths that belong to a particular session and look at this "merged" random process. We can define a random variable d_{TA} for a specific *traffic allocation* that reflects the instantaneous delay corresponding to this merged process in the appropriate sense. We shall define d_{TA} as:

$$d_{TA} = \sum_{j=1}^{n} f_j d_j \tag{4.1}$$

Note that d_{TA} is a weighted linear combination of the delays experienced across different paths, where the weights correspond to the fraction of the packets that were assigned to that path. In other words, d_{TA} denotes the instantaneous delay averaged across all paths, and thus conveys the notion of average delay experienced by packets at a particular instance of time. We should emphasize that once a traffic allocation $\mathbf{F}$ has been adopted by the source node, it can continuously and repeatedly assign the corresponding fraction of packets on each path. We can then consider the temporal realizations of this instantaneous average delay (i.e. d_{TA}) over a period of time. It is reasonable to say that in a stable system we hope to observe fairly uniform realizations of d_{TA}. Of course we should remember that delay variations of the first type are the reason for degradation of this consistency, but we wish to choose the fractional flows (i.e. vector $\mathbf{F}$) in such a way as to guarantee a fairly stable set of temporal realizations for d_{TA}. Note that the mean and variance of d_{TA} can be easily computed as

$$\begin{aligned}
\mathbb{E}\left[d_{TA}\right] &= \mathbb{E}\left[\sum_{j=1}^{n} f_j d_j\right] = \sum_{j=1}^{n} f_j \mu_j = \mathbf{F}^T \boldsymbol{\mu} \\
\mathrm{Var}\left[d_{TA}\right] &= \sum_{i=1}^{n}\sum_{j=1}^{n} f_i f_j \mathrm{Cov}(d_i, d_j) = \mathbf{F}^T \boldsymbol{\Sigma} \mathbf{F}
\end{aligned}$$

Given the aforementioned quantities, how can we define the "optimal" traffic allocation? An optimal allocation may refer to one that minimizes the expected value or variance of d_{TA}. Since expectation is a linear operation, the expected value corresponding to a specific allocation is simply the weighted linear combination of individual mean delays and is thus minimized if the entire traffic is assigned to the path with the lowest expected delay. On the other hand, variance of d_{TA} is a quadratic function of the traffic allocation $\mathbf{F}$, and the allocation that minimizes this variance depends on the covariance matrix $\mathbf{\Sigma}$. One natural way to incorporate both criteria into an optimization framework is to find the minimum-variance allocation that achieves a pre-specified expected value, μ^*. Hence the optimization can be written as,

$$\begin{aligned} \underset{\mathbf{F}}{\text{minimize}} \quad & \mathbf{F}^T \mathbf{\Sigma} \mathbf{F} \\ \text{subject to} \quad & \mathbf{e}^T \mathbf{F} = 1 \\ & \mathbf{F}^T \boldsymbol{\mu} = \mu^* \\ & 0 \leq f_i \leq 1, \; i = 1, \ldots, n. \end{aligned}$$

where $\mathbf{e}$ denotes a vector of all ones, and the constraint $\mathbf{e}^T \mathbf{F} = 1$ ensures that fractional flows sum-up to one. Algorithmically speaking, the application specifies a pair of numbers (μ^*, σ^*) to the NMC system, representing the maximum acceptable values for the expected value and variance of this instantaneous average delay. The NMC system evaluates the aforementioned optimization to determine the feasibility of the request. If a feasible traffic allocation exists, the application's request is accepted and appropriate routing information is provided by recommending a specific traffic allocation. If the request is infeasible, the NMC system will either reject the application's request or will reconfigure the network in such a way to accommodate the original request. Network reconfiguration tactics are outside the scope of this work and is left as future work.

4.3.2 Solution

The investigation of minimum variance allocations for a desired average performance, as expressed above, was originally proposed by Harry Markowitz in the context of portfolio theory and allocation of financial assets [36]. In that context, he expressed the goal of rational investors as hoping to allocate/invest their assets in such a way as to achieve the lowest risk (lowest standard deviation) for a desired expected return on investments. In a similar manner, we wish to identify a traffic allocation that achieves the lowest variance for a desired expected instantaneous average delay.

Before studying the details of the optimization, let us consider the simplest possible setting whereby an OD pair is connected via two paths. Let us suppose that the first path, P_1, has a mean delay of 300 ms and delay variation of 15 ms while the second path, P_2, has a mean delay of 100 ms but a delay variation of 20 ms. We would like to depict the characteristics of d_{TA} for a range of traffic allocations. The best way to accomplish this would be to visualize d_{TA} as a point on the Cartesian plane, where its first two moments (i.e. $\mathbb{E}[d_{TA}]$ and $\sqrt{\mathrm{Var}[d_{TA}]}$) are used as its coordinates as shown in Figure 4-5. Note that if the entire traffic is allocated to one of the paths, d_{TA} would have the same first two moments as the delay associated with that path. Hence, the red points in the figure correspond to allocating the entire traffic to one of the paths, and reflects their specific mean delay and delay variations.

In general, a traffic allocation vector is $\mathbf{F} = (f_1, f_2)^T$, where $f_2 = 1 - f_1$. Hence, we can determine the performance of all traffic allocations by sweeping the f_1 parameter between 0 and 1. Each line in Fig. 4-5 traces the set of achievable mean and standard deviation combinations of d_{TA} for a specific correlation coefficient. Note that a properly chosen traffic allocation can result in an overall standard deviation for d_{TA} that is significantly lower than that afforded by either of the individual paths. In particular, if the delay on the two paths are perfectly anti-correlated, *i.e.* $\rho = -1$, there exists a traffic allocation whose corresponding instantaneous delay averaged across paths (*i.e.* d_{TA}) will exhibit a standard deviation of zero. For our example, this point occurs at $(f_1, f_2) = (0.57, 0.43)$.

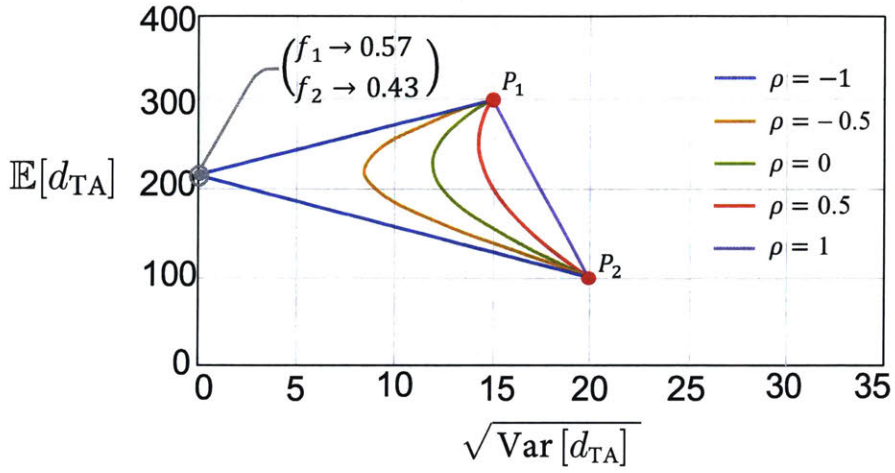


Figure 4-5: Impact of diversity routing on the first two moments of d_{TA} for various correlation coefficients for an OD pair connected via two paths.

We should also note that a particular standard deviation requirement for d_{TA} can be satisfied at two different mean delays (corresponding to two different traffic allocations). In the absence of additional selection criteria, we should always choose the traffic allocation that has a smaller mean delay. In other words, we will always be interested in the bottom portion of the aforementioned traces. The set of traffic allocations that constitute the bottom portion of the curves will be referred to as the set of *Efficient Allocations*, following a similar naming convention in [36]. Figure 4-6 depicts efficient allocations corresponding to the previously discussed case of an OD pair connected via exactly two paths.

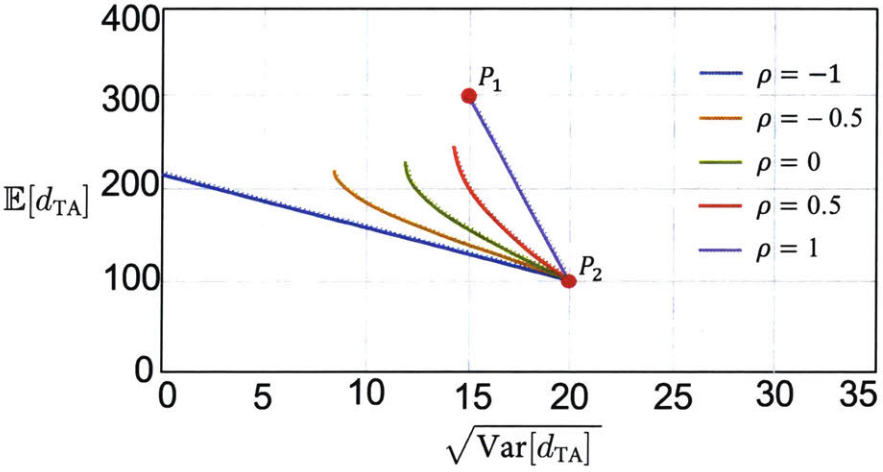
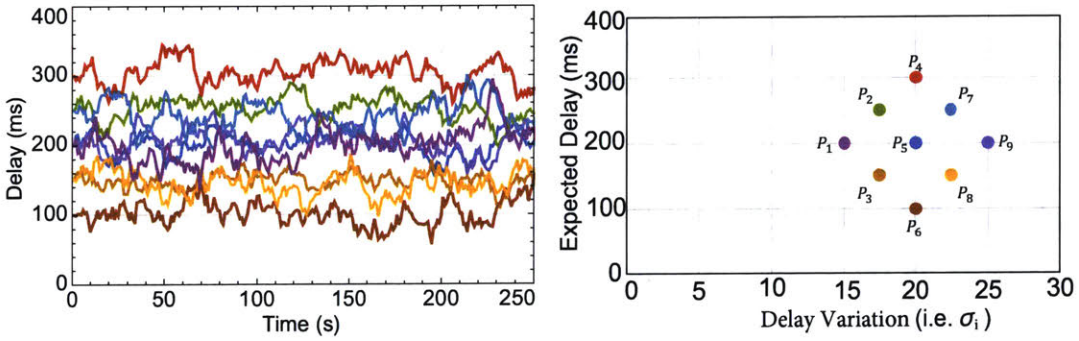


Figure 4-6: Efficient allocations corresponding to an OD pair connected via two paths.

Fortunately, the same basic behavior is observed when the number of paths increases, as shown in the following simulated scenario. Suppose that the NMC system has observed the instantaneous delay corresponding to 9 paths that connect a particular origin-destination pair over a long period of time. A possible realization of the observed instantaneous delays for a 250 second period is depicted in Figure 4-7a. The traces correspond to a set of 9 correlated random processes whose mean and standard deviation (*i.e.* delay variations) are depicted as a grid in Figure 4-7b and the sample realizations are color-coded to match the corresponding points on the grid. [4]



(a) Sample realization of the instantaneous delay corresponding to 9 correlated paths.

(b) Cartesian coordinates corresponding to delay performance of each individual path.

Figure 4-7: Depiction of correlated delay of 9 paths connecting an OD pair.

Given these observations, the NMC system can readily compute the corresponding correlation matrix (or equivalently the covariance matrix). The computed correlation matrix for this specific example is depicted in Figure 4-8. We can then numerically solve the following convex optimization problem, for all feasible values of μ^*. Feasible values of μ^* are those that fall between the minimum mean-delay and the maximum mean-delay of the 9 paths (*i.e.* between 100-300 ms),

$$\begin{aligned} \underset{\mathbf{F}}{\text{minimize}} \quad & \mathbf{F}^T \boldsymbol{\Sigma} \mathbf{F} \\ \text{subject to} \quad & \mathbf{e}^T \mathbf{F} = 1 \\ & \mathbf{F}^T \boldsymbol{\mu} = \mu^* \\ & 0 \leq f_i \leq 1, \; i = 1, \ldots, n. \end{aligned} \tag{4.2}$$

[4] More specifically, these sample paths were drawn from a set of correlated Ornstein-Uhlenbeck processes with pre-specified means and standard deviations.

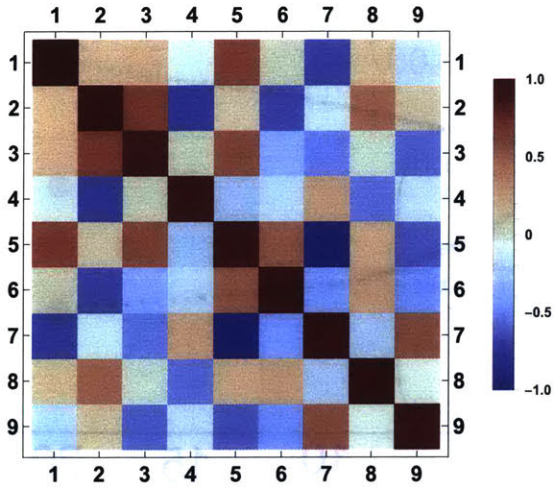


Figure 4-8: Correlation matrix corresponding to the delay observations of the 9 paths shown in Figure 4-7.

Figure 4-9 depicts the solution of the aforementioned optimization for all feasible values of μ^*. Once again, each colored dot on the right-hand side of the figure corresponds to the first two moments of d_{TA} if the entire traffic was assigned to a specific path. Not surprisingly, the coordinates of these dots correspond to the average delay and delay variation of each of the individual paths as was depicted in Fig. 4-7b. Each black point on the left corresponds a specific traffic allocation vector, and the resulting characteristics of d_{TA}. Note that the black points present a significantly reduced standard deviations in comparison to the original paths. Finally, the red dot corresponds to the "minimum-variation allocation". The network cannot support an application that requires more stringent delay variation than that afforded by the allocation corresponding to the red dot. As shown in the figure, the minimum-variation allocation sends most of its traffic through paths P_5 and P_7 as denoted by $f_5 = 0.51, f_7 = 0.45$.

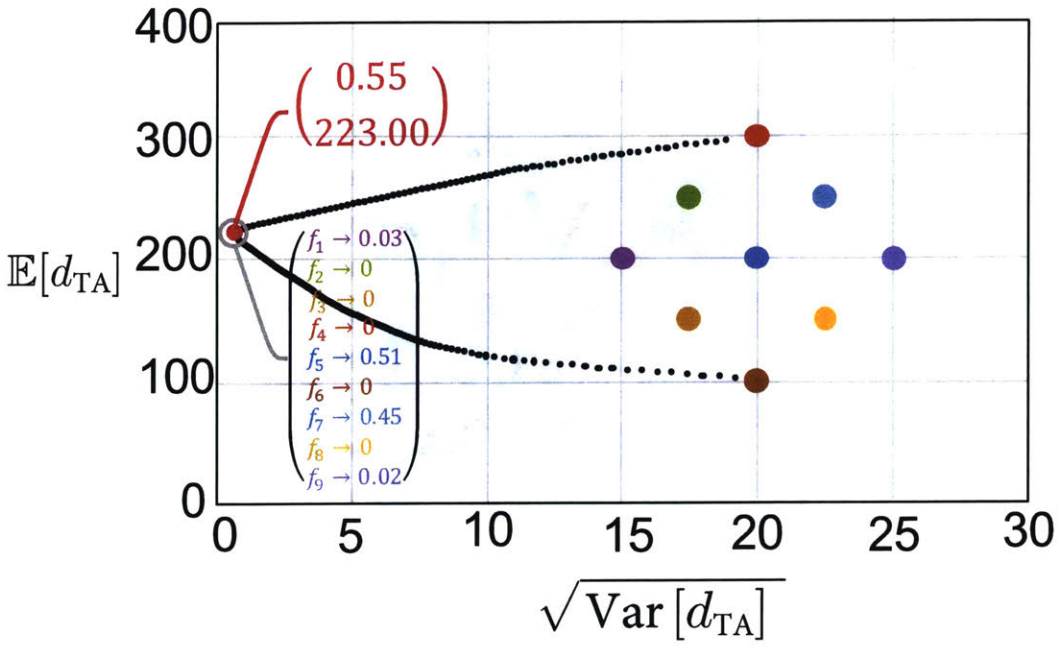


Figure 4-9: Characteristics of the optimal instantaneous delay averaged across paths (d_{TA}).

To get some insight about this, we draw your attention to the significant negative correlation between paths P_5 and P_7 as illustrated in Figure 4-8. Not surprisingly, the overall variations (as measured by standard deviation) is reduced when the flow is split amongst negatively correlated paths. Negative correlations can arise in many situations in real networks. One such instance is exemplified by Autonomous Systems (AS) that compete for traffic share by advertising different costs to a given destination. When an AS advertises a cheaper route, it will attract traffic from other AS's, resulting in negatively correlated delay on the respective paths. Another example is caused by the cyclical nature of traffic demand which corresponds to the time of day, hence, geographical areas that are offset by certain time differences will exhibit negatively correlated behaviors. We shall wrap up this section by reiterating that the lower half of the plot corresponds to *efficient allocations*.

4.4 Incorporating a Path with no delay variations

One way to avoid delay variations is to utilize a dedicated path between an OD pair. If the packet arrival rate into this path is constant, and if it is not shared with other users, we expect it to exhibit a constant delay for all packets. In other words, this path acts as a static pipe that delivers every packet to the destination node after a fixed and deterministic amount of time. One may think of circuit switched architectures as an example of a variation-free communication link. In such networks, once the circuit is established it will be dedicated to carrying the information between a specific OD pair and will provide a fixed-delay path without any variations between them.

In this section, we will discuss how the presence of such an independent variation-free path fits into our diversity routing framework. For simplicity, let us assume that the OD pair is connected via n paths, one of which is variation-free and independent of the others. Figure 4-10 depicts such a scenario whereby the path with no delay variations is denoted by a red $\otimes$ on the vertical axis. It is important to note that there is a fundamental limit on the minimum expected delay of any path. This limit is governed by the physical length of this path and the speed of light in that medium.

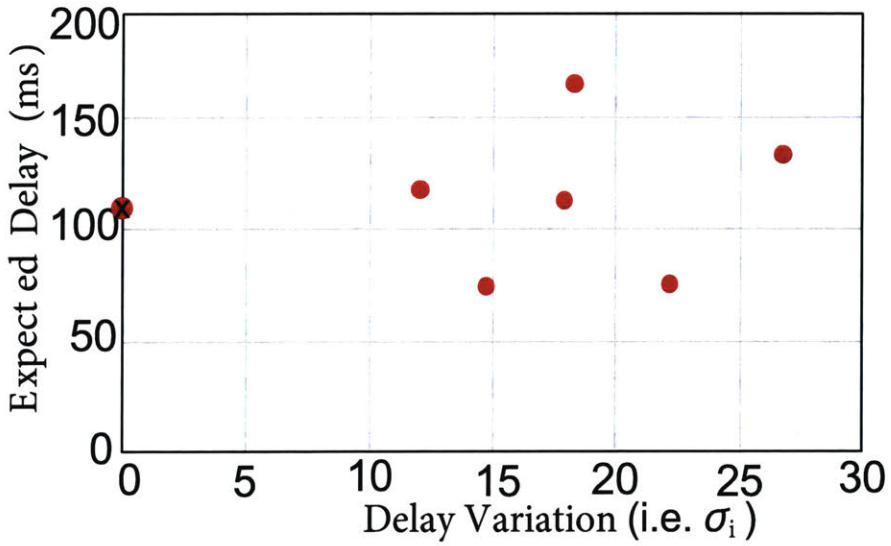


Figure 4-10: Including an independent path with no delay variations denoted by a red $\otimes$ on the vertical axis.

Let us consider the set of optimal traffic allocations which are achievable by utilizing every path except the variation-free path. This set is illustrated as a blue curve in Fig 4-11 below,

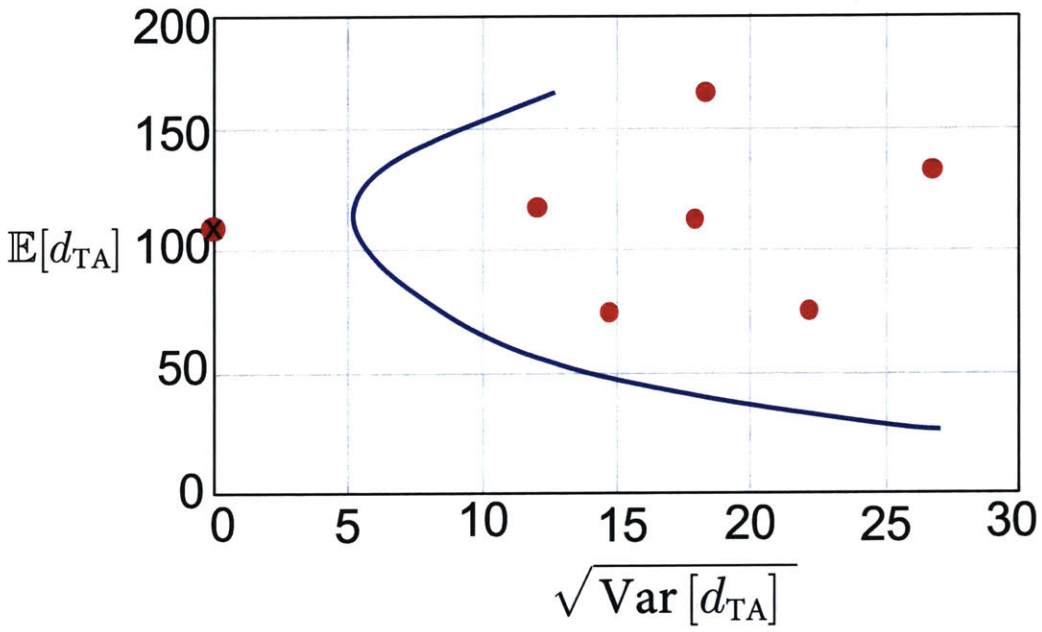


Figure 4-11: Optimal traffic allocations when the variation-free path is excluded.

Recall that our discussion has been focused on a path with no delay variations that is independent of all other paths! As a result, the performance of any "optimal" diversity routing scheme that incorporates the variation-free path will be a linear combination of some point on the blue curve and the $\otimes$ point corresponding to the variation-free path. Not surprisingly, incorporating the this path enables us to improve the set of efficient allocations. The best way to visualize the delay characteristics of such traffic allocations is by drawing a line that starts from $\otimes$ and is tangent to the portion of the blue curve corresponding to the original efficient allocations as shown in Figure 4-12. We can identify the performance of this expanded set of efficient allocation by choosing the minimum point amongst all available combinations of the plot. This is depicted in Figure 4-13. Note that this expanded efficient allocation has two segments, a linear segment from $\otimes$ to the tangency point and a secondary segment that is unaffected by the availability of the variation free path.

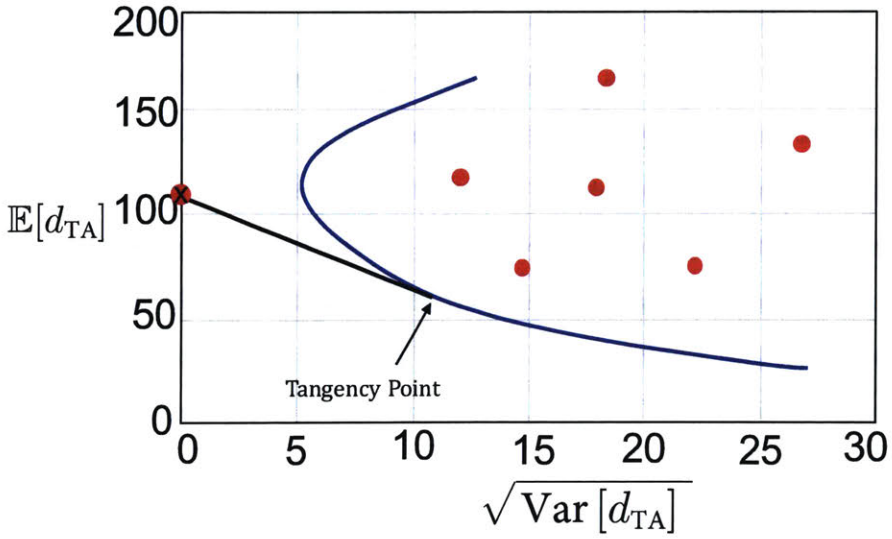


Figure 4-12: Feasible allocations when the variation-free path is used in conjunction to the previously optimal traffic allocations.

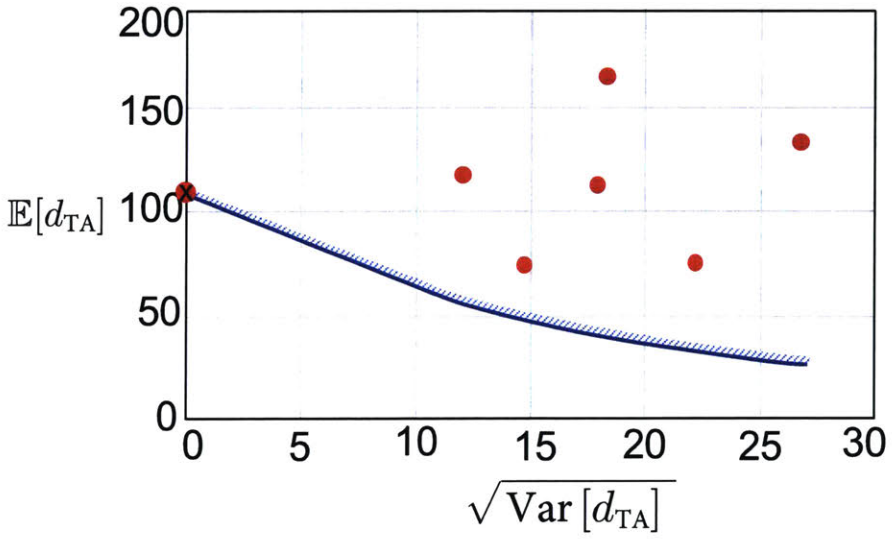


Figure 4-13: The expanded set of efficient allocations.

If the variation-free path has an expected delay that is less than or equal to the minimum expected delay of all other paths, then this path will be strictly better that any other path and all traffic should be assigned to it. Under such singular circumstances the curve corresponding to the set of efficient allocations will exhibit a non-negative slope!

We shall conclude this section by noting that the corresponding plot in financial literature is often referred to as the risk-return spectrum, and the point corresponding to variation-free paths denotes risk-free assets such as short-term Treasury bills. Unlike financial engineering that allows for "shorting" of an asset, in diversity routing we cannot assign negative flows to a given path. If one could assign a negative fraction of flow to a given path, the aforementioned linear segment would be extended beyond the tangent point and would remain linear! The slope of that line is often referred to as the Sharpe Ratio of the Tangency Portfolio, and denotes the "fair" linear tradeoff between the risk and return or in our case expected delay and delay variation.

While the diversity routing model does not generally result in linear set of efficient allocations, we can still interpret the slope of the linear segment as a fundamental quantity. Ignoring the singular case discussed previously, the slope of this line is negative and has the interpretation of expected delay per unit 'delay-variation' and thus assigns a "fair" tradeoff between these two quantities. Said another way, we should expect a fixed reduction in expected delay for each additional unit of variations that we can handle, or alternatively we should expect a fixed increase in mean-delay for each unit reduction in variation.

4.5 Limits to Diversification

The successful examples of previous sections may lead us to wonder whether we could completely eliminate temporal variations of the instantaneous delay averaged over paths (*i.e.* d_{TA}) by using diversity routing and employing additional paths. As we will see, there are limits to the diversification effect afforded to us through diversity routing. The discussions of this section will closely follow the developments of similar material in section 7.3 of [39] which showed the limits of diversification in context of Modern Portfolio Theory.

Recall the expression for the variance of a given traffic allocation, and rewrite it as

$$\begin{aligned} \mathrm{Var}\left[d_{TA}\right] &= \sum_{i=1}^{n}\sum_{j=1}^{n} f_i f_j \mathrm{Cov}(d_i, d_j) \\ &= \sum_{i=1}^{n} f_i^2 \sigma_i^2 + 2\sum_{i<j} f_i f_j \sigma_{i,j} \end{aligned}$$

where $\sigma_{i,j}$ is the covariance between delays of path i and path j respectively. Clearly, variance of delay in individual paths, as denoted by σ_i^2, contributes n terms to the sum while the covariances contribute approximately n^2 terms to the sum. This simple observation signifies the importance and contribution of covariances/correlations between various paths which can easily outweigh the delay variations of individual paths! It can be shown that the contribution of the variances can be eliminated through the introduction of additional paths, but the covariances will dominate and constitute the bulk of the remaining variations. The following example is often used to convey the aforementioned idea. Let us consider the case of equal-splitting of the traffic amongst all n paths, *i.e.* $f_i = 1/n$. Then:

$$\begin{aligned} \mathrm{Var}\left[d_{TA}\right] &= \sum_{i=1}^{n} f_i^2 \sigma_i^2 + 2\sum_{i<j} f_i f_j \sigma_{i,j} \\ &= \sum_{i=1}^{n} \left(\frac{1}{n}\right)^2 \sigma_i^2 + 2\sum_{i<j} \left(\frac{1}{n}\right)^2 \sigma_{i,j} \\ &= \frac{\text{(Avg. Var)}}{n} + \left(1 - \frac{1}{n}\right) \text{(Avg. Covar)} \end{aligned}$$

Interestingly, as $n \to \infty$, the contribution of variances of delay will become negligible and approach zero, and thus average covariance of delay becomes the dominant term. In other words, diversity routing reduces the variance of d_{TA} by incorporating paths whose average delay covariance is negligible. The aforementioned analysis is the basis of the common practice that dictates "diversification reduces risk" in financial literature.

Contribution of each path to the variance of d_{TA} can be quantified by rewriting the expressions as,

$$\begin{aligned}
\mathrm{Var}\left[d_{TA}\right] &= \sum_{i=1}^{n} f_i \mathrm{Cov}\Big(d_i, \sum_{j=1}^{n} f_j d_j\Big) \\
&= \sum_{i=1}^{n} f_i \mathrm{Cov}\Big(d_i, \sum_{j=1}^{n} f_j d_j\Big) \\
&= \sum_{i=1}^{n} f_i \mathrm{Cov}\Big(d_i, \mathbb{E}\left[d_{TA}\right]\Big)
\end{aligned}$$

which shows that the total variance of d_{TA} is a weighted average of the covariance of delay on each path and the average instantaneous delay. We conclude this section by presenting the following simple bound,

$$\frac{1}{\mathbf{e}^T \Sigma^{-1} \mathbf{e}} \leq \mathrm{Var}\left[d_{TA}\right]$$

This bound is derived in Appendix 4.A and can act as the first check to determine whether an application requirements can be met or not. The NMC system will reject any application's request for network resource if they are not obtainable. Furthermore, network architects should use the aforementioned metric as a barometer to decide if network resources should be reconfigured (including addition of new wavelengths, rejection of lower priority traffic, etc) in order to satisfy user demands. Last but not least, application designers can use such metrics in their feasibility analysis before deployment of their application on unknown networks.

4.6 Generalized Cost Function and Indifference Maps

It should come as no surprise that in real networks, moments of d_{TA} are not the sole criteria for path selection in diversity routing; but how can we incorporate additional cost criteria into the model? The most natural way of adding cost criteria is to realize that transmission over different paths may have different "costs". One reason for this may be the heterogeneity of the underlying physical layer. For example, a given path may be a fiber optic while another one is a satellite link. Even in homogeneous networks where transmission over all links have the same cost, we can associate a cost with the "length" of a given path. Clearly, a path consisting of 3 links will have 3 times the cost as a path with 1 segment. Last but not least, we should recognize that real networks (e.g. the US fiber backbone) are often a collection of independently owned and operated subnets (aka Autonomous Systems). Hence, we should expect various vendors to charge different amounts for using their systems. Either way, we can easily associate and incorporate the respective costs with each path.

Let us use $\mathbf{C}$ to denote a cost vector, whose i^{th} element c_i denotes the cost per unit flow over path P_i. We can then rewrite our original optimization problem as,

$$\begin{aligned} \underset{\mathbf{F}}{\text{minimize}} \quad & \mathbf{C}^T\mathbf{F} + \mathbf{F}^T\mathbf{\Sigma}\mathbf{F} \\ \text{subject to} \quad & \mathbf{e}^T\mathbf{F} = 1 \\ & \mathbf{F}^T\boldsymbol{\mu} = \mu^* \\ & 0 \leq f_i \leq 1,\ i = 1, \ldots, n. \end{aligned} \tag{4.3}$$

One way to interpret this new formulation is to think of a service provider that has to balance two competing goals. The first goal is to reduce the transportation cost as captured by $\mathbf{C}^T\mathbf{F}$ and the second goal is to reduce the potential loss of revenue associated with delivering lower QoS. This loss of revenue may reflect the immediate drop in customer satisfaction or the eventual customer defection caused by subpar QoS. In effect, we have taken variance of instantaneous average delay, $\mathbf{F}^T\mathbf{\Sigma}\mathbf{F}$, as a stand-in for this loss of revenue, reflecting our preference for reduced temporal

variations in d_{TA}. It is clear that the formulation could be further generalized by using a convex function of $\mathbf{F}^T\mathbf{\Sigma}\mathbf{F}$ as the second term of the objective function, but we shall sacrifice that generality in favor of simplicity, for now.

Before investigating the impact of the generalized cost functions on the solution space, we shall explore how user preferences can be mapped into reasonable cost vectors. One of the best ways of visualizing such preferences is through an *Indifference Map*, an example of which is shown in Figure 4-14. An indifference map is a collection of indifference curves, whereby a given curve connects the set of points that provide the same level of utility to users. As a result, it is reasonable to assume that users would be willing to incur the same cost for points on an indifference curve. For our purposes, it is clear that users would always prefer reduced average delay and delay variations over higher ones and hence they would be willing to pay a higher cost as we move to the bottom-left corner of the indifference map shown in Figure 4-14. On the other hand, users may be willing to trade a slightly higher mean delay for reduced delay variations, or alternatively slightly higher delay variations for lower mean delay and thus the indifference curves depicted in the Figure 4-14 can be considered reasonable surfaces of equal utility and cost to users.

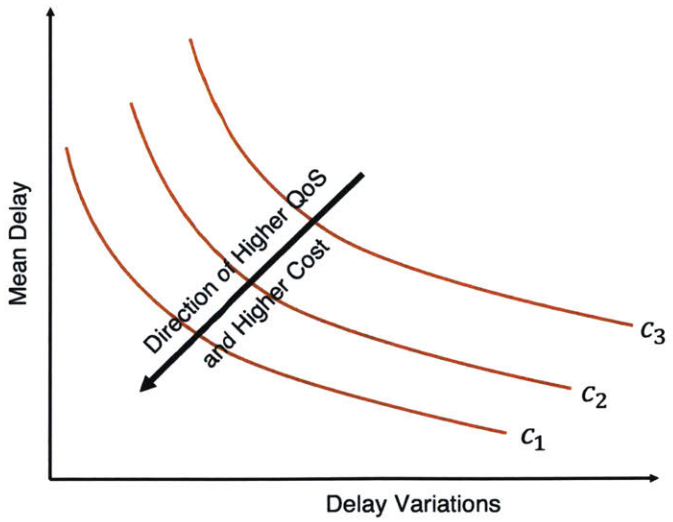


Figure 4-14: A potential indifference map for combination of average delay and delay-variations of specific paths. Note that the utility and cost increases as we move to the bottom-left corner of the plot and thus $c_1 \geq c_2 \geq c_3$.

With this characterization in mind, let us return to the generalized optimization problem of Eq. 4.3. Note that this new formulation is still a convex quadratic optimization whose solution is obtained as easily as before, and hence we will refrain from additional discussion of the solution space except for the following example. Let us revisit the routing example used in section 4.3.2 and incorporate a specific cost vector $\mathbf{C}$ as shown in Figure 4-15. We have assigned the paths to 3 different cost groups, (150, 100, and 50). While the numbers were chosen arbitrarily, they reflect our previous discussion regarding increasing cost as we move towards the bottom-left corner of the plot.

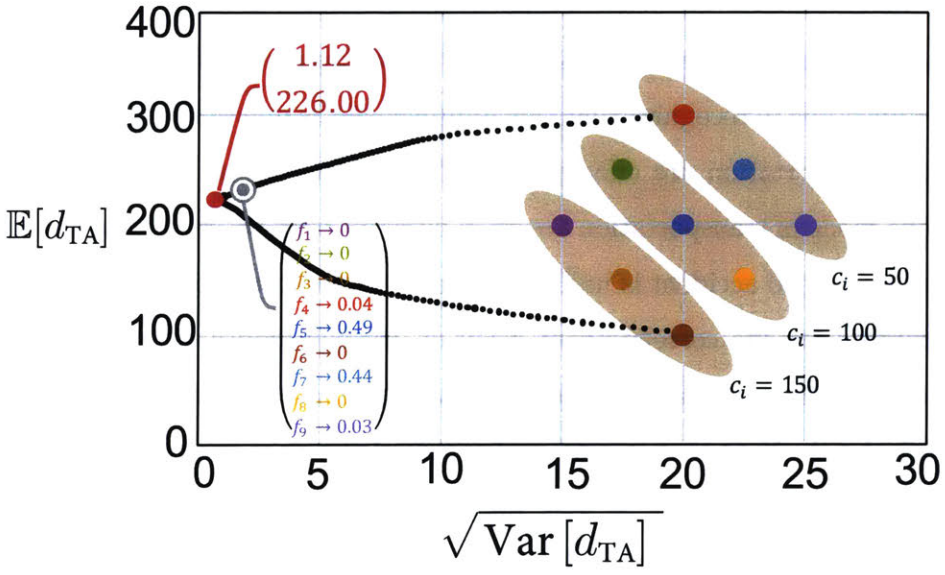


Figure 4-15: Delay characteristics of optimal traffic allocations, with indifference regions that correspond to path costs of 150, 100, and 50 per unit flow.

Once again, each black dots constitutes an "optimal" traffic allocations for each value of expected delay, with the caveat that the minimum-cost allocation is no longer on the leftmost point on the plot. The coordinates of the leftmost point is $(1.12, 226)$ and is denoted in red. The figure also identifies the traffic allocation vector corresponding to the minimum-cost allocation.

4.7 Generalization to Transportation Networks

Our analysis has so far focused on the importance of diversity routing in communication networks. Fortunately, our proposed mechanism can also be utilized when transporting physical goods over general transportation networks.

As an example, consider a retail store in Boston that wants to receive a steady supply of a given product from New York City. Commonly, retail stores contract a logistics and transportation company to transport the products from NYC to Boston. If the logistics company uses one mode of transportation (*e.g.* trucks), the exact delivery time can be impacted by the often unpredictable road conditions. On the other hand, if multiple modes of transportation (such as air, sea, railroad, etc.) are used the uncertainty in the delivery time of the products, as defined by the standard deviation of the instantaneous arrival time averaged over all paths, can be minimized.

It is important to recognize that this improvement is due to the fact that different modes of transportation are affected by different factors, and thus conditions that impact one mode of transportation are often different from those that impact another. For example, a road accident is unlikely to be related to conditions of shipping lanes. By accounting for the correlation of delay on various modes of transportation (or various roads), the logistics company can deliver the goods on a more regular basis (*i.e.* lower variations). Note that regular and steady delivery of goods can be immensely important to the retail store as well, because it will eliminate the cost of excessive local storage and warehousing for the retail company. In fact warehousing of goods at local facilities serves a similar purpose to packet buffering of communication networks.

As we noted in Section 4.3, delay is not the only consideration in a general transportation network. This is particularly true when we consider the following two emerging technologies/solutions: 1) the adoption of autonomous cars and the race by many companies to operate large fleets of autonomous vehicles 2) new initiatives within large cities and urban areas that try to incentivize certain traffic patterns through dynamic congestion pricing.

With respect to the first technology we can point to companies such as Uber and Lyft which currently operate large fleets of cars with independent drivers and are investing in the development of fully autonomous vehicles. With respect to the second technology, we can point to London, Stockholm, and Singapore which have already adopted various congestion pricing models as well as U.S. mega-cities such as New York City which are on the verge of developing their own version of congestion pricing [40].

Congestion pricing models range wildly in sophistication from simple fixed-price tolls at the entrance of certain highways to dynamic models that can automatically charge a vehicle based on a multitude of variables including make/model, weight, time of the day, fuel type, geographical location, etc. One of the main reasons for the current popularity of fixed-price tolls is their simplicity and consistency which are of paramount importance to (human) drivers. On the other hand, variable rates or complex pricing models can be easily analyzed by a computer and are thus more likely to become prevalent as autonomous vehicles gain popularity. With that in mind, let us consider a futuristic scenario where the operator of a large autonomous fleet has to make routing decisions while considering the total cost of fleet operation including the uncertain and dynamic congestion pricing.

Example 2. Suppose that on daily basis a fleet operator has to transport 1000 cars from point A to point B, which are connected via two non-overlapping roads. From prior experience, the operator knows that sending a car over the first road incurs a cost of C_1 dollars and sending it over the second road incurs a cost of C_2 dollars. Where C_1 and C_2 are two random variables with expected value and standard deviation of $(15, 50)$, and $(10, 50)$ respectively. For this example, let us suppose that our prior data shows that the two random variables are anti-correlated (*i.e.* $\rho_{1,2} = -1$ or equivalently $\mathrm{Cov}[C_1, C_2] = -2500$). Clearly, the cost of taking the cars from point A to point B depends on the realizations of C_1 and C_2.

A rational operator is often interested in minimizing its cost of operation, and we may (wrongly) think that it would prefer to send all 1000 cars through the second road with a lower expected cost (i.e. $\mathbb{E}[C_2] = 10$). In reality, the true difficulty for a fleet

operator in such a scenario is that it has to maintain a positive cash flow to support its business and may not be able to sustain more than a few days of negative cash flow. Unfortunately, because of the inherent uncertainly in the daily cost of operations, it may be faced with a realization of C_2 that is too large. For example, suppose that the operator is faced with a realization of C_2 that is one standard deviation larger than the mean, i.e. $c_2 = 10 + 50 = \$60$. In other words, the operator has to pay a total of $\$60,000$ on that particular day to transport its 1000 cars. Not surprisingly it won't be able to sustain its business if it is faced with a multi-day streak of bad luck.

In this scenario, the question for the operator is whether or not it can operate its fleet on a daily basis in such a way as to minimize the variations in the daily cost of its fleet. What our formulation does, is to split the traffic in such a way as to minimize the variance of this daily cost. Let us consider the scenario where the operator splits the traffic equally between the two roads, *i.e.* 500 cars in each road. We can see that the total daily cost of operation for this traffic allocation would be $C_{TA} = 500\ C_1 + 500\ C_2$, which would have the following expected value and variance:

$$\begin{aligned}
\mathbb{E}\left[C_{TA}\right] &= 500\ \mathbb{E}\left[C_1\right] + 500\ \mathbb{E}\left[C_2\right] = 500 * 15 + 500 * 10 = \$12500 \\
\text{Var}\left[C_{TA}\right] &= 500^2\ \text{Var}[C_1] + 500^2\ \text{Var}[C_2] + 2 * 500 * 500\ \text{Cov}[C_1, C_2] \\
&= 500^2(50^2) + 500^2(50^2) + 2 * 500 * 500 * (-2500) = 0
\end{aligned}$$

Let us take a moment to appreciate the importance of the two expressions above. What it has to say is that if we split the cars equally between the two paths, we will incur a daily cost of $\$12,500$ every day, (no more, no less)! In other words, despite the dynamic congestion pricing, the daily cost of operation is fixed and has zero variance. Of course we should note that this specific traffic allocation (*i.e.* equal splitting) minimizes the variance only when we have a perfectly anti-correlated scenario as we had assumed. Furthermore, we should note that the fleet operator may be okay with a certain amount of uncertainly and may be willing to split the traffic in such a way as to reduce the expected daily cost of operations in exchange for a small increase in the daily variance of its cost.

We should caution that our simple example of dynamic congestion pricing overlooked many other aspects of a real transportation system. First and foremost, we should note that a transportation network is best described as a multi-party enterprise where multiple (potentially independent) agents are interacting with one another. In this sense, a game theoretic framework, such as that described in Section 1.1.1, may be needed to properly examine and optimize the interaction of various agents. Furthermore, each agent may have a distinct objective and thus the various costs associated with the whole system will be borne unevenly by different parties. Proper accounting of these issues will be essential in any practical extensions of our work. We should also emphasize that our example uses a correlation coefficient of -1 for illustration purposes but real traffic networks may exhibit an arbitrary correlation.

Last but not least, we should note that a similar argument can be used to reduce uncertainty in other aspects of supply-chain management. For example, a company can order raw materials from multiple suppliers in such a way as to reduce the uncertainty in their arrival rate or cost, where the fraction ordered from each supplier is computed according to our formulation. We believe that our proposed method can be used to systematically achieve high-level managerial goals which are often referred to as "just-in-time manufacturing" or "lean manufacturing". These terms refer to practices that enable a typical manufacturing plant to operate continuously without the need for significant storage of raw materials. To achieve these goals each unit within the manufacturing plant will only request items (and/or raw material) that can be immediately used.

4.8 Discussion and Future Work

In this section we discuss some of the overarching principles which should be considered with regards to the adoption of our diversity routing mechanism. Recall that our treatment of diversity routing started with a network management and control system that has visibility to all layers. This included the ability to monitor the state and performance characteristics of various elements, as well as orchestration and resource reconfiguration capabilities. Resource reconfiguration may include tasks such as addition or removal of wavelengths on a particular fiber connection, which is currently carried out by human operators. More importantly, the NMC system will interact with applications to identify appropriate routes that can deliver a desired level of service. This challenges the conventional wisdom that networks should avoid any coordination or interaction with applications. This long-held strategy has forced a whole host of responsibilities to the end-user terminal. For example, rate control, congestion control and backoff algorithms are largely delegated to the communication end-point and the rapid growth of internet access is often attributed to this choice.

We challenge this paradigm by promoting a user-centric view that expects the network to do its best to deliver the desired quality of service to the user. Of course, this approach comes at the expense of additional complexity to the network, but we believe that this added complexity can be justified when it enables the rapid adoption of next-generation applications. Simply put, current networking practices may impede the arrival of new applications that will constitute the next wave of innovation. Of course, introduction of additional complexity will have diminishing return and thus the appropriate level of complexity should be investigated.

On a related note, we should point-out that we have not addressed the security issues that arise when the network interface is opened to various applications. Not surprisingly, malicious applications may leverage this ability to manipulate and/or attack the network by making requests which can result in misallocation of resource and ultimately resource exhaustion within the network. This topic is of immense importance and should be the focus of future investigations.

We should emphasize that diversity routing or multiple-path routing is not an entirely new idea. As far back as 1998, the Internet Engineering Task Force was considering the use of multiple paths to achieve QoS-based routing [41]. Singh *et al.* provide a detailed survey of various such routing schemes in [42], and we shall address a few major differences in our approach vs. the prior art. Most prior work concentrate on throughput maximization as their central objective and not surprisingly using *all* available paths is the simplest way to achieve this goal. Furthermore, most of their analyses considers a static network as opposed to a truly dynamic network. Note that from an optimal routing perspective, static vs. dynamic is simply a matter of the precision by which network state (*e.g.* congestion/load) is known. Clearly, optimal decisions can be made if the precise network state is known at all times. The overarching assumption in the prior work is that the network state is either fixed or varies slowly enough to ensure that underlying routing algorithms have a precise and consistent view of the network. As a result, their formulation does not account for the unavoidable uncertainty in the state of the network and overlooks the fact that routing decisions should be made despite this uncertainty. In our approach, the uncertainty in delay characteristic of a link/path is captured by variance of delay on each path, denoted by σ_i^2, which can be computed from the historical behavior of a given link. Another unique feature of our development is that we account for and utilize the correlation between various links to achieve higher quality of service. This is in contrast to traditional approaches that disregard the presence of correlated behavior and often assume independence to achieve/design simpler operating paradigms.

Additionally, most authors employ a narrow network-centric approach in their formulation. These approaches lead network architects to attach undue value to goals that are certainly reasonable but secondary in nature. For example, multiple-path routing is often used to achieve load balancing and avoid undesirable oscillatory behavior in the network. But load balancing should be a byproduct of clever network design and not its primary purpose. Our formulation uses a combination of factors, such as expected delay and delay-variations as the primary design parameters and achieves a certain level of load-balancing as a byproduct of our solution.

Last but not least, we should mention that communication networks can suffer from out-of-order packet delivery associated with multiple-path routing. It suffices to say that resequencing of packets can be handled separately and in fact error correcting codes can be utilized to significantly reduce the effects of out-of-order packet delivery.

4.9 Conclusion

In this chapter, we introduced a new mechanism for efficient allocation of traffic across a diversified set of paths. This allocation allows the network to deliver customizable quality of service to different users and can potentially reduce the need for buffers at various network elements. Our work focused on the tradeoff between mean-delay and delay variation as the main contributors to QoS. An important feature of this approach is its ability to achieve the desired QoS despite the relative uncertainty about the state of the network. Noting that the introduction of demanding (and data hungry) applications often outpace that of network upgrades, we have argued that our innovative solution can accelerate the adoption of these applications without the need for immediate capital expenditure. We concluded our remarks by extending our findings to general transportation networks and argued that this approach can significantly improve the supply chain predictability and reduce the need for storage facilities.

Appendix

4.A Lower Bound for the Minimum Variance Allocation

By relaxing the positivity constraints on f_i's, we obtain an analytical solution to the relaxed optimization problem via a Lagrange multiplier. Let us write the Lagrangian as

$$\mathcal{L}(\mathbf{F}, \ell) = \frac{1}{2} \mathbf{F}^T \mathbf{\Sigma} \mathbf{F} + \ell(1 - \mathbf{e}^T \mathbf{F})$$

which can be solved as the solution to $\frac{\partial \mathcal{L}}{\partial f_i} = \frac{\partial \mathcal{L}}{\partial \ell} = 0$. Where

$$\frac{\partial \mathcal{L}}{\partial f_i} = f_i \sigma_i^2 + \sum_{j \neq i} f_j \sigma_{i,j} - \ell = 0$$

rewriting the solution as a matrix gives us $\mathbf{\Sigma F} = \ell \mathbf{e}$ or equivalently, $\mathbf{F} = \ell \Sigma^{-1} \mathbf{e}$. Noting that $\mathbf{e}^T \mathbf{F} = 1$, we get

$$\begin{aligned} \mathbf{e}^T \mathbf{F} &= \mathbf{e}^T \left(\ell \Sigma^{-1} \mathbf{e} \right) = \ell \mathbf{e}^T \Sigma^{-1} \mathbf{e} = 1 \\ \ell &= \frac{1}{\mathbf{e}^T \Sigma^{-1} \mathbf{e}} \end{aligned}$$

which gives us the following allocation

$$\mathbf{F} = \frac{\Sigma^{-1} \mathbf{e}}{\mathbf{e}^T \Sigma^{-1} \mathbf{e}}$$

Let us use U to denote this unconstrained traffic allocation. Then we have the following mean and variance for the delay:

$$\begin{aligned}
\mathbb{E}[d_U] &= \mathbf{F}^T\boldsymbol{\mu} = \left(\ell\Sigma^{-1}\mathbf{e}\right)^T\boldsymbol{\mu} \\
&= \ell\mathbf{e}^T\left(\Sigma^{-1}\right)^T\boldsymbol{\mu} = \ell\mathbf{e}^T\Sigma^{-1}\boldsymbol{\mu} = \frac{\mathbf{e}^T\Sigma^{-1}\boldsymbol{\mu}}{\mathbf{e}^T\Sigma^{-1}\mathbf{e}} \\
\mathrm{Var}[d_U] &= \mathbf{F}^T\boldsymbol{\Sigma}\mathbf{F} = \mathbf{F}^T\ell\mathbf{e} = \ell\mathbf{e}^T\mathbf{F} = \ell
\end{aligned}$$

Recall that we ignored the positivity constraints on f_i, and hence the aforementioned variance, is a lower bound to the achievable minimum variance. If we use minVar to denote the minimum achievable variance for operationally feasible traffic allocations we have

$$\frac{1}{\mathbf{e}^T\Sigma^{-1}\mathbf{e}} \leq \mathrm{Var}[d_{\mathrm{minVar}}]$$

Bibliography

[1] A. Rezaee and V. Chan, "Cognitive Network Management and Control with Significantly Reduced State Sensing," in *Global Telecommunications Conference (GLOBECOM 2018)*, pp. 1–7, IEEE, 2018.

[2] A. Rezaee and V. W. Chan, "Cognitive Network Management and Control with Significantly Reduced State Sensing," *IEEE Trans. Cognitive Commun. and Networking*, 2019.

[3] Y. Shao, A. Rezaee, S. C. Liew, and V. W. Chan, "Significant Sampling for Shortest Path Routing: A Deep Reinforcement Learning Solution," in *IEEE Global Commun. Conf. (Globecom)*, IEEE, 2019.

[4] A. Rezaee and V. W. Chan, "Diversity Routing to Improve Delay-Jitter Tradeoff in Uncertain Network Environments," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, 2019.

[5] "IEEE Future Networks – Enabling 5G and Beyond." `https://futurenetworks.ieee.org/standards`, 2019. [Online; Accessed 11-Nov-2019].

[6] M. Series, "IMT Vision–Framework and overall objectives of the future development of IMT for 2020 and beyond," *Recommendation ITU*, pp. 2083–0, 2015.

[7] Y. Wen, *Scalable fault management architecture for dynamic optical networks: an information-theoretic approach.* PhD thesis, Massachusetts Institute of Technology, 2008.

[8] R. Sutton, A. Barto, and F. Bach, *Reinforcement learning: an introduction.* MIT Press, 2018.

[9] "The Zettabyte Era: Trends and Analysis." `https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html`, June 2017.

[10] "Cisco VNI: Forecast and Methodology, 2016-2021." `https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html`, June 2017.

[11] International Telecommunication Union, "Minimum requirements related to technical performance for IMT-2020 radio interface (s)." `https://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-M.2410-2017-PDF-E.pdf`, February 2017.

[12] V. W. Chan and E. Jang, "Cognitive all-optical fiber network architecture," in *Transparent Optical Networks (ICTON), 2017 19th International Conference on*, pp. 1–4, IEEE, 2017.

[13] M. P. Agaskar, *Architectural constructs for time-critical networking in the smart city.* PhD thesis, Massachusetts Institute of Technology, 2018.

[14] D. Braess, A. Nagurney, and T. Wakolbinger, "On a paradox of traffic planning," *Transportation science*, vol. 39, no. 4, pp. 446–450, 2005.

[15] D. Acemoglu, A. Makhdoumi, A. Malekian, and A. Ozdaglar, "Informational braess' paradox: The effect of information on traffic congestion," *Operations Research*, vol. 66, no. 4, pp. 893–917, 2018.

[16] M. D. Rinehart, *The value of information in shortest path optimization.* PhD thesis, Massachusetts Institute of Technology, 2010.

[17] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*, vol. 6. Athena Scientific Belmont, MA, 1997.

[18] L. Zhang, *Network management and control of flow-switched optical networks: joint architecture design and analysis of control plane and data plane with physical-layer impairments.* PhD thesis, Massachusetts Institute of Technology, 2014.

[19] D. Bertsekas and R. Gallager, *Data Networks (Second Edition.).* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.

[20] L. Kleinrock, *Queueing Systems Volume I: Theory*, vol. 1. New York: John Wiley & Sons, 1975.

[21] L. Kleinrock, *Queueing Systems Volume II: Computer Applications*, vol. 2. New York: John Wiley & Sons, 1976.

[22] J. F. C. Kingman, "The single server queue in heavy traffic," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 57, no. 4, pp. 902–904, 1961.

[23] S. Halfin and W. Whitt, "Heavy-traffic limits for queues with many exponential servers," *Operations research*, vol. 29, no. 3, pp. 567–588, 1981.

[24] P. Mörters and Y. Peres, *Brownian motion*, vol. 30. Cambridge University Press, 2010.

[25] S. Floyd and V. Jacobson, "The synchronization of periodic routing messages," *IEEE/ACM transactions on networking*, vol. 2, no. 2, pp. 122–136, 1994.

[26] J. M. Wozencraft and I. M. Jacobs, *Principles of communication engineering.* New York, Wiley [1965], 1965.

[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[29] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[30] Y. Li, "Deep reinforcement learning: an overview," *arXiv:1701.07274*, 2017.

[31] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, pp. 1057–1063, 2000.

[32] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Int. Conf. Mach. Learning (ICML)*, 2014.

[33] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv:1509.02971*, 2015.

[34] C. E. Shannon, "Communication in the presence of noise," *Proc. Inst. Radio Eng*, vol. 371, 1934.

[35] C. Demichelis and P. Chimento, "RFC 3393: IP packet delay variation metric for IP performance metrics (IPPM)"," *IETF, November*, vol. 2002, 2002.

[36] H. Markowitz, "Portfolio selection," *The journal of finance*, vol. 7, no. 1, pp. 77–91, 1952.

[37] H. M. Markowitz, "The early history of portfolio theory: 1600–1960," *Financial Analysts Journal*, vol. 55, no. 4, pp. 5–16, 1999.

[38] E. Elton, M. Gruber, S. Brown, and W. Goetzmann, *Modern Portfolio Theory and Investment Analysis.* John Wiley & Sons, 2009.

[39] R. A. Brealey, S. C. Myers, F. Allen, and P. Mohanty, *Principles of corporate finance.* Tata McGraw-Hill Education, 2012.

[40] "Congestion Pricing in NYC: Getting it right." `http://library.rpa.org/pdf/RPA-CongestionPricingNYC_GettingItRight.pdf`, Sep 2019. [Online; Accessed 20-Nov-2019].

[41] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, "RFC2386: A Framework for QoS-Based Routing," *Network Working Group*, 1998.

[42] S. K. Singh, T. Das, and A. Jukan, "A survey on internet multipath routing and provisioning," *IEEE Communications Surveys Tutorials*, vol. 17, pp. 2157–2175, Fourthquarter 2015.