Delivering Perishable Information and Cargos:

Delay-Constrained Communication and Transportation

DENG, Lei

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Information Engineering

The Chinese University of Hong Kong
May 2017

Abstract

Nowadays, provisioning delay-constrained services becomes more and more important in communication and transportation systems. However, most existing communication and transportation systems are best-effort and do not provide any delay guarantee. To address such an issue, in this thesis, we study two important yet different *delay-constrained* problems: (i) how to maximize the system performance in terms of network utility of a wireless communication system where every packet has a hard delivery delay constraint? and (ii) how to minimize the fuel consumption of a heavy-duty truck delivering cargos in a highway transportation system where the truck has a hard delivery delay constraint?

The first problem is important in multimedia communication systems, cyber-physical systems (CPSs), and Internet-of-Things (IoT) systems with real-time applications over wireless networks. The second problem is important for truck operators who aim at both saving fuel cost and ensuring timely delivery. These two problems share a common perishable feature that the considered entity (the packet/information or the truck/cargos) is required to reach the destination on/before a given hard deadline. We remark that the delay-unconstrained counterparts of these two problems (i.e., without the perishable feature) were well studied and understood while their approaches cannot be applied to our delay-constrained problems as shown later in the thesis.

For the first problem, recently, Hou and Kumar provided a novel idle-time-based framework to solve it for a special frame-synchronized traffic pattern. However, the problem remains largely open for *general traffic patterns*. To tackle this problem, we have two critical issues: one is how to characterize the timely capacity region, and the other is how to maximize the network utility (i.e., achieve the timely capacity region). In this thesis, we propose a general framework to address these two critical issues with general traffic patterns in a single-hop downlink access-point wireless network. We first show that the timely wireless flow problem is fundamentally an infinite-horizon Markov Decision Process (MDP). Then we apply two simplification methods to prove that the timely capacity region is a convex polygon specified by a finite number of linear constraints. This allows us *for the first time* to characterize the timely capacity region of wireless flows with general

traffic patterns, addressing the first critical issue. To address the second critical issue, we further design three scheduling policies to maximize network utility and/or support feasible timely throughput vectors for general traffic patterns. The first policy achieves the optimal network utility and supports any feasible timely throughput vector but suffers from the curse of dimensionality. The second and third policies are heuristic that are inspired by our MDP framework and are of much lower complexity. Simulation results show that both achieve near-optimal performance and outperform other existing alternatives.

For the second problem, we minimize the fuel consumption of the heavy-duty truck by exploring the full design space of both route planning and speed planning. We remark that this new and practically important problem is a generalization of the classical Restricted Shortest Path (RSP) problem. We first show that the problem is NP-complete and then design a fully polynomial time approximation scheme (FPTAS) to solve it. While achieving highly-preferred theoretical performance guarantee, the proposed FPTAS still suffers from long running time for large-scale national highway systems. Leveraging insights from studying the dual problem, we design a heuristic with much lower complexity, which can be applied to large-scale national highway systems. We further characterize a condition under which our heuristic generates an optimal solution. We observe that the condition holds in most of practical instances in numerical experiments, justifying the superior empirical performance of our heuristic. Numerical experiments using real-world truck data over the actual U.S. highway network show that our proposed solutions reduce the fuel consumption by up to 17%, as compared to the shortest/fastest path algorithm adapted from common practice.

Overall, in the communication and transportation systems, we observe that both the problem structure and the problem-solving methodology of delay-constrained problems are completely different from those of the well-understood delay-unconstrained ones. This thesis serves as a first step towards studying delay-constrained communication and transportation systems and calls for further investigation participation.

摘要

现在的通信系统和交通系统越来越需要提供满足时延要求的服务。但是，大部分已有系统都只能提供尽可能的服务而不能保证时延要求。因此，本论文研究以下两个重要的具有严格时延要求的问题：(i) 在无线通信系统中，如果每个数据包有严格传输时延要求，如何最大化网络吞吐量？(ii) 在高速交通网络中，如果一辆运输货物的重型卡车有严格运输时延要求，如何最小化其油耗？

第一个问题对于通过无线网络传输实时数据的多媒体通信系统，信息物理融合系统以及物联网系统非常重要。第二个问题对于想同时节省油耗和保证及时运输的卡车管理者非常重要。这两个问题的共同点是：所研究对象（数据包或者卡车）需要在给定时间点之前到达目的地。需要强调的是，如果没有严格时延要求，这两个问题都已经被深入研究并且有很好的解决方法了。

针对第一个问题，已有的分析框架只适用于一种帧同步的特殊流量模型。本论文则提出了一种通用框架来分析一般流量模型。我们首先证明第一个问题本质上是一个无限时间马尔科夫决策过程。然后，我们采用两个简化方法证明了实时吞吐量区域是一个由有限个线性约束条件组成的多边形。针对一般流量模型，我们进一步设计了三个调度算法来最大化网络效益和支持可行实时吞吐量向量。第一个算法可以达到最优的网络效益并且支持任何可行实时吞吐量向量，但复杂度很高。第二个算法和第三个算法则是启发式算法但复杂度低。仿真结果显示两个低复杂度算法都可以达到接近最优的性能并且优于已有算法。

针对第二个问题，我们通过优化路径和行驶速度来最小化重型卡车的油耗。我们首先证明此问题是NP完全的，然后提出一个完全多项式时间近似算法。尽管此算法有很好的理论性能保证，但是其复杂度对于实际的大规模国家级高速交通网络仍然很高。因此，我们设计了另一个适用于大规模网络的低复杂度启发式算法。我们进一步证明了在某个条件下，此启发式算法可以得到最优解。而且，这个条件在我们仿真的大部分实际场景中都成立。最后，基于实际数据的仿真结果显示，与现实生活中经常采用的最短路径算法和最快路径算法相比，我们提出的启发式算法可以节省17%的油耗。

整体而言，通过研究这两个问题，我们发现具有严格时延要求的通信和交通系统不管是从问题结构，还是从解决方法上来看，和没有严格时延要求的通信和交通系统相比都很不相同。因此，我们也希望更多研究人员来参与研究这一课题。

Acknowledgement

First and foremost, I would like to express my deep gratitude to my supervisor, Prof. Minghua Chen, for his invaluable guidance and continuous support during my PhD journey. I was very fortunate to work closely with him. He, with great patience, taught me how to choose the right and important problems, how to work out technical details, how to gain and explore insights of the results, and finally how to write and present papers. His research enthusiasm and research attitude have greatly influenced me and deeply shaped my understanding on how to do high-quality research. He is and will always be the good researcher that I strive to become.

I would also like to sincerely thank Prof. Chih-Chun Wang for hosting and advising me when I visited Purdue University. During the six-month visit, I learned a lot of research skills from him when we worked closely on the timely wireless flow problem which eventually becomes the first part of this thesis. He is always patient, encouraging, and has so many inspiring ideas.

It was my great pleasure to work with all my other collaborators: Prof. Haibo Zeng, Prof. Zongpeng Li, Prof. Jack Y. B. Lee, Prof. Ying Jun (Angela) Zhang, Prof. Lingyang Song, Dr. Mohammad Hassan Hajiesmaili, Dr. Shizhen Zhao, and Ying Zhang. I benefited a lot during many insightful discussions with them.

I also thank all DREAMS labmates: Guanlin Zhang, Wenjie Zhang, Peijian Wang, Mohammad Hassan Hajiesmaili, Qi Chen, Shaoquan Zhang, Hanxu Hou, Jinlong Tu, Jincheng Zhang, Ying Zhang, Yang Yang, Lichao Yan, Hanling Yi, Benedit Max, Qiulin Lin, and Ali Menati. I really enjoyed the life with them at CUHK in the last four years. Thank all my friends at Purdue University who made my Purdue life colorful. They are Shizhen Zhao, Jia Zhang, Tianqiong Luo, Yihan Zou, Jaemin Han, Imad Ahmad, and Chih-Hua Chang.

Finally, I cannot find any words to express my gratitude to my mother Yanqin Wen, my father Huashan Deng, my brother Cong Deng, and my dear girlfriend Kerry. This thesis would not have happened without their unconditional love and support.

This work is dedicated to my family.

Contents

List of Figures

List of Tables

Part I

Motivation and Thesis Structure

Nowadays, provisioning delay-constrained services becomes more and more important in communication and transportation systems. However, most existing communication and transportation systems are best-effort and do not provide any delay guarantee. To address such an issue, in this thesis, we study two important yet different *delay-constrained* problems:

- Timely Wireless Flow Problem: how to maximize the system performance in terms of the network utility in a wireless communication system with general traffic patterns where every packet has a hard delivery delay constraint?
- Energy-Efficient Timely Transportation Problem: how to minimize the fuel consumption of a heavy-duty truck delivering cargos in a highway transportation system where the truck has a hard delivery delay constraint?

The first timely wireless flow problem is important for multimedia communication systems such as real-time streaming and video conferencing over cellular networks, and cyber-physical systems (CPSs) or Internet-of-Things (IoT) systems such as real-time surveillance and control over wireless sensor networks. The second energy-efficient timely transportation problem is important for current truck operators who aim at both saving fuel cost and ensuring timely delivery. These two problems share a common perishable feature that the considered entity (the packet/information or the truck/cargos) is required to reach the destination on/before a given hard deadline. We remark that the delay-unconstrained counterparts of these two problems (i.e., without the perishable feature) were well studied and understood while their approaches cannot be applied to our delay-constrained problems as shown later in the thesis.

This thesis is divided into two parts to solve these two problems. The first part is from Chapter 1 to Chapter 9 focusing on the first timely wireless flow problem. The second part is from Chapter 10 to Chapter 17 focusing on the second energy-efficient timely transportation problem. We preview each chapter as follows.

Chapters 1–9 study the first timely wireless flow problem. Part of the results was published in [40].

Chapter 1 introduces necessary backgrounds, challenges, fundamental problems, and our contributions for the first timely wireless flow problem.

Chapter 2 discusses the related work in the timely wireless flow research area.

Chapter 3 presents our system model and formulates our timely wireless flow problem. We also prove that the network utility maximization problem for timely wireless flow is co-NP-hard in the strong sense.

Chapter 4 explains how to cast our timely wireless flow problem as an infinite-dimension infinite-horizon multi-reward Markov Decision Process (MDP) problem.

Chapter 5 demonstrates two simplification methods to reduce the state space and address the infinite-horizon issue by observing that our MDP is actually almost cyclostationary. We prove that the timely capacity region is a convex polygon which is characterized by a finite set of linear constraints. We also design a scheduling policy by solving a finite-size convex program, which is feasibility-optimal and maximizes network utility. But the scheduling policy suffers from the curse of dimensionality.

Chapter 6 proposes two low-complexity heuristic algorithms to address the curse of dimensionality of our MDP formulation.

Chapter 7 presents numerical performances of our solutions on characterizing timely capacity region, maximizing network utility, and supporting feasible timely throughput vectors. It shows that the two proposed low-complexity heuristics achieve near-optimal performance and outperform other existing alternatives.

Chapter 8 concludes the first part and introduces possible future research directions for our first timely wireless flow problem.

Chapter 9 gives the detailed proofs of our theoretical results and presents some other supplementary analysis for our first timely wireless flow problem.

Chapters 10–17 study the second energy-efficient timely transportation problem. Part of the results was published in [39].

Chapter 10 introduces the importance of both saving fuel cost and ensuring timely delivery for heavy-duty trucks, and motivates our second energy-efficient timely transforation problem. We further summarize our contributions.

Chapter 11 shows the system model and formulates our energy-efficient timely transportation problem. We prove that this problem is NP-complete.

Chapter 12 designs a fully polynomial time approximation scheme (FPTAS) to solve our energy-efficient timely transportation problem. The FPTAS attains an approximation

ratio of $(1+\epsilon)$ with a network-size induced complexity of $O(mn^2/\epsilon^2)$, where m and n are the numbers of nodes and edges of the highway transportation network, respectively.

Chapter 13 proposes another low-complexity dual-based heuristic algorithm to solve our energy-efficient timely transportation problem. The network-sized induced complexity is $O(m + n \log n)$, much lower than that of the FPTAS. The proposed heuristic algorithm allows us to tackle the energy-efficient timely transportation problem on large-scale national highway systems. We further characterize a condition under which our heuristic generates an optimal solution.

Chapter 14 discusses some possible extensions of our problem by considering more practical constraints for the heavy-duty trucks.

Chapter 15 uses real-world data to evaluate the performance of our algorithms. It shows that our heuristic algorithm guarantees timely delivery and can save up to 17% of fuel consumption as compared to the fastest/shortest path algorithm adapted from common practice.

Chapter 16 concludes the second part and introduces possible future research directions for our second energy-efficient timely transportation problem.

Chapter 17 gives the detailed proofs of our theoretical results and presents some other supplementary analysis for our second energy-efficient timely transportation problem.

Part II

Timely Wireless Flows with General Traffic Patterns: Capacity Region and Scheduling Algorithms

Chapter 1

Introduction

1.1 Background

Real-time wireless communication systems that require delay guarantee have become prevalent. Typical systems of this kind include multimedia communication systems such as real-time streaming and video conferencing over cellular networks, and cyber-physical systems (CPSs) or Internet-of-Things (IoT) systems such as real-time surveillance and control over wireless sensor networks. As a consequence, real-time wireless traffic has experienced a phenomenal growth in recent years [80], and is predicted to increase its volume by another 7-fold in 2016–2021 [33].

A common characteristic of these systems is that they have a strict deadline for packet delivery. Packets traversing the wireless network need to be delivered before their deadlines, otherwise they expire and are deemed useless. For example, mobile video conferencing may require bounded delay on video delivery [87]. Similarly, in CPSs, time-critical applications impose latency constraints within which data or control messages must reach their targeting entities [25]. Additionally, real-time wireless communication systems often require performance on the *timely throughput*, defined as the throughput of packets that are delivered on time [53].

1.2 Challenges

Serving delay-constrained traffic over wireless networks is uniquely challenging due to the inherent coupling of space, time, and unreliable transmission.

Space: Wireless networks differ from wired networks in the presence of spatial interference, wherein the transmission over a link can upset other transmissions in its neighborhood. An optimal scheduler needs to carefully decide which link/flow to serve at a

given time slot.

Time: To ensure timely packet delivery, one also has to keep track of deadlines of individual packets and properly account for delivery urgency in scheduling link transmissions. Such unique feature in the time domain often introduces high-dimensional system state.

Unreliable Transmission: Wireless transmissions are unreliable because of shadowing and fading. The channel quality may also differ from link to link. This could result in significant delay when a delay-oblivious scheduling scheme is used.

1.3 Fundamental Problems

In delay-constrained wireless communications, there are three fundamental problems.

Capacity Region Problem: How to characterize the capacity region in terms of timely throughput? This problem is important because: (i) it provides the fundamental benchmark to evaluate any scheduling policy, and (ii) it lays down the necessary foundation for network utility maximization in terms of timely throughput.

Network Utility Maximization (NUM) Problem: How to design scheduling policies to maximize network utility in terms of timely throughput? This is the *delay-constrained* counterpart of the celebrated NUM framework for *delay-unconstrained* wireless flows, which has been widely used as both a modeling language and solution tools [62, 76, 32].

Feasibility-Optimal Policy Design Problem: A common by-product of solving the capacity region problem is that one can obtain a scheduling policy to support *one* feasible throughput vector in the region, by solving an optimization problem (a linear one if the capacity region is a polyhedron). This approach, however, may result in using (many) different policies to support different feasible rate vectors, one policy for each or multiple rate vectors. In practice, it is more desirable to implement only one policy that can support any feasible rate vectors.

For the delay-unconstrained scenario, the celebrated back-pressure algorithm [89] can support *any* feasible rate vector within the delay-unconstrained capacity region, and it is termed *throughput-optimal*. For the delay-constrained scenario studied in this thesis, following the terminology coined in [53], we call a policy *feasibility-optimal* if it can support

Table 1.1: Our contributions and comparisons with existing works. All of them consider a single-hop downlink AP scenario.

Traffic Pattern	Capacity Region	Scheduling Policy for Network Utility Maximization	Feasibility-Optimal Scheduling Policy Design
Frame-Synchronized ([53, 56])	[53]	[56]	[53]
General (this work)	A convex polygon (Chap. 5)	RAC, optimal, high complexity (Chap. 5)	RAC, optimal, high complexity (Chap. 5)
	A fast outer bound (Chap. 6)	RAC-Approx, heuri., low compl. (Chap. 6)	RAC-Approx, heuri., low compl. (Chap. 6)
			L-LDF, heuristic, low complexity (Chap. 6)

any feasible throughput vector within the timely capacity region. A central problem of scheduling delay-constrained traffic is to design a feasibility-optimal policy.[1]

Systematically solving these three fundamental problems calls for a framework that both captures the challenges in Sec. 1.2 of delay-constrained wireless communications and offers tractable solutions.

1.4 Our Contributions

Recently, researchers devoted much effort to studying real-time wireless communications [53, 56, 54, 57, 55, 68, 58, 59, 60, 65]. Among them, Hou and Kumar [53, 56, 54, 57, 55] developed an elegant idle-time-based framework to solve all the three fundamental problems in Sec. 1.3 for a special frame-synchronized traffic pattern, over single-hop downlink access-point (AP) wireless networks. Inspiring as it is, their idle-time-based framework apparently only applies to flows with the special traffic pattern, which can only capture a limited number of practical scenarios. Overall, the three fundamental problems in Sec. 1.3 remain largely open for general traffic patterns.

In this thesis, we take a first step towards solving these three fundamental problems for *general traffic patterns* by establishing a framework based on Markov Decision Process (MDP). The structure of the timely wireless flow problem makes MDP a natural candidate for establishing such framework. We summarize our contributions about how we solve these problems and compare them with existing works in Tab. 1.1. Specifically, we make

[1]Note that the Largest Debt First (LDF) policy proposed by Hou, Borkar, and Kumar in [53] is feasibility-optimal, for a special traffic (arrival and expiration) pattern. In this thesis, we design a feasibility-optimal policy for general traffic patterns.

the following contributions:

▷ In Chap. 3, we model general traffic patterns. Then in Chap. 4, we show that the timely wireless flow problem with general traffic patterns is fundamentally an MDP problem. This new observation allows us to systematically explore the *full* design space, beyond those in previous studies [53, 56, 54, 57, 55].

▷ The MDP formulation is challenging to solve. In particular, it is of infinite-horizon, infinite state space, and time-heterogeneous. In Chap. 5, by leveraging the underlying structure of the MDP formulation, we apply two simplification methods to show that the timely capacity region is a finite-size convex polygon. Our results build upon the rich literature of MDP to *judiciously formulate the problem and adapt several existing techniques of MDP in a coherent way so as to fully answer the fundamental problem: "What is the capacity region for timely wireless flows with general traffic patterns?"* As a by-product of the capacity region analysis, we obtain a provably optimal scheduling policy, called RAC, for network utility maximization. We also show that RAC is feasibility-optimal.

▷ Our capacity region characterization and the optimal RAC scheduler suffer from the curse of dimensionality rooted in the MDP approach. To address this issue, in Chap. 6, we first propose a relaxed but computationally-efficient convex polygon characterization, serving as a fast outer bound of the capacity region. Based on the outer bound analysis, we propose a low-complexity heuristic scheduling policy, called RAC-Approx, for optimizing network utility and supporting feasible timely throughput vectors. Motivated by our model for system state, we also propose another low-complexity heuristic scheduling policy, called L-LDF, for supporting feasible timely throughput vectors.

▷ In Chap. 7, we carry out extensive simulations to verify the optimality of our RAC scheduler and show that our proposed heuristic scheduler are near-optimal and outperform other conceivable alternatives.

Chapter 2

Related Work

Supporting delay-constrained traffic over wireless networks has been a very active research area and we review the most relevant works in the following by categorizing them according to the three fundamental problems in Sec. 1.3.

Capacity Region Problem: For the special frame-synchronized traffic pattern, Hou *et al.* in the seminal paper [53] proposed an idle-time based approach to characterize the capacity region of timely flows over a single-hop downlink AP scenario. The approach has been further extended to variable-bit-rate applications in [54] and time-varying channels in [55]. However, to the best of our knowledge, there are no results to characterize the capacity region for general traffic patterns beyond the special frame-synchronized traffic pattern. In this work, we fill this gap and give a complete characterization of the capacity region for general traffic patterns.

NUM Problem: Still for the special frame-synchronized traffic pattern, Hou *et al.* in [56] solved the NUM problem efficiently for the single-hop downlink AP scenario where each user has a general and valid utility function in terms of the achieved timely throughput. Later in [68] Lashgari *et al.* generalized the single-AP scenario to a multi-AP scenario, but still focused on the frame-synchronized traffic pattern and only considered a linear utility function for each user. They proposed a relaxed bin-packing problem with elegant insights for the original complicated network utility maximization problem, and provided some theoretical guarantees for such relaxation. However, there is little result on network utility maximization beyond the special frame-synchronized traffic pattern. Our work addresses this open issue.

Feasibility-Optimal Scheduling Policy Design Problem: For the special frame-synchronized traffic pattern, Hou *et al.* in [53] proposed the celebrated largest-deficit-first (LDF) scheduling policy and proved that it is feasibility-optimal in the sense that it can support any feasible timely throughput vectors. However, it turns out LDF is not

feasibility-optimal for general traffic patterns [65, 58]. There have been two lines of efforts to study the feasibility-optimal policy design problem for general traffic patterns. One is to study the performance of LDF. Kang *et al.* in [65] proposed a theoretical lower bound for the quality of service (QoS) efficiency ratio, i.e., the fraction of capacity region that can be achieved by LDF. Further, in [64], Kang *et al.* derived theoretical upper and lower bounds for the capacity efficiency ratio, i.e., the minimum link capacity required by LDF to achieve the capacity region in the same network with unit-capacity links. Both [65] and [64] considered an "i.i.d." traffic pattern. The other line is to propose new scheduling algorithms to support feasible timely throughput vectors. In [58], Hou *et al.* proposed the *Earliest-Positive-deficit-Deadline-First* (EPDF) scheduling policy, which is shown to outperform LDF numerically for the frame-based heterogenous-delay traffic pattern. However, currently there is not yet feasibility-optimal scheduling policy for general traffic patterns. This work fills in this gap and proposes a feasibility-optimal policy for general traffic patterns.

Chapter 3

System Model and Problem Formulation

3.1 The Communication Model

Network Topology and Scheduling Model: We consider a single-hop downlink access-point (AP) scenario where the AP aims to transmit K independent timely traffic to K users,[1] one for each user. The traffic (resp. channel) between the AP and user $k \in [1, K]$ is denoted as flow (resp. link) k. Assume slotted transmission. In each slot, only one link can be scheduled and can only send one packet. At the beginning of slot t, the action of the AP, denoted by A_t, thus decides which flow/link to schedule.[2] At the beginning of slot $(t+1)$, the AP can choose a different A_{t+1} and the process starts over. For easier reference, we use "at time (slot) t" to refer to "at the beginning of slot t" and use "in time (slot) t" to refer to "in the time span of slot t."

Propagation Delay and Random Erasure: To model propagation delay, we assume that if link k is scheduled at time t, then the transmitted packet can be received by user k at the *end* of time t. To model unreliable transmission of wireless channels, we assume that along any link k successful delivery happens with some probability $p_k \in (0, 1]$,

[1] In this work, each user represents one delay-constrained application, e.g., video streaming, video conferencing, etc. A physical user/device can simultaneously run multiple delay-constrained applications and thus host multiple users.

[2] After scheduling which flow to transmit, one also needs to choose which packet of the selected flow to transmit if there are multiple packets in the current queue. However, as one can show by a realization-based argument, it is optimal to always transmit the packet of the selected flow that is of the earliest deadline. If there is no packet of the selected flow in the AP's data queue, the AP just remains idle (or equivalently transmits nothing), which will not contribute to the timely throughput. In this work, we assume that the AP always chooses one flow to transmit while implicitly allowing the AP to remain idle when the queue of the chosen flow is empty.

the random delivery events are independently and identically distributed (i.i.d.) over time, and the events for different links are independent.

We also assume that at the end of time t, the scheduled user will inform the AP through a separate control channel whether it has received the transmitted packet or not (ACK/NACK). The information will then be used for scheduling at time $(t+1)$ and onward.

The above model captures the practical Wi-Fi networks and is also widely adopted in the real-time wireless communications literature, e.g., [53, 57, 55, 56, 68]. We also remark that although we consider an ON-OFF channel model in this thesis, our results can be extended to the general multi-state channel model [29].

3.2 Traffic Pattern

We assume *periodic-i.i.d.* packet arrivals with hard delay constraints for each flow k, which can be best described by the following concept of "arrival and expiration (A&E) profile." For any flow k, its A&E profile can be described by a 4-dim. vector

$$(\mathsf{offset}_k, \mathsf{prd}_k, \mathsf{D}_k, \mathsf{B}_k),$$

where offset_k denotes the time offset for the start of the arrival process of flow k; prd_k is the inter-arrival period of flow k; D_k is the deadline for each flow-k packet; and $\mathsf{B}_k \in (0,1]$ is the arrival probability of each flow-k packet. For flow k with an A&E profile $(\mathsf{offset}_k, \mathsf{prd}_k, \mathsf{D}_k, \mathsf{B}_k)$, we denote the arrival time of the m-th[3] flow-k packet as $t_{\rm arr}^{[k]}(m)$, which can be computed as

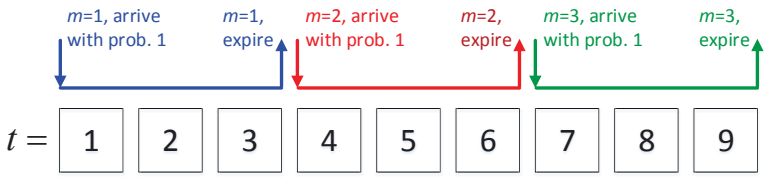
$$t_{\rm arr}^{[k]}(m) = \mathsf{offset}_k + (m-1)\mathsf{prd}_k + 1. \tag{3.1}$$

The m-th packet arrives with probability B_k. If it indeed arrives, it expires after D_k slots, and the expiration time is denoted as $t_{\rm exp}^{[k]}(m)$, which can be computed as

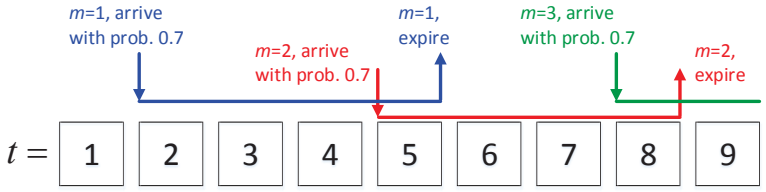
$$t_{\rm exp}^{[k]}(m) = t_{\rm arr}^{[k]}(m) + \mathsf{D}_k. \tag{3.2}$$

The expired packets are removed from the system as they are no longer useful to the application.

[3] We slightly abuse the notation and still call the packet arriving at $t_{\rm arr}^{[k]}(m)$ the m-th packet, even though on average only $(m-1)\mathsf{B}_k$ out of the first $m-1$ packets have actually "arrived."



(a) $(\mathsf{offset}_1, \mathsf{prd}_1, \mathsf{D}_1, \mathsf{B}_1) = (0, 3, 3, 1)$



(b) $(\mathsf{offset}_2, \mathsf{prd}_2, \mathsf{D}_2, \mathsf{B}_2) = (1, 3, 4, 0.7)$

Figure 3.1: The illustration of two arrival and expiration (A&E) profiles.

An illustration of two A&E profiles is provided in Fig. 3.1. For example, the first flow-1 packet will always arrive at time 1 since $\mathsf{offset}_1 = 0$ and $\mathsf{B}_1 = 1$ and will expire at time 4. The second flow-2 packet will arrive at time 5 with probability 0.7 and expire at time 9.

We then define the *traffic pattern* as the collection of A&E profiles of all K flows. We remark that our traffic model is quite general because it captures not only the special frame-synchronized traffic pattern, but also other practical traffic patterns, as shown in the following.

The frame-synchronized traffic pattern can be captured by our traffic model as

$$(\mathsf{offset}_k, \mathsf{prd}_k, \mathsf{D}_k, \mathsf{B}_k) = (0, T, T, 1), \forall k \in [1, K]$$

where T is called the *frame length*. As we can see, all K flows start at slot 1 and have the same arrival period T, and the same deadline T. Thus, every T slots, all flows have a packet arrival simultaneously, and all these packets will expire simultaneously after T slots. All three fundamental problems in Sec. 1.3 have been solved by [53, 56] for this special traffic pattern.

However, the frame-synchronized traffic pattern cannot model many important practical scenarios. For example, consider a typical mobile video conferencing scenario. Suppose that packets arrive every 20ms with a hard delay of 200ms.[4] Since the delay is

[4]If we assume that every packet has 1000 bytes (1kB), such arriving process means a sending rate

larger than the period, such flow cannot be modeled by the frame-synchronized traffic pattern. However, our traffic model can capture such traffic profile: if we assume that each slot spans 20ms and the first packet arrives at time 1, the A&E profile would be $(\mathsf{offset}, \mathsf{prd}, \mathsf{D}, \mathsf{B}) = (0, 1, 10, 1)$.

Note that our traffic pattern also suggests that we do not need an infinite-size data queue in the AP. Since all expired flow-k packets will be removed from the system, there exist at most $\left\lceil \frac{\mathsf{D}_k}{\mathsf{prd}_k} \right\rceil$ flow-k packets in the data queue of the AP. The total number of packets in the queue (from all K flows) is thus at most $\sum_{k=1}^{K} \left\lceil \frac{\mathsf{D}_k}{\mathsf{prd}_k} \right\rceil$. To avoid overflow, we therefore require that the data queue of the AP can at least hold $\sum_{k=1}^{K} \left\lceil \frac{\mathsf{D}_k}{\mathsf{prd}_k} \right\rceil$ packets. Again consider a video-conferencing scenario with ten flows where each flow's packets arrive every 20ms with a hard delay of 200ms. If every packet has a size of 1000 bytes (1kB), then the minimal data queue size requirement is $1\text{kB} \times 10 \times \left\lceil \frac{200\text{ms}}{20\text{ms}} \right\rceil = 100\text{kB}$.

Remark: Although our work is described only for the periodic-i.i.d. traffic patterns, the same principle can be readily extended to the much more general cyclostationary Markovian arrivals with observable states. Moreover, although we assume that any flow has at most one packet arrival in each period, our work can be generalized to the case that a flow may have multiple packet arrivals in a batch [93]. Our analysis can also be generalized to the case that different flows could have different packet lengths by treating a large packet as multiple sub-packets of the same length.

3.3 The Objective

The timely throughput R_k of flow k is defined as

$$R_k \triangleq \liminf_{\mathsf{T}\to\infty} \frac{\mathsf{E}\{\#\text{ of flow-}k\text{ pkts delivered before expiration in } [1,\mathsf{T}]\}}{\mathsf{T}}, \tag{3.3}$$

which computes the long-term average number of flow-k packets delivered before expiration per slot. Obviously, R_k depends on how to schedule the links/flows for $t = 1$ to ∞.

As we introduced in Sec. 1.3, our objective is to solve the following three fundamental problems.

of 400kbps. All settings, including packet size, sending rate and delay, are in line with practical video conferencing systems [94].

Capacity Region Problem: Characterize the capacity region,

$$\mathcal{R} \triangleq \{\vec{R} = (R_1, R_2, \cdots, R_K) | \text{ there exists a scheduling policy achieving timely throughput } R_k, \forall k \in [1, K]\}. \quad (3.4)$$

NUM Problem: Design scheduling policies such that the resulting timely throughput vector solves the NUM problem,

$$(\mathbf{P1}) \quad \max_{\vec{R} \in \mathcal{R}} \sum_{k=1}^{K} U_k(R_k)$$

where $U_k(\cdot)$ is the utility function for flow k, which is assumed to be increasing, concave, and continuously differentiable.

Feasibility-Optimal Scheduling Policy Design Problem: Design one scheduling policy, so that for any feasible timely throughput vector $\vec{R} = (R_1, R_2, \cdots, R_K) \in \mathcal{R}$, the achieved flow-$k$ timely throughput under this policy is at least R_k for any $k \in [1, K]$.

3.4 Complexity Hardness

It is valuable to first examine the computational complexity of our problems. Since only the NUM problem (**P1**) is a well-defined optimization problem, we show its hardness.

Theorem 3.1. (**P1**) *is co-NP-hard in the strong sense.*

Proof. Please see Appendix 9.1. □

This shows that it is computationally prohibitive to get the exact solution unless P=co-NP. Note that it only shows the hardness but does not suggest any exact solution to solve (**P1**).

In next two chapters (Chap. 4 and Chap. 5), we will propose an exact solution based on MDP to (**P1**) (and also to capacity region problem and feasibility-optimal scheduling policy design problem) with exponential complexity, meaning that all these three problems are in principle solvable. Theorem 3.1 shows that this could be the best that we can achieve if we desire an exact solution. To address the complexity issue, we must sacrifice the optimality. In Chap. 6, we therefore propose some heuristic solutions with much lower complexity.

Chapter 4

An Infinite-Dimension Infinite-Horizon MDP

This section explains how to cast our timely wireless flow problem as an infinite-dimension infinite-horizon multi-reward MDP problem. In Chap. 5, we will further simplify the infinite-size MDP and characterize the complete timely capacity region and propose a scheduling policy that is feasibility-optimal and maximizes network utility.

An MDP problem [79, 31] can be described in many different forms. The MDP used in this work is described by a tuple $(\mathcal{S}, \{\mathcal{A}_s : s \in \mathcal{S}\}, \{P_t\}, \{r_k\})$ where $\mathcal{S}$ is the state space, $\mathcal{A}_s$ is the set of possible actions when the state is $s \in \mathcal{S}$, and P_t is the transition probabilities in time t:

$$P_t(S_{t+1} = s' | S_t = s, A_t = a), \quad \forall t, \forall s, s' \in \mathcal{S}, \forall a \in \mathcal{A}_s, \tag{4.1}$$

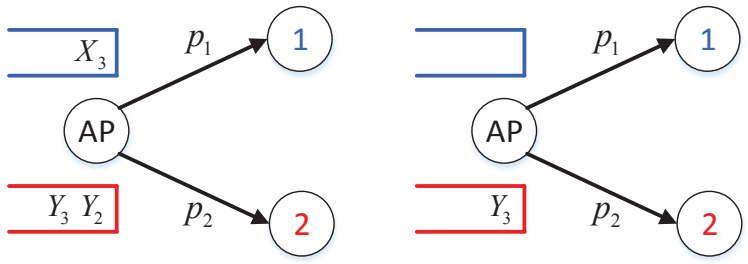
and r_k is the flow-k reward function, i.e., $r_k(s, a)$ denotes the per-slot (additive) flow-k reward of taking the action $A_t = a$ when the system state is $S_t = s$. In our problem, since we should characterize the capacity region in terms of all K flows' timely throughput, we thus define K reward functions. This is called an MDP with multiple rewards [31]. We now describe how the timely wireless flow problem can be cast as an MDP by describing the corresponding $(\mathcal{S}, \{\mathcal{A}_s : s \in \mathcal{S}\}, \{P_t\}, \{r_k\})$.

Definition of the State: We define the (network) state S_t of the MDP as the *snapshot* of all the network queues at time t. More specifically, define

$$S_t \triangleq (S_t^1, S_t^2, \cdots, S_t^K),$$

where S_t^k, the state of flow k at time t, is the collection of all non-expired flow-k packets in the AP's queue.

For example, suppose that there are only $K = 2$ flows with the corresponding A&E profiles depicted in Figs. 3.1(a) and 3.1(b), respectively. Then a possible network state



(a) $S_8 = (\{X_3\}, \{Y_2, Y_3\})$ (b) $S_9 = (\emptyset, \{Y_3\})$

Figure 4.1: Two examples for network states in slots 8 and 9, respectively.

at slot 8 is illustrated in Fig. 4.1(a). Specifically, for flow 1, at slot 8, packets $m = 1$ and $m = 2$ have expired and packet $m = 3$ has arrived at the AP. If the packet $m = 3$ has not been delivered successfully, it will remain in the queue and the state of flow 1 is $S_8^1 = \{X_3\}$. For flow 2, at slot 8, packet $m = 1$ has expired (no matter whether it showed up at the AP or not), and thus it does not appear in the queue. Packets $m = 2$ and $m = 3$ could have arrived at the AP. Suppose that these two packets have not been delivered successfully. The state of flow 2 is thus $S_8^2 = \{Y_2, Y_3\}$. The network state at slot 8 is $S_8 = (S_8^1, S_8^2) = (\{X_3\}, \{Y_2, Y_3\})$. Clearly, this is just one of many possibilities. Fig. 4.1(b) depicts another possible network state $S_9 = (S_9^1, S_9^2) = (\emptyset, \{Y_3\})$ at slot 9.

By enumerating all possible network states, we can explicitly construct the state space $\mathcal{S}$.

Definition of the Action: As mentioned in Sec. 3.1, an action A_t represents the selection of which flow to transmit in time t. After selecting the flow, say flow k, the AP will transmit the oldest flow-k packet if there exist packet(s) in the data queue or remain idle otherwise. For example, if the network state at slot 8 is as Fig. 4.1(a), then there are 2 possible actions:

- Action 1: schedule link 1 (and then transmit the oldest packet of flow 1, which is X_3);
- Action 2: schedule link 2 (and then transmit the oldest packet of flow 2, which is Y_2).

One can quickly see that there are at most K actions for any state s. Even though

not all K actions will contribute to timely throughput, for ease of exposition, in this thesis we denote the action set for any state s as $\mathcal{A}_s = \{1, 2, \cdots, K\}$ and then simply write $\mathcal{A} = \{1, 2, \cdots, K\}$ by omitting the subscript s. Thus we get the action space $\mathcal{A} = \{1, 2, \cdots, K\}$.

Definition of the Transition Probabilities: We observe that the transition probability P_t from $S_t = s$ to $S_{t+1} = s'$ if taking action $A_t = a$ in slot t depends on (i) the channel success probabilities $\{p_k : k = 1, \cdots, K\}$, and (ii) the arrival and expiration events at the end of time t (or equivalently at the beginning of time $(t+1)$). For example, at slot $(t+1)$, some packet may be successfully delivered in time t, some old packets may expire and no longer remain in the queue, and some new packets may arrive, all of which will affect the network state S_{t+1}. By carefully examining (i) and (ii), we can explicitly construct the transition probability P_t in (4.1) for all t, s, s' and a. For example, considering the scenario in Fig. 4.1, we have

$$P_8(S_9 = (\emptyset, \{Y_3\}) | S_8 = (\{X_3\}, \{Y_2, Y_3\}), A_8 = \text{Action 1}) = p_1.$$

The reason is as follows. When the AP takes "Action 1: schedule link 1 (and transmit packet X_3)" in slot 8, if the transmission is successful, then X_3 will arrive at user 1 and will thus be removed from the queue. At the same time, since Y_2 will always expire at slot 9, it will also be removed from the queue. The network state at slot 9 thus becomes $S_9 = (\emptyset, \{Y_3\})$. The probability of this transition is thus p_1.

Note that since the packet arrival/expiration event depends on the time index t, the transition probabilities are time-inhomogeneous.

Definition of the Reward: In our problem, we care about the timely throughput of all K flows. Thus, we associate K reward functions for each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ [31]. More specifically, for any flow $k \in [1, K]$, we define a reward function

$$r_k(s, a) \triangleq p_k \cdot 1_{\{\text{a flow-}k \text{ pkt is transmitted under state } s \text{ \& action } a\}}. \tag{4.2}$$

The indicator function $1_{\{\cdot\}}$ returns value 1 if the action a schedules a flow-k packet and the corresponding queue, specified in s, is not empty, and returns 0 otherwise. Notation p_k is the probability that the scheduled packet is successfully delivered. Eq. (4.2) calculates the expected value of the flow-k contribution for a given (s, a). Note that since our definition of

state s only keeps those *unexpired* packets, any successful transmission is always unexpired and will contribute to $r_k(s,a)$.

For example, at slot 8, if $S_8 = (\{X_3\}, \{Y_2, Y_3\})$ (see Fig. 4.1(a)) and A_8 is "Action 1: schedule link 1 (and transmit packet X_3)", then the respective flow-1/flow-2 rewards are

$$r_1(S_8, A_8) = p_1, \quad r_2(S_8, A_8) = 0,$$

MDP-based Equivalence: From our MDP formulation, we can see that the timely throughput of any flow k is exactly the flow-k (long-term) average reward, i.e.,

$$R_k = \liminf_{\mathsf{T}\to\infty} \frac{\sum_{t=1}^{\mathsf{T}} \mathsf{E}\{r_k(S_t, A_t)\}}{\mathsf{T}}. \tag{4.3}$$

The capacity region $\mathcal{R}$ defined in (3.4) can then be rewritten as the following reward region of our MDP formulation, i.e.,

$$\mathcal{R} = \{\vec{R} = (R_1, R_2, \cdots, R_K) | \text{ there exists a scheduling policy of the MDP achieving average reward } R_k, \forall k\}.$$

It is straightforward to verify that the NUM problem and the feasibility-optimal scheduling policy design problem can also be rewritten as an MDP-based formulation in a similar manner.

Remark: Note that the essence/difficulty of the timely throughput problem is that when we schedule a particular flow k at time t, the remaining packets in the queues are getting "older" and some may even expire. Therefore, the decision of sending which flow not only affects the instantaneous "reward" in time t, but it will also change the subsequent network state at time $(t+1)$. The effect of a decision at time t can even propagate over multiple time slots, which makes it difficult to find the optimal solution. Such a phenomenon is captured naturally by our new MDP formulation where the action A_t not only affects $r_k(S_t, A_t)(\forall k \in [1, K])$ but also affects the next network state S_{t+1} through the transition probability (4.1).

Chapter 5

Simplification and Optimal Solutions

The first contribution of this work is to observe that the the timely wireless flow problem is fundamentally an MDP problem. However, our MDP formulation in Chap. 4 is difficult to handle, because it has an infinite number of states, and it is time-inhomogeneous with infinite horizon. In this section, we demonstrate two simplification methods to reduce the state space $\mathcal{S}$, and address the time-inhomogeneity by observing that our MDP is actually *almost cyclostationary*. We then prove that the timely capacity region is a convex polygon which is characterized by a finite set of linear constraints. Our analytical results also allow us to design a scheduling policy, by solving a convex program, which is feasibility-optimal and maximizes network utility.

5.1 Reduce the State Space

Define the *lead time* (see [66] for further discussion) of the m-th flow-k packet at slot $\tau \in [t^{[k]}_{\text{arr}}(m), t^{[k]}_{\text{exp}}(m) - 1]$ as

$$t^{[k]}_{\text{lead}}(m) = t^{[k]}_{\text{exp}}(m) - \tau. \tag{5.1}$$

Clearly, we have $t^{[k]}_{\text{lead}}(m) \in [1, \mathsf{D}_k]$, which can be interpreted as the remaining time before expiration. Moreover, at any slot t, there exists at most one flow-k packet in the queue whose lead time is τ, for any $\tau \in [1, \mathsf{D}_k]$. Therefore, the state of flow k, which was originally defined as the set of unexpired flow-k packets in the queue, can be rewritten as an equivalent binary string, $S^k_t \triangleq l^k_1 l^k_2, \cdots l^k_{\mathsf{D}_k}$ where

$$l^k_i = \begin{cases} 1, & \text{if } \exists \text{ a flow-}k \text{ packet with lead time } i \text{ at } t; \\ 0, & \text{otherwise.} \end{cases}$$

For example, for flow 2 in Fig. 4.1(a), the state at slot 8 is $S_8^2 = 1001$. The reason is that both Y_2 and Y_3 are in the queue. The lead time of Y_2 is $t_{\text{lead}}^{[2]}(2) = t_{\text{exp}}^{[2]}(2) - 8 = 1$ and the lead time of Y_3 is $t_{\text{lead}}^{[2]}(3) = t_{\text{exp}}^{[2]}(3) - 8 = 4$. At slot 9, the state becomes $S_9^2 = 0010$ since only Y_3 remains and its lead time is now changed to $t_{\text{lead}}^{[2]}(3) = t_{\text{exp}}^{[2]}(3) - 9 = 3$. For Fig. 4.1, similar reasoning can be used to show $S_8^1 = 010$ and $S_9^1 = 000$. The network state thus becomes $S_8 = (S_8^1, S_8^2) = (010, 1001)$ and $S_9 = (S_9^1, S_9^2) = (000, 0010)$.

The new binary-string-based representation is equivalent to the original set-based representation since for any time t, we can use (3.2) and (5.1) to infer whether the m-th flow-k packet is in the queue or not.

Since each state s^k is a binary string of length D_k, if we denote $\mathcal{S}^k$ as the set of all possible s^k, then we have $|\mathcal{S}^k| \leq 2^{\mathsf{D}_k}$. The total number of network states is thus

$$|\mathcal{S}| = |\mathcal{S}^1| \cdot |\mathcal{S}^2| \cdot \cdots \cdot |\mathcal{S}^K| \leq 2^{\mathsf{D}_1 + \mathsf{D}_2 + \cdots + \mathsf{D}_K} < \infty. \tag{5.2}$$

The new lead-time-based state space $\mathcal{S}$ is therefore bounded. Note that even with the lead-time-based $\mathcal{S}$, the MDP is still of infinite horizon.

The reason that (5.2) is only an upper bound is that for any given traffic pattern, some binary strings do not represent any state. This fact can be used to further reduce the state space for some special traffic patterns. For example, for the *frame-synchronized traffic pattern* in Sec. 3.2, at each time t, the flow-k state $S_t^k = l_1^k l_2^k, \cdots l_T^k$, where

$$\begin{cases} l_i^k = 0, \forall i \in [1, T], & \text{if no flow-}k\text{ packet;} \\ l_{g(t)}^k = 1, l_i^k = 0, \forall i \neq g(t), & \text{if } \exists \text{ a flow-}k\text{ packet.} \end{cases} \tag{5.3}$$

and $g(t) = T - ((t-1) \mod T)$. Since there are only two possible states for each flow-k at any slot, we can perform a "lossless compression" and use $S_t^k = 0$ to represent the first case, and use $S_t^k = 1$ for the second case in (5.3). In this way, the state space is further reduced and we have $|\mathcal{S}^k| = 2$. The number of network states is then equal to $|\mathcal{S}| = 2^K$, much smaller than the upper bound (5.2). A similar compression method can be used to reduce the bound to $|\mathcal{S}| \leq 2^{\sum_{k=1}^{K} \lceil \mathsf{D}_k / \mathsf{prd}_k \rceil}$ when the traffic pattern is not frame-synchronized.

5.2 Reduce the Horizon

With the simplification in Sec. 5.1, the new MDP is of finite dimension now. However, it is still time inhomogeneous with infinite horizon, which makes it difficult to apply the existing techniques that solve time-homogeneous infinite-horizon average-reward MDP [79]. To circumvent this difficulty, we make another critical observation:

Lemma 5.1. *Using the new network state representation, the transition probabilities P_t are* almost cyclostationary. *Namely, define*

$$\mathsf{Prd} \triangleq \mathit{Least.Common.Multiple}(\mathsf{prd}_1, \mathsf{prd}_2, \cdots, \mathsf{prd}_K),$$

i.e., Prd *is is the smallest positive integer that is divisible by all* prd_k*, and choose* L *as a constant positive integer such that*

$$L \cdot \mathsf{Prd} \geq \max_{k\in[1,K]} (\mathsf{offset}_k + \mathsf{D}_k).$$

Then, for any $\tau \in [1, \mathsf{Prd}], l \geq L$*, the transition probability* $P_{l\cdot\mathsf{Prd}+\tau}$ *for slot* $t = l \cdot \mathsf{Prd} + \tau$ *is identical to the transition probability* $P_{(l+1)\mathsf{Prd}+\tau}$ *for slot* $t' = (l+1) \cdot \mathsf{Prd} + \tau$*.*

Proof. Please see Appendix 9.2. □

The reason behind is that when $l \geq L$, then at time $t = (l \cdot \mathsf{Prd} + \tau)$, the first packet of flow k has expired for all k. Therefore, all K flows have left their transient "initialization phase" and entered their "steady state". Also, since Prd is the least common multiple of all prd_k, then after every Prd time slots the arrival and expiration patterns of all K flows will repeat themselves. Since the inhomogeneity of the transition probability P_t is only caused by different arrival and expiration events for each time t, the transition probability P_t will also repeat itself after every Prd time slots since the traffic pattern are periodic. For future reference, we define $\tau_{\text{trans}} = L \cdot \mathsf{Prd}$ and call the time interval $[1, \tau_{\text{trans}}]$ the *transient duration*.

The fact that the transition probability P_t is almost periodic prompts the following intuition. If we can focus on the those time slots after the transient duration, then the network controller faces a periodic environment. As a result, in terms of finding the asymptotic average reward, we can consider a single period instead of the infinite horizon

from 1 to ∞, as long as the period of interest is beyond the transient duration. For example, one such period could be the interval $[L \cdot \mathsf{Prd} + 1, (L+1)\mathsf{Prd}]$. We will make this intuition rigorous in the next subsection.

5.3 Optimal Solutions

In this subsection, we make use of the two simplification methods in Sec. 5.1 and Sec. 5.2 to characterize the capacity region, based on which we further propose a scheduling policy that is feasibility-optimal and maximizes network utility.

Towards that end, we first define the randomized almost cyclostationary (RAC) policy as follows.

Definition 5.1. *A scheduling policy π is* randomized *if for every time t under state $S_t = s$, the action A_t is chosen randomly according to a probability mass function*

$$\mathsf{Prob}_{A_t|S_t}(a|s) = \mathsf{Prob}(A_t = a|S_t = s), \quad \forall a \in \mathcal{A}.$$

For our given MDP, a randomized policy π is almost cyclostationary *if the following two conditions hold: (i)*

$$\mathsf{Prob}_{A_t|S_t}(a|s) = \mathsf{Prob}_{A_{t+\mathsf{Prd}}|S_{t+\mathsf{Prd}}}(a|s),$$

for all $s \in \mathcal{S}, a \in \mathcal{A}, t > \tau_{trans}$, that is, for all $t > \tau_{trans}$, the conditional probabilities repeat themselves after Prd slots, and (ii) the random process[1] of the MDP state after time τ_{trans}, $\{S_{\tau_{trans}+t} : \forall t \geq 1\}$, is cyclostationary with period Prd.

We now present our main result.

Theorem 5.1. *(i) Any feasible timely throughput vector $\vec{R} \in \mathcal{R}$ can be achieved by an RAC policy.*

(ii) The capacity region $\mathcal{R}$ can be characterized by the following convex polygon,

$$\mathcal{R} = \{\vec{R} = (R_1, R_2, \cdots, R_K) | \text{ there exits an } \vec{x}$$
$$\text{such that the following conditions } (5.4a) - (5.4e) \text{ hold}\},$$

[1] Condition (i) requires that the way we make the decision is periodic and condition (ii) requires that the resulting state distribution is periodic. Although condition (i) generally means that condition (ii) is satisfied asymptotically when $t \to \infty$, here we require condition (ii) to be satisfied for small finite t.

where the conditions are

$$\sum_{a\in\mathcal{A}} x_{t+1}(s',a) = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} P_t(s'|s,a)x_t(s,a), \quad \forall s'\in\mathcal{S}, t\in[T_1,T_2-1] \tag{5.4a}$$

$$\sum_{a\in\mathcal{A}} x_{T_1}(s',a) = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} P_{T_2}(s'|s,a)x_{T_2}(s,a), \quad \forall s'\in\mathcal{S} \tag{5.4b}$$

$$R_k \le \sum_{t=T_1}^{T_2}\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \frac{r_k(s,a)x_t(s,a)}{\mathsf{Prd}}, \quad \forall k\in[1,K] \tag{5.4c}$$

$$\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} x_t(s,a) = 1, \quad \forall t\in[T_1,T_2] \tag{5.4d}$$

$$\vec{x}\ge 0, \vec{R}\ge 0 \tag{5.4e}$$

with $[T_1,T_2]\triangleq[L\cdot\mathsf{Prd}+1,(L+1)\mathsf{Prd}]$.

(iii) For any $\vec{x}$ *and* $\vec{R}=(R_1,R_2,\cdots,R_K)$ *that satisfy* (5.4a) – (5.4e)*, the RAC policy*[2] *with conditional probability,*

$$\begin{cases} \mathsf{Prob}_{A_t|S_t}(a|s) = \frac{x_t(s,a)}{\sum_{a'\in\mathcal{A}} x_t(s,a')}, & \forall t\in[T_1,T_2]; \\ \mathsf{Prob}_{A_t|S_t}(a|s) = \mathsf{Prob}_{A_{t-\mathsf{Prd}}|S_{t-\mathsf{Prd}}}(a|s), & \forall t>T_2, \end{cases} \tag{5.5}$$

and state distribution,

$$\mathsf{Prob}_{S_t}(s) = \sum_{a\in\mathcal{A}} x_t(s,a), \quad \forall t\in[T_1,T_2]. \tag{5.6}$$

achieves the timely throughput R_k*, for any* $k\in[1,K]$.

Proof. Please see Appendix 9.3. □

Note that here $[T_1,T_2]=[L\cdot\mathsf{Prd}+1,(L+1)\mathsf{Prd}]$ is the period that we mentioned in Sec. 5.2.

Part (i) of Theorem 5.1 shows that RAC polices achieve any feasible timely throughput vector. This greatly reduces the policy space since, if we ignore the transient phase, an RAC scheme can be specified by the conditional probability $\mathsf{Prob}_{A_t|S_t}(a|s)$ and the resulting state distribution $\mathsf{Prob}_{S_t}(s)$ for one period of Prd slots. The design space is now bounded and the resulting RAC policy can fully solve an infinite-horizon MDP problem. This justifies our intuition in Sec. 5.2.

[2]For ease of exposition, we omit the design of the transient state $t\le\tau_{\text{trans}}=T_1-1$. The complete RAC design can be found in the proof, i.e., Appendix 9.3.

Part (ii) of Theorem 5.1 shows that the complete timely capacity region can be characterized by a finite-size convex polygon in (5.4). The intuition of (5.4) is as follows. The variable $x_t(s,a) = \mathsf{Prob}_{S_t,A_t}(s,a)$ is the probability that the system state is s and the action is a at slot t under an RAC policy, which is why the total sum of $x_t(s,a)$ is 1 (see (5.4d)). The right-hand side of (5.4c) computes the average reward of flow k under such an RAC policy. Eq. (5.4a) is the consistency condition for time $[T_1, T_2-1]$. The left-hand side of (5.4a) is the marginal probability $\mathsf{Prob}_{S_{t+1}}(s')$. The right-hand side of (5.4a) starts from the joint distribution Prob_{S_t,A_t} and uses the transition probability $P_t(S_{t+1}|S_t,A_t)$ to compute $\mathsf{Prob}_{S_{t+1}}(s')$. Similarly, (5.4b) is the consistency condition from time T_2 back to T_1 since we require the periodicity condition $\mathsf{Prob}_{S_{T_2+1}}(s') = \mathsf{Prob}_{S_{T_1}}(s')$.

Part (iii) of Theorem 5.1 gives the corresponding RAC policy to achieve any feasible timely throughput vector based on the solution of (5.4).

Theorem 5.1 solves the first fundamental problem in Sec. 1.3, i.e., characterizing the capacity region. To the best of our knowledge, this is the first timely capacity characterization for general traffic patterns.

Based on the capacity region in (5.4), we can optimally solve the other two fundamental problems proposed in Sec. 1.3. More specifically, the NUM problem (**P1**) is equivalent to the following convex one:

$$\begin{aligned} (\mathbf{P2}) \quad \max \quad & \sum_{k=1}^{K} U_k(R_k) \\ \text{s.t.} \quad & \text{(5.4a)} - \text{(5.4e)} \\ \text{var.} \quad & \vec{x}, \vec{R} \end{aligned}$$

Similarly, to design a feasibility-optimal scheduling policy, we can solve the following linear programming (LP),[3]

$$\begin{aligned} (\mathbf{P3}) \quad \max \quad & 1 \\ \text{s.t.} \quad & \text{(5.4a)} - \text{(5.4e)} \\ \text{var.} \quad & \vec{x} \end{aligned}$$

[3]We write (**P3**) as a maximization problem for consistence, but we should note that in (**P3**), we only need to find a solution $\vec{x}$ such that (5.4a) – (5.4e) hold.

Note that in (**P2**), the achieved timely throughput R_k is an optimization variable while in (**P3**), R_k is given as an input.

RAC (Optimal) Scheduling Policy: Once we solve (**P2**) or (**P3**), we obtain the optimal state-action frequency $\vec{x}^*$. We then replace $\vec{x}$ in (5.5) and (5.6) by $\vec{x}^*$. This gives the optimal RAC policy that is feasibility-optimal and maximizes network utility, as shown in part (iii) of Theorem 5.1.

Remark: The existing elegant framework in [53, 56] is based on the frame-synchronized setting. In contrast, our framework applies to general traffic patterns. Further, the existing framework is based on first deriving an idle-time-based outer bound. Then a largest-deficit-first (LDF) scheme is proposed that attains any point within the outer bound. However, for general settings, how to find a tight outer bound is highly non-trivial and remains open as of today. Instead of finding an outer bound and an achievability scheme separately as in [53], our approach is fundamentally different. By proposing a new MDP framework, we first establish that any optimal point can always be achieved by an RAC policy. Then we search for the optimal RAC by solving a finite-size convex program. The solution is thus simultaneously an outer bound (no scheme can do better) and an inner bound (as it explicitly leads to an optimal design). In the broadest sense, our approach can be viewed as directly finding the *maximum flow* instead of indirectly finding the *minimum cut*.

Chapter 6

Low-complexity Heuristics

Thus far we have characterized the timely capacity region and designed the corresponding optimal scheduling policy that is feasibility-optimal and maximizes network utility. However, our capacity region characterization (5.4) suffers from high complexity, which involves $O(\mathsf{Prd} \cdot K \cdot 2^{\sum_{k=1}^{K} \mathsf{D}_k})$ variables and constraints. This makes it less appealing for practical implementation.[1]

In this section, we address the complexity issue by proposing two low-complexity heuristics, both of which are inspired by our MDP-based formulation. In the first one in Sec. 6.1, we derive a computationally-efficient outer bound for the capacity region in (5.4), based on which we further propose a heuristic scheduling policy to optimize network utility or support feasible timely throughput vectors. In the second one in Sec. 6.2, we improve the LDF scheduling policy [53] to support feasible timely throughput vectors by combining both deficit information and *urgency* information. Both heuristics are based on the insights derived in Theorem 5.1 and achieve good performance in the numerical evaluation, as shown later in Chap. 7.

6.1 RAC Approximation

In our original system, the AP schedules one and only one flow at each slot. We can equivalently convert this 1-to-many system to K parallel 1-to-1 systems $(\mathrm{src}_k, \mathrm{dst}_k)$ for each k in the following way. We allow each src_k to make their own decision $A_t^k \in \{1, \cdots, K\}$

[1]The main complexity is due to the computation of the optimal $x_t^*(s,a)$ in (**P2**) or (**P3**). Once $x_t^*(s,a)$ is known, the actual RAC scheduler is simple and involves generating random variables with probability distribution in (5.5). Therefore, for a relatively stable system, the optimal RAC policy can still be implemented by computing the optimal $x_t^*(s,a)$ offline. For references, using off-the-shelf solvers, (**P2**) or (**P3**) can be solved in a few seconds with $K = 7$ flows and moderate D_k values for linear utility functions.

and src_k transmits only when $A_t^k = k$ and remains idle whenever $A_t^k \neq k$. We further impose that $A_t^{k_1} = A_t^{k_2}$ for any two flows k_1 and k_2. This ensures that even though we have K parallel 1-to-1 systems, their decisions are strictly synchronized, and only one of them can be active in any time t. Therefore, the K parallel 1-to-1 systems are equivalent to the original 1-to-many AP network.

Now we relax this *synchronized action constraint* $A_t^{k_1} = A_t^{k_2}$ to a *common scheduling frequency constraint* $\mathsf{Prob}(A_t^{k_1} = a) = \mathsf{Prob}(A_t^{k_2} = a) \triangleq z_t(a)$. Namely, for each parallel system k, we use $z_t^k(s^k, a)$ to denote the probability that flow k is in state s^k and the action is a at slot t. The *common scheduling frequency constraint* imposes that the K parallel systems must share a common scheduling frequency, i.e.,

$$\sum_{s^k \in \mathcal{S}^k} z_t^k(s^k, a) = z_t(a), \forall k \in [1, K], t, a \in \mathcal{A}. \tag{6.1}$$

Clearly, the sample-path-based *synchronized action constraint* $A_t^{k_1} = A_t^{k_2}$ implies the distribution-based *common scheduling frequency constraint* (6.1). This motives us to define the following outer bound $\mathcal{R}^{\mathsf{outer}}$ of the capacity region $\mathcal{R}$ in (5.4).

$$\begin{aligned}\mathcal{R}^{\mathsf{outer}} \triangleq \{\vec{R} = (R_1, R_2, \cdots, R_K) | \text{ there exits an } \vec{z} \\ \text{such that the following conditions (6.2a) } - \text{ (6.2f) hold}\}.\end{aligned}$$

where the conditions are

$$\sum_{a \in \mathcal{A}} z_{t+1}^k(\tilde{s}^k, a) = \sum_{s^k \in \mathcal{S}^k} \sum_{a \in \mathcal{A}} P_t^k(\tilde{s}^k | s^k, a) z_t^k(s^k, a), \quad \forall k \in [1, K], \tilde{s}^k \in \mathcal{S}^k, t \in [T_1, T_2 - 1] \tag{6.2a}$$

$$\sum_{a \in \mathcal{A}} z_{T_1}^k(\tilde{s}^k, a) = \sum_{s^k \in \mathcal{S}^k} \sum_{a \in \mathcal{A}} P_{T_2}^k(\tilde{s}^k | s^k, a) z_{T_2}^k(s^k, a), \quad \forall k \in [1, K], \tilde{s}^k \in \mathcal{S}^k \tag{6.2b}$$

$$R_k \leq \sum_{t=T_1}^{T_2} \sum_{s^k \in \mathcal{S}^k} \sum_{a \in \mathcal{A}} \frac{r_k(s^k, a) z_t^k(s^k, a)}{\mathsf{Prd}}, \quad \forall k \in [1, K] \tag{6.2c}$$

$$\sum_{s^k \in \mathcal{S}^k} z_t^k(s_k, a) = z_t(a), \quad \forall k \in [1, K], t \in [T_1, T_2] \tag{6.2d}$$

$$\sum_{s^k \in \mathcal{S}^k} \sum_{a \in \mathcal{A}} z_t^k(s^k, a) = 1, \quad \forall k \in [1, K], t \in [T_1, T_2] \tag{6.2e}$$

$$\vec{z} \geq 0, \vec{R} \geq 0 \tag{6.2f}$$

In (6.2), $P_t^k(\tilde{s}^k|s^k,a)$ is the transition probability from state s^k to state $\tilde{s}^k$ for flow k if taking action $A_t^k = a$ at slot t, $r_k(s^k,a)$ is the flow-k per-slot reward under state s^k and action a (defined similarly as (4.2)), and T_1 and T_2 are defined as the same in (5.4). One can see that the form of (6.2) is very close to that of (5.4) except that (6.2) deals with each 1-to-1 system separately and links them through the common scheduling frequency constraint (6.2d).

We can regard (6.2) as a relaxed version of (5.4). In return for the relaxation, we can handle it more efficiently since the state of each flow k is considered separately (rather than considered as a joint network state). The new complexity (in terms of number of variables and constraints) thus becomes,

$$O\left((2^{\mathsf{D}_1}+2^{\mathsf{D}_2}+\cdots+2^{\mathsf{D}_K})\cdot K\cdot \mathsf{Prd}\right).$$

This allows us to handle significantly large K, Prd and very reasonable practical D_k values. If we further use the lossless simplification method in (5.3), then the complexity can be further reduced to

$$O\left(\left(2^{\left\lceil \frac{\mathsf{D}_1}{\mathsf{prd}_1}\right\rceil}+2^{\left\lceil \frac{\mathsf{D}_2}{\mathsf{prd}_2}\right\rceil}+\cdots+2^{\left\lceil \frac{\mathsf{D}_K}{\mathsf{prd}_K}\right\rceil}\right)\cdot K\cdot \mathsf{Prd}\right), \tag{6.3}$$

which is quite manageable for almost all practical system parameters. If we aim to solve (6.2)[2] approximately rather than exactly, we can further *unwind* the arrival of packets from K flows and find an approximation solution of (6.2) with a complexity of $O((\sum_{k=1}^{K}\lceil \mathsf{D}_k/\mathsf{prd}_k\rceil)\cdot K\cdot \mathsf{Prd})$. Please see the details in Appendix 9.5. We thus call $\mathcal{R}^{\mathsf{outer}}$ a fast outer bound of the capacity region $\mathcal{R}$.

Next we use the outer bound $\mathcal{R}^{\mathsf{outer}}$ to design a heuristic scheduling policy, called RAC-Approx, to either maximize the network utility or support feasible timely throughput vectors. More concretely, for the NUM problem, we solve the following convex program,

$$\begin{aligned}(\mathbf{P4})\quad \max\quad & \sum_{k=1}^{K} U_k(R_k)\\ \text{s.t.}\quad & (6.2a)-(6.2f)\\ \text{var.}\quad & \vec{z},\vec{R}\end{aligned}$$

[2]Precisely, we mean solving (**P4**) or (**P5**) with constraints (6.2) which will be mentioned soon.

and for designing low-complexity scheduling policy for supporting feasible throughput vectors, we solve the following size-reduced LP with the timely throughput vector $\vec{R}$ as an input:

$$\begin{aligned} (\mathbf{P5}) \quad \max \quad & 1 \\ \text{s.t.} \quad & (6.2a) - (6.2f) \\ \text{var.} \quad & \vec{z} \end{aligned}$$

We can regard (**P4**) (resp. (**P5**)) as the relaxed problem of (**P2**) (resp. ((**P3**)) with much lower complexity. We then use the optimal solution of (**P4**) or (**P5**), denoted by $\tilde{z}_t^k(s^k, a)$, to design the control probability of an RAC policy, i.e., we will replace (5.5) by a new formula.

RAC-Approx Scheduling Policy: At slot t, suppose that the system state is $S_t = (S_t^1, S_t^2, \cdots S_t^K) = (s^1, s^2, \cdots s^K)$, first compute the following conditional probability for each action $a \in \mathcal{A}$ and each flow $k \in [1, K]$,

$$\begin{cases} \mathsf{Prob}_{A_t|S_t^k}(a|s^k) = \frac{\tilde{z}_t^k(s^k,a)}{\sum_{a' \in \mathcal{A}} \tilde{z}_t^k(s^k,a')}, & \forall t \in [T_1, T_2]; \\ \mathsf{Prob}_{A_t|S_t^k}(a|s^k) = \mathsf{Prob}_{A_{t-\mathsf{Prd}}|S_{t-\mathsf{Prd}}^k}(a^k|s), & \forall t > T_2. \end{cases} \tag{6.4}$$

and then select action a with probability

$$\frac{\prod_{k=1}^K \mathsf{Prob}_{A_t|S_t^k}(a|s^k)}{\sum_{a' \in \mathcal{A}} \prod_{k=1}^K \mathsf{Prob}_{A_t|S_t^k}(a'|s^k)}. \tag{6.5}$$

The intuition of (6.5) is as follows. Eq. (6.4) is the probability that the k-th parallel system will choose a specific action a. Since all parallel systems choose their actions independently, the numerator of (6.5) is the probability that all flows of the auxiliary K parallel systems choose the same action a. When all flows choose the same action a, we let the AP of our actual system take such action a. Note that it is possible that all flows choose a different action a'. By normalizing over the probability of all possible a' in the denominator of (6.5), it is as if we directly let the AP randomly choose an action a with probability (6.5).

Our RAC-Approx policy, using (**P4**)/(**P5**), (6.4), and (6.5), is very efficient since all the computation can be performed on a per-flow base, as opposed to the network-wide computation in (**P2**)/(**P3**) and (5.5). We further show the convergence result of our RAC-Approx policy under a mild condition.

Lemma 6.1. *Suppose that the arrival probability is strictly less than 1, i.e.,* $\mathsf{B}_k < 1$*, for any flow* $k \in [1, K]$*. Then the timely throughput of the RAC-Approx policy converges and the* $\liminf$ *in* (3.3) *can be replaced by* $\lim$*.*

Proof. Please see Appendix 9.4. □

6.2 Deficit-based Scheduling Algorithm

In this subsection, we propose a low-complexity heuristic deficit-based scheduling policy to support feasible timely throughput vectors. Our MDP formulation shows that the lead-time-based state representation is critical to finding the optimal solution. In the following, we show that by incorporating the concept of lead time, we can further improve the performance of the existing deficit-based policies.

In general, the flow-k deficit at slot t is the difference between the desired number of delivered flow-k packets[3] and the actual number of delivered flow-k packets up to slot t [53]. Intuitively, the AP should schedule the flow with largest deficit, which is the celebrated *Largest-Deficit-First* (LDF) scheduler. The authors in [53] proved that LDF is feasibility-optimal for the frame-synchronized traffic pattern. The reason why LDF is optimal in the frame-synchronized traffic pattern is that all non-expired packets are equally urgent because they have the same deadline. However, when different packets have different deadlines, they have different levels of urgency. Since the deficit does not contain any *urgency* information, the LDF policy is no longer optimal for general traffic patterns [65, 58].

To handle heterogeneous deadlines, [58] proposed the *Earliest-Positive-deficit-Deadline-First* (EPDF) scheduler. In EPDF, at any slot, the AP focuses on those flows with strictly positive deficit and among them selects the flow which has the earliest deadline. Unfortunately, when evaluated numerically, EPDF is strictly sub-optimal, see Sec. 7.3 for more detailed discussion of the sub-optimality of EPDF.

Inspired by our lead-time-based MDP study, we propose the following *Lead-time-normalized-Largest-Deficit-First* (L-LDF) scheduler, which combines both deficit and urgency information.

[3]More specifically, the desired number of delivered flow-k packets up to t is $q_k t$, where q_k is the flow-k timely throughput requirement.

L-LDF Scheduling Policy: Suppose flow-k timely throughput requirement is $q_k \in (0, 1]$. At each slot t, among all flows that currently have packets to send, the AP computes the *lead-time-normalized deficit* $\bar{d}_k(t)$ for each flow k:

$$\bar{d}_k(t) \triangleq \frac{d_k(t) \cdot p_k}{\text{smallest-lead-time}(k, t)},$$

where $d_k(t)$ is the flow-k deficit at slot t defined as,

$$d_k(t) \triangleq [d_k(t-1) - 1_{\{\text{a flow-}k\text{ packet is delivered at slot } t\}}]^+ + q_k,$$

with $d_k(0) \triangleq 0$, $[x]^+ \triangleq \max\{x, 0\}$, and smallest-lead-time$(k, t)$ is the smallest lead time among all flow-k packets at slot t. Note that smallest-lead-time(k, t) is no smaller than 1 according to the definition of lead time in (5.1) and the remark right below the equation. Then, the AP selects the flow with the largest $\bar{d}_k(t)$.

Note that L-LDF collapses to the existing LDF in the frame-synchronized traffic pattern, since in that setting all non-expired packets at time t will have the same smallest lead time. However, the additional normalization according to the smallest lead time will better reflect the urgency of each individual flow for general traffic patterns.

Chapter 7

Simulation

In this section, we demonstrate numerical performances of our solutions on characterizing timely capacity region, maximizing network utility, and supporting feasible timely throughput vectors.

7.1 Characterizing Capacity Region

Since the existing idle-time-based analysis [53] can only characterize the capacity region for the frame-synchronized traffic pattern, we also apply our MDP-based computation $\mathcal{R}$ in (5.4) to such a simple setting. Specifically, we consider the following frame-synchronized traffic pattern:

$$(\mathsf{offset}_1, \mathsf{prd}_1, \mathsf{D}_1, \mathsf{B}_1, p_1) = (0, 3, 3, 1, 0.8),$$
$$(\mathsf{offset}_2, \mathsf{prd}_2, \mathsf{D}_2, \mathsf{B}_2, p_2) = (0, 3, 3, 1, 0.6).$$

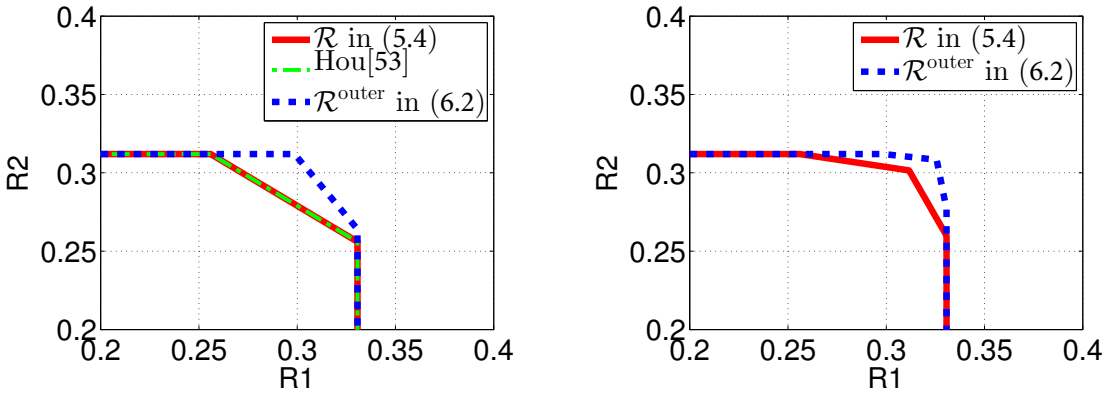
Fig. 7.1(a) shows the capacity region of this traffic pattern. As expected, both [53] and our MDP-based computation $\mathcal{R}$ in (5.4) successfully characterize the same capacity region.

Next we offset flow-2 by 2 slots. Namely, the two flows are non-synchronized now:

$$(\mathsf{offset}_1, \mathsf{prd}_1, \mathsf{D}_1, \mathsf{B}_1, p_1) = (0, 3, 3, 1, 0.8),$$
$$(\mathsf{offset}_2, \mathsf{prd}_2, \mathsf{D}_2, \mathsf{B}_2, p_2) = (2, 3, 3, 1, 0.6).$$

The idle-time-based analysis does not hold anymore. However, our MDP-based computation $\mathcal{R}$ in (5.4) can still characterize the capacity region, see Fig. 7.1(b), which contains three corner points, as opposed to only two corner points in Fig. 7.1(a). Such a phenomenon is observed for the first time in the literature.

In both Figs. 7.1(a) and 7.1(b), we also evaluate our fast outer bound $\mathcal{R}^{\mathsf{outer}}$ in (6.2). We can see that empirically it is a reasonably tight outer bound of the capacity region.



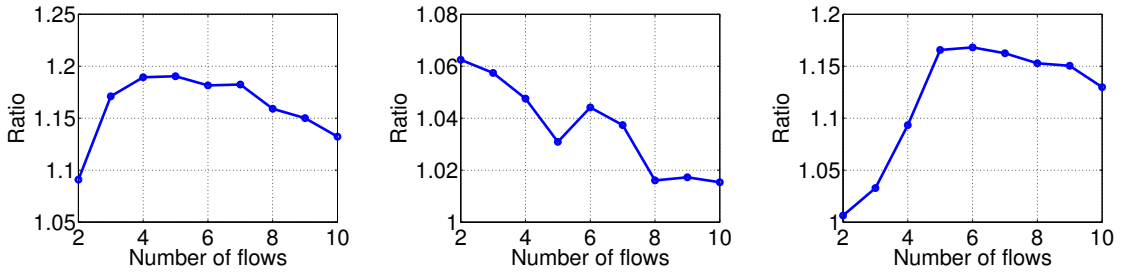
(a) Two synchronized flows

(b) Two non-synchronized flows

Figure 7.1: Capacity regions of two examples in Sec. 7.1.

We further show the gap between our outer bound and the capacity region as follows. We consider the linear utility function $U_k(R_k) = w_k R_k$. For any particular weight vector $\vec{w} = (w_1, w_2, \cdots, w_K)$, we solve the RAC problem (**P2**) and the RAC-Approx problem (**P3**), respectively. We denote the optimal value/utility of (**P2**) (resp. (**P4**)) as $u_2^*(\vec{w})$ (resp. $u_4^*(\vec{w})$). We then define $r(\vec{w}) \triangleq \frac{u_4^*(\vec{w})}{u_2^*(\vec{w})}$, which measures the gap between the outer bound and the capacity region *in the direction* $\vec{w}$. Now we define the *ratio of the outer bound to the capacity region* as $r^* \triangleq \max_{\vec{w} \in \mathbb{R}_+^K} r(\vec{w})$ where $\mathbb{R}_+^K$ is the set of all nonnegative directions, i.e., $\mathbb{R}_+^K \triangleq \{\vec{w} = (w_1, w_2, \cdots, w_K) : w_k \geq 0, \forall k \in [1, K]\}$. If $r^* = 1$, then the outer bound is exactly the capacity region, and smaller r^* means tighter outer bound. We thus use r^* to measure the gap between our outer bound and the actual capacity region.

We measure r^* by using Monte Carlo method, i.e., randomly generating 1000 different direction $\vec{w}$'s. We show *ratio vs. number of flows* results in Fig. 7.2 for three different traffic patterns. As we can see, the ratio r^* does not monotonically change when the number of flows increases. When we compare the ratios for different cases with different flows, the traffic patterns may not change in a "monotonic" way though all flows in those different cases share similar A&E profile. Thus, we may not be able to observe the monotonicity. But we can observe the decreasing trend of r^* when the number of flows becomes larger. Note that due to the high complexity, it requires more significant computing resources than those we can access to solve the RAC problem (**P2**) for more than 10 flows, and thus



(a) Case 1: $(\mathsf{offset}_k, \mathsf{prd}_k, \mathsf{D}_k, \mathsf{B}_k, p_k) = (0, 2, 2, 0.5, 0.8)$ for all k.

(b) Case 2: $(\mathsf{offset}_k, \mathsf{prd}_k, \mathsf{D}_k, \mathsf{B}_k, p_k) = (0, 3, 3, 1, p_k)$ for all k where $p_k = 0.8$ (resp. 0.6) for odd (resp. even) k.

(c) Case 3: $(\mathsf{offset}_k, \mathsf{prd}_k, \mathsf{D}_k, \mathsf{B}_k, p_k) = (\mathsf{offset}_k, 3, 3, 0.5, 0.8)$ for all k where $\mathsf{offset}_k = 0$ (resp. 1) for odd (resp. even) k.

Figure 7.2: Ratio of the outer bound to the capacity region, defined as $r^* \triangleq \max_{\vec{w}=(w_1,w_2,\cdots,w_K)\in\mathbb{R}_+^K} \frac{u_4^*(\vec{w})}{u_2^*(\vec{w})}$, where $u_2^*(\vec{w})$ (resp. $u_3^*(\vec{w})$) is the optimal value/utility of (**P2**) (resp. (**P4**)) if taking utility function $U_k(R_k) = w_k R_k$ for all k.

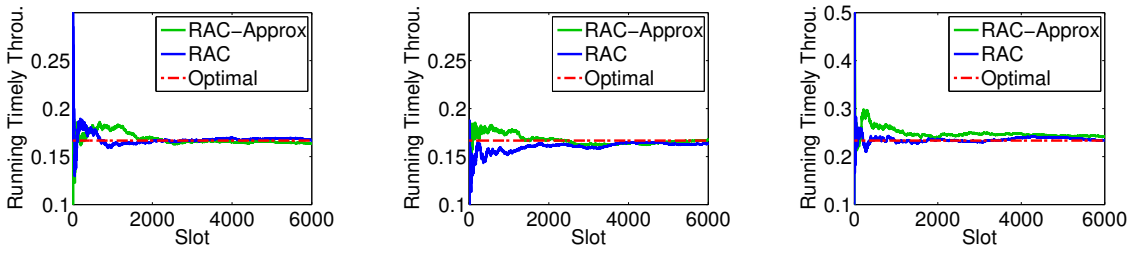
we only show the results up to 10 flows.

7.2 Maximizing Network Utility

In this subsection we evaluate the two proposed scheduling policies to maximize the network utility: one is the provably optimal RAC scheduling policy; the other is the low-complexity heuristic RAC-Approx scheduling policy. We show their performances by considering the following 3-flow traffic pattern:

$$(\mathsf{offset}_1, \mathsf{prd}_1, \mathsf{D}_1, \mathsf{B}_1, p_1) = (0, 4, 4, 1, 0.5),$$
$$(\mathsf{offset}_2, \mathsf{prd}_2, \mathsf{D}_2, \mathsf{B}_2, p_2) = (2, 4, 4, 1, 0.5),$$
$$(\mathsf{offset}_3, \mathsf{prd}_3, \mathsf{D}_3, \mathsf{B}_3, p_3) = (0, 1, 3, 0.9, 0.7).$$

We first set the utility functions as $U_k(R_k) = \log R_k, \forall k \in [1, 3]$. Note that here flow 3 is simply the traditional i.i.d. arrival since $\mathsf{prd}_3 = 1$. By solving (**P2**), we get the optimal timely throughput vector $(R_1^*, R_2^*, R_3^*) = (0.1667, 0.1667, 0.2333)$, which maximizes the network utility. To see concretely how our optimal RAC policy works, in Appendix 9.6 in the supplementary materials, we also show the conditional probability $\mathsf{Prob}_{A_t|S_t^k}(a|s^k)$



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

Figure 7.3: Comparison of different scheduling policies for maximizing network utility for three flows in Sec. 7.2 with utility functions, $U_1(R_1) = \log(R_1)$, $U_2(R_2) = \log(R_2)$, and $U_3(R_3) = \log(R_3)$. Both RAC and RAC-Approx scheduling policies converge to the optimal solution.

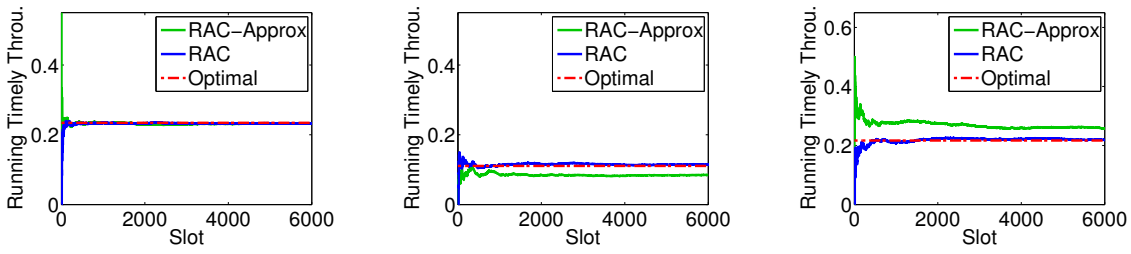
of the optimal RAC policy (see (5.5)). Fig. 7.3 shows that both RAC and RAC-Approx converge to the optimal solution. This verifies the optimality of RAC scheduling policy and demonstrates the good empirical performance of RAC-Approx scheduling policy. Note that in Fig. 7.3 and later figures in this section, we define the flow-k *running timely throughput* at slot t as

$$\frac{1}{t} \cdot \{\#\text{ of flow-}k\text{ pkts delivered before expiration in } [1, t]\}.$$

We also evaluate RAC and RAC-Approx policies with different utility functions. Specifically, we set $U_1(R_1) = 2\sqrt{R_1}$, $U_2(R_2) = \sqrt{R_2}$, and $U_3(R_3) = \sqrt{R_3}$. The results are shown in Fig. 7.4. As we can see, our provably optimal RAC policy again converges to the optimal timely throughput vector $(R_1^*, R_2^*, R_3^*) = (0.2344, 0.1107, 0.2169)$ with optimal utility $u^* = 2\sqrt{R_1^*} + \sqrt{R_2^*} + \sqrt{R_3^*} = 1.7667$. Our proposed low-complexity RAC-Approx policy converges to a sub-optimal timely throughput vector $(\bar{R}_1, \bar{R}_2, \bar{R}_3) = (0.2328, 0.0848, 0.2573)$ with utility $\bar{u} = 2\sqrt{\bar{R}_1} + \sqrt{\bar{R}_2} + \sqrt{\bar{R}_3} = 1.7634$. Though RAC-Approx does not achieve the optimal utility, it has quite good performance with a utility ratio $\bar{u}/u^* = 99.81\%$.

7.3 Supporting Feasible Timely Throughput Vectors

In this subsection we evaluate the three proposed scheduling polices to support feasible throughput vectors: the first is the provably optimal RAC scheduling policy; the second is the low-complexity heuristic RAC-Approx scheduling policy; the last is the low-complexity



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

Figure 7.4: Comparison of different scheduling policies for maximizing network utility for three flows in Sec. 7.2 with utility functions, $U_1(R_1) = 2\sqrt{R_1}$, $U_2(R_2) = \sqrt{R_2}$, and $U_3(R_3) = \sqrt{R_3}$. RAC scheduling policy converges to the optimal solution and RAC-Approx scheduling policy converges to a near-optimal solution which achieves 99.81% of the optimal utility.

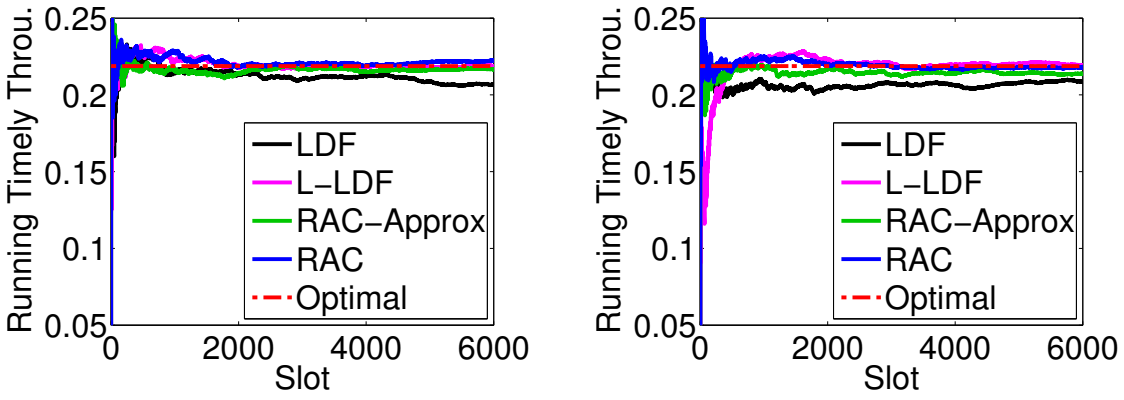
deficit-based L-LDF scheduling policy. We will compare them to existing LDF [53] and EPDF [58] scheduling policies.

Since all these scheduling policies require a feasible timely throughput vector as an input, we will also set a utility function $U_k(R_k)$ for each flow k and solve (**P2**) to get a timely throughput vector on the boundary of the capacity region. We then use it as the input to the scheduling policies. In the following, we use two simple scenarios to show that both LDF and EPDF can be strictly suboptimal while our proposed RAC policy is guaranteed to achieve optimality in all scenarios. Our heuristic solutions RAC-Approx and L-LDF also outperform LDF and EPDF in these two examples.

LDF is Sub-optimal: Consider a 2-flow case with

$$(\mathsf{offset}_1, \mathsf{prd}_1, \mathsf{D}_1, \mathsf{B}_1, p_1) = (0, 4, 4, 1, 0.5), U_1(R_1) = R_1,$$
$$(\mathsf{offset}_2, \mathsf{prd}_2, \mathsf{D}_2, \mathsf{B}_2, p_2) = (2, 4, 4, 1, 0.5), U_2(R_2) = R_2.$$

By solving (**P2**), the optimal timely throughput vector is $(R_1^*, R_2^*) = (0.2187, 0.2187)$. We use (R_1^*, R_2^*) as the timely throughput requirements for the scheduling policies to be evaluated. Fig. 7.5 shows their performances. As we can see, RAC converges to (R_1^*, R_2^*) as proven in Theorem 5.1. Our proposed heuristics, RAC-Approx and L-LDF, also converge to (R_1^*, R_2^*). Meanwhile, the LDF algorithm cannot support this particular timely throughput vector; indeed, the achieved rates for both flows are strictly smaller



(a) Flow 1 (b) Flow 2

Figure 7.5: An example showing that LDF is strictly sub-optimal.

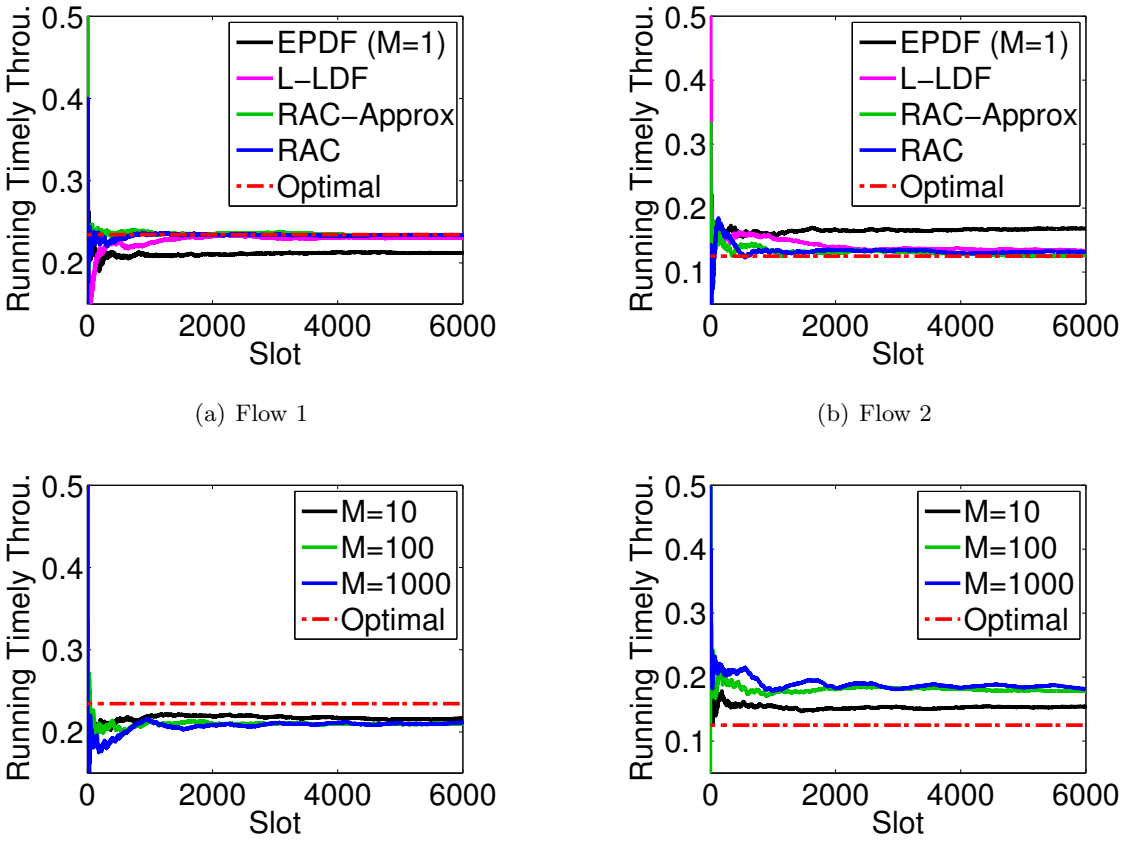
than (R_1^*, R_2^*).

EPDF is Sub-optimal: Consider a 2-flow case with

$$(\mathsf{offset}_1, \mathsf{prd}_1, \mathsf{D}_1, \mathsf{B}_1, p_1) = (0, 4, 4, 1, 0.5),$$
$$(\mathsf{offset}_2, \mathsf{prd}_2, \mathsf{D}_2, \mathsf{B}_2, p_2) = (0, 4, 3, 1, 0.5).$$

We set the utility function as $U_k(R_k) = w_k R_k$ with weights $(w_1, w_2) = (1, 10^{-5})$. Choosing $w_2 = 10^{-5}$ means that we give absolute priority to flow 1. The optimal rate is $(R_1^*, R_2^*) = (0.2344, 0.1250)$, which we input to all scheduling policies. Fig. 7.6(a) and Fig. 7.6(b) show that the achieved timely throughputs of our proposed RAC, RAC-Approx, and L-LDF scheduling policies all converge to (R_1^*, R_2^*); hence, they can all support this timely throughput vector. On the contrary, EPDF cannot support this particular timely throughput vector, and thus is strictly sub-optimal. The observation holds for a wide range of different M values as shown in Fig. 7.6(c) and Fig. 7.6(d) where M is a tuning parameter of EPDF [58]. The reason is as follows. The choice of $U_1(R_1)$ and $U_2(R_2)$ implies that to achieve the optimal (R_1^*, R_2^*), we must always give priority to flow 1. However, in EPDF, the *periodic virtual injection of every M time slots* ensures that for a constant fraction of time slots, the deficit of flow 2 will be strictly positive. Since flow 2 has an earlier deadline, EPDF will favor flow 2 for a constant fraction of time slots. This is strictly sub-optimal since an optimal policy must always give precedence to flow 1.

In both Fig. 7.5 and Fig. 7.6, we verify that our RAC scheduling policy is feasibility-



(a) Flow 1

(b) Flow 2

(c) Flow 1 under EPDF

(d) Flow 2 under EPDF

Figure 7.6: An example showing that EPDF is strictly sub-optimal.

optimal. We also show that, for the instances considered in this set of simulations, our proposed low-complexity heuristic RAC-Approx and L-LDF scheduling polices support the given timely throughput vector and thus outperform existing alternatives.

7.4 Average Performance Comparison of Scheduling Policies Over A Large Number of Problem Instances

In this subsection, we compare the performance of the two scheduling policies for maximizing network utility, RAC and RAC-Approx, and five scheduling policies for supporting feasible timely throughput vectors, RAC, RAC-Approx, LLDF, LDF and EPDF, over a large number of randomly generated problem instances with up to 10 flows.

Table 7.1: Performance comparison of two scheduling polices for maximizing network utility, in terms of the average utility gap $\delta_1(u, u^*)$ over 1000 problem instances.

Policy \ K	2	4	6	8	10
RAC	0.00%	0.00%	0.01%	0.01%	0.02%
RAC-Approx	0.82%	1.36%	1.76%	2.82%	3.31%

Our experiments consider K flows where $K \in \{2, 4, 6, 8, 10\}$ and we randomly generate the A&E profile and the successful delivery probability for any flow $k \in [1, K]$, i.e., $(\mathsf{offset}_k, \mathsf{prd}_k, \mathsf{D}_k, \mathsf{B}_k, p_k)$, where offset_k and prd_k are integers uniformly drawn from $[1, 5]$, D_k is an integer uniformly drawn from $[1, \mathsf{prd}_k]$, and B_k and p_k are real numbers uniformly drawn from $[0.5, 1]$. For each flow k, we choose the utility function $U_k(R_k) = \log R_k$.

For the utility-maximization problem, we first solve the utility-maximization problem (**P2**) and get the optimal network utility u^*. We then evalute the empirical network utility u for RAC and RAC-Approx scheduling polices. We measure the utility gap by

$$\delta_1(u, u^*) \triangleq \frac{u^* - u}{|u^*|}.$$

To evaluate whether a scheduling policy is feasibility-optimal, i.e., capable of supporting any feasible throughput vector in the timely capacity region, we first solve the utility-maximization problem (**P2**) and get the optimal rate vector $\vec{R}^* = (R_1^*, R_2^*, \cdots, R_K^*)$. We input $\vec{R}^*$ as the timely throughput requirements for RAC-Approx (see (**P3**)), LDF, EPDF, and LLDF scheduling policies. We then evaluate the empirical timely throughput vector $\vec{R}$ for RAC, RAC-Approx, LDF, EPDF, and LLDF scheduling policies. We measure the throughput gap by

$$\delta_2(\vec{R}, \vec{R}^*) \triangleq \frac{\sum_{k=1}^{K} [R_k^* - R_k]^+}{\sum_{k=1}^{K} R_k^*},$$

where $[x]^+ \triangleq \max\{x, 0\}$.

For each $K \in \{2, 4, 6, 8, 10\}$, the empirical performance of each instance is measured over 1000000 time slots and we repeat the experiment for 1000 problem instances. We run all evaluations in MATLAB in a cluster of 40 Linux servers, each of which has an 8-core Intel Core-i7 3770 3.4Ghz CPU and up to 61GB memory, running CentOS 6.4. We compute the average utility gap $\delta_1(u, u^*)$ for the two scheduling policies for maximizing

Table 7.2: Performance comparison of five scheduling polices for supporting feasible timely throughput vector, in terms of the average throughput gap $\delta_2(\vec{R}, \vec{R}^*)$ over 1000 problem instances.

Policy \ K	2	4	6	8	10
RAC	0.04%	0.06%	0.09%	0.14%	0.16%
LLDF	0.85%	1.31%	0.80%	1.13%	1.20%
LDF	1.91%	3.34%	1.95%	1.76%	1.76%
RAC-Approx	1.77%	3.94%	5.25%	6.24%	6.43%
EPDF	1.81%	6.02%	6.27%	9.38%	8.99%

network utility, RAC and RAC-Approx, as shown in Tab. 7.1. We compute the average rate gap $\delta_2(\vec{R}, \vec{R}^*)$ for the five scheduling policies for supporting feasible timely throughput vectors, RAC, RAC-Approx, LDF, EPDF, and LLDF, as shown in Tab. 7.2.

For maximizing network utility, Tab. 7.1 verifies that our proposed high-complexity RAC policy achieves the optimal network utility, and also shows that our proposed low-complexity RAC-Approx policy achieves near-optimal performance. For supporting feasible timely throughput vector, Tab. 7.2 verifies that our proposed high-complexity RAC policy is feasibility-optimal. Our proposed L-LDF policy achieves the smallest throughput gap among the remaining four low-complexity scheduling policies.

Chapter 8

Conclusion and Future Work

In this part (Chap. 1–9), we study three fundamental problems of timely wireless flows under *general traffic patterns*: capacity region problem, network utility maximization problem and feasibility-optimal policy design problem. All of them remained largely open. We propose a new MDP-based framework to formulate the timely wireless flow problem with general traffic patterns, which allows us to systematically explore the full design space beyond the existing synchronized-frame-based studies. By applying two problem-structure-inspired simplification methods, *for the first time* we show that all these three fundamental problems can be solved in principle though suffering the curse of dimensionality. Therefore, this thesis serves as a benchmark to evaluate any scheduling policies for timely wireless flows under general traffic patterns. We also take a first step toward addressing the curse of dimensionality by proposing two low-complexity heuristic solutions. Simulation results show that they achieve near-optimal performance and outperform existing alternatives.

For the future work, an interesting and important direction is to design efficient scheduling algorithms with performance guarantee for general traffic patterns. To achieve such goal, one possibility is to apply the approximate MDP (or more general approximate dynamic programming) solutions [82, 36, 48, 78] to our timely wireless flow problem. However, to the best of our knowledge, there does not exist direct results that can address the curse of dimensionality while providing performance guarantee for our timely wireless flow problem. Thus, it requires further efforts to adapt existing works on approximate MDP to our problem.

Another practical direction is to study how to generalize our solution to the scenario that the channel successful-delivery probability is unknown and/or the traffic pattern is unknown. In our work, we assume that both the channel successful-delivery probability and the traffic pattern are known a prior to the system operator. However, in practice,

both of them could be unknown but need to be learned during the transmission/scheduing phase. It is important to design algorithms to simultaneously learn these system parameters and do scheduling so that our solution can be applied to more practical scenarios.

Chapter 9

Appendix

9.1 Proof of Theorem 3.1

We turn to prove the hardness of the corresponding decision problem of (**P1**): given x, determine whether there exists a scheduling policy such that $\sum_{k=1}^{K} U_k(R_k) \geq x$. If we take $x = \sum_{k=1}^{K} U_k\left(\frac{1}{\mathsf{prd}_k}\right)$, since $R_k \leq \frac{1}{\mathsf{prd}_k}$, the decision problem is equivalent to the following feasibility-check problem: given timely throughput vector $\vec{R} = (R_1, \cdots, R_K) = (\frac{1}{\mathsf{prd}_1}, \cdots, \frac{1}{\mathsf{prd}_K})$, determine whether $\vec{R}$ is in the capacity region or not.

Note that our timely wireless flow problem can be regarded as a stochastic version of the preemptive scheduling problem of periodic real-time tasks on one processor [26]. In [26], the authors proved that the feasibility-check problem for arbitrary periodic real-time tasks on one processor is co-NP-hard in the strong sense. We largely follow their results to prove our Theorem 3.1.

In [26], authors leveraged the hardness result of the Simultaneous Congruences Problem (SCP): given a set $A = \{(a_1, b_1), (a_2, b_2), \cdots, (a_n, b_n)\} \subset \mathbb{N} \times \mathbb{N}^+$ and an integer $k \in [2, n]$, determine whether there are a subset $A' \subset A$ of k pairs and a positive integer x such that for every $(a_i, b_i) \in A'$, $x \equiv a_i \pmod{b_i}$. The hardness of SCP is shown in the following lemma.

Lemma 9.1 (Theorem 3.2, [26])**.** *SCP is NP-complete in the strong sense.*

Now for any instance of SCP, i.e., $A = \{(a_1, b_1), (a_2, b_2), \cdots, (a_n, b_n)\} \subset \mathbb{N} \times \mathbb{N}^+$ and an integer $k \in [2, n]$, we construct (in polynomial time) the following instance of our feasibility-check problem: for any $i \in [1, K]$, let

$$\begin{aligned} &(\mathsf{offset}_i, \mathsf{prd}_i, \mathsf{D}_i, \mathsf{B}_i) = ((k-1)a_i, (k-1)b_i, k-1, 1), \\ &p_i = 1, R_i = \frac{1}{\mathsf{prd}_i} = \frac{1}{(k-1)b_i}. \end{aligned} \tag{9.1}$$

Next we show that the timely throughput vector $\vec{R} = (R_1, \cdots, R_K) = (\frac{1}{(k-1)b_1}, \cdots, \frac{1}{(k-1)b_K})$ is in the capacity region if and only if the SCP answers No.

"If" Since the SCP answers No, then there are at most $k-1$ flows with a packet arrival at any slot. At slot 1, there are at most $k-1$ packets arriving in the system. All these packets can be scheduled with any work-conserving policy in the next $k-1$ slots during which no new packets arrive (since both $\mathsf{offset}_i = (k-1)a_i$ and $\mathsf{prd}_i = (k-1)b_i$ are a multiple of $k-1$ for any $i \in [1, K]$). After $k-1$ slots (i.e., at slot k), again there are at most $k-1$ packets arriving to the system, and thus the process can repeat. Therefore, all packets of all K flows can be delivered successfully before expiration. This yields to flow-i's timely throughput $\frac{1}{(k-1)b_i} = R_i$.

"Only if" Suppose that the SCP answers Yes, i.e, there exist a subset $A' \subset A$ of k pairs and a positive integer x such that $x \equiv a_i \pmod{b_i}, \forall (a_i, b_i) \in A'$, which equivalently means that $(k-1)x \equiv (k-1)a_i \pmod{(k-1)b_i}, \forall (a_i, b_i) \in A'$. Also, if we denote the common period $M = \text{Least.Common.Multiple}\{(k-1)b_i : (a_i, b_i) \in A'\}$, we easily obtain that $(k-1)x + jM \equiv (k-1)a_i \pmod{(k-1)b_i}, \forall (a_i, b_i) \in A'$ for any $j \in \mathbb{N}$. Thus, at slot $(k-1)x + jM + 1 (\forall j \in \mathbb{N})$, at least k packets arrive in the system, all of which had a deadline $k-1$. Thus, at least 1 packet will be expired every M slots, which means that at least a $\frac{1}{M} > 0$ timely throughput will be lost among all K flows. Therefore, $\vec{R} = (R_1, \cdots, R_K) = (\frac{1}{(k-1)b_1}, \cdots, \frac{1}{(k-1)b_K})$ cannot be achieved by any scheduling policy and thus is not in the capacity region.

Parts **"If"** and **"Only if"** prove that the timely throughput vector $\vec{R} = (R_1, \cdots, R_K) = (\frac{1}{(k-1)b_1}, \cdots, \frac{1}{(k-1)b_K})$ is in the capacity region if and only if the SCP answers No. Since SCP is NP-complete as shown in Lemma 9.1, we proves that it is co-NP-hard to determine whether or not the timely throughput vector $\vec{R} = (R_1, \cdots, R_K) = (\frac{1}{\mathsf{prd}_1}, \cdots, \frac{1}{\mathsf{prd}_K}) = (\frac{1}{(k-1)b_1}, \cdots, \frac{1}{(k-1)b_K})$ is in the capacity region. This completes the proof that the corresponding decision problem of (**P1**) is co-NP-hard in the strong sense and thus (**P1**) is also co-NP-hard in the strong sense.

9.2 Proof of Lemma 5.1

We notice that for any slot t, the following conditional independence relation among different flows holds,

$$P_t((\tilde{s}^1, \cdots, \tilde{s}^K)|(s^1, \cdots, s^K), a) = \prod_{k=1}^{K} P_t^k(\tilde{s}^k|s^k, a). \tag{9.2}$$

To show that

$$\begin{aligned} &P_t((\tilde{s}^1, \cdots, \tilde{s}^K)|(s^1, \cdots, s^K), a) = \\ &P_{t'}((\tilde{s}^1, \cdots, \tilde{s}^K)|(s^1, \cdots, s^K), a), \end{aligned} \tag{9.3}$$

where $t = l \cdot \mathsf{Prd} + \tau$ and $t' = (l+1) \cdot \mathsf{Prd} + \tau$, it suffices to show that

$$P_t^k(\tilde{s}^k|s^k, a) = P_{t'}^k(\tilde{s}^k|s^k, a), \forall k \in [1, K], s^k \in \mathcal{S}^k, a \in \mathcal{A}. \tag{9.4}$$

In the following, we will prove (9.4). Let us focus on an arbitrary $k \in [1, K]$. The proof will be divided into two steps. The first step is to show that the set of all possible states of flow k at slot t is *equal* to the set of all possible states of flow k at slot t', both of which are a subset of $\mathcal{S}^k$. The second step is to show that the transition probabilities at slots t and t' from any state $s = l_1^k l_2^k \cdots l_{\mathsf{D}_k}^k$ to any state $\tilde{s} = \tilde{l}_1^k \tilde{l}_2^k \cdots \tilde{l}_{\mathsf{D}_k}^k$ under the same action a are the same.

Step 1: (i) For any state in t, we can prove that it is also a state in t'. More specifically, if $l_i^k = 1$ under the state in t, we denote the corresponding packet as m. Clearly, if packet $m+1$ arrived the system and has not been delivered at t', the state at slot t' also has $l_i^k = 1$. Similar reasoning can be applied for $l_i^k = 0$. This proves that any state in t is also a possible state in t'. (ii) On the other hand, if $l_i^k = 1$ under the state in t', we denote the corresponding packet as m. Note that $t' \geq \tau + \mathsf{Prd}$, which means that[1] $m \geq 3$. Then, if packet $m - 1 \geq 2$ arrived the system and had not been delivered at t, the state at slot t also had $l_i^k = 1$. Similar reasoning can be applied for $l_i^k = 0$. This proves that any state in t' is also a possible state in t. Thus, (i) and (ii) show that the sets of all possible state at slots t and t' are the same.

[1] See the definition of L in Lemma 5.1.

Step 2: The transition from state s to next state s' consists of three independent parts: (i) transmission, (ii) lead time evolution (including expiration), and (iii) arrival. Denote the lead-time-based binary string representation[2] after part (i) by $s_1 = m_1^k m_2^k \cdots m_{\mathsf{D}_k}^k$, after parts (i) and (ii) by $s_2 = n_1^k n_2^k \cdots n_{\mathsf{D}_k}^k$, and after parts (i), (ii) and (iii) by $s_3 = o_1^k o_2^k \cdots o_{\mathsf{D}_k}^k$. Part (i) characterizes the effect of transmission. If $l_i^k = 1$ and the corresponding packet is scheduled for transmission under action a, then $m_i^k = 0$ with probability p_k and $m_i^k = 1$ with probability $1 - p_k$. If the corresponding packet for the bit l_i^k is not scheduled for transmission under action a, then $m_i^k = l_i^k$. Part (ii) characterizes the evolution of lead time. From the beginning of slot t to the beginning of slot $(t+1)$, the lead time of every packet will decrease by 1 and the packet with lead time 1 at slot t will expire at slot $(t+1)$. This evolution is equivalent to a left-shift operation for s_1. Namely, $s_2 = \mathsf{left\text{-}shift}(s_1)$, or $s_2 = n_1^k n_2^k \cdots n_{\mathsf{D}_k}^k = m_2^k m_3^k \cdots m_{\mathsf{D}_k}^k 0$. Part (iii) characterizes the effect of new arrival. At the beginning of slot $(t+1)$, it is possible that a new packet will come to the system if $(t+1)$ is the arrival time of some packet of flow k (see (3.1)). Note that the new arrived packet will always have lead time D_k and thus it only affects the last bit, i.e., $o_{\mathsf{D}_k}^k$. More specifically, if $(t+1)$ is not the arrival time of some packet, then $o_{\mathsf{D}_k}^k = 0$. If $(t+1)$ is the arrival time of some packet, then $o_{\mathsf{D}_k}^k = 1$ with probability B_k and $o_{\mathsf{D}_k}^k = 0$ with probability $1 - \mathsf{B}_k$. Other bits remain unchanged from part (ii) to part (iii), i.e., $s_3 = o_1^k o_2^k \cdots o_{\mathsf{D}_k - 1}^k o_{\mathsf{D}_k}^k = n_1^k n_2^k \cdots n_{\mathsf{D}_k - 1}^k o_{\mathsf{D}_k}^k$.

From the above analysis, we can easily see that parts (i) and (ii) are time-invariant and part (iii) is periodic with period Prd, which more specifically means that the arrival probability of a new packet at the beginning of slot $(t+1)$ is the same as that of at the beginning of slot $(t' + 1 = t + \mathsf{Prd} + 1)$. Therefore, the transition probabilities at slots t and t' from any state $s = l_1^k l_2^k \cdots l_{\mathsf{D}_k}^k$ to any state $\tilde{s} = \tilde{l}_1^k \tilde{l}_2^k \cdots \tilde{l}_{\mathsf{D}_k}^k$ under the same action a are the same.

Step 1 and *Step 2* complete the proof for (9.4) and the whole proof of Lemma 5.1 is thus completed.

[2]Here we use different letters, i.e., m, n and o to represent the letter l for ease of exposition. They should be clear under the context here.

9.3 Proof of Theorem 5.1

For ease of exposition, we define the rate vector $\vec{R} \triangleq (R_1, R_2, \cdots, R_k)$ and the following two regions:

$$\mathcal{R}^{\text{RAC}} \triangleq \{\vec{R} : \vec{R} \text{ can be achieved by some RAC scheduling policy}\},$$

$$\mathcal{R}^{\text{LP}} \triangleq \{\vec{R} : \vec{R} \text{ (together with } \vec{x}) \text{ is a feasible solution of (5.4a)} - \text{(5.4e)}\}.$$

Next we will prove the following two claims. **Claim 1:** $\mathcal{R}^{\text{RAC}} = \mathcal{R}^{\text{LP}}$. **Claim 2:** $\mathcal{R} = \mathcal{R}^{\text{LP}}$.

Clearly, **Claim 2** completes the proof for part (ii) of Theorem 5.1. And **Claim 1** & **Claim 2** show that $\mathcal{R} = \mathcal{R}^{\text{RAC}}$, which means that RAC can achieve any feasible rate vector. This completes the proof for part (i) of Theorem 5.1. Part (iii) of Theorem 5.1 will be proved when we prove **Claim 1**. The whole Theorem 5.1 is thus proved.

9.3.1 Proof of Claim 1: $\mathcal{R}^{\text{RAC}} = \mathcal{R}^{\text{LP}}$

Step 1: Show that $\mathcal{R}^{RAC} \subset \mathcal{R}^{LP}$. This is trivially true because of Definition 5.1 for an RAC scheduling policy. In Definition 5.1, for any RAC scheduling policy π, condition (i) specifies a conditional probability $\mathsf{Prob}_{A_t|S_t}(a|s)$, and condition (ii) specifies a series of state distribution $\mathsf{Prob}_{S_t}(s)$ where $\mathsf{Prob}_{S_{T_2+1}}(s) = \mathsf{Prob}_{S_{T_1+\mathsf{Prd}}}(s) = \mathsf{Prob}_{S_{T_1}}(s)$. Now if we define $x_t(s,a) = \mathsf{Prob}_{A_t|S_t}(a|s) \cdot \mathsf{Prob}_{S_t}(s), \forall t \in [T_1, T_2]$ and define $\vec{R}$ as the achieved rate vector of π, we can easily check that $\vec{R}$ together with $\vec{x}$ is a feasible solution of (5.4a) – (5.4e). This proves that $\mathcal{R}^{\text{RAC}} \subset \mathcal{R}^{\text{LP}}$.

Step 2: Show that $\mathcal{R}^{LP} \subset \mathcal{R}^{RAC}$. We will base on the solution of the LP (5.4a) – (5.4e), denoted by $x_t(s,a)$ and $\vec{R}$, to construct an RAC scheduling policy such that the constructed RAC can achieve the rate vector $\vec{R}$. Note that the following construction is also the complete design of the RAC scheme in (5.5) and (5.6), and will also be used to prove part (iii) of Theorem 5.1. In the following, we define $\mathsf{Prob}_{A_t|S_t}(a|s) = \frac{x_t(s,a)}{\sum_{a' \in \mathcal{A}} x_t(s,a')}$ and $\mathsf{Prob}_{S_t}(s) = \sum_{a \in \mathcal{S}} x_t(s,a)$, same in (5.5) and (5.6), respectively. The construction contains two phases.

Phase 1: Initialization. The goal of this phase is to make the state distribution at slot T_1 to be the steady state distribution $\mathsf{Prob}_{S_{T_1}}(s)$ with the help of artificial *dummy packets*.

At the beginning of slot $T_0 \triangleq 1 + \max_{k\in[1,K]} \mathsf{offset}_k$, the arrival process of all flows has started. We first remove all arrived packets and then insert dummy packets randomly into the system such that the probability that the network state at slot T_0 is s is $\mathsf{Prob}_{S_{g(T_0)}}(s)$ for all $s \in \mathcal{S}$, where $g(T_0) = L \cdot \mathsf{Prd} + [1 + ((T_0 - 1) \mod \mathsf{Prd})]$ is the corresponding same-position slot in the optimized period $[T_1, T_2]$ in (5.4a) – (5.4e). Note that to achieve such goal, we should set the lead time of the dummy packets carefully (and randomly), but any inserted flow-k dummy packet will expire at/before slot $T_0 + \mathsf{D}_k \leq T_1$.

Phase 2: Making Scheduling Decisions. At the beginning of any slot $t \geq T_0$, we make the decision in the following way. Define $g(t) \triangleq L \cdot \mathsf{Prd} + [1 + ((t-1) \mod \mathsf{Prd})]$ as the corresponding same-position slot in the optimized period $[T_1, T_2]$ in (5.4a) – (5.4e). Let s^t denote the current network state at time t, which counts both real packets and dummy packets. We use a *random scheduler* that chooses the flow k with probability

$$\mathsf{Prob}_{A_{g(t)}|S_{g(t)}}(k|s_t). \tag{9.5}$$

One can easily see that the resulting random process of the network state, i.e., $\{S_t : \forall t = 1, 2, \cdots\}$, is cyclostationary with period Prd, due to (5.4a) and (5.4b). Thus, the network state at slot $T_1 = L \cdot \mathsf{Prd} + 1$ also has the same distribution of the network state at slot 1, i.e., $\{\mathsf{Prob}_{S_{T_1}}(s) : s \in \mathcal{S}\}$. Note that here the network state $\{S_t\}$ counts both real packets and dummy packets. However, since any inserted flow-k dummy packet will expire at/before slot T_1 as shown in **Phase 1**, from slot T_1 on, only real packets exist in the system.

Now let us focus on the time interval $t \in [T_1, \infty)$. We can see that the scheduling strategy (9.5) satisfies condition (i) in Definition 5.1. And the random process of the network state, i.e., $\{S_{\tau_{\text{trans}}+t} : \forall t = 1, 2, \cdots\}$, is cyclostationary with period Prd. This shows that condition (ii) in Definition 5.1 also holds. Therefore, our two-phase construction indeed produces an RAC scheduling policy. Further, we can readily see that the constructed RAC can achieve the rate vector[3] $\vec{R}$ due to (5.4c), completing the proof for part (iii) of Theorem 5.1 and also the proof for $\mathcal{R}^{\text{LP}} \subset \mathcal{R}^{\text{RAC}}$.

Step 1 and *Step 2* complete the proof for **Claim 1**.

[3]We can ignore the finite transient duration $[1, \tau_{\mathsf{trans}}] = [1, T_1 - 1]$ but only focus on the time interval $[T_1, \infty)$ to calculate the (long-term) timely throughput.

9.3.2 Proof of Claim 2: $\mathcal{R} = \mathcal{R}^{\mathbf{LP}}$

Step 1: Show that $\mathcal{R}^{LP} \subset \mathcal{R}$. This is trivially true because $\mathcal{R}^{\mathrm{LP}} = \mathcal{R}^{\mathrm{RAC}} \subset \mathcal{R}$ from **Claim 1**.

Step 2: Show that $\mathcal{R} \subset \mathcal{R}^{LP}$. We prove this by showing that for any scheduling policy Ψ that achieves rate vector $\vec{R} = (R_1, \cdots, R_K) \geq 0$, we can always find a $\vec{x} = \{x_t(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}, t \in [T_1, T_2]\}$ such that $\vec{R}$ and $\vec{x}$ jointly satisfy (5.4a) $-$ (5.4e).

For any scheduling policy Ψ, we choose an arbitrary integer $F \geq 1$ and compute the following quantity for all $t \in [1, \mathsf{Prd}]$, $s \in \mathcal{S}$, and $a \in \mathcal{A}$,

$$y_t^F(s,a) \triangleq \frac{\sum_{f=0}^{F-1} \mathsf{E}\left\{1_{\{(\text{state,action}) = (s,a) \text{ at time } (t + f \cdot \mathsf{Prd})\}}\right\}}{F}, \tag{9.6}$$

where $1_{\{\cdot\}}$ is the indicator function. Intuitively, $y_t^F(s,a)$ in (9.6) quantifies how frequently we are going to see the network state being s and action being a at a (relative-position) slot t under policy Ψ, where the average is performed over both the probability space (due to the expectation operator) and over the time axis (due to the summation over different f's). Obviously, $y_t^F(s,a)$ depends on the underlaying policy Ψ. On the other hand, given Ψ and F, $y_t^F(s,a)$ always exists.

We observe that $y_t^F(s,a)$ in (9.6) is defined for all $t \in [1, \mathsf{Prd}]$. We can also define the $y_t^F(s,a)$ value for $t = \mathsf{Prd} + 1$ in the following way,

$$y_{\mathsf{Prd}+1}^F(s,a) \triangleq \frac{\sum_{f=0}^{F-1} \mathsf{E}\left\{1_{\{(\text{state,action}) = (s,a) \text{ at time } (\mathsf{Prd} + 1 + f \cdot \mathsf{Prd})\}}\right\}}{F}. \tag{9.7}$$

It is worth emphasizing that in general $y_{\mathsf{Prd}+1}^F(s,a) \neq y_1^F(s,a)$ since the given policy Ψ may not be cyclostationary. We will use this statement soon.

Now we define the following

$$x_t^F(s,a) \triangleq y_{t-L\cdot\mathsf{Prd}}^F(s,a), \forall t \in [T_1, T_2], s \in \mathcal{S}, a \in \mathcal{A}. \tag{9.8}$$

Clearly, $x_t^F(s,a)$ is just a time shift of $y_t^F(s,a)$ such that the time indices are within the optimized period $[T_1, T_2]$ in (5.4a) $-$ (5.4e).

We now find a sequence $\{F_n : n \geq 1\}$, which is a subsequence of the positive integer sequence $\{n : n \geq 1\}$, such that $\{y_t^{F_n}(s,a) : n \geq 1\}$ converges for *all* $s \in \mathcal{S}$, $a \in \mathcal{A}$, and $t \in [1, \mathsf{Prd}]$. Toward that end, we first consider a tuple (s,a,t) where $s \in \mathcal{S}$, $a \in \mathcal{A}$, and

$t \in [1, \mathsf{Prd}]$. Since $y_t^n(s,a) \leq 1$, i.e., $\{y_t^n(s,a) : n \geq 1\}$ is a bounded sequence, we can find a convergent subsequence $\{y_t^{F_n^1}(s,a) : n \geq 1\}$. Clearly $\{F_n^1 : n \geq 1\}$ is a subsequence of $\{n : n \geq 1\}$. Now for another tuple (s', a', t') different from (s, a, t), since $\{y_{t'}^{F_n^1}(s', a') : n \geq 1\}$ is a bounded sequence, we can again find a convergent subsequence $\{y_{t'}^{F_n^2}(s', a') : n \geq 1\}$. Clearly $\{F_n^2 : n \geq 1\}$ is a subsequence of $\{F_n^1 : n \geq 1\}$, and both $\{y_t^{F_n^2}(s,a) : n \geq 1\}$ and $\{y_{t'}^{F_n^2}(s', a') : n \geq 1\}$ converge. By iterating over all $|\mathcal{S}| \cdot |\mathcal{A}| \cdot \mathsf{Prd}$ tuples for (s, a, t), we can find sequences $\{F_n^1 : n \geq 1\}$, $\{F_n^2 : n \geq 1\}$, $\cdots$, $\{F_n^{|\mathcal{S}|\cdot|\mathcal{A}|\cdot\mathsf{Prd}} : n \geq 1\}$, where $\{F_n^{i+1} : n \geq 1\}$ is a subsequence of $\{F_n^i : n \geq 1\}$ and $\{F_n^1 : n \geq 1\}$ is a subsequence of $\{n : n \geq 1\}$. We let $F_n = F_n^{|\mathcal{S}|\cdot|\mathcal{A}|\cdot\mathsf{Prd}}$, and thus $\{F_n : n \geq 1\}$ is a subsequence of $\{n : n \geq 1\}$. We also see that $\{y_t^{F_n}(s,a) : n \geq 1\}$ converges for *all* (s, a, t) tuples. We then define $y_t^\infty(s,a) \triangleq \lim_{n\to\infty} y_t^{F_n}(s,a)$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, $t \in [1, \mathsf{Prd}]$. We also define $x_t^\infty(s,a) \triangleq y_{t-\mathsf{Prd}}^\infty(s,a)$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, $t \in [T_1, T_2]$.

Now we will show that $x_t^\infty(s,a)$ and $\vec{R}$ can satisfy (5.4d), (5.4a), (5.4b) and (5.4c), respectively. Eq. (5.4e) is trivially true.

(i) It is easy to show that (5.4d) holds for $x_t^F(s,a)$ for any $F \geq 1$ (see (9.6)) and therefore (5.4d) holds for $x_t^\infty(s,a)$.

(ii) To show that (5.4a) holds, we should notice the following equation due to the total probability theorem where we define events $A(s') = \{\text{state} = s' \text{ at time } (t+1+f\cdot\mathsf{Prd})\}$ and $B(s,a) = \{\text{(state,action)} = (s,a) \text{ at time } (t+f\cdot\mathsf{Prd})\}$,

$$\begin{aligned}\mathsf{E}\{1_{\{A(s')\}}\} = \mathsf{Prob}(A(s')) &= \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \mathsf{Prob}(A(s')|B(s,a))\mathsf{Prob}(B(s,a)) \\ &= \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} P_t(s'|s,a)\mathsf{Prob}(B(s,a)) = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} P_t(s'|s,a)\mathsf{E}\{1_{\{B(s,a)\}}\}. \qquad (9.9)\end{aligned}$$

Now due to the definition of $y_t^F(s,a)$ in (9.6), we get

$$\sum_{a\in\mathcal{A}} y_t^F(s,a) = \frac{\sum_{f=0}^{F-1} \mathsf{E}\left\{1_{\{\text{state} = s \text{ at time } (t+f\cdot\mathsf{Prd})\}}\right\}}{F}. \qquad (9.10)$$

By inserting (9.9) into (9.10) (with slot and state modification) and doing some simple deductions, we get,

$$\sum_{a\in\mathcal{A}} y_{t+1}^F(s',a) = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} P_t(s'|s,a) y_t^F(s,a). \qquad (9.11)$$

This shows that (5.4a) holds for the time-shifted $y_t^F(s,a)$, i.e., $x_t^F(s,a)$, for any $F \geq 1$, and therefore (5.4a) holds for $x_t^\infty(s,a)$.

(iii) To show that (5.4b) holds, we cannot argue that it holds for $x_t^F(s,a)$ for any $F \geq 1$. That is because the following equation is generally not true,

$$y_1^F(s,a) = y_{\mathsf{Prd}+1}^F(s,a), \quad \forall F \geq 1.$$

Instead we will show that

$$y_1^\infty(s,a) = y_{\mathsf{Prd}+1}^\infty(s,a). \tag{9.12}$$

We first quantify the difference between $y_1^F(s,a)$ and $y_{\mathsf{Prd}+1}^F(s,a)$ with the following lemma.

Lemma 9.2. *For any scheduling policy Ψ, we always have*

$$|y_1^F(s,a) - y_{\mathsf{Prd}+1}^F(s,a)| \leq \frac{1}{F}, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}.$$

Proof. By the definitions of (9.6) and (9.7), we have

$$\begin{aligned} y_1^F(s,a) - y_{\mathsf{Prd}+1}^F(s,a) = &\frac{\mathsf{E}\left\{1_{\{\text{(state,action) = } (s,a) \text{ at time } 1\}}\right\}}{F} - \\ &\frac{\mathsf{E}\left\{1_{\{\text{(state,action) = } (s,a) \text{ at time } (1+F\cdot \mathsf{Prd})\}}\right\}}{F}. \end{aligned} \tag{9.13}$$

Denote the right-hand side of (9.13) by $\mathsf{term}_1 - \mathsf{term}_2$. Since both term_1 and term_2 are non-negative, we have

$$|\mathsf{term}_1 - \mathsf{term}_2| \leq \max\{\mathsf{term}_1, \mathsf{term}_2\}.$$

Since the numerators of term_1 and term_2 are upper bounded by 1, we have $\max\{\mathsf{term}_1, \mathsf{term}_2\} \leq \frac{1}{F}$. The proof is thus completed. □

By Lemma 9.2, we thus conclude that (9.12) holds. Then from (9.12) and (9.11), we have

$$\sum_{a\in\mathcal{A}} y_1^\infty(s',a) = \sum_{a\in\mathcal{A}} y_{\mathsf{Prd}+1}^\infty(s',a) = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} P_{\mathsf{Prd}}(s'|s,a) y_{\mathsf{Prd}}^\infty(s,a),$$

which shows that (5.4b) holds for $x_t^\infty(s,a)$.

(iv) Due to the definition of R_k in Sec. 3.3, we have

$$\begin{aligned} R_k &= \liminf_{\mathsf{T}\to\infty} \frac{\mathsf{E}\{\#\text{ of flow-}k\text{ pkts delivered before exp. in }[1,\mathsf{T}]\}}{\mathsf{T}} \\ &= \liminf_{n\to\infty} \frac{\sum_{t=1}^{\mathsf{Prd}} \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} r_k(s,a) y_t^n(s,a)}{\mathsf{Prd}} \\ &\leq \lim_{n\to\infty} \frac{\sum_{t=1}^{\mathsf{Prd}} \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} r_k(s,a) y_t^{F_n}(s,a)}{\mathsf{Prd}} \\ &= \frac{\sum_{t=1}^{\mathsf{Prd}} \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} r_k(s,a) y_t^{\infty}(s,a)}{\mathsf{Prd}}. \end{aligned} \tag{9.14}$$

Therefore, (5.4c) also holds for $x_t^{\infty}(s,a)$ and $\vec{R}$.

Thus, (i) $-$ (iv) show that for any scheduling policy Ψ that achieves rate vector $\vec{R} = (R_1, \cdots, R_K)$, we can always find an $\vec{x} = \{x_t(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}, t \in [T_1, T_2]\}$ such that $\vec{R}$ and $\vec{x}$ are a feasible solution of (5.4a) $-$ (5.4e).

This completes the proof for $\mathcal{R} \subset \mathcal{R}^{\mathrm{LP}}$.

Step 1 and *Step 2* complete the proof for **Claim 2**.

9.4 Proof of Lemma 6.1

We consider the random process $\{S_t : \forall t \geq 1\}$ for the system state induced by our RAC-Approx scheduling policy. We then consider the sub-process $\{Z_i : \forall i \geq 1\}$ where $Z_i \triangleq S_{i\cdot L\cdot \mathsf{prd}+1}$. Since the decision rules of our RAC-Approx scheduling policy using (6.4) and (6.5) repeat themselves every prd slots staring at slot $(L \cdot \mathsf{prd} + 1)$, it is easy to see that $\{Z_i : \forall i \geq 1\}$ is a Markov chain.

We first show that the Markov chain $\{Z_i : \forall i \geq 1\}$ is an ergodic unichain [43]. We denote $\mathcal{S}_z$ as the state space of the Markov chain $\{Z_i : \forall i \geq 1\}$. Clearly $\mathcal{S}_z$ is a subset of the state space $\mathcal{S}$, and thus is finite. Then the Markov chain $\{Z_i : \forall i \geq 1\}$ must have at least one positive recurrent class $\mathcal{C}_z \subset \mathcal{S}_z$. We further denote the set of other states as $\bar{\mathcal{C}_z} = \mathcal{S}_z \backslash \mathcal{C}_z$. Since we assume that the packet arrival probability $\mathsf{B}_k < 1$, there exists a strictly positive probability that no packets of any flow arrive in the system during the time interval $[i \cdot L \cdot \mathsf{prd} + 2, (i+1) \cdot L \cdot \mathsf{prd} + 1]$. In addition, all packets arrived at slot $(i \cdot L \cdot \mathsf{prd} + 1)$ will expire at slot $((i+1) \cdot L \cdot \mathsf{prd} + 1)$ since $L \cdot \mathsf{Prd} \geq \max_{k\in[1,K]}(\mathsf{offset}_k + \mathsf{D}_k)$.[4]

[4] See the definition of L in Lemma 5.1.

Therefore, it is possible from any state in $\mathcal{S}_z$ to the empty state, denoted as $s_\emptyset \in \mathcal{S}_z$, i.e.,

$$\mathsf{Prob}(Z_{i+1} = s_\emptyset | Z_i = s) > 0, \quad \forall s \in \mathcal{S}_z. \tag{9.15}$$

Now consider a positive recurrent state $s \in \mathcal{C}_z$. Since s is positive recurrent and $s_\emptyset$ is accessible from s due to (9.15), s must also be accessible from state $s_\emptyset$. Thus, $s_\emptyset$ and s communicate and we have that $s_\emptyset \in \mathcal{C}_z$. We further show that all states in $\bar{\mathcal{C}_z}$ are transient. Suppose that state $s' \in \bar{\mathcal{C}_z}$ is not transient, which means that it must be positive recurrent. Now again since $s_\emptyset$ is accessible from s' due to (9.15), s' must also be accessible from state $s_\emptyset$. Thus, $s_\emptyset$ and s' communicate and we have that $s' \in \mathcal{C}_z$, which is a contradiction that $s' \in \bar{\mathcal{C}_z} = \mathcal{S}_z \backslash \mathcal{C}_z$. Therefore, the Markov chain $\{Z_i : \forall i \geq 1\}$ has only one positive recurrent class $\mathcal{C}_z$, and thus is a unichain [43]. In addition, since $\mathsf{Prob}(Z_{i+1} = s_\emptyset | Z_i = s_\emptyset) > 0$, the empty state $s_\emptyset$ is aperiodic and thus the positive recurrent class $\mathcal{C}_z$ is also aperiodic. Therefore, the Markov chain $\{Z_i : \forall i \geq 1\}$ is an ergodic unichain.

Then according to Theorem 4.3.7 in [43], the Markov chain $\{Z_i : \forall i \geq 1\}$ has a unique steady state distribution, i.e., there exists a π_s such that

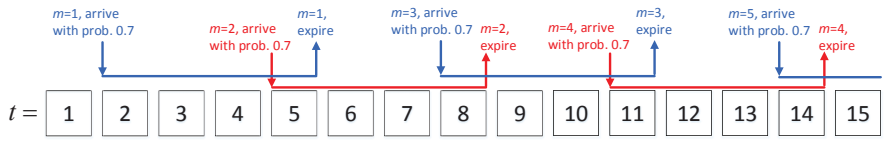
$$\lim_{i \to \infty} \mathsf{Prob}(Z_i = s) = \pi_s, \forall s \in \mathcal{S}_z. \tag{9.16}$$

Based on the initial state distribution $\{\pi_s : s \in \mathcal{C}_z\}$, for any flow k, we can get the expected number of delivered flow-k packets during the interval $[i \cdot L \cdot \mathsf{prd}+1, (i+1) \cdot L \cdot \mathsf{prd}+1]$ as $i \to \infty$, denoted by n_k. Then we can easily see that the achieved timely throughput for any flow k up to slot T, i.e., $\frac{\mathsf{E}\{\#\text{ of flow-}k\text{ pkts delivered before expiration in } [1,\mathsf{T}]\}}{\mathsf{T}}$, converges to $\frac{n_k}{L \cdot \mathsf{prd}}$ as $\mathsf{T} \to \infty$. This completes the proof.

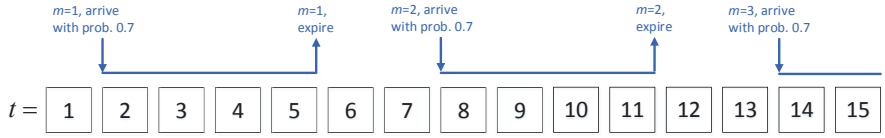
9.5 How to Solve (P4) and (P5) Approximately in Polynomial Time by Unwinding Traffic Patterns

For any flow k with A&E profile $(\mathsf{offset}_k, \mathsf{prd}_k, \mathsf{D}_k, \mathsf{B}_k)$ and $\mathsf{prd}_k < \mathsf{D}_k$, we unwind it into $\lceil \mathsf{D}_k / \mathsf{prd}_k \rceil$ flows, indexed from k_1, k_2, to $k_{\lceil \mathsf{D}_k/\mathsf{prd}_k \rceil}$: flow k_i $(1 \leq i \leq \lceil \mathsf{D}_k/\mathsf{prd}_k \rceil)$ has A&E profile $(\mathsf{offset}_{k_i}, \mathsf{prd}_{k_i}, \mathsf{D}_{k_i}, \mathsf{B}_{k_i})$ where

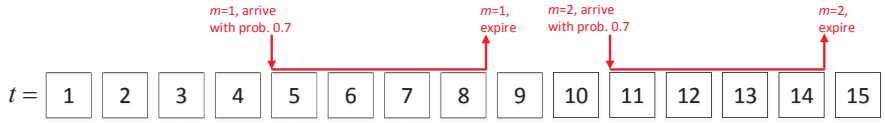
$$\mathsf{offset}_{k_i} = \mathsf{offset}_k + (i-1)\mathsf{prd}_k, \qquad \mathsf{D}_{k_i} = \mathsf{D}_k,$$
$$\mathsf{prd}_{k_i} = \lceil \mathsf{D}_k/\mathsf{prd}_k \rceil \mathsf{prd}_k, \qquad \mathsf{B}_{k_i} = \mathsf{B}_k.$$



(a) Original flow k with A&E profile: $(\mathsf{offset}_k, \mathsf{prd}_k, \mathsf{D}_k, \mathsf{B}_k) = (1, 3, 4, 0.7)$



(b) Unwound flow k_1 with A&E profile: $(\mathsf{offset}_{k_1}, \mathsf{prd}_{k_1}, \mathsf{D}_{k_1}, \mathsf{B}_{k_1}) = (1, 6, 4, 0.7)$



(c) Unwound flow k_2 with A&E profile: $(\mathsf{offset}_{k_2}, \mathsf{prd}_{k_2}, \mathsf{D}_{k_2}, \mathsf{B}_{k_2}) = (4, 6, 4, 0.7)$

Figure 9.1: An illustrating example of unwinding traffic patterns. We unwind the original flow k in (a) into flow k_1 in (b) and flow k_2 in (c).

One unwinding example is shown in Fig. 9.1. Now for each flow k_i, the delay D_{k_i} is no more than the period prd_{k_i}. Thus, the size of the state space of flow k_i can be reduced to 2 by using the compression method in (5.3). In addition, any flow-k_i's packet will be successfully delivered with probability p_k.

After such unwinding, in the new system, all flows have size-2 state space. We further denote R_{k_i} as the timely throughput of flow k_i in the new system, and define

$$R_k = \sum_{i=1}^{\lceil \mathsf{D}_k/\mathsf{prd}_k \rceil} R_{k_i},$$

as the timely throughput for flow k in the original system. Now if we use the relaxation method in Sec. 6.1 for the new system, the complexity (in terms of number of variables and constraints) of solving (**P4**) and (**P5**) becomes $O((\sum_{k=1}^{K} \lceil \mathsf{D}_k/\mathsf{prd}_k \rceil) \cdot K \cdot \mathsf{Prd})$. Therefore, by unwinding traffic patterns, we can solve (**P4**) and (**P5**) for the original system *approximately* in polynomial time.

Table 9.1: The optimal RAC policy for the setting in Fig. 7.3 in Sec. 7.2. We show the conditional probability $\mathsf{Prob}_{A_t|S_t}(a|s)$ for each state $s = (s^1, s^2, s^3)$, each action $a \in \{1, 2, 3\}$ in the period $[T_1, T_2] = [9, 12]$. **Note 1:** the flow-1 state $s^1 = 1$ (resp. 0) means one (resp. no) flow-1 packet; the flow-2 state $s^2 = 1$ (resp. 0) means one (resp. no) flow-2 packet; the flow-3 state $s^3 = l_1^3 l_2^3 l_3^3$ where $l_i^k = 1$ (resp. 0) means one (resp. no) flow-3 packet with lead time i. For example, state $s = (s^1, s^2, s^3) = (0, 1, 001)$ means that in the AP's queue, there is no flow-1 packet, one flow-2 packet, and one flow-3 packet with lead time 3. **Note 2:** The hyphen (-) notation means that the corresponding state is impossible at the corresponding slot under this optimal RAC policy. **Note 3:** We only show 2 digits after the decimal point. Thus $\sum_{a=1}^{3} \mathsf{Prob}_{A_t|S_t}(a|s)$ may not be 1.

	$t = T_1 = 9$			$t = 10$			$t = 11$			$t = T_2 = 12$		
$s = (s^1, s^2, s^3)$	$a = 1$	$a = 2$	$a = 3$	$a = 1$	$a = 2$	$a = 3$	$a = 1$	$a = 2$	$a = 3$	$a = 1$	$a = 2$	$a = 3$
(0, 0, 000)	-	-	-	0.33	0.33	0.33	-	-	-	0.33	0.33	0.33
(0, 0, 001)	-	-	-	0.00	0.00	1.00	-	-	-	0.00	0.00	1.00
(0, 0, 010)	-	-	-	0.00	0.00	1.00	-	-	-	0.00	0.00	1.00
(0, 0, 011)	-	-	-	0.00	0.00	1.00	-	-	-	0.00	0.00	1.00
(0, 0, 100)	-	-	-	-	-	-	-	-	-	-	-	-
(0, 0, 101)	-	-	-	0.00	0.00	1.00	-	-	-	0.00	0.00	1.00
(0, 0, 110)	-	-	-	0.00	0.00	1.00	-	-	-	0.00	0.00	1.00
(0, 0, 111)	-	-	-	0.00	0.00	1.00	-	-	-	0.00	0.00	1.00
(0, 1, 000)	-	-	-	-	-	-	0.00	1.00	0.00	0.00	1.00	0.00
(0, 1, 001)	-	-	-	0.00	0.98	0.02	0.00	0.83	0.17	0.00	0.85	0.15
(0, 1, 010)	-	-	-	0.00	1.00	0.00	0.00	0.03	0.97	0.00	0.01	0.99
(0, 1, 011)	-	-	-	0.00	0.97	0.03	0.00	0.48	0.52	0.00	0.62	0.38
(0, 1, 100)	-	-	-	-	-	-	0.00	0.00	1.00	0.00	0.00	1.00
(0, 1, 101)	-	-	-	0.00	0.50	0.50	0.00	0.01	0.99	0.00	0.00	1.00
(0, 1, 110)	-	-	-	0.00	0.50	0.50	0.00	0.00	1.00	0.00	0.00	1.00
(0, 1, 111)	-	-	-	0.00	0.52	0.48	0.00	0.00	1.00	0.00	0.87	0.13
(1, 0, 000)	1.00	0.00	0.00	1.00	0.00	0.00	-	-	-	-	-	-
(1, 0, 001)	0.83	0.00	0.17	0.85	0.00	0.15	-	-	-	0.98	0.00	0.02
(1, 0, 010)	0.03	0.00	0.97	0.01	0.00	0.99	-	-	-	1.00	0.00	0.00
(1, 0, 011)	0.48	0.00	0.52	0.62	0.00	0.38	-	-	-	0.97	0.00	0.03
(1, 0, 100)	0.00	0.00	1.00	0.00	0.00	1.00	-	-	-	-	-	-
(1, 0, 101)	0.01	0.00	0.99	0.00	0.00	1.00	-	-	-	0.50	0.00	0.50
(1, 0, 110)	0.00	0.00	1.00	0.00	0.00	1.00	-	-	-	0.50	0.00	0.50
(1, 0, 111)	0.00	0.00	1.00	0.87	0.00	0.13	-	-	-	0.52	0.00	0.48
(1, 1, 000)	0.00	1.00	0.00	0.00	1.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
(1, 1, 001)	0.01	0.99	0.00	0.00	1.00	0.00	0.99	0.01	0.00	1.00	0.00	0.00
(1, 1, 010)	0.00	0.37	0.63	0.00	1.00	0.00	0.37	0.00	0.63	1.00	0.00	0.00
(1, 1, 011)	0.00	0.50	0.50	0.22	0.39	0.38	0.50	0.00	0.50	0.39	0.22	0.38
(1, 1, 100)	0.00	0.00	1.00	0.00	0.52	0.48	0.00	0.00	1.00	0.52	0.00	0.48
(1, 1, 101)	0.00	0.00	1.00	0.00	0.65	0.35	0.00	0.00	1.00	0.65	0.00	0.35
(1, 1, 110)	0.00	0.55	0.45	0.00	0.62	0.38	0.55	0.00	0.45	0.62	0.00	0.38
(1, 1, 111)	0.00	0.91	0.09	0.14	0.77	0.09	0.91	0.00	0.09	0.77	0.14	0.09

9.6 The Optimal RAC Policy for the Setting in Fig. 7.3 in Sec. 7.2

In this part, we show the optimal RAC policy for the setting in Fig. 7.3 in Sec. 7.2, i.e.,

$$(\mathsf{offset}_1, \mathsf{prd}_1, \mathsf{D}_1, \mathsf{B}_1, p_1) = (0, 4, 4, 1, 0.5), U_1(R_1) = \log(R_1),$$
$$(\mathsf{offset}_2, \mathsf{prd}_2, \mathsf{D}_2, \mathsf{B}_2, p_2) = (2, 4, 4, 1, 0.5), U_2(R_2) = \log(R_2),$$
$$(\mathsf{offset}_3, \mathsf{prd}_3, \mathsf{D}_3, \mathsf{B}_3, p_3) = (0, 1, 3, 0.9, 0.7), U_3(R_3) = \log(R_3).$$

In Tab. 9.1, we show the conditional probability $\mathsf{Prob}_{A_t|S_t}(a|s)$ for each state $s = (s^1, s^2, s^3)$, each action $a \in \{1, 2, 3\}$, and each slot $t \in [T_1, T_2] = [9, 12]$, which is the optimization period in (**P2**).

Note that since $\mathsf{D}_k \leq \mathsf{prd}_k$ for $k = 1, 2$, we can use the compression method similar in (5.3) to denote the state of flow 1 and flow 2. More specifically, we use $s^1 = 1$ (resp. 0) to denote the flow 1's state: flow 1 has one (resp. no) packet, and we use $s^2 = 1$ (resp. 0) to denote the flow 2's state: flow 2 has one (resp. no) packet. We remark that though we do not have the lead time information in the state representation for flow 1 and flow 2, we can recover the lead time from the current slot. For flow 3, since $\mathsf{D}_k > \mathsf{prd}_k$, we still use the lead-time-representation to denote its state, i.e., $s^3 = l_1^3 l_2^3 l_3^3$ where $l_i^k = 1$ (resp. 0) is the state that flow 3 has one (resp. no) packet with lead time i. For example, state $s = (s^1, s^2, s^3) = (0, 1, 001)$ means that in the AP's queue, there is no flow-1 packet, one flow-2 packet, and one flow-3 packet with lead time 3.

Now we show several examples to illustrate the optimal RAC policy in Tab. 9.1.

- Consider state $s = (s^1, s^2, s^3) = (0, 0, 000)$ at slot $t = 9$, i.e., all flows have no packet. This state is impossible (we use the hyphen (-) notation in Tab. 9.1 to denote an impossible state at the corresponding slot) because at the beginning of slot 9, flow 1 has a packet arrival with probability $\mathsf{B}_1 = 1$ and thus the flow-1 state must be 1.

- Consider state $s = (s^1, s^2, s^3) = (0, 0, 000)$ at slot $t = 10$, i.e., all flows have no packet. Though our optimal RAC policy will still schedule each flow with equal probability (0.33), no packets will be transmitted and thus the system actually remains idle.

- Consider state $s = (s^1, s^2, s^3) = (0, 1, 100)$ at slot $t = 11$, i.e., flow 2 has a packet with lead time 4 and flow 3 has a packet with lead time 1. Our optimal RAC policy will schedule flow 3 with probability 1, i.e, give the complete priority to flow 3. This is reasonable because the flow-3 packet will be expired in the next slot and thus is more urgent than the flow-2 packet.

- Consider state $s = (s^1, s^2, s^3) = (1, 0, 111)$ at slot $t = 12$, i.e., flow 1 has a packet with lead time 1, and flow 2 has a packet with lead time 1, a packet with lead time 1, and a packet with lead time 3. Our optimal RAC policy will schedule flow 1 with probability 0.52 and schedule flow 3 with probability 0.48, i.e., gives a little bit more priority to flow 1. This is reasonable because though both flows have a very urgent packet (with lead time 1), flow 3 has more packet candidates than flow 1.

Part III

Energy-Efficient Timely Transportation of Long-Haul Heavy-Duty Trucks

Chapter 10

Introduction

In the U.S., heavy-duty trucks haul more than 70% of all freight tonnage [6], and they consume 17.6% of energy in transportation sector [35, Tab. 2.8] and contribute to about 5% of the greenhouse gas emission [17]. Fuel cost is the largest operating cost (34%) of truck owners/operators [42], and reducing fuel consumption is critical for cost-effective and environment-friendly heavy-duty truck operations.

Currently there are mainly two lines of efforts to reduce fuel consumption of heavy-duty trucks. The first line is to operate with more fuel efficient trucks, from better designs for engines, drivetrains, aerodynamics, and tires [49, 74, 14], to better management of truck parts such as maintaining optimal tire pressures [7]. The second line is to operate heavy-duty trucks more economically. This explores several possibilities, e.g., reducing idling energy consumption [85], platooning more than one heavy-duty trucks [20, 67], route planning [38, 86, 92], and speed planning [52, 51, 12, 4]. In this thesis, we focus on route and speed planning. Different routes could have different mileages, levels of congestion, road grades, and surface types, etc., all of which would largely affect the fuel consumption. Real-world studies [92] show that choosing a more efficient route for a heavy-duty truck can improve its fuel economy by 21%. Speed planning is another well-recognized approach to effectively reduce fuel consumption: As a rule of thumb for truck operations on highway, every one mile per hour (mph) increase in speed incurs about 0.14 mile per gallon (mpg) penalty in fuel economy [12, 4].

However, operating at low speed may result in excessive travel time and the goods carried by the truck cannot be delivered on time. We remark that timely delivery is critical for truck operators [71, 16]. As estimated by the U.S. Federal Highway Administration (FHWA) in [71], unexpected delay can increase freight cost by 50% to 250%. Multiple reasons can explain the importance of timely delivery. First, some freight goods are perishable, such as food [24], which definitely require timely delivery. Second, to ensure

Table 10.1: Comparisons of our work and existing work on energy-efficient truck operation.

Paper	Route Planning	Speed Planning	Hard Deadline
[83]	✓	✗	✗
[52]	✗	✓	✗
[51]	✗	✓	✓
This work	✓	✓	✓

customers' satisfaction, some companies, e.g., Amazon, may have a service-level agrement (SLA) with users, under which the delivery delay is guaranteed [11]. Finally, violating scheduled delay can introduce difficulties for global logistic decisions and even increase the uncertainty and inefficiency of supply chains [71]. Overall, it is crucial to ensure timely goods delivery for truck operators, and considering timely delivery in fuel cost minimization poses a unique challenge.

Motivated by the above observations, in this thesis, we study the problem of *energy-efficient timely transportation* for heavy-duty trucks. We aim to minimize the heavy duty truck's fuel consumption while satisfying a hard deadline constraint, under which we take both route planning and speed planning into account to exploit complete design space of reducing fuel consumption. Since heavy-duty trucks are mainly operated for *long-haul* delivery and most of time run on highways [35, Tab. 5.2 and Fig. 5.1], we focus our model on their operation in the highway transportation network system. We summarize our contributions in the following.

▷ We formulate an energy-efficient timely transportation problem of minimizing the fuel consumption subject to a hard deadline constraint for a heavy-duty truck running on a highway transportation network, with design spaces of both route planning and speed planning in Chap. 11. To the best of our knowledge, as compared to existing work on energy-efficient truck operation [83, 52, 51], our work is the first one that simultaneously considers route planning, speed planning, and hard deadline (see Tab. 10.1).

▷ We show that our problem is NP-Complete and then design a fully polynomial time approximation scheme (FPTAS) in Chap. 12 to solve it. The proposed FPTAS attains an approximation ratio of $(1 + \epsilon)$ with a network-size induced complexity of $O(mn^2/\epsilon^2)$, where m and n are the numbers of nodes and edges, respectively.

Table 10.2: Comparisons of our problem and existing problems on performance optimization in various transportation systems with delay taken into consideration. Here RSP stands for Restricted Shortest Path problem, VRPTW stands for Vehicle Routing Problem with Time Windows, and BSP stands for Bi-objective Shortest Path problem.

		Our problem	**RSP** [50, 45, 63]	**VRPTW** [81, 44, 34, 88]	**Tramp Ship Operation** [77]	**BSP** [84, 91, 30, 46]
Settings	Objective	Cost minimization	Cost minimization	Cost minimization	Profit maximization	Cost and delay minimization
	Constraints	A hard deadline	A hard deadline	Time window*, Other constraints	Time window*, Other constraints	N/A
	Design Spaces	Route planning, Speed planning	Route planning	Route planning	Route planning, Speed planning	Route planning
Results	Hardness	NP-Complete	NP-Complete [45]	NP-Complete [81]	N/A	NP-Complete [84]
	Algorithms	FPTAS, Heuristic†	FPTAS [50], Heuristic [63]	Heuristic [44, 34, 88]	Heuristic	FPTAS [91, 30], Heuristic [46]

$\star$ The time window constraint captures the hard deadline constraint in our problem and RSP as a special case.

† We further characterize a condition under which the heuristic outputs the optimal solution to our problem.

▷ While achieving highly-preferred theoretical performance guarantee, the proposed FPTAS still suffers from long running time when applying to national-wide highway systems with tens of thousands of nodes and edges. In Chap. 13, by leveraging elegant insights from studying the dual of the original problem, we design a fast heuristic solution with $O(m + n \log n)$ complexity. The proposed heuristic scheme allows us to tackle the energy-efficient timely transportation problem on large-scale national highway systems. We further characterize a condition under which our heuristic generates an optimal solution. We observe that the condition holds in most of the practical instances in numerical experiments in Chap. 15, justifying the superior empirical performance of our heuristic.

▷ We carry out extensive numerical experiments using real-world truck data over the U.S. highway network in Chap. 15. The results show that our proposed solutions achieve 17% (resp. 14%) fuel consumption reduction, as compared to a fastest path (resp. shortest path) algorithm adapted from common practice. The amount of fuel consumption saving is enough to power up more than 90% of the entire transportation sector in New York State [3].

Comparison with existing problems. Theoretically, we also compare our problem with some existing problems in Tab. 10.2. First, our energy-efficient timely transportation problem is a generalized version of Restricted Shortest Path problem (RSP) [50, 45, 63], with an extra design space of speed planning. Therefore, we generalize the FPTAS design

and the dual-based design of RSP to our problem. Second, for the well-studied Vehicle Routing Problem with Time Windows (VRPTW) [81, 44, 34, 88], if we only consider one vehicle and one customer with departure deadline, then it becomes the RSP problem, which is a special case of our problem without speed planning. Third, our problem can be regarded as a special case of the problem on energy-efficient tramp ship operation [77], a different context from our focus on trucks. However, [77] does not prove the hardness of the problem and only proposes a heuristic algorithm without performance guarantee to solve the problem. Our work, instead, shows that our studied problem is NP-complete, and proposes an FPTAS and a heuristic algorithm with characterizing an optimality condition to solve the problem. Finally, another related problem is the Bi-objective Shortest Path problem (BSP) [84, 91, 30, 46], which needs to find all Pareto-optimal paths to simultaneously minimize the travel cost and travel time. Clearly, BSP is different from our problem because it regards the travel time as one objective and it does not have the design space of speed planning.

Chapter 11

System Model and Problem Formulation

11.1 System Model

Consider a highway transportation network as exemplified in Fig. 11.1. We model it as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the vertex/node set and $\mathcal{E}$ is the edge/road set. We define $n \triangleq |\mathcal{V}|$ as the number of nodes and $m \triangleq |\mathcal{E}|$ as the number of edges. For each edge $e \in \mathcal{E}$, we denote $D_e > 0$ as its distance (unit: mile), and $R_e^{\mathsf{lb}} > 0$ (resp. $R_e^{\mathsf{ub}} \geq R_e^{\mathsf{lb}}$) as its minimal (resp. maximal) speed (unit: mph). (Governments usually set the maximal speed for all highways and the minimal speed for some highways. For the sake of both safety and fuel efficiency, lower speed limits than passenger cars may be applied to large commercial vehicles like heavy-duty trucks and buses.) Now consider a long-haul heavy-duty truck who aims to ship cargos from a source node $s \in \mathcal{V}$ to a destination node $d \in \mathcal{V}$. The goal is to minimize the energy/fuel[1] consumption subject to a *hard* delay requirement $T > 0$ (unit: hour).

Fuel consumption and travel delay are usually in conflict with each other, both of which are related to the speed profile of the truck. High travel speed obviously decreases the travel delay, but it can also increase the fuel consumption significantly [12, 4]. To analyze the performance tradeoff between energy and delay, we need to model the relationship between the fuel consumption and the travel speed. There are an intensive number of such models (see a survey in [37]). In this thesis, we use the instantaneous fuel consumption model [37, 19] which generally depends on three factors: (i) static vehicle/road/environment properties, (ii) instantaneous acceleration/deceleration, and (iii) instantaneous speed. As we consider a specific vehicle running over a specific

[1] We interchangeably use *fuel* and *energy* in this thesis.

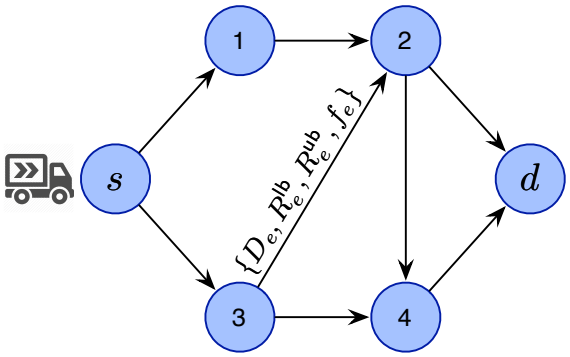


Figure 11.1: System model.

network, static vehicle/road/environment properties are fixed. The instantaneous acceleration/deceleration reflects the speed variation. However, since we consider a highway model, the truck spends most of time to maintain a relatively constant cruise speed [23, 72] such that the fuel consumption caused by acceleration/deceleration would be negligible. This motivates us to model the instantaneous fuel consumption as a function of the instantaneous speed.

We thus define $f_e : [R_e^{\mathsf{lb}}, R_e^{\mathsf{ub}}] \to \mathbb{R}^+$ as the (instantaneous) *fuel-rate-speed function* of the truck running on edge e: if the vehicle's speed on edge e is r_e (unit: mph), the fuel consumption rate is $f_e(r_e)$ (unit: gallons per hour (gph)), and then the total fuel consumption for driving time τ (unit: hour) with the constant speed r_e is $f_e(r_e) \cdot \tau$ (unit: gallon). Since many existing models [23, 21, 19, 75, 27] use polynomial functions to model the fuel consumption which are also strictly convex in a reasonable speed limit region, in this thesis, we assume that $f_e(\cdot)$ is a polynomial function and is strictly convex[2] over $[R_e^{\mathsf{lb}}, R_e^{\mathsf{ub}}]$. This assumption also holds in the physical interpretation of fuel-rate-speed function as shown in Appendix 17.1, and is further verified in our simulation using real-world data (see Fig. 15.2(a)).

[2]The strict convexity can be relaxed to convexity. For simplicity, we use the strict convexity in this thesis.

11.2 Problem Formulation

We consider two design spaces: path selection (route planning) and speed optimization (speed planning). For path selection, we define a binary variable x_e for any $e \in \mathcal{E}$,

$$x_e = \begin{cases} 1, & \text{Edge } e \text{ is on the selected path;} \\ 0, & \text{otherwise.} \end{cases} \tag{11.1}$$

For the speed optimization, the truck needs to determine a speed profile (speeds at all travel time) over any selected edge. This is a functional variable, but the convexity of fuel-rate-speed function can simplify the speed profile significantly based on the following lemma.

Lemma 11.1. *For any edge e, if the travel time t_e is given, i.e., the truck must pass edge e with exactly t_e hours, then the optimal speed profile to minimize the fuel consumption is to maintain constant speed D_e/t_e during the whole trip.*

Proof. See Appendix 17.2. □

Lemma 11.1 shows that for any edge, any non-constant speed profile is dominated by another constant speed profile in terms of fuel consumption without sacrificing the delay performance. Therefore, without loss of optimality, the truck only needs to follow a constant speed for any edge. As explained in Sec. 11.1, since we consider a long-haul highway scenario, we will ignore the speed transition period between two adjacent edges. Thus, for the speed optimization, we consider the travel time $t_e > 0$ over each edge e as the design variable, which equivalently implies a constant speed D_e/t_e over e. We then define a *fuel-time* function $c_e(\cdot)$ for each road e,

$$c_e(t_e) \triangleq t_e \cdot f_e(\frac{D_e}{t_e}), \tag{11.2}$$

which is the total fuel consumption for the truck traveling edge e with travel time t_e.

By vectorizing our decision variables as $\boldsymbol{x} \triangleq \{x_e : e \in \mathcal{E}\}$ and $\boldsymbol{t} \triangleq \{t_e : e \in \mathcal{E}\}$, now we are ready to formulate our PAth selection and Speed Optimization (PASO) problem,

$$\text{PASO:} \min_{\boldsymbol{x}\in\mathcal{X},\boldsymbol{t}\in\mathcal{T}} \quad \sum_{e\in\mathcal{E}} x_e \cdot c_e(t_e) \tag{11.3}$$

$$\text{s.t.} \quad \sum_{e\in\mathcal{E}} x_e t_e \le T \tag{11.4}$$

In PASO, set $\mathcal{X}$ restricts that one and only one $s-d$ path is selected, defined as

$$\mathcal{X} \triangleq \{\boldsymbol{x} : x_e \in \{0,1\}, \forall e \in \mathcal{E}, \text{ and } \sum_{e \in \mathsf{out}(v)} x_e - \sum_{e \in \mathsf{in}(v)} x_e = 1_{\{v=s\}} - 1_{\{v=d\}}, \forall v \in \mathcal{V}\}, \tag{11.5}$$

where $1_{\{\cdot\}}$ is the indicator function, $\mathsf{in}(v) \triangleq \{(u,v) : (u,v) \in \mathcal{E}\}$ is the set of incoming edges of node $v \in \mathcal{V}$, and $\mathsf{out}(v) \triangleq \{(v,u) : (v,u) \in \mathcal{E}\}$ is the set of outgoing edges of node v. Set $\mathcal{T}$ captures the speed limits of all roads, defined as

$$\mathcal{T} \triangleq \{\boldsymbol{t} : t_e^{\mathsf{lb}} \leq t_e \leq t_e^{\mathsf{ub}}, \forall e \in \mathcal{E}\}, \tag{11.6}$$

where $t_e^{\mathsf{lb}} \triangleq \frac{D_e}{R_e^{\mathsf{ub}}}$ and $t_e^{\mathsf{ub}} \triangleq \frac{D_e}{R_e^{\mathsf{lb}}}$ are the minimal and maximal travel time due to the speed limits on edge e, respectively. Constraint (11.4) is to satisfy the hard delay requirement. Objective (11.3) is to minimize the total fuel consumption over the selected path.

11.3 Complexity Hardness

PASO has both integer variables and continuous variables. Thus it is worth understanding its hardness first. It turns out that a special case of PASO is the well-known Restricted Shortest Path (RSP) problem [50, 45]. In RSP, a directed graph is given where each edge has a *fixed* travel time and travel cost, and the goal is to find a minimal-cost path subject to a hard path delay requirement. Clearly, our problem PASO generalizes RSP where we allow a varying edge cost and edge time because of the design space of speed optimization. Since RSP is NP-complete [45], we can thus easily prove that our problem PASO is also NP-complete.

Theorem 11.1. *PASO is NP-complete.*

Proof. We can prove it by setting $R_e^{\mathsf{lb}} = R_e^{\mathsf{ub}}$ to an appropriate value for each edge e in PASO, and using the result that RSP is NP-complete [45]. □

11.4 Preprocessing and Some Notations

We first check the feasibility of our problem PASO. We can use the shortest path algorithm where each edge e has cost t_e^{lb} to find the fastest path. If the travel time of the fastest

path is larger than the delay requirement T, PASO is infeasible. In the rest of this thesis, we thus assume that the delay constraint T is at least the travel time of the fastest path such that the problem is feasible.

We then analyze properties of the fuel-time function $c_e(\cdot)$.

Lemma 11.2. *$c_e(t_e)$ is strictly convex over $[t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]$. Also, there exists a point $\hat{t}_e \in [t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]$[3] such that $c_e(t_e)$ is first strictly decreasing over $[t_e^{\mathsf{lb}}, \hat{t}_e]$ and then strictly increasing over $[\hat{t}_e, t_e^{\mathsf{ub}}]$.*

Proof. See Appendix 17.3. □

Based on Lemma 11.2, we can easily prove that the travel time over edge e, i.e., t_e, in any optimal solution of PASO must be in the region $[t_e^{\mathsf{lb}}, \hat{t}_e]$. Otherwise, we can decrease the trave time from t_e to $\hat{t}_e$ and at the same time decrease the fuel consumption, which violates the optimality of t_e. Thus, without loss of optimality, we can reset the travel time limit from $[t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]$ to $[t_e^{\mathsf{lb}}, \hat{t}_e]$, which equivalently implies that we reset the speed limit from $[R_e^{\mathsf{lb}}, R_e^{\mathsf{ub}}]$ to $[D_e/\hat{t}_e, R_e^{\mathsf{ub}}]$. After such preprocessing, in the rest of the thesis, $c_e(t_e)$ can be assumed to be *strictly convex and strictly decreasing* over $t_e \in [t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]$ *without loss of optimality.*

In the rest of the thesis, we define an $s-d$ path p as the set of all *edges* over p and $\boldsymbol{t}_p \triangleq \{t_e : e \in p\}$ as the corresponding travel time set. Moreover, we define $c(p, \boldsymbol{t}_p) \triangleq \sum_{e\in p} c_e(t_e)$ as the fuel consumption of path p with travel time set $\boldsymbol{t}_p$, and OPT as the optimal value of PASO.

Next, we will propose a fully polynomial time approximation scheme (FPTAS) in Chap. 12 and a fast dual-based heuristic scheme in Chap. 13 to solve our problem PASO.

[3]Note that $\hat{t}_e$ can be on the boundary.

Chapter 12

An FPTAS for PASO

Since PASO generalizes RSP, which is well-known to have an FPTAS [50, 70], it is natural to ask whether we can extend RSP's FPTAS for our problem PASO. In this section, by carefully tackling the difference between PASO and RSP, we "reformulate" PASO such that we can adapt RSP's FPTAS to construct an FPTAS for PASO. More specifically, in this section, we propose an approximation scheme (Algorithm 3) such that for any given $\epsilon \in (0, 1)$, it can find a $(1+\epsilon)$-approximate solution in the sense that the solution is *feasible* and the corresponding fuel consumption is at most $(1 + \epsilon)\mathsf{OPT}$, and the time complexity is polynomial in both the problem size and $\frac{1}{\epsilon}$.

The essence of RSP's FPTAS [50, 70] is a test procedure. For any input value $X > 0$ and any input accuracy parameter $\delta > 0$, the test procedure can "approximately" compare X and the optimal value OPT in the sense that it can tell either $\mathsf{OPT} > X$ or $\mathsf{OPT} \leq (1 + \delta)X$ in polynomial time. Based on this test procedure, starting with some arbitrary lower bound LB and upper bound UB for OPT, a binary search scheme is designed [50, 70] to exponentially narrow down the bounding interval $[\mathsf{LB}, \mathsf{UB}]$ and finally a $(1 + \epsilon)$-approximate solution is outputted.

To solve our problem PASO, we adapt RSP's FPTAS by designing our own test procedure. In RSP, [50] and [70] use the *rounding and scaling* technique, where each *fixed* edge cost is rounded into certain (polynomial) number of cost levels controlled by the accuracy parameter δ. As we only require an "approximate" comparison, rounding into certain number of cost levels is enough to perform such a task. However, as opposed to a fixed edge cost in RSP, in PASO each edge has a fuel-time function. Hence, instead of rounding a fixed cost in RSP, we *quantize* the continuous fuel-time function $c_e(\cdot)$ into another staircase fuel-time function $\tilde{c}_e(\cdot)$ according to the input value X and the input accuracy parameter δ, which can be further characterized by a polynomial number of *representative points*. We then prove that such quantization can perform the "approximate" comparison.

Later on we will describe our algorithms in a bottom-up fashion. We first describe the quantizing procedure (Algorithm 1) in Sec. 12.1. Then we present our own test procedure (Algorithm 2) which invokes Algorithm 1 in Sec. 12.2. Finally, we describe the whole FPTAS (Algorithm 3) which invokes Algorithm 2 in Sec. 12.3.

12.1 Quantizing Fuel-Time Function

For any $V > 0$ and $N \in \mathbb{Z}^+$, we quantize the edge-e fuel-time function $c_e(t_e)$ to be

$$\tilde{c}_e(t_e) \triangleq \min\left\{\left\lfloor \frac{c_e(t_e)}{V} \right\rfloor + 1, N\right\}, \forall t_e \in [t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]. \tag{12.1}$$

Since we have assumed that $c_e(t_e)$ is strictly decreasing in Sec. 11.4, $\tilde{c}_e(t_e)$ thus becomes a *staircase* function with at most N stairs. During the quantization, parameter V is to control the accuracy, which is the vertical span of each stair. Larger V means rougher quantization and lower accuracy but smaller complexity. Parameter N is to control the maximal number of stairs. Since $c_e(t_e)$ could take an arbitrarily large value, the number of stairs could be unbounded, which definitely incurs high complexity. To design a polynomial time test procedure where we only need to perform an "approximate" comparison, we truncate $c_e(t_e)$ by putting a ceil VN. This truncation is sufficient for use in the test procedure (see Sec. 12.2). Clearly, $\tilde{c}_e(t_e)$ is a *quantized and truncated* version of $c_e(t_e)$. An example is shown in Fig. 12.1. Here we set $V = 20, N = 4$. Thus, each stair spans 20 and $c_e(t_e)$ is truncated by the ceil $VN = 80$. The resulting curve $\tilde{c}_e(t_e)$ is a non-increasing staircase function, which jumps from 4 to 3 at $t_e = 1.8$ and jumps from 3 to 2 at $t_e = 2.8$.

Moreover, since $\tilde{c}_e(t_e)$ is a staircase function and only takes integer values, we can use an N-dim vector $\boldsymbol{\tau}_e$ to represent it without any information loss. We define it as $\boldsymbol{\tau}_e \triangleq (\tau_e^1, \tau_e^2, \cdots, \tau_e^N)$ where τ_e^i is the minimal travel time over $[t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]$ such that $\tilde{c}_e(\cdot) = i$ and is defined as nan if $\tilde{c}_e(\cdot) = i$ has no solution. For the example in Fig. 12.1, we have $\tau_e = (\tau_e^1, \tau_e^2, \tau_e^3, \tau_e^4) = (\mathsf{nan}, 2.8, 1.8, 1)$.

We call (τ_e^i, i) the i-th *representative point* of $\tilde{c}_e(\cdot)$. Thus $\tilde{c}_e(\cdot)$ is characterized by at most N representative points, which will play a key role in our test procedure in Sec. 12.2. We summary the quantizing procedure $\texttt{QUANTIZE}(e, V, N)$ in Algorithm 1. The basic idea is to first find the range of the stair levels, i.e., $[n_{\min}, n_{\max}]$ and then find τ_e^i for any level i in this range by solving an equation $c_e(t_e) = iV$.

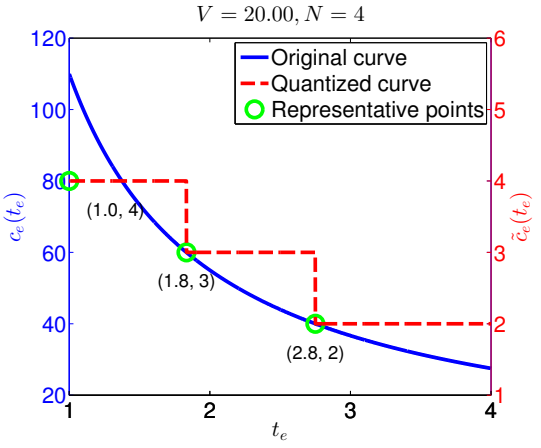


Figure 12.1: An example for quantizing $c_e(\cdot)$.

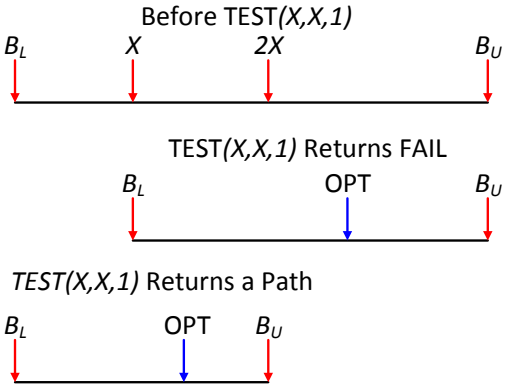


Figure 12.2: Binary search (*Step 2*) of Algorithm 3.

Time Complexity: (i) When $n_{\min} = n_{\max}$ (e.g., if $t_e^{\mathsf{ub}} = t_e^{\mathsf{lb}}$), the loop in lines 7-12 will not be executed. Thus, the total complexity of $\texttt{QUANTIZE}(e, V, N)$ is $O(N)$ due to the initial loop in lines 1-3. (ii) When $n_{\min} < n_{\max}$, we need to solve an equation for each i in the range $[n_{\min}, n_{\max} - 1]$ as shown in line 8. Since we have assumed that $c_e(t_e)$ is a strictly decreasing function, we can use a binary search to solve this equation, which has time complexity $O\left(\log\left\lceil \frac{t_e^{\mathsf{ub}} - t_e^{\mathsf{lb}}}{\mathsf{tol}} \right\rceil\right)$ where tol is the tolerance level for termination. The total complexity of $\texttt{QUANTIZE}(e, V, N)$ is $O\left(N + (n_{\max} - n_{\min}) \log\left\lceil \frac{t_e^{\mathsf{ub}} - t_e^{\mathsf{lb}}}{\mathsf{tol}} \right\rceil\right) = O\left(N + N \log\left\lceil \frac{t_e^{\mathsf{ub}} - t_e^{\mathsf{lb}}}{\mathsf{tol}} \right\rceil\right) = O\left(N \log\left(2\left\lceil \frac{t_e^{\mathsf{ub}} - t_e^{\mathsf{lb}}}{\mathsf{tol}} \right\rceil\right)\right)$. To unify the expression of time complexity for both (i) and (ii), we define

$$\xi_e \triangleq \max\left\{2, 2\left\lceil \frac{t_e^{\mathsf{ub}} - t_e^{\mathsf{lb}}}{\mathsf{tol}} \right\rceil\right\},$$

then the complexity of $\texttt{QUANTIZE}(e, V, N)$ is $O(N \log \xi_e)$. Also, if we define

$$\xi \triangleq \max_{e \in \mathcal{E}} \xi_e, \tag{12.2}$$

then the complexity of $\texttt{QUANTIZE}(e, V, N)$ is $O(N \log \xi)$ for any $e \in \mathcal{E}$.

12.2 The Test Procedure

As introduced above, the test procedure should "approximately" compare X and the optimal value OPT such that it can answer either $\mathsf{OPT} > X$ or $\mathsf{OPT} \leq (1 + \delta)X$ in

Algorithm 1 A Quantizing Procedure $\texttt{QUANTIZE}(e, V, N)$

1: **for** $i = 1, 2, \cdots, N$ **do**
2: Set $\tau_e^i = \mathsf{nan}$
3: **end for**
4: Set $n_{\min} = \tilde{c}_e(t_e^{\mathsf{ub}}) = \min\{\lfloor \frac{c_e(t_e^{\mathsf{ub}})}{V} \rfloor + 1, N\}$
5: Set $n_{\max} = \tilde{c}_e(t_e^{\mathsf{lb}}) = \min\{\lfloor \frac{c_e(t_e^{\mathsf{lb}})}{V} \rfloor + 1, N\}$
6: Set $\tau_e^{n_{\max}} = t_e^{\mathsf{lb}}$
7: **for** $i = n_{\min}, n_{\min} + 1, \cdots, n_{\max} - 1$ **do**
8: Solve the equation $c_e(t_e) = iV$ over $t_e \in [t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]$
9: **if** the equation has a solution t_e **then**
10: Set $\tau_e^i = t_e$
11: **end if**
12: **end for**
13: **return** $\boldsymbol{\tau}_e = (\tau_e^1, \tau_e^2, \cdots, \tau_e^N)$

polynomial time. Inspired by [70], which improves the FPTAS of RSP in [50], we adopt a more powerful test procedure, denoted by $\texttt{TEST}(L, U, \delta)$. It can answer either $\mathsf{OPT} > U$ or $\mathsf{OPT} \leq U + \delta L$. Clearly, if we set $L = U = X$, $\texttt{TEST}(X, X, \delta)$ can answer either $\mathsf{OPT} > X$ or $\mathsf{OPT} \leq (1 + \delta)X$, which exactly completes the "approximate" comparison. The reason to adopt a more powerful test procedure, similar to [70], is that we will also use it to finally output a $(1 + \epsilon)$-approximate solution. We will discuss it soon in Sec. 12.3.

The details of $\texttt{TEST}(L, U, \delta)$ are shown in Algorithm 2. As we mentioned before, the major difference between our problem PASO and the existing problem RSP is that PASO has a continuous fuel-time function for each edge instead of a fixed cost. Thus, different from the test procedure for RSP (see [70, Fig. 1]), we have a step to invoke the quantizing procedure (Algorithm 1) to quantize the fuel-time function, as shown in lines 3-5 in Algorithm 2. More importantly, since our test procedure $\texttt{TEST}(L, U, \delta)$ aims to check either $\mathsf{OPT} > U$ or $\mathsf{OPT} \leq U + \delta L$, roughly speaking, we do not need to quantize the portion of each fuel-time function with high fuel cost, i.e., larger than $U + \delta L$. Hence, to ensure polynomial time complexity eventually, we put a ceil $V(N+1)$ for $c_e(t_e)$ as shown in line 4 of the algorithm, where V and N are appropriately set such that $V(N + 1) \geq U + \delta L$.

After such quantization, the fuel-time function $c_e(t_e)$ for each edge e consists of at

most $N+1$ representative points. Therefore, conceptually we can construct a new graph $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{E}})$. Each edge $e \in \mathcal{E}$ in the original graph corresponds to at most $N+1$ edges in the new graph $\tilde{\mathcal{E}}$. For each edge $e \in \tilde{\mathcal{E}}$, the edge cost $\tilde{c}_e$ is a positive integer, as shown in (12.1). This is exactly an RSP problem. Therefore, the remaining steps follow the test procedure for RSP on the new graph $\tilde{\mathcal{G}}$. Specifically, since each edge $e \in \mathcal{E}$ has at most $N+1$ possible cost values all of which are positive integers (each edge e in the new graph $\tilde{\mathcal{E}}$ has a positive integer cost), we can use dynamic programming to complete such test. Similar to [50, 70], we define $g_v(c)$ as the minimal path travel time among all $s-v$ paths whose path cost is at most $c \in \mathbb{Z}^+$, and define $g_v(c) = \infty$ if no such path. The optimality condition (or Bellman's equation) becomes, for any $c = 1, 2, \cdots,$

$$g_v(c) = \min\{g_v(c-1), \min_{u,i:e=(u,v)\in\mathcal{E}, i=1,\cdots,N, \tau_e^i \neq \mathsf{nan}} \{g_u(c-i) + \tau_e^i\}\}, \tag{12.3}$$

which is shown in line 10 in Algorithm 2. Since we only need to answer either $\mathsf{OPT} > U$ or $\mathsf{OPT} \leq U + \delta L$, we do not have to process large c. Instead, iterating c from 1 to N is enough for us to complete this task. This dynamic programming procedure is shown in lines 6-15 of Algorithm 2.

In PASO, we should carefully design the quantizing and the dynamic programming procedures jointly to guarantee performance, as shown in the following lemmas, which are the counterparts to Lemma 2 and Lemma 3 for RSP in [70].

Lemma 12.1. *If Algorithm 2 returns a path p and travel time set $\boldsymbol{t}_p$, then we have*

$$\mathit{OPT} \leq c(p, \boldsymbol{t}_p) \leq U + L\delta. \tag{12.4}$$

Proof. Please see Appendix 17.4. □

Lemma 12.2. *If $U \geq \mathit{OPT}$, then Algorithm 2 must return a feasible path p and travel time set $\boldsymbol{t}_p$, which satisfy*

$$c(p, \boldsymbol{t}_p) \leq \mathit{OPT} + L\delta. \tag{12.5}$$

Proof. Please see Appendix 17.5. □

Lemma 12.3. *If Algorithm 2 returns FAIL, then we have*

$$\mathit{OPT} > U. \tag{12.6}$$

Algorithm 2 A Test Procedure $\texttt{TEST}(L, U, \delta)$

1: Set $V = \frac{L\delta}{n+1}$
2: Set $N = \lfloor \frac{U}{V} \rfloor + n + 1$
3: **for** $e \in \mathcal{E}$ **do**
4: Get $\boldsymbol{\tau}_e = \texttt{QUANTIZE}(e, V, N+1)$
5: **end for**
6: Set $g_s(c) = 0$, $\forall c = 0, 1, \cdots, N$
7: Set $g_v(0) = \infty$, $\forall v \neq s, v \in \mathcal{V}$
8: **for** $c = 1, 2, \cdots, N$ **do**
9: **for** $v \in \mathcal{V}$ **do**
10: Set $g_v(c)$ according to (12.3)
11: **end for**
12: **if** $g_d(c) \leq T$ **then**
13: **return** the corresponding path p and travel time set $\boldsymbol{t}_p = \{t_e : e \in p\}$
14: **end if**
15: **end for**
16: **return** FAIL

Proof. This directly follows Lemma 12.2. □

Our test procedure either returns a path p and travel time set $\boldsymbol{t}_p$ in line 13, which implies that $\mathsf{OPT} \leq U + L\delta$ from Lemma 12.1, or returns FAIL in line 16, which implies $\mathsf{OPT} > U$ from Lemma 12.3. Therefore, Lemma 12.1 and Lemma 12.3 justify that our test procedure (Algorithm 2) completes the "approximate" comparison, i.e., answers either $\mathsf{OPT} > U$ or $\mathsf{OPT} \leq U + L\delta$.

Thus, for the purpose of the test procedure, Lemma 12.1 and Lemma 12.3 are enough. However, we present Lemma 12.2, which is stronger than Lemma 12.3, to provide a sufficient condition such that our test procedure returns a path p and travel time set $\boldsymbol{t}_p$. We will use Lemma 12.2 shortly in Sec. 12.3 to finally output a $(1+\epsilon)$-approximate solution.

Time Complexity: The quantizing procedures for all edges in lines 3-5 require $O(mN \log \xi)$. The dynamic programming procedure in lines 6-15 requires $O(mN^2)$. Since $N = \lfloor \frac{U}{V} \rfloor + n + 1 = \lfloor \frac{U}{L} \cdot \frac{n+1}{\delta} \rfloor + n + 1 = O(\frac{U}{L} \cdot \frac{n}{\delta} + n)$, the total time complexity of Algorithm

Algorithm 3 An FPTAS

1: Get a lower bound LB and upper bound UB for OPT
2: Set $B_L = \mathsf{LB}$
3: Set $B_U = \mathsf{UB}$
4: **while** $\frac{B_U}{B_L} > 16$ **do**
5: $X = \sqrt{B_L \cdot B_U}$
6: Call $\texttt{TEST}(X, X, 1)$
7: **if** $\texttt{TEST}(X, X, 1)$ returns FAIL **then**
8: Set $B_L = X$
9: **else**
10: Set $B_U = 2X$
11: **end if**
12: **end while**
13: Call $\texttt{TEST}(B_L, B_U, \epsilon)$

2 is $O(mN \log \xi + mN^2) = O(m(\frac{U}{L} \cdot \frac{n}{\delta} + n) \log \xi + m(\frac{U}{L} \cdot \frac{n}{\delta} + n)^2)$.

12.3 The Proposed FPTAS

Based on our own test procedure (Algorithm 2), we then follow the FPTAS for RSP in [70, Fig. 2] by replacing its test procedure with ours. For completeness, we present the FPTAS in Algorithm 3 and explain it with the following three steps.

Step 1 (line 1): To initialize the bound interval, we need to first obtain a lower bound LB and an upper bound UB for the optimal value OPT. Define that the minimal single-edge fuel cost is $\mathsf{C}^{\mathsf{lb}} \triangleq \min_{e \in \mathcal{E}} c_e(t_e^{\mathsf{ub}})$ and the maximal single-edge fuel cost is $\mathsf{C}^{\mathsf{ub}} \triangleq \max_{e \in \mathcal{E}} c_e(t_e^{\mathsf{lb}})$. Simply, we can use the minimal single-edge fuel consumption C^{lb} as the lower bound LB and use the maximal single-path[1] fuel consumption $n\mathsf{C}^{\mathsf{ub}}$ as the upper bound UB. Also, in Chap. 13, we will propose a heuristic scheme which can always output a set of LB and UB.

Step 2 (lines 2-12): Using the initial lower bound LB and upper bound UB, we design a binary search scheme, which repeatedly invokes our test procedure (Algorithm 2) to

[1] A simple path can have at most n edges.

exponentially narrow down the bound interval $[B_L, B_U]$ until $B_U/B_L \leq 16$. The binary search step is visualized in Fig. 12.2. Note that we always keep B_L as a lower bound and B_U as an upper bound for OPT. Whenever $B_U/B_L > 16$, we input the geometric mean $X = \sqrt{B_L \cdot B_U}$ and $\delta = 1$ to the test procedure, as shown in lines 5 and 6. If $\texttt{TEST}(X, X, 1)$ returns FAIL, then according to Lemma 12.2, we must have $X < \mathsf{OPT}$. In this case, we reset the lower bound B_L to be V in line 8. Otherwise, $\texttt{TEST}(X, X, 1)$ returns a feasible path p and travel time set $\boldsymbol{t}_p$. According to Lemma 12.1, we must have $\texttt{OPT} \leq X + \delta X = 2X$. We reset the upper bound to be $2X$ in line 10. It can be easily shown that this binary search returns a lower bound B_L and an upper bound B_U for OPT such that $B_U/B_L \leq 16$ in $O(\log\log \frac{\mathsf{UB}}{\mathsf{LB}})$ iterations.

Step 3 (line 13): When $\frac{B_U}{B_L} \leq 16$, we call our test procedure again but we use $L = B_L$ and $U = B_U$ and $\delta = \epsilon$. Since $B_U \geq \mathsf{OPT}$, according to Lemma 12.2, $\texttt{TEST}(B_L, B_U, \epsilon)$ must return a feasible path p and travel time $\boldsymbol{t}_p$ such that

$$c(p, \boldsymbol{t}_p) \leq \mathsf{OPT} + \epsilon B_L \leq \mathsf{OPT} + \epsilon\mathsf{OPT} = (1+\epsilon)\mathsf{OPT}.$$

Therefore, we get a $(1+\epsilon)$-approximate solution to PASO.

Time Complexity: *Step 1* requires $O(m)$ to get an initial lower bound LB and upper bound UB. *Step 2* invokes the test procedure $O(\log\log \frac{\mathsf{UB}}{\mathsf{LB}})$ times and each invocation takes $O(mn\log\xi + mn^2)$ time by using $L = U = X$ and $\delta = 1$. Thus *Step 2* takes $O((mn\log\xi + mn^2)\log\log \frac{\mathsf{UB}}{\mathsf{LB}})$. *Step 3* also invokes the test procedure, and it takes $O(\frac{mn\log\xi}{\epsilon} + \frac{mn^2}{\epsilon^2})$ time by using $\delta = \epsilon < 1$ and $O(\frac{U}{L}) = O(\frac{B_U}{B_L}) = O(1)$ because $\frac{B_U}{B_L} \leq 16 = O(1)$. Here we can also see why we need to use a binary search to obtain $\frac{B_U}{B_L} \leq 16$ in *Step 2*. This is because $\frac{B_U}{B_L} = O(1)$ ensures polynomial time complexity in *Step 3*. Therefore, the total complexity is $O((mn\log\xi + mn^2)\log\log \frac{\mathsf{UB}}{\mathsf{LB}} + \frac{mn\log\xi}{\epsilon} + \frac{mn^2}{\epsilon^2})$.

We summarize our results for the approximate scheme in the following theorem.

Theorem 12.1. *Algorithm 3 returns a $(1+\epsilon)$-approximate solution for PASO in time $O((mn\log\xi + mn^2)\log\log \frac{\mathsf{UB}}{\mathsf{LB}} + \frac{mn\log\xi}{\epsilon} + \frac{mn^2}{\epsilon^2})$. In addition, when we use $\mathsf{LB} = \mathsf{C}^{\mathsf{lb}}$ and $\mathsf{UB} = n\mathsf{C}^{\mathsf{ub}}$ where $\mathsf{C}^{\mathsf{lb}} \triangleq \min_{e\in\mathcal{E}} c_e(t_e^{\mathsf{ub}})$ and $\mathsf{C}^{\mathsf{ub}} \triangleq \max_{e\in\mathcal{E}} c_e(t_e^{\mathsf{lb}}) = c_{e_1}(t_{e_1}^{\mathsf{lb}})$, we have $\log\log \frac{\mathsf{UB}}{\mathsf{LB}} = \max\{O(\log\log n), O(I_{e_1})\}$ where I_{e_1} is the input size of all parameters of edge e_1. Thus, Algorithm 3 has time complexity polynomial in the input size of the problem PASO and therefore is an FPTAS.*

Proof. Please see Appendix 17.6. □

Although we generalize the FPTAS design from RSP to PASO, such an FPTAS (Algorithm 3) still has high complexity for a large-scale highway network with tens of thousands of nodes and edges. In the next chapter, we propose a heuristic scheme with substantially lower complexity.

12.4 Compare FPTAS with a Simple Layered Algorithm

For those who are familiar with the FPTAS of RSP [50], it appears that our FPTAS of PASO shares some similarities with the FPTAS of RSP. One may ask whether we can use the FPTAS of RSP as a blackbox to design an FPTAS of PASO. In this section, we propose such a simple layered algorithm, which first quantizes the fuel-time function $c_e(t_e)$, and then constructs a new graph with fixed travel time and cost, and finally applies the FPTAS of RSP for the new graph. We show that this simple layered algorithm outputs a $(1+\epsilon)$-approximate solution to PASO, but its time complexity is no longer polynomial in the problem size, which means that it is not an FPTAS of PASO.

The idea of the simple layered algorithm is as follows. We first quantize the fuel-time function $c_e(t_e)$ to a staircase function $\tilde{c}_e(t_e)$, as shown in Sec. 12.1. If the quantization is accurate enough, we can show that the optimal solution to PASO for graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with quantized fuel-time function $\tilde{c}_e(t_e)$ is also a near-optimal solution to PASO for graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with original fuel-time function $c_e(t_e)$. Thus we can resort to solving PASO for graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with quantized fuel-time function $\tilde{c}_e(t_e)$. Since $\tilde{c}_e(t_e)$ is a staircase function , we can prove that there exists an optimal solution to PASO for graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\tilde{c}_e(t_e)$, where the optimal travel time of any edge $e \in \mathcal{E}$ belongs to one of the representative points of edge e (see the definition of representative points in Sec. 12.1). Hence, though $\tilde{c}_e(t_e)$ is a continuous function, it suffices to focus on those discrete representative points to solve PASO. Then we can construct another graph $\tilde{\mathcal{G}} = (V, \tilde{\mathcal{E}})$ with only fixed travel time and fixed travel cost based on representative points. If edge $e = (u, v) \in \mathcal{E}$ has k discrete representative points, then there are k parallel edges from u to v in the new graph $\tilde{\mathcal{G}}$, all of which have different (but fixed) travel time and cost. Solving PASO for graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\tilde{c}_e(t_e)$ is equivalent to solving RSP for graph $\tilde{\mathcal{G}} = (V, \tilde{\mathcal{E}})$. Finally,

Algorithm 4 A Simple Layered Algorithm

1: Set $\epsilon_1 = \frac{\epsilon}{2}$
2: Set $V = \frac{\mathsf{LB}\cdot\epsilon_1}{n}$
3: Set $N = \lfloor \frac{\mathsf{C^{ub}}}{V} \rfloor + 1 = \lfloor \frac{1}{\epsilon_1} \cdot \frac{\mathsf{UB}}{\mathsf{LB}} \rfloor + 1$
4: **for** $e \in \mathcal{E}$ **do**
5: Get $\boldsymbol{\tau}_e = \texttt{QUANTIZE}(e, V, N)$
6: **end for**
7: Construct another graph $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{E}})$ and initially set $\tilde{\mathcal{E}} = \emptyset$
8: **for** $e = (u, v) \in \mathcal{E}$ **do**
9: **for** $i = 1, 2, \cdots, N$ **do**
10: **if** $\tau_e^i \neq \mathsf{nan}$ **then**
11: Add edge (u, v) with fixed travel time τ_e^i and fixed travel cost $\tilde{c}_e(\tau_e^i)$ (see (12.1) for the definition of $\tilde{c}_e(\cdot)$) into $\tilde{\mathcal{E}}$
12: **end if**
13: **end for**
14: **end for**
15: Set $\epsilon_2 = \frac{\epsilon}{2+\epsilon}$
16: Call the FPTAS of RSP in [50, Section 4] for graph $\tilde{\mathcal{G}}$ and approximate ratio ϵ_2
17: Return the output path p and travel time set $\boldsymbol{t}_p$.

we can use the FPTAS of RSP to solve RSP problem for graph $\tilde{\mathcal{G}} = (V, \tilde{\mathcal{E}})$. The output solution is an approximate solution to PASO for graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with quantized fuel-time function $\tilde{c}_e(t_e)$, which is further an approximate solution to PASO for graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with original fuel-time function $c_e(t_e)$. Next, we present this simple layered algorithm.

Same in Sec. 12.3, we define the minimal single-edge fuel cost $\mathsf{C^{lb}} \triangleq \min_{e\in\mathcal{E}} c_e(t_e^{\mathsf{ub}})$ and the maximal single-edge fuel cost $\mathsf{C^{ub}} \triangleq \max_{e\in\mathcal{E}} c_e(t_e^{\mathsf{lb}})$. We then define $\mathsf{LB} \triangleq \mathsf{C^{lb}}$ as the lower bound for OPT and $\mathsf{UB} \triangleq n\mathsf{C^{ub}}$ as the upper bound for OPT. The simple layered algorithm is shown in Algorithm 4.

Step 1 (lines 1-6): We quantize the fuel-time function $c_e(t_e)$ for all edges $e \in \mathcal{E}$ with $V = \frac{\mathsf{LB}\cdot\epsilon_1}{n}$ and $N = \lfloor \frac{\mathsf{C^{ub}}}{V} \rfloor + 1 = \lfloor \frac{1}{\epsilon_1} \cdot \frac{\mathsf{UB}}{\mathsf{LB}} \rfloor + 1$. The time complexity of *Step 1* is $O(mN \log \xi)$.

Step 2 (lines 7-14): We construct another graph $\tilde{\mathcal{G}} = (V, \tilde{\mathcal{E}})$ based on representative points. Note that if edge $e = (u, v) \in \mathcal{E}$ in the original graph $\mathcal{G}$ has $k \leq N$ discrete

representative points, then there are k parallel edges from u to v in the new graph $\tilde{\mathcal{G}}$. The new graph $\tilde{\mathcal{G}}$ has $|\mathcal{V}| = n$ nodes and $|\tilde{\mathcal{E}}| = O(mN)$ edges. The time complexity of *Step 2* is $O(n + mN)$.

Step 3 (lines 15-16): We apply the FPTAS in [50, Section 4] to solve the RSP problem for graph $\tilde{\mathcal{G}}$. The output solution is an $(1 + \epsilon_2)$-approximate solution to PASO for graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with quantized fuel-time function $\tilde{c}_e(t_e)$, and further is a $(1 + \epsilon)$-approximate solution to PASO for graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with original fuel-time function $c_e(t_e)$. Note that we should get a lower bound $\tilde{\mathsf{LB}}$ and an upper bound $\tilde{\mathsf{UB}}$ for the optimal value. It is easy to see that we can set $\tilde{\mathsf{LB}} = \lfloor \frac{\mathsf{C^{lb}}}{V} \rfloor + 1 = \lfloor \frac{n}{\epsilon_1} \rfloor + 1$ and $\tilde{\mathsf{UB}} = n(\lfloor \frac{\mathsf{C^{ub}}}{V} \rfloor + 1) = n(\lfloor \frac{1}{\epsilon_1} \cdot \frac{\mathsf{UB}}{\mathsf{LB}} \rfloor + 1)$, implying that $\frac{\tilde{\mathsf{UB}}}{\tilde{\mathsf{LB}}} = O(\frac{\mathsf{UB}}{\mathsf{LB}})$. Thus, according to [50, Section 4], the time complexity of *Step 3* is $O(\log\log \frac{\tilde{\mathsf{UB}}}{\tilde{\mathsf{LB}}}(\frac{|\tilde{\mathcal{E}}| \cdot n}{\epsilon_2} + \log\log \frac{\tilde{\mathsf{UB}}}{\tilde{\mathsf{LB}}})) = O(\log\log \frac{\mathsf{UB}}{\mathsf{LB}}(\frac{mnN}{\epsilon_2} + \log\log \frac{\mathsf{UB}}{\mathsf{LB}}))$.

Time Complexity: According to the above three-step analysis, the total time complexity of the simple layered algorithm (Algorithm 4) is $O(mN \log \xi + \log\log \frac{\mathsf{UB}}{\mathsf{LB}}(\frac{mnN}{\epsilon_2} + \log\log \frac{\mathsf{UB}}{\mathsf{LB}}))$. From line 3 in Algorithm 4, we get that $N = O(\frac{1}{\epsilon_1} \cdot \frac{\mathsf{UB}}{\mathsf{LB}})$. Thus, the total time complexity is $O(\frac{m}{\epsilon_1} \cdot \frac{\mathsf{UB}}{\mathsf{LB}} \cdot \log \xi + \log\log \frac{\mathsf{UB}}{\mathsf{LB}}(\frac{mn}{\epsilon_1 \cdot \epsilon_2} \cdot \frac{\mathsf{UB}}{\mathsf{LB}} + \log\log \frac{\mathsf{UB}}{\mathsf{LB}}))$ where $\epsilon_1 = \frac{\epsilon}{2}$ and $\epsilon_2 = \frac{\epsilon}{2+\epsilon}$, as defined in line 1 and line 15 in Algorithm 4, respectively.

We summarize the time complexity and prove the approximate ratio of Algorithm 4 in the following theorem.

Theorem 12.2. *Algorithm 4 returns a $(1 + \epsilon)$-approximate solution for PASO in time $O(\frac{m}{\epsilon_1} \cdot \frac{UB}{LB} \cdot \log \xi + \log\log \frac{UB}{LB}(\frac{mn}{\epsilon_1 \cdot \epsilon_2} \cdot \frac{UB}{LB} + \log\log \frac{UB}{LB}))$ where $\epsilon_1 = \frac{\epsilon}{2}$ and $\epsilon_2 = \frac{\epsilon}{2+\epsilon}$.*

Proof. Please see Appendix 17.7. □

Theorem 12.2 shows that our simple layered algorithm (which uses the FPTAS of RSP as a blackbox) indeed can output a $(1 + \epsilon)$-approximate solution. But the time complexity is no longer polynomial in the problem size since $N = O(\frac{\mathsf{UB}}{\mathsf{LB}})$ is not polynomial in the problem size. The reason is as follows. In the simple layered algorithm, we only quantize the fuel-time functions once. In order to guarantee the approximate ratio, we must quantize the fuel-time functions accurately enough. This makes the number of stairs N too large, i.e., in the order of the ratio of the initial upper bound UB to the initial lower bound LB, which is not polynomial in the problem size any more.

As compared to this simple layered algorithm (Algorithm 4), our FPTAS (Algorithm 3)

first iteratively finds a better lower bound and upper bound until the ratio of the upper bound to the lower bound is within a constant (see lines 2-12 in Algorithm 3) and then performs an accurate quantization to output the $(1+\epsilon)$-approximate solution. During the procedure to narrow down the ratio of the upper bound to the lower bound, our FPTAS utilizes the test procedure `test`$(X, X, 1)$ (see line 6 in Algorithm 3) which also needs to perform a quantization but the number of stars N is always guaranteed to be polynomial in the problem size.[2] Therefore, our FPTAS avoids quantizations with too many stairs and thus can guarantee polynomial time complexity.

[2] According to lines 1-2 in Algorithm 2, when $L = U = X, \delta = 1$, we have $N = \lfloor n + 1 \rfloor + n + 1 = O(n)$.

Chapter 13

A Fast Dual-Based Heuristic

In this section, we present a heuristic scheme for our problem PASO based on Lagrangian relaxation. Such a heuristic scheme, as we will show later in Sec. 13.3, runs much faster than the FPTAS (Algorithm 3). Also, it always outputs a lower bound LB and an upper bound UB on OPT, which implements *Step 1* in Algorithm 3. Moreover, in most practical scenarios as shown in Chap. 15, this heuristic scheme outputs an optimal (or at least near-optimal) solution, i.e., $\mathsf{LB} = \mathsf{UB} = \mathsf{OPT}$ (or at least $\mathsf{LB} \approx \mathsf{OPT} \approx \mathsf{UB}$).

13.1 Lagrangian Relaxation and Dual Problem

In our problem PASO, since the hard delay constraint (11.4) couples path selection variable $\boldsymbol{x}$ with speed optimization variable $\boldsymbol{t}$, we relax it and introduce a Lagrangian dual variable $\lambda \geq 0$, which can be interpreted as a (per-unit) delay price *over the entire network*.

Based on such relaxation, we can get the corresponding Lagrangian,

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{t}, \lambda) &\triangleq \sum_{e \in \mathcal{E}} x_e \cdot c_e(t_e) + \lambda \left(\sum_{e \in \mathcal{E}} x_e t_e - T \right) \\ &= \sum_{e \in \mathcal{E}} x_e \cdot \left(c_e(t_e) + \lambda t_e \right) - \lambda T, \end{aligned} \tag{13.1}$$

and the corresponding dual function is defined as $D(\lambda) \triangleq \min_{\boldsymbol{x} \in \mathcal{X}, \boldsymbol{t} \in \mathcal{T}} L(\boldsymbol{x}, \boldsymbol{t}, \lambda)$. Then the dual problem of PASO is formulated as

$$(\mathsf{PASO\text{-}Dual}) \quad \max_{\lambda \geq 0} D(\lambda)$$

13.2 Obtain Dual Function

Before we solve the dual problem, let us first show how to obtain the dual function for a given λ as follows,

$$\begin{aligned}
D(\lambda) &= \min_{\boldsymbol{x}\in\mathcal{X},\boldsymbol{t}\in\mathcal{T}} L(\boldsymbol{x},\boldsymbol{t},\lambda) \\
&= -\lambda T + \min_{\boldsymbol{x}\in\mathcal{X},\boldsymbol{t}\in\mathcal{T}} \sum_{e\in\mathcal{E}} x_e\cdot(c_e(t_e)+\lambda t_e) \\
&\overset{(E_1)}{=} -\lambda T + \min_{\boldsymbol{x}\in\mathcal{X}} \left[\min_{\boldsymbol{t}\in\mathcal{T}} \sum_{e\in\mathcal{E}} x_e\cdot(c_e(t_e)+\lambda t_e)\right] \\
&\overset{(E_2)}{=} -\lambda T + \min_{\boldsymbol{x}\in\mathcal{X}} \sum_{e\in\mathcal{E}} x_e\cdot \min_{t_e^{\mathsf{lb}}\le t_e\le t_e^{\mathsf{ub}}} (c_e(t_e)+\lambda t_e) \\
&\overset{(E_3)}{=} -\lambda T + \min_{\boldsymbol{x}\in\mathcal{X}} \sum_{e\in\mathcal{E}} x_e\cdot [c_e(t_e^*(\lambda))+\lambda t_e^*(\lambda)] \\
&\overset{(E_4)}{=} -\lambda T + \min_{\boldsymbol{x}\in\mathcal{X}} \sum_{e\in\mathcal{E}} x_e\cdot w_e(\lambda) \\
&\overset{(E_5)}{=} -\lambda T + \sum_{e\in p^*(\lambda)} w_e(\lambda). \qquad (13.2)
\end{aligned}$$

We explain $(E_1)-(E_5)$ in (13.2) one by one. Equality (E_1) is because no coupled constraints exist for $\boldsymbol{x}$ and $\boldsymbol{t}$. Equality (E_2) is because no coupled constraints exist for the travel time at different edges in $\mathcal{T}$. Recall that $\mathcal{T} \triangleq \{\boldsymbol{t} : t_e^{\mathsf{lb}} \le t_e \le t_e^{\mathsf{ub}}, \forall e\in\mathcal{E}\}$ as defined in (11.6).

In equality (E_3), $t_e^*(\lambda)$ is defined as

$$t_e^*(\lambda) \triangleq \arg\min_{t_e^{\mathsf{lb}}\le t_e\le t_e^{\mathsf{ub}}} (c_e(t_e)+\lambda t_e)\,. \qquad (13.3)$$

Note that since we have assumed that $c_e(t_e)$ is strictly convex and strictly decreasing over $[t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]$ in Sec. 11.4, $t_e^*(\lambda)$ is unique and thus (13.3) is well defined. Specifically, $t_e^*(\lambda)$ can be obtained analytically as follows.

Lemma 13.1. *Define $c_e'^{-1}(\cdot)$ as the inverse function of $c_e'(\cdot)$. Then we have*

$$t_e^*(\lambda) = \begin{cases} t_e^{\mathsf{ub}}, & \text{If } 0\le\lambda<-c_e'(t_e^{\mathsf{ub}}); \\ c_e'^{-1}(-\lambda), & \text{If } -c_e'(t_e^{\mathsf{ub}})\le\lambda\le -c_e'(t_e^{\mathsf{lb}}); \\ t_e^{\mathsf{lb}}, & \text{If } \lambda > -c_e'(t_e^{\mathsf{lb}}). \end{cases} \qquad (13.4)$$

Proof. Please see Appendix 17.8. □

In addition, (13.3) has a nice economic interpretation. As we have relaxed the hard delay constraint, we penalize each edge e with a delay cost, which is the product of the travel time t_e and the (per-unit) delay price λ. Then for a given delay price λ, each edge selects the optimal travel time to minimize its *generalized* cost, including both fuel cost $c_e(t_e)$ and delay cost λt_e. Thus, $t_e^*(\lambda)$ is the *best response* of edge e for a given delay price λ.

In equality (E_4), $w_e(\lambda)$ is defined as

$$w_e(\lambda) \triangleq c_e(t_e^*(\lambda)) + \lambda t_e^*(\lambda), \tag{13.5}$$

which can be interpreted as the minimal generalized cost (including both fuel cost and delay cost) of edge e for a given delay price λ. Obviously, $w_e(\lambda)$ is the generalized cost under the best response $t_e^*(\lambda)$.

In equality (E_5), since $\mathcal{X}$ restricts that an $s-d$ path is selected, $\min_{\boldsymbol{x}\in\mathcal{X}} \sum_{e\in\mathcal{E}} x_e \cdot w_e(\lambda)$ is exactly a shortest path problem where each edge e has a generalized cost $w_e(\lambda)$. We define $p^*(\lambda)$ as the resulting shortest-generalized-cost path.

In summary, (13.2) shows that for any dual variable λ, we only need to solve a shortest path problem to obtain the dual function value $D(\lambda)$, which is much easier than PASO.

13.3 The Heuristic Algorithm

Our heuristic scheme relies on one key observation. Define

$$\delta(\lambda) \triangleq \sum_{e\in p^*(\lambda)} t_e^*(\lambda), \tag{13.6}$$

which is the total travel time of the resulting shortest-generalized-cost path $p^*(\lambda)$ for a given λ. Our key observation is the following theorem (see an example in Fig. 15.3).

Theorem 13.1. *$\delta(\lambda)$ is non-increasing over $\lambda \in [0, +\infty)$.*

Proof. Please see Appendix 17.9. □

Theorem 13.1 shows that increasing λ will decrease the total travel time of the selected path based on the best responses of all edges. Intuitively, since λ can be interpreted as

a delay price, increasing λ will force all edges to select a shorter travel time and further force the resulting shortest-generalized-cost path to have a shorter travel time.

Based on Theorem 13.1, we can use a simple dual variable λ to *coordinate* the total travel time. For example, when $\delta(\lambda) > T$, we can increase λ such that $\delta(\lambda)$ can be decreased to finally satisfy the hard delay requirement. On the other hand, when $\delta(\lambda) < T$, it means that the truck travels very fast and there still exists some room to increase the travel time and thus decrease the fuel consumption. Then we decrease λ such that $\delta(\lambda)$ can be increased to reach T. This is called a *coordination mechanism* [28, Ch. 5.1.6]. Therefore, we aim to find a λ_0 such that $\delta(\lambda_0) = T$. However, our problem PASO is not convex but has a combinatorial difficulty. Thus it is not guaranteed to find such a λ_0. We thus call our binary search for λ_0 (Algorithm 5) as a heuristic scheme.

In Algorithm 5, we first set an initial lower bound $\lambda_L = 0$ and an initial upper bound $\lambda_U = \lambda_{\max}$ for the targeted λ_0. We discuss how to set $\lambda_{\max}$ in Sec. 13.4.1. Then we do binary search in lines 3-19, where tol in line 3 is the tolerance level for termination which is close to zero. During the binary search, based on the non-increasing property of $\delta(\lambda)$ (Theorem 13.1), we keep updating the lower bound λ_L and its corresponding solution $(p^*(\lambda_L), \{t_e^*(\lambda_L) : e \in p^*(\lambda_L)\})$, as well as the upper bound λ_U and its corresponding solution $(p^*(\lambda_U), \{t_e^*(\lambda_U) : e \in p^*(\lambda_U)\})$.

This algorithm has two possible results:

▷ **Case 1:** If it returns in line 9, then we have found a λ_0 such that $\delta(\lambda_0) = T$. We prove that the returned solution is optimal for PASO in Theorem 13.2.

▷ **Case 2:** If it returns in line 20, then we have found a λ_0 such that $\delta(\lambda_L) > T$ and $\delta(\lambda_U) < T$. With a small enough tolerance level tol, $\lambda_L = \lambda_0 - \mathsf{tol}/2 \to \lambda_0^-$. Likewise, $\lambda_U = \lambda_0 + \mathsf{tol}/2 \to \lambda_0^+$. Roughly speaking, this means that $\delta(\lambda)$ is not continuous at $\lambda = \lambda_0$. Although this return does not guarantee optimality, we prove in Theorem 13.3 that the returned solutions $(p^*(\lambda_L), \{t_e^*(\lambda_L) : e \in p^*(\lambda_L)\})$ and $(p^*(\lambda_U), \{t_e^*(\lambda_U) : e \in p^*(\lambda_U)\})$ give a lower bound LB and an upper bound UB for OPT, respectively. We thus use $(p^*(\lambda_U), \{t_e^*(\lambda_U) : e \in p^*(\lambda_U)\})$ as the output solution to PASO in this case.

Theorem 13.2. *If Algorithm 5 returns in line 9, then the returned solution* $(p^*(\lambda_0), \{t_e^*(\lambda_0) : e \in p^*(\lambda_0)\})$ *is an optimal solution of PASO.*

Proof. Please see Appendix 17.10. □

Algorithm 5 A Heuristic Scheme

1: Set $\lambda_L = 0$
2: Set $\lambda_U = \lambda_{\max}$
3: **while** $\lambda_U - \lambda_L >$ tol **do**
4: Set $\lambda_0 = \frac{\lambda_L + \lambda_U}{2}$
5: Get $t_e^*(\lambda_0)$ from Lemma 13.1 for all $e \in \mathcal{E}$
6: Get $w_e(\lambda_0) = c_e(t_e^*(\lambda_0)) + \lambda_0 t_e^*(\lambda_0)$ for all $e \in \mathcal{E}$
7: Get the shortest path $p^*(\lambda_0)$ in terms of $w_e(\lambda_0)$
8: **if** $\delta(p^*(\lambda_0)) = T$ **then**
9: **return** $(p^*(\lambda_0), \{t_e^*(\lambda_0)\})$
10: **else if** $\delta(p^*(\lambda_0)) > T$ **then**
11: Set $\lambda_L = \lambda_0$
12: Set $p^*(\lambda_L) = p^*(\lambda_0)$
13: Set $t_e^*(\lambda_L) = t_e^*(\lambda_0), \forall e \in \mathcal{E}$
14: **else**
15: Set $\lambda_U = \lambda_0$
16: Set $p^*(\lambda_U) = p^*(\lambda_0)$
17: Set $t_e^*(\lambda_U) = t_e^*(\lambda_0), \forall e \in \mathcal{E}$
18: **end if**
19: **end while**
20: **return** $(p^*(\lambda_L), \{t_e^*(\lambda_L)\})$ and $(p^*(\lambda_U), \{t_e^*(\lambda_U)\})$

As a by-product, Theorem 13.2 also shows that the strong duality for the combinatorial problem PASO holds in this case. Also, λ_0 is the optimal solution to the dual problem PASO-Dual.

Theorem 13.3. *If Algorithm 5 returns in line 20, and define* $LB \triangleq \sum_{e \in p^*(\lambda_L)} c_e(t_e^*(\lambda_L))$ *and* $UB \triangleq \sum_{e \in p^*(\lambda_U)} c_e(t_e^*(\lambda_U))$*, then we have* $LB \leq OPT \leq UB$.

Proof. Please see Appendix 17.11. □

Since $(p^*(\lambda_U), \{t_e^*(\lambda_U) : e \in p^*(\lambda_U)\})$ is a feasible solution to PASO, we use it as the output solution to our problem PASO if Algorithm 5 returns in line 20. Though this solution is not optimal to PASO, we evaluate its performance later in Sec. 13.4.2.

We should further note that the LB and UB returned by Algorithm 5 in line 20 can be used for *Step 1* of Algorithm 3. For the case that Algorithm 5 returns in line 9, we use the returned optimal solution as both a lower bound and an upper bound with $\mathsf{LB} = \mathsf{UB} = \mathsf{OPT}$. After such unification, Algorithm 5 always outputs a LB and UB for the optimal solution OPT.

Time Complexity: If we use Dijkstra's shortest-path algorithm with a min-priority queue in line 7 in Algorithm 5, Algorithm 5 has complexity
$O((m + n \log n) \log \lambda_{\max})$, much faster than the FPTAS (Algorithm 3).

Remark: A similar dual-based heuristical approach for RSP is proposed in [63]. However, as mentioned in Chap. 12, different from RSP, our problem PASO has an extra design space of speed optimization. Therefore, theoretically our contribution in this section is to generalize the dual-based heuristical design from RSP [63] to PASO.

13.4 Discussions

13.4.1 How to Set $\lambda_{\max}$?

Since our heuristic algorithm has complexity $O(\log \lambda_{\max}(m + n \log n))$, in this subsection, we show how to set $\lambda_{\max}$.

Define $\mathcal{P}$ as the set of all $s - d$ paths. Suppose each edge $e \in \mathcal{E}$ has a fixed travel time t_e^{lb} and a fixed travel/fuel cost $c_e^{\mathsf{ub}} = c_e(t_e^{\mathsf{lb}})$.[1] Define $C_p \triangleq \sum_{e \in p} c_e^{\mathsf{ub}}$ and $T_p \triangleq \sum_{e \in p} t_e^{\mathsf{lb}}$ as the travel cost and travel time of any path $p \in \mathcal{P}$. Define T^{lb} as the smallest travel time from s to d, i.e.,

$$T^{\mathsf{lb}} \triangleq \min_{p \in \mathcal{P}} T_p. \tag{13.7}$$

It is possible that there exist multiple fastest paths. We then define $\mathcal{P}_F$ as the set of all fastest $s - d$ paths, i.e.,

$$\mathcal{P}_F \triangleq \{p \in \mathcal{P} : T_p = T^{\mathsf{lb}}\}. \tag{13.8}$$

Among all fastest paths, we define p^* as an arbitrary path with minimal fuel cost, i.e.,

$$p^* \in \arg\min_{p \in \mathcal{P}_F} C_p. \tag{13.9}$$

[1] These are the travel time and fuel cost when the truck runs at the highest speed over edge e.

Clearly, $T_{p^*} = T^{\mathsf{lb}}$ and $C_{p^*} \leq C_p, \forall p \in \mathcal{P}_F$.

Now we define the following three λ values:

$$\lambda_1 \triangleq \max_{e \in \mathcal{E}} \left\{ -c'_e(t_e^{\mathsf{lb}}) \right\}, \tag{13.10}$$

$$\lambda_2 \triangleq \max_{p \in \mathcal{P} \setminus \mathcal{P}_F} \frac{C_{p^*} - C_p}{T_p - T_{p^*}}, \tag{13.11}$$

$$\bar{\lambda} \triangleq \max\{\lambda_1, \lambda_2\}. \tag{13.12}$$

We has the following result.

Lemma 13.2. *For all $\lambda \geq \bar{\lambda}$, $\delta(\lambda) = T^{lb}$.*

Proof. See Appendix 17.12 □

Lemma 13.2 suggests that when $\lambda \geq \bar{\lambda}$, $\delta(\lambda)$ will not change anymore. Therefore, we can set $\lambda_{\max}$ to be $\bar{\lambda}$. However, it could be difficult to obtain λ_2 (and also $\bar{\lambda}$) because the number of $s-d$ paths, i.e., $|\mathcal{P}|$, could exponentially increase when the the network size increases. We then find an easy-to-compute upper bound for λ_2,

$$\lambda_2 = \max_{p \in \mathcal{P} \setminus \mathcal{P}_F} \frac{C_{p^*} - C_p}{T_p - T_{p^*}} \leq \max_{p \in \mathcal{P} \setminus \mathcal{P}_F} \frac{C_{p^*}}{T_p - T_{p^*}} = \frac{C_{p^*}}{\min\limits_{p \in \mathcal{P} \setminus \mathcal{P}_F} T_p - T_{p^*}} \triangleq \lambda_3. \tag{13.13}$$

Note that $\min\limits_{p \in \mathcal{P} \setminus \mathcal{P}_F} T_p$ is the minimal travel time among all $s-d$ paths excluding those fastest paths, namely, the second smallest travel time among all $s-d$ paths. To compute λ_3, we can use the polynomial-time algorithm[2] in [41] to find k shortest path in terms of edge cost t_e^{lb}. As we do not know the number of fastest paths, i.e., $|\mathcal{P}_F|$, we cannot specify k. Heuristically, we can try different k (e.g., increase k one by one) until we find a path with the second smallest travel time. Note that we also find all fastest paths. Thus, we can obtain C_{p^*},T_{p^*} and $\min\limits_{p \in \mathcal{P} \setminus \mathcal{P}_F} T_p$, and therefore compute λ_3.

We can then set $\lambda_{\max}$ as

$$\lambda_{\max} \triangleq \max\{\lambda_1, \lambda_3\} \geq \max\{\lambda_1, \lambda_2\} = \bar{\lambda}. \tag{13.14}$$

Remark: We note that λ_3 (and also $\lambda_{\max}$) depends on (i) the fuel consumption of the fastest path (when the truck runs at the highest speed), and (ii) the total travel time

[2]More specifically, the time complexity of the algorithm in [41] to find k shortest paths is $O(m+n\log n+kn)$ for a directed graph with n nodes and m edges.

difference between the fastest path and the second fastest path (when the truck runs at the highest speed).

The analysis above is about how to set $\lambda_{\max}$ theoretically. Practically, since we are considering the fuel consumption and λ can be interpreted as a delay price, $\lambda_{\max}$ can be reasonably set to be an upper bound of the fuel consumption per hour. In our simulation in Chap. 15, we set $\lambda_{\max} = 100$, which works for all settings.

13.4.2 How to Characterize the Optimality Gap of the Heuristic Algorithm?

As we proved in Theorem 13.2, if our heuristic algorithm (Algorithm 5) returns in line 9, then the returned solution $(p^*(\lambda_0), \{t_e^*(\lambda_0) : e \in p^*(\lambda_0)\})$ is an optimal solution of PASO. But if Algorithm 5 returns in line 20, we only get a feasible (but not necessarily optimal) solution to our problem PASO, i.e., $(p^*(\lambda_U), \{t_e^*(\lambda_U) : e \in p^*(\lambda_U)\})$. In this subsection, we evaluate the performance of this solution.

Suppose that our heuristic algorithm (Algorithm 5) returns in line 20. For simplicity, we assume that $\mathsf{tol} = 0$ and thus we have $\lambda_L = \lambda_U = \lambda_0$. We further let $p_1 = p^*(\lambda_L)$ and $p_2 = p^*(\lambda_U)$ as the $s - d$ path corresponding to the lower bound solution and the upper bound solution, respectively. Denote $C_p = \sum_{e \in p} c_e(t_e^*(\lambda_0))$ and $T_p = \sum_{e \in p} t_e^*(\lambda_0)$ as the fuel cost and travel time of path p when $\lambda = \lambda_0$. Since $\delta(\lambda)$ function has a jump (switch from T_{p_1} to T_{p_2}) at the point $\lambda = \lambda_0$, we must have[3]

$$C_{p_1} + \lambda_0 T_{p_1} = C_{p_2} + \lambda_0 T_{p_2}, \tag{13.15}$$

i.e., the generalized cost of path p_1 is equal to the generalized cost of path p_2 when $\lambda = \lambda_0$.

As proved in Theorem 13.3, we have that $C_{p_1} = \mathsf{LB} \le \mathsf{OPT} \le \mathsf{UB} = C_{p_2}$. Therefore, (13.15) implies that the performance gap (in terms of the total fuel cost) between the output solution $(p_2, \{t_e^*(\lambda_U) : e \in p_2\})$ and the optimal solution can be upper bounded as

$$C_{p_2} - \mathsf{OPT} \le C_{p_2} - \mathsf{LB} = C_{p_2} - C_{p_1} = \lambda_0(T_{p_1} - T_{p_2}), \tag{13.16}$$

[3]It is easy to prove that the edge-e generalized cost $w_e(\lambda)$ defined in (13.5) is a continuous function with respect to λ. Therefore, the path-p generalized cost $\sum_{e \in p} w_e(\lambda)$ is also a continuous function with respect to λ. Then we can show that the generalized cost of path p_1 is equal to the generalized cost of path p_2 when $\lambda = \lambda_0$.

where $(T_{p_1} - T_{p_2})$ is the jump range of $\delta(\lambda)$ at the point $\lambda = \lambda_0$.

According to our simulations later in Chap. 15, both the jump range and the performance gap are usually very small. For example, in Fig. 15.3, the jump range is only 0.2 and the performance gap $\lambda_0(T_{p_1} - T_{p_2}) = 11.5 \times 0.2 = 2.3$, which is negligible to OPT, usually more than 300 (gallons of fuels) for a 40-hour trip.

We should further note that the upper bound $\lambda_0(T_{p_1} - T_{p_2})$ can only be determined after we run our heuristic algorithm. To obtain an upper bound before running our heuristic algorithm, we can further upper bound $\lambda_0(T_{p_1} - T_{p_2})$ as,

$$\lambda_0(T_{p_1} - T_{p_2}) \leq \lambda_{\max}(T^{\mathsf{ub}} - T^{\mathsf{lb}}), \tag{13.17}$$

where $T^{\mathsf{ub}} \triangleq \max_{p\in\mathcal{P}}(\sum_{e\in p} t_e^{\mathsf{ub}})$ is the maximal travel time, and $T^{\mathsf{lb}} \triangleq \min_{p\in\mathcal{P}}(\sum_{e\in p} t_e^{\mathsf{lb}})$ is the minimal travel time (same as (13.7)). To avoid solving a NP-hard longest path problem to obtain T^{ub}, we can further upper bound T^{ub} as $T^{\mathsf{ub}} \leq n \cdot \max_{e\in\mathcal{E}} t_e^{\mathsf{ub}}$. Thus, the performance gap of our heuristic solution is upper bounded by

$$C_{p_2} - \mathsf{OPT} \leq \lambda_0(T_{p_1} - T_{p_2}) \leq \lambda_{\max}(T^{\mathsf{ub}} - T^{\mathsf{lb}}) \leq \lambda_{\max}(n \cdot \max_{e\in\mathcal{E}} t_e^{\mathsf{ub}} - T^{\mathsf{lb}}), \tag{13.18}$$

which can be obtained before we run our heuristic algorithm.

13.4.3 How to Generalize Our Approach?

For our NP-complete mixed discrete-continuous problem PASO, our dual-based heuristic algorithm has much lower complexity than the FPTAS, and it outputs an optimal solution under a condition (see Theorem 13.2). It is interesting to see whether our approach/techinque can be generalized to solve other problems. In this subsection, we generalize Theorem 13.2 for a class of mixed discrete-continuous problems.

Consider a general mixed discrete-continuous optimization problem, called $(\mathbf{P})$,

$$\begin{aligned}
(\mathbf{P}): \quad \min \quad & f(\boldsymbol{x}, \boldsymbol{t}) && (13.19)\\
\text{s.t.} \quad & (\boldsymbol{x}, \boldsymbol{t}) \in \mathcal{C} && (13.20)\\
& g(\boldsymbol{x}, \boldsymbol{t}) \leq 0 && (13.21)\\
\text{var.} \quad & \boldsymbol{x} \in \{0,1\}^m, \boldsymbol{t} \in \mathbb{R}^n && (13.22)
\end{aligned}$$

Here we assume that $m, n \in \mathbb{Z}^+$. If $m = 0$, then $(\mathbf{P})$ is a pure continuous optimization problem. If $n = 0$, then $(\mathbf{P})$ is a pure discrete optimization problem. Set $\mathcal{C}$ is a constraint

set for the optimization variables $(\boldsymbol{x}, \boldsymbol{t})$. Also, $f : \{0,1\}^m \times \mathbb{R}^+ \to \mathbb{R}$ is the objective function and $g : \{0,1\}^m \times \mathbb{R}^+ \to \mathbb{R}$ is the extra constraint function. Note that we assume that f, g, and $\mathcal{C}$ are general. We denote the optimal value of problem $(\mathbf{P})$ as OPT. One can easily see that our problem PASO in (11.3)-(11.4) is a concrete example in the format of problem $(\mathbf{P})$ where the hard delay constraint (11.4) is the extra constraint (13.21).

We assume that problem $(\mathbf{P})$ is hard to solve due to the extra constraint (13.21), but becomes easy to solve if we relax (13.21). More specifically, we define another mixed discrete-continuous optimization problem, called $(\tilde{\mathbf{P}}(\lambda))$,

$$(\tilde{\mathbf{P}}(\lambda)): \quad \min \quad f(\boldsymbol{x}, \boldsymbol{t}) + \lambda g(\boldsymbol{x}, \boldsymbol{t}) \tag{13.23}$$

$$\text{s.t.} \quad (\boldsymbol{x}, \boldsymbol{t}) \in \mathcal{C} \tag{13.24}$$

$$\text{var.} \quad \boldsymbol{x} \in \{0,1\}^m, \boldsymbol{t} \in \mathbb{R}^n \tag{13.25}$$

The only difference between $(\tilde{\mathbf{P}}(\lambda))$ and $(\mathbf{P})$ is that we put the extra constraint (13.21) in $(\mathbf{P})$ into the objective function in $(\tilde{\mathbf{P}}(\lambda))$. We denote the optimal value of $(\tilde{\mathbf{P}}(\lambda))$ as $\tilde{\mathsf{OPT}}(\lambda)$. We assume that it is easy to solve $(\tilde{\mathbf{P}}(\lambda))$ for any $\lambda \geq 0$ in the sense that we can find an α-approximate ($\alpha \geq 1$) solution $(\boldsymbol{x}^*(\lambda), \boldsymbol{t}^*(\lambda))$ to $(\tilde{\mathbf{P}}(\lambda))$ in polynomial time. Namely, $(\boldsymbol{x}^*(\lambda), \boldsymbol{t}^*(\lambda))$ is a feasible solution to $(\tilde{\mathbf{P}}(\lambda))$ and

$$f(\boldsymbol{x}^*(\lambda), \boldsymbol{t}^*(\lambda)) + \lambda g(\boldsymbol{x}^*(\lambda), \boldsymbol{t}^*(\lambda)) \leq \alpha \tilde{\mathsf{OPT}}(\lambda). \tag{13.26}$$

Now we try to solve the hard problem $(\mathbf{P})$ by solving the easy problem $(\tilde{\mathbf{P}}(\lambda))$. We relax (13.21) for $(\mathbf{P})$ by introducing a dual variable $\lambda \geq 0$. The corresponding dual function becomes

$$D(\lambda) \triangleq \min_{(\boldsymbol{x},\boldsymbol{t}):(\boldsymbol{x},\boldsymbol{t})\in\mathcal{C}, \boldsymbol{x}\in\{0,1\}^m, \boldsymbol{t}\in\mathbb{R}^n} [f(\boldsymbol{x}, \boldsymbol{t}) + \lambda g(\boldsymbol{x}, \boldsymbol{t})]. \tag{13.27}$$

Clearly, for any given λ, $D(\lambda)$ can be computed by solving the easy problem $(\tilde{\mathbf{P}}(\lambda))$. Then we have the following generalized result of Theorem 13.2.

Theorem 13.4. *If we can find a λ_0 such that $\lambda_0 \cdot g(\boldsymbol{x}^*(\lambda_0), \boldsymbol{t}^*(\lambda_0)) = 0$ and $g(\boldsymbol{x}^*(\lambda_0), \boldsymbol{t}^*(\lambda_0)) \leq 0$, then $(\boldsymbol{x}^*(\lambda_0), \boldsymbol{t}^*(\lambda_0))$ is an α-approximate solution to problem $(\mathbf{P})$.*

Proof. Please see Appendix 17.13. □

Note that if we can find an optimal solution $(\boldsymbol{x}^*(\lambda), \boldsymbol{t}^*(\lambda))$ to problem $(\tilde{\mathbf{P}}(\lambda))$, i.e., $\alpha = 1$, then $(\boldsymbol{x}^*(\lambda_0), \boldsymbol{t}^*(\lambda_0))$ is optimal to problem $(\mathbf{P})$ in Theorem 13.4. This is the case for our problem PASO.

Theorem 13.4 is based on the condition that we can find a λ_0 such that $\lambda_0 \cdot g(\boldsymbol{x}^*(\lambda_0), \boldsymbol{t}^*(\lambda_0)) = 0$ and $g(\boldsymbol{x}^*(\lambda_0), \boldsymbol{t}^*(\lambda_0)) \leq 0$. But there are two major challenges by applying Theorem 13.4 to solve the hard problem $(\mathbf{P})$:

- First, it is possible that such a λ_0 does not exist;
- Second, it is highly non-trivial to find such a λ_0.

For our problem PASO, indeed, it is possible that such a λ_0 does not exist (see Fig. 15.3 with $T \in (37.62, 37.83)$ as an example). If such a λ_0 exists, since we have found a monotonic property of the extra constraint $g(\boldsymbol{x}^*(\lambda), \boldsymbol{t}^*(\lambda)) = \delta(\lambda) - T$ with respect to λ (see Theorem 13.1), we can design an efficient binary search scheme to find this λ_0. However, for general problem $(\mathbf{P})$, it is not clear whether there also exists such a monotonic property for the extra constraint $g(\boldsymbol{x}^*(\lambda), \boldsymbol{t}^*(\lambda))$ with respect to λ. We should exploit the problem structure to design an efficient algorithm to find such a λ_0.

We leave it as an interesting future direction to use/generalize our approach to solve other concrete problems in the format of $(\mathbf{P})$.

Chapter 14

Extensions

The algorithmic solutions can be extended for consideration of other constraints in long-haul heavy-duty truck operations, including: (i) truck-restricted roads and toll fees, (ii) a set of destinations instead of one single destination, (iii) hours of service restriction rules for truck drivers, and (iv) real-time traffic information.

In this section, we discuss several extensions of our solutions by considering more realistic constraints for long-haul heavy-duty trucks.

14.1 Truck-Restricted Roads and Toll Fees

In our problem PASO, we only consider fuel consumption. However, in reality, some highways have restrictions for truck's weight and size (e.g., [13]) and/or toll fees. Our solutions can be optimally extended to both cases.

For any truck-restricted road, we just check whether our heavy-duty truck meets its restrictions. If not, this road cannot be selected during the routing, and we can remove it from the graph and solve the problem using the remaining graph. For road-dependent toll fees, we should consider a monetary cost including both fuel cost and toll fees. However, the toll fee for each road is fixed and independent with speed optimization. Thus we can define the monetary cost for road $e \in \mathcal{E}$ as $m_e(t_e) = P^{\mathsf{fuel}} c_e(t_e) + P_e^{\mathsf{toll}}$ where P^{fuel} is the price per gallon of fuel and P_e^{toll} is the toll fee of road e, which is 0 if e is free. Since $m_e(t_e)$ is still polynomial and strictly convex, we can use both the FPTAS and the heuristic scheme to solve PASO where the objective is replaced by $\min \sum_{e \in \mathcal{E}} x_e \cdot m_e(t_e)$.

14.2 Balance Fuel Cost and Time Cost

In our problem PASO, we aim to minimize the fuel cost as long as the travel time is within the delay constraint T. However, in reality, the driver may prefer to finish the delivery as

soon as possible such that he or she can have more leisure time. Our solution can also be optimally extended to this case.

Similar to Sec. 14.1, we can model the edge cost as $\tilde{c}_e(t_e) = c_e(t_e) + wt_e$, where $c_e(t_e)$ is the fuel cost, t_e is the time cost, and $w \geq 0$ is a parameter to balance the fuel cost and the time cost. The constraints are still the same as that in PASO where a path should be selected and the travel time is within the hard delay requirement T. Since the new cost-time function $\tilde{c}_e(t_e)$ is still polynomial and strictly convex, all our results hold optimally. Note that letting $w = 0$ (i.e., only considering the fuel cost) reduces to our problem PASO, and letting $w \to \infty$ (i.e., only considering the time cost) leads to the result that the fastest path with maximal speed is the optimal solution.

14.3 A Set of Destinations

Another natural extension of PASO is to take into account multiple destinations. In this case, a set of destinations are given as the inputs to the problem, each with an individual hard delay. The problem is to select a path and optimize the speed so as to reach every destination within its hard delay, while minimizing the fuel consumption. First, note that this extended problem without speed optimization is the well-known traveling salesman problem (TSP) [22] and vehicle routing problems (VRP) [47], both of which are known to be NP-complete. Second, to the best of our knowledge, there is no FPTAS solutions for TSP or VRP in their basic formulations. Consequently, as a heuristic approach, one way to tackle the problem is to decompose the general problem into a set of sub-problems, each with one source-destination pair. We can solve any sub-problem using our approach, and then solve the next sub-problem by contemplating the destination of the current sub-problem as the source of next sub-problem.

14.4 Hours of Service Restriction Rules

Another extension to our problem originates from the real world regulations. In several countries there are so-called hours-of-service (HOS) [5] safety rules issued by governmental organizations (like Federal Motor Carrier Safety Administration in USA), regarding the limits on non-stop driving hours for the truck driver. As an example, in Australia the

driver must rest for 30 minutes every 5 hours and stop for 10 hours of sleep for every 14 hours of work [9]. In this scenario, the path and the speed must be chosen not only to meet the deadline and minimize the fuel consumption, but also to satisfy the HOS constraints. In the most general scenario, one may consider that a subset of nodes (cities) in the highway transportation network are the resting nodes. And during traveling from source to destination, the non-stop driving duration between two resting node must respect HOS constraint (5 hours, say). As a heuristic approach, similar to Sec. 14.3, we divide the problem into some sub-problems. More specifically, we follow three steps: (i) we first select some rest nodes to form a *rest-node-line*; (ii) we then divide the total delay into all segments in the rest-node-line such that the HOS constraint is satisfied for each segment; (iii) we finally use our solution to solve the sub-problem for each segment. Note that steps (i) and (ii) could be done reasonably according to experience or any other criteria (e.g., distance).

14.5 Real-Time Traffic

In our problem formulation, we consider a static speed limit for any road segments. It is a reasonable first-order model for highway scenarios. However, in reality, some highways near urban cities could also suffer from the traffic congestion, which significantly affects the travel delay and also the fuel consumption. Therefore, taking real-time speed limits into account is also an important extension. Since real-time traffic is generally unknown in advance, this is an online problem. One nature approach is to solve the online problem in an offline manner. Namely, we can use our approach to solve the problem based on the current speed limits and then follow the selected path and speed. After traveling some time or distance, we use our approach to solve the problem again with the updated speed limits.

Chapter 15

Performance Evaluation

In this section, we use real-world data to evaluate the performance of our algorithms. Our objectives are three-fold: (i) collect realistic dataset and model the fuel-rate-speed function, (ii) evaluate and compare the performance of our FPTAS and heuristic, (iii) compare our algorithms with baseline algorithms, including both shortest path algorithm and fastest path algorithm adapted from common practice, and (iv) investigate the energy-delay tradeoff of long-haul heavy-duty trucks by evaluating how much fuel can be saved by increasing the hard deadline.

15.1 Dataset

Transportation Network: We construct the U.S. National Highway Systems (NHS) from the dataset of Clinched Highway Mapping (CHM) Project [90]. The whole highway network graph file is specified in [2], which consists of 84504 nodes (waypoints) and 89119 (one-direction) edges.

Elevation: In this thesis, we consider the grade/slope effect when modeling the road-dependent fuel-rate-speed function. To obtain the road grades, we use the Elevation Point Query Service [18] provided by the U.S. Geological Survey (USGS) to query elevations of all nodes in the NHS graph.

Speed Limits: We use the historical average speed as the maximal speed R_e^{ub} for each road e. HERE map [15] has put speed detectors over many countries including U.S., and it provides APIs to query location-based real-time speed information. We collect the real-time speed information from HERE map [15] for two weeks and use the average as R_e^{ub} for each road e in the NHS graph[1]. For the minimal speed limit R_e^{lb}, we manually set it to be $R_e^{\mathsf{lb}} = \min\{30, R_e^{\mathsf{ub}}\}$.

[1] Due to the truck's gradeability, it may not achieve the average speed and thus later we also update R_e^{ub} based on the maximal speed that the truck can achieve at road e's grade.

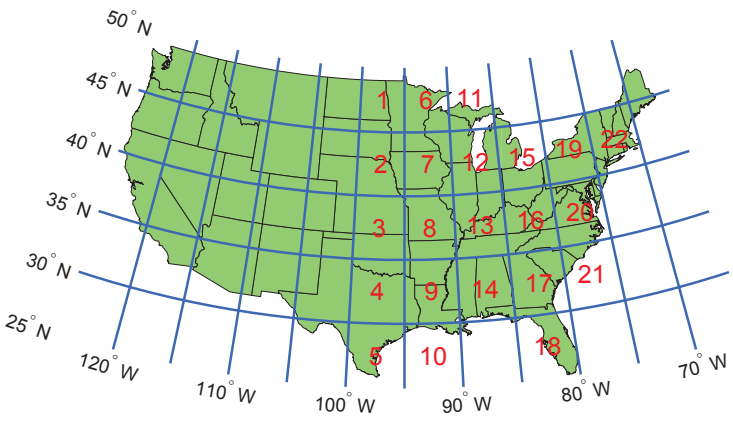


Figure 15.1: U.S. map and 22 regions.

Fuel Consumption Data: It is hard for us to get suitable real-world fuel consumption data. In this thesis, we instead leverage the widely-used ADVISOR simulator [73] to collect fuel consumption data (see Sec. 15.2).

Heavy-Duty Truck: Fuel consumption highly depends on the truck type. Another benefit of using ADVISOR is that it also provides some heavy-duty truck configurations. In this simulation, we use the Kenworth T800 Vehicle [8], a Class 8 heavy-duty truck, with 36-ton full load. It is specified in files `VEH_KENT800Trailer.m` and `HeavyTruck_in.m`[2] in ADVISOR with the following parameters in Tab. 15.1.

Table 15.1: Truck parameters (Kenworth T800).

Drag Coefficient c_d	**Frontal area** A_f	**Glider Mass**	**Cargo Mass**
0.7	8.5502 m^2	2,552kg	33,234kg

Preprocessing Highway Network: In the original NHS graph from CHM [2], we observe that: (i) most roads are in the "eastern" U.S., and (ii) many roads are very short with degree-1 endpoints (non-intersection roads). To create a network with more diverse paths, we first cut the whole NHS graph to the "eastern" part with longitude to the east of $100°W$ (see Fig. 15.1). We further merge the non-intersection roads with the same level of grades[3] into a single road. Some network statistics after these two kinds of preprocessing

[2] We replace `vinf.vehicle.name` by `VEH_KENT800Trailer`.

[3] In this simulation, we use 0.4% as the span of a grade level.

Table 15.2: Network statistics. "O" is the original NHS graph, "E" is the "eastern" graph (to the east of $100°W$), and "M" is the merged one. θ is the grade.

$\mathcal{G}$	n	m	**avg** D_e **(mile)**	**avg** R_e^{lb} **(mph)**	**avg** R_e^{ub} **(mph)**	**avg** $\lvert\theta\rvert$ **(%)**
O	84504	178238	2.08	37.4	55.97	0.64
E	65520	137521	1.97	37.3	55.55	0.58
M	38213	82781	3.26	36.43	54.19	0.82

are shown in Tab. 15.2. Note that since the average distance for each edge is 3.26 miles after preprocessing, it is reasonable to ignore the speed transition over two adjacent edges, which justifies the assumption in our fuel consumption model.

Moreover, to better visualize and evaluate the results, we divide the major "eastern" U.S. into 22 regions, as shown in Fig. 15.1. In each region $i \in [1, 22]$, we find the node in the graph which is nearest to the region's center. We also call it node i for convenience. Later on, we will use these 22 nodes as the source and destination nodes.

15.2 Model Fuel-Rate-Speed Function

We model the fuel-rate-speed function as

$$f_e(x) = a_e x^3 + b_e x^2 + c_e x + d_e, \forall e \in \mathcal{E}. \tag{15.1}$$

Here x is the speed (unit: mph) and $f_e(x)$ is the fuel rate consumption (unit: gph (gallons per hour)). Although our model (15.1) can capture any road-dependent features/factors, e.g., grade, rolling resistance, and air density, etc., we only consider the road grade θ in this simulation, which is the major factor for truck fuel consumption [19].

To learn the parameters a_e, b_e, c_e and d_e in (15.1) in terms of functions of θ, we use ADVISOR by invoking function `adv_no_gui(action,input)` where we specify `action=drive_cycle` to run a driving cycle test [1, Ch. 2.3]. As mentioned in Sec. 15.1, we choose the default vehicle file `HeavyTruck_in` where we use vehicle type `VEH_KENT800`, which specifies Kenworth T800 in Tab. 15.1. For our purpose, we generate a driving cycle file where we use a constant speed (say x (mph)) profile over a total of 4 hours and a constant grade (say θ) over the whole speed profile. Then after running ADVISOR, we can get the total fuel consumption (say w (gallons)) over a 4-hour driving time with speed x over a grade-θ

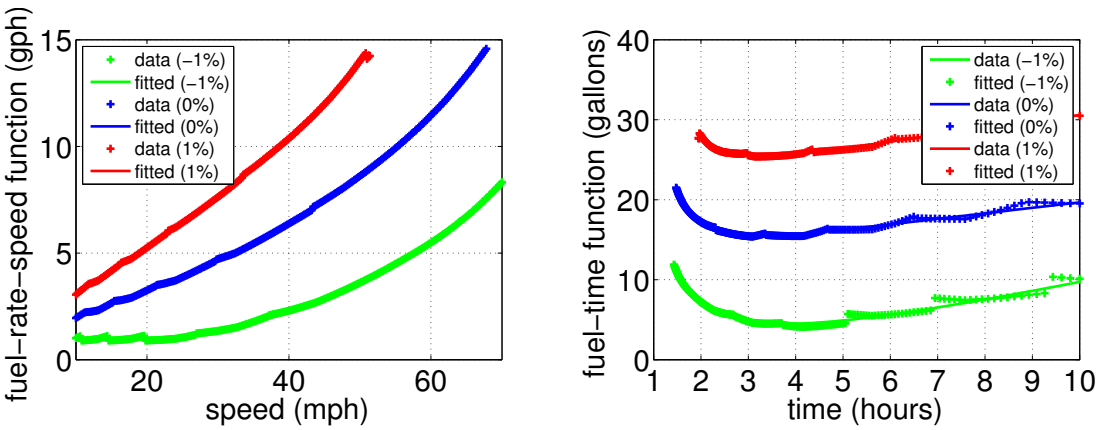
Table 15.3: Fitting parameters. For the convex region, $\leq x$ is the interval $[0, x]$ and $\geq x$ is the interval $[x, \infty)$.

Grade (%)	a_e	b_e	c_e	d_e	**Convex Region**
-2.0	5.5679e-06	-1.0839e-04	-0.0064	1.0655	$\geq$6.49
-1.0	1.0778e-05	1.2960e-03	-0.0456	1.2879	$\geq$0.00
0.0	3.3057e-05	-1.4102e-03	0.1476	0.5985	$\geq$14.22
1.0	4.9559e-05	-2.3563e-03	0.2583	0.6624	$\geq$15.85
2.0	5.9418e-05	-2.2194e-03	0.3404	0.8741	$\geq$12.45

road. Since almost all the time the truck runs with constant speed x, we can get the corresponding fuel-rate consumption as $w/4$ (gph). By enumerating x from 10 mph to 70 mph with a step of 0.2 mph, and enumerating θ from -10.0% to 10.0% with a step of 0.1%, we collect many $(x, \theta, w/4)$ data points.

For each grade θ from -10.0% to 10.0% with a step of 0.1%, we use all $(x, w/4)$ points to fit the model (15.1) by invoking MATLAB's `fit` function. We sample several grade points in Tab. 15.3, where we also put the strictly convex region for the fitted fuel-rate-speed function $f_e(x)$. As we can see, all fuel-rate-speed functions $f_e(\cdot)$ are strictly convex in reasonable speed limit regions. For example, when grade is 0 (a flat road), $f_e(\cdot)$ is strictly convex if the speed is larger than 14.22 mph, which holds generally in reality. This justifies our assumption that the fuel-rate-speed function is polynomial and strictly convex over the speed limit region.

More concretely, we visualize the fuel-rate-speed function $f_e(x)$ and fuel-time function $c_e(t_e)$ for three sampled grades, -1.0%, 0.0%, and 1.0%, as shown in Fig. 15.2. We can see that both of them are strictly convex in reasonable regions. We also verify that $c_e(t_e)$ will first strictly decrease and then strictly increasing and thus we only need to focus on the decreasing interval without loss of optimality, as discussed in Sec. 11.2. From Fig. 15.2(b), we also observe that the fuel-time curve is not smooth but has some glitches. This is due to the gear switch of the truck.



(a) Fuel-rate-speed function $f_e(x)$.

(b) Fuel-time function $c_e(t_e)$ over a 100-mile road.

Figure 15.2: Fit curve vs. data for grades 0%, ±1%.

15.3 Evaluate/Compare FPTAS and Heuristic

We implement our algorithms with C++ where we use the SNAP graph structure [69]. We evaluate on a server with an 8-core Intel Core-i7 3770 3.4 Ghz CPU and 16 GB memory, running CentOS 6.4. To evaluate and compare our FPTAS (Algorithm 3) and heuristic scheme (Algorithm 5), we consider 4 different settings, S1, S2, S3, and S4, as shown in Tab. 15.4. Note that since we aim to compare them, we use LB = 1 and UB = 1000 in *Step 1* of Algorithm 3.

In terms of the minimized fuel cost of the algorithms, Tab. 15.4 shows that the heuristic scheme always outputs the optimal solution (LB = UB, hence LB = UB = OPT), and the FPTAS also outputs a near-optimal solution (e.g., in S1, 74.812 is only a little bit larger than OPT = 74.811). This demonstrates that both FPTAS and the heuristic scheme have good performance. However, in terms of time/space complexity, the heuristic scheme is much better than FPTAS. As we can see, the FPTAS only works fine for the small-scale settings (S1 and S4), where the transportation network in regions 1 and 2 in Fig. 15.1 is considered, with only 1185 nodes and 2568 edges. When we use a little bit larger scale setting S2, it runs for nearly 1 hour and consumes 14.76 GB memory (out of 16 GB in total). Our server cannot run any other setting whose scale is larger than S2. We also note that the complexity of the FPTAS increases significantly as we decrease ϵ from 0.1 to 0.05,

Table 15.4: Comparisons of FPTAS and heuristic. Here an instance is the tuple (source, destination, delay), i.e., (s, d, T). For example, in S1, (1,2,8) means that the source (resp. destination) node is 1 (resp. 2), which is the nearest node to the center of region 1 (resp. region 2) in Fig. 15.1, and the total delay is 8 hours.

No.	Network			Input		Performance (gallon)		Time (second)		Memory (GB)	
	Reg.	n	m	Instance	ϵ	Heuri. LB/UB	FPTAS	Heuri.	FPTAS	Heuri.	FPTAS
S1	1&2	1185	2568	(1,2,8)	0.1	74.811/74.811	74.812	1	50	0.29	2.73
S2	17&18	3274	7465	(18,17,10)	0.1	60.2795/60.2795	60.2798	2	3511	0.29	14.76
S3	1-22	38213	82781	(4,22,40)	0.1	290.744/290.744	-	365	-	0.29	-
S4	1&2	1185	2568	(1,2,8)	0.05	74.811/74.811	74.812	1	126	0.29	6.84

as shown in settings S1&S4. Contrarily, our heuristic scheme can handle all 22 regions (setting S3) with 38213 nodes and 82781 edges easily with low time/space complexity.

Tab. 15.4 verifies that the FPTAS is not necessarily scalable to practical large-scale highway networks, but our heuristic scheme works very well in terms of both performance and complexity. To see why the heuristic scheme performs well, we examine an example source-destination pair in the setting S3, $(s, d) = (4, 22)$, and plot its $\delta(\lambda)$ function (the total travel time of the shortest-generalized-cost path, see (13.6)) in Fig. 15.3. We observe that function $\delta(\lambda)$ is non-increasing, which verifies Theorem 13.1. Moreover, $\delta(\lambda)$ has only a few small non-continuous jumps (e.g., a jump at point $\lambda = 11.47$ from 37.83 to 37.62). Whenever a (feasible) delay is not within such jump regions, we can always find a λ_0 such that $\delta(\lambda_0) = T$. According to Theorem 13.2, the output solution must be optimal. For example, when $T = 40$, we can find $\lambda_0 = 4.48$ such that $\delta(\lambda_0) = 40$, as shown in Fig. 15.3. The optimal solution can be derived as $(p^*(\lambda_0), \{t_e^*(\lambda_0)\})$. Even when T is within one of such jump regions (e.g., $T \in (37.62, 37.83)$), since the length of the delay region (e.g., $(37.62, 37.83)$ has a length of 0.21 hours) is often negligible as compared to a nearly 40-hour travel, the output LB and UB would be very close. Hence, our heuristic scheme outputs an optimal (at least near-optimal) solution for any input T. We will further justify this observation with more instances in Sec. 15.4.

15.4 Compare Performance with Baselines

In this section, we compare the performance of our heuristic scheme with the following 4 baseline algorithms: (i) fastest (time) path algorithm with maximal speed, (ii) fastest path

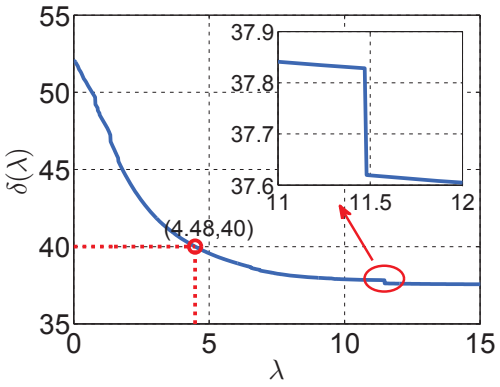


Figure 15.3: An example for $\delta(\lambda)$ when $(s, d) = (4, 22)$.

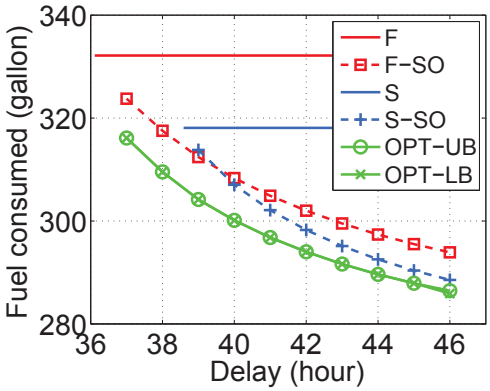


Figure 15.4: The delay effect when $(s, d) = (9, 22)$.

Table 15.5: Description of 6 solutions.

Solution	Description	Benchmark
F	Sol. of fastest path with maximal speed	Time
F-SO	Sol. of fastest path with optimal speed	-
S	Sol. of shortest path with maximal speed	Distance
S-SO	Sol. of shortest path with optimal speed	Distance
OPT-LB	Sol. of LB of our heuristic scheme	Fuel
OPT-UB	Sol. of UB of our heuristic scheme	-

algorithm with optimal speed, (iii) shortest (distance) path algorithm with maximal speed, and (iv) shortest path algorithm with optimal speed. Each of them outputs one solution for PASO. Since our heuristic scheme outputs two solutions respectively corresponding to the LB and UB, we have 6 solutions in total, as summarized in Tab. 15.5.

In later comparison, since the travel time of F is the minimal time for any feasible solution of PASO, we will use it as the *time benchmark*. For example, a solution SOL (e.g., SOL could be OPT-UB) with time increment 10% means that $\frac{\text{Travel time of SOL} - \text{Travel time of F}}{\text{Travel time of F}} = 10\%$. Similarly, we use the travel distance of S/S-SO as the *distance benchmark*, and use the fuel consumption of OPT-LB as the *fuel benchmark*.

Now we input all 22 regions in Fig. 15.1 as the underlaying highway network and use

Table 15.6: Value ranges for (s, d, T) tuples.

Parameter	**Value Range**
s	$[1, 22]$
d	$[1, 22]$
T	$\left[\lceil T^{\mathsf{f}} \rceil, \lceil T^{\mathsf{f}} \rceil + 9\right]$

Note: T^{f} is the fastest travel time from s to d.

all permutations of the 22 nodes (the nearest points to each individual region's center) as (s, d) pairs. For each (s, d) pair, we use ten different deadlines, from $\lceil T^{\mathsf{f}} \rceil$ to $\lceil T^{\mathsf{f}} \rceil + 9$ where T^{f} is the fastest travel time from s to d. In total, we have 2704 valid (s, d, T) tuples, as shown in Tab. 15.6.

A Single Instance: We first consider one instance $(s, d, T) = (9, 22, 40)$. Tab. 15.7 compares the 6 solutions[4]. As we can see, our heuristic scheme again outputs the optimal solution. It consumes 300.1 gallons of fuel, runs 10.76% slower than the time benchmark (F), and 0.3% longer than the distance benchmark (S/S-SO). Also, without speed optimization, the fastest path (F) consumes 32 more gallons (10.67%) and the shortest path (S) consumers 18 more gallons (5.99%). But with speed optimization, both fastest path and shortest path have near-optimal performance.

For $(s, d) = (9, 22)$, we also evaluate the effect of input delay T as shown in Fig. 15.4. Considering speed optimization, when the input delay $T \in [36.11, 38.58)$, the shortest path is infeasible, which shows that fastest path outperforms shortest path. The shortest path becomes feasible when $T \geq 38.58$, and it outperforms the fastest path when $T > 39$. This figure thus shows that the shortest path becomes better and better as the delay constraint increases. Intuitively, when the hard delay constraint can be satisfied, the travel distance would be critical for the total fuel consumption.

The OPT-UB curve in Fig. 15.4 is the *energy-delay tradeoff* of $(s, d) = (9, 22)$. We see that increasing delay can save fuel consumption, and the saving has a "diminishing" property. For example, the truck can save 6.6 gallons of fuel if it increases its delay from 37 to 38 hours, but the saving reduces to 1.46 gallons if its delay is relaxed from 45 to 46 hours. We give a more complete study on energy-delay tradeoff in Sec. 15.5.

[4] Readers can visualize our solutions in [10] in the HERE map.

Table 15.7: Performance of instance $(s, d, T) = (9, 22, 40)$.

Sol.	Time (hour)	Incre. (%)	Dist. (mile)	Incre. (%)	Fuel (gal.)	Incre. (%)
F	36.11	-	1821	2.71	332.1	10.67
F-SO	40	10.76	1821	2.71	308.3	2.73
S	38.58	6.85	1773	-	318.0	5.99
S-SO	40	10.76	1773	-	307.0	2.30
OPT-LB	40	10.76	1778	0.30	300.1	-
OPT-UB	40	10.76	1778	0.30	300.1	0

Table 15.8: Average performance of all instances.

Sol.	Avg Time Incre.(%)	Avg Dist. Incre.(%)	Avg Fuel Incre.(%)	Avg Fuel Econ.(mpg)
F	-	1.71	20.14	5.05
F-SO	32.80	1.71	2.00	5.94
S	2.82	-	16.40	5.13
S-SO	32.80	-	0.31	5.94
OPT-LB	32.95	0.17	-	5.96
OPT-UB	32.89	0.18	0.02	5.96

All Instances: Similar to Tab. 15.7, we can get the time, distance, and fuel of the 6 solutions for all source-sink pairs. We evaluate the average performance of all running instances in terms of time/distance/fuel increments compared to the benchmark numbers, as summarized in Tab. 15.8. Note that in 4.84% of instances, shortest path is infeasible. Tab. 15.8 only has the average performance over the instances where the shortest path is feasible.

Tab. 15.8 shows that on average OPT-UB only consumes 0.02% more fuels than the fuel benchmark (OPT-LB). This again shows that our heuristic scheme outputs a near-optimal solution in all instances.

For the baseline algorithms, Tab. 15.8 shows that the fastest path (resp. shortest path) algorithm without speed optimization consumes 20.14% (resp. 16.40%) of more fuels than

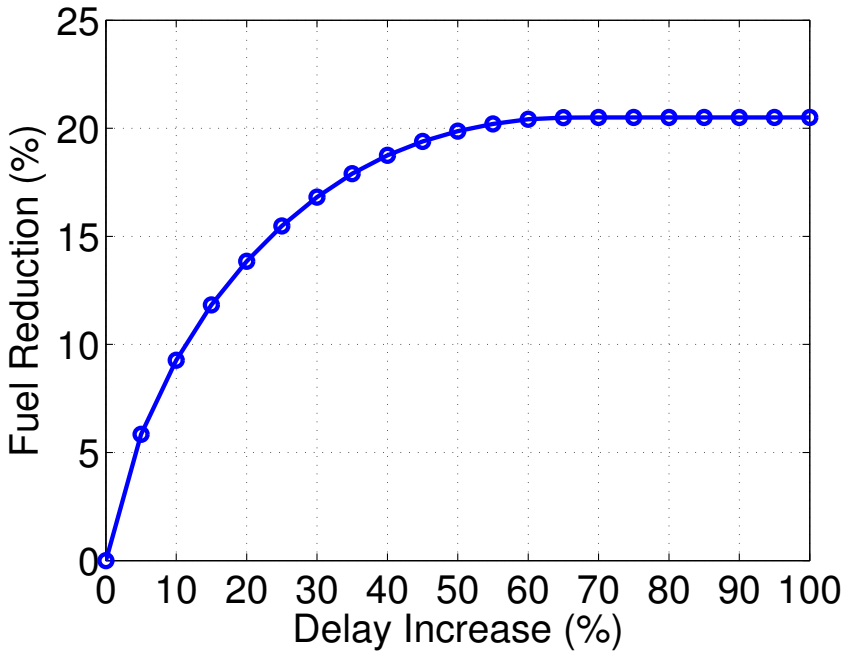


Figure 15.5: The energy-delay tradeoff.

our solution. In other words, our heuristic solution achieves 16.76% (resp. 14.09%) fuel consumption reduction, as compared to the fastest path (resp. shortest path) algorithm. Our heuristic solution also improves the *36-ton*-truck's fuel economy from 5.05 for the fastest path and 5.13 for the shortest path to 5.96. Considering its significant portion of energy consumption, our solution can indeed save much fuel cost for the long-haul heavy-duty trucks.

When we allow speed optimization for the fastest path and the shortest path, we find that on average both of them are close to the optimal solution. More specifically, F-SO consumes 2.00% of more fuels and S-SO only consumes 0.31% of more fuels than OPT-LB. This apparently suggests that in the U.S., it is good enough to first choose the shortest or fastest path and then do speed optimization. However, in our simulation, the shortest path is infeasible among 4.84% of all instances, and the fastest path with speed optimization can consume 21.32% of more fuels in the worst instance. As opposed to them, our PASO solution is robust in the sense that it always outputs a solution that is both feasible and near-optimal. We also leave it as a future work to understand under which conditions the

fastest/shortest path with speed optimization is close to the optimal solution.

15.5 Energy-Delay Tradeoff

In this subsection, we evaluate all (s, d) pairs where s and d range from 1 to 22. For each (s, d) pair, we first get the fastest travel time T^{f} and get the corresponding fuel consumption C^{f}. Now we increase the deadline by $x\%$ and evaluate the fuel consumption $C(x\%)$ when $T = (1 + x\%)T^{\mathsf{f}}$, and get the fuel consumption reduction $\frac{C^{\mathsf{f}} - C(x\%)}{C^{\mathsf{f}}}$. By changing the percentage of delay increase, i.e., x, we get different percentages of fuel consumption reduction. The average energy-delay tradeoff performance among all (s, d) pairs is shown in Fig. 15.5.

As we can see, the fuel consumption reduction has a "diminishing" property. As compared to the fastest travel time, if we increase the hard deadline by 10%, we can reduce the fuel consumption by 10%. If we increase the hard deadline by 50%, we can reduce the fuel consumption by 20%. If we further increase the hard deadline after 70%, there is little extra benefit, mainly because of the lower bound of the speeds over all edges.

Chapter 16

Conclusion and Future Work

Provisioning both energy-efficient and timely delivery is of great importance for logistic operators. This part (Chap. 10–17) presents a first step to study the energy-efficient timely transportation problem with an emphasis for long-haul heavy-duty trucks. We propose two algorithms: the first one is an FPTAS and the second one is a heuristic with lower complexity and near-optimal empirical performance. Our real-world data-driven simulations show that our solution guarantees timely delivery and can save up to 17% of fuel consumption as compared to a fastest/shortest path algorithm adapted from common practice. An interesting and important future direction is to generalize our results beyond the highway setting to cover more sophisticated local driving scenarios.

Chapter 17

Appendix

17.1 Physical Interpretation of Fuel-Rate-Speed Function

A truck running on a road with grade/slope θ (positive if moving up and negative if moving down) faces three resistances: aerodynamic (air) resistance, rolling resistance and grade resistance [72]. The air resistance is the friction of air, which is modeled as

$$F_a(v) = \frac{1}{2}\rho A_f c_d v^2, \tag{17.1}$$

where ρ is the air density and A_f the frontal area of the truck and c_d is drag coefficient of the truck (see Tab. 15.1 for c_d and A_f) and v is the speed of the truck. The rolling resistance is the friction between the tires and the ground, which is modeled as

$$F_r = c_r mg\cos\theta, \tag{17.2}$$

where c_r is the coefficient of rolling resistance (friction coefficient) between the tires and the ground, m is the truck mass and g is gravitational acceleration. The grade resistance is the force of the gravity on the opposite direction of truck movement, i.e.,

$$F_g = mg\sin\theta. \tag{17.3}$$

Then the tractive force is

$$F_t(v) = F_a(v) + F_r + F_g, \tag{17.4}$$

which yields to the power consumption

$$P_f(v) = F_t(v)\cdot v = \frac{1}{2}\rho A_f c_d v^3 + (c_r\cos\theta + \sin\theta)mgv. \tag{17.5}$$

We can regard $P_f(v)$ as the power demand to move the truck on the road with constant speed v. To provision such power demand, the internal combustion engine (ICE) needs to convert fuel into mechanical energy. There are a substantial number of models for ICE

[75]. For the purpose of this physical interpretation, we use the following relationship (see [75, Equation 10]),

$$P_f = f(v) \cdot \mathsf{LHV} \cdot \eta, \tag{17.6}$$

where $f(v)$ is the fuel rate consumption (unit: gallon per hour), LHV is the lower heating value of the fuel (unit: KJ per gallon), and η is the fuel efficiency[1]. Eq. (17.6) gives the fuel-rate-speed function $f(v)$ as follows,

$$f(v) = \frac{P_f}{\mathsf{LHV} \cdot \eta} = \frac{\frac{1}{2}\rho A_f c_d v^3 + (c_r \cos\theta + \sin\theta) mgv}{\mathsf{LHV} \cdot \eta}, \tag{17.7}$$

which shows that the fuel-rate-speed function is polynomial with speed v and also strictly convex.

Therefore, such physical interpretation justifies our assumption for the fuel-rate-speed function in Sec. 11.1.

17.2 Proof of Lemma 11.1

We can prove this lemma by using the Jensen's inequality. For any speed profile $v : [0, t_e] \to \mathbb{R}^+$ over road/edge e, the incurred fuel consumption is $\int_0^{t_e} f_e(v(t))dt$, and the travelled distance is $\int_0^{t_e} v(t)dt$. As we require that the truck must pass edge e with exactly t_e hours, we must have

$$\int_0^{t_e} v(t)dt = D_e. \tag{17.8}$$

Since $f_e(\cdot)$ is convex, according to the continuous Jensen's inequality [61, Ch. 12.411], we have

$$\frac{\int_0^{t_e} f_e(v(t))dt}{t_e} \geq f_e\left(\frac{\int_0^{t_e} v(t)dt}{t_e}\right) = f_e\left(\frac{D_e}{t_e}\right), \tag{17.9}$$

which means

$$\int_0^{t_e} f_e(v(t))dt \geq t_e f_e\left(\frac{D_e}{t_e}\right), \tag{17.10}$$

with equality when $v(t) = \frac{D_e}{t_e}$ for all $t \in [0, t_e]$.

The proof is completed.

[1]The unit of power demand P_f would be KW. We can appropriately make all units consistent.

17.3 Proof of Lemma 11.2

Since the fuel-rate-speed function $f_e(v)$ is a polynomial function (and thus twice differentiable) with respect to v, we can thus obtain the first and second-order derivative of $c_e(t_e) = t_e f_e\left(\frac{D_e}{t_e}\right)$ with respect to t_e, i.e.,

$$c'_e(t_e) = f_e\left(\frac{D_e}{t_e}\right) - \frac{D_e}{t_e} f'_e\left(\frac{D_e}{t_e}\right), \tag{17.11}$$

and

$$\begin{aligned} c''_e(t_e) &= f'_e\left(\frac{D_e}{t_e}\right)\left(-\frac{D_e}{t_e^2}\right) - \left[\left(-\frac{D_e}{t_e^2}\right) f'_e\left(\frac{D_e}{t_e}\right) + \frac{D_e}{t_e} f''_e\left(\frac{D_e}{t_e}\right)\left(-\frac{D_e}{t_e^2}\right)\right] \\ &= \frac{D_e^2}{t_e^3} f''_e\left(\frac{D_e}{t_e}\right). \end{aligned} \tag{17.12}$$

Since $f_e(\cdot)$ is strictly convex over the speed limit region, we have $f''_e(\frac{D_e}{t_e}) > 0$ and thus we conclude that

$$c''_e(t_e) > 0, \tag{17.13}$$

which proves that $c_e(\cdot)$ is strictly convex with respect to t_e over $[t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]$.

For the second part of this lemma, we first observe that $c'_e(t_e)$ is a differentiable (and thus continuous) and strictly increasing function. Thus we will consider the following three cases.

Case 1 $0 \le c'_e(t_e^{\mathsf{lb}})$**:** In this case, we know that $c_e(t_e)$ is strictly increasing over $[t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]$ and we can set $\hat{t}_e = t_e^{\mathsf{lb}}$.

Case 2 $0 \in (c'_e(t_e^{\mathsf{lb}}), c'_e(t_e^{\mathsf{ub}}))$**:** In this case, we can find a $\hat{t}_e \in (c'_e(t_e^{\mathsf{lb}}), c'_e(t_e^{\mathsf{ub}}))$ such that $c'_e(\hat{t}_e) = 0$ due to the continuity of $c'_e(t_e)$.

Case 3 $0 \ge c'_e(t_e^{\mathsf{ub}})$**:** In this case, we know that $c_e(t_e)$ is strictly decreasing over $[t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]$ and we can set $\hat{t}_e = t_e^{\mathsf{ub}}$.

In all three cases, we obtain that $c_e(t_e)$ is first strictly decreasing over $[t_e^{\mathsf{lb}}, \hat{t}_e]$ and then strictly increasing over $[\hat{t}_e, t_e^{\mathsf{ub}}]$. Note that $\hat{t}_e$ could be on the boundary of $[t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]$, as shown in **Case 1** and **Case 3**.

The proof is completed.

17.4 Proof of Lemma 12.1

First, since p and $\boldsymbol{t}_p$ is a feasible solution to PASO, we have $\mathsf{OPT} \leq c(p, \boldsymbol{t}_p)$.

Second, since Algorithm 2 returns in line 13, the path cost will be no greater than some $c \leq N$, thus we have

$$\tilde{c}(p, \boldsymbol{t}_p) \triangleq \sum_{e \in p} \tilde{c}_e(t_e) = \sum_{e \in p} \min\{\lfloor \frac{c_e(t_e)}{V} \rfloor + 1, N+1\} \leq N,$$

which clearly implies that

$$\tilde{c}_e(t_e) = \lfloor \frac{c_e(t_e)}{V} \rfloor + 1, \forall e \in p.$$

Then we have

$$\tilde{c}(p, \boldsymbol{t}_p) = \sum_{e \in p} \tilde{c}_e(t_e) = \sum_{e \in p} \left[\lfloor \frac{c_e(t_e)}{V} \rfloor + 1 \right] \geq \sum_{e \in p} \frac{c_e(t_e)}{V} = \frac{c(p, t_p)}{V},$$

which yields to

$$\begin{aligned} c(p, \boldsymbol{t}_p) \leq \tilde{c}(p, \boldsymbol{t}_p) V \leq NV &= \left(\lfloor \frac{U}{V} \rfloor + n + 1 \right) V \\ &\leq \left(\frac{U}{V} + n + 1 \right) V = U + (n+1)V = U + L\delta. \end{aligned}$$

The proof is completed.

17.5 Proof of Lemma 12.2

For PASO, let us denote $(p^*, \boldsymbol{t}_{p^*})$ as an optimal solution. Namely, p^* is an optimal path and $\boldsymbol{t}_{p^*}$ is the corresponding optimal travel time set. For each edge $e \in p^*$, we must have

$$\min\{\lfloor \frac{c_e(t_e)}{V} \rfloor + 1, N+1\} = \lfloor \frac{c_e(t_e)}{V} \rfloor + 1.$$

Suppose not. Then

$$\lfloor \frac{c_e(t_e)}{V} \rfloor + 1 > N + 1,$$

which means

$$c_e(t_e) \geq V \lfloor \frac{c_e(t_e)}{V} \rfloor > VN = V(\lfloor \frac{U}{V} \rfloor + n + 1) > U \geq \mathsf{OPT}.$$

This is a contradiction to $c_e(t_e) \leq \sum_{e\in p^*} c_e(t_e) = \mathsf{OPT}$.

Then we have

$$\begin{aligned}\tilde{c}(p^*, \boldsymbol{t}_{p^*}) &= \sum_{e\in p^*} \tilde{c}_e(t_e) = \sum_{e\in p^*} \min\{\lfloor \frac{c_e(t_e)}{V} \rfloor + 1, N+1\} \\ &= \sum_{e\in p^*} \left[\lfloor \frac{c_e(t_e)}{V} \rfloor + 1 \right] \leq \sum_{e\in p^*} \left[\frac{c_e(t_e)}{V} + 1 \right] \\ &\leq \frac{\mathsf{OPT}}{V} + n \leq \frac{U}{V} + n \leq (\lfloor \frac{U}{V} \rfloor + 1) + n = N. \end{aligned} \tag{17.14}$$

Here is a critical step which is different from Lemma 3 in [70] for RSP problem. For each edge $e \in p^*$, t_e may not be a representative point in vector $\boldsymbol{\tau}_e$. However, we can consider the representative point $\tilde{t}_e = \tau_e^i$ where $i \triangleq \tilde{c}_e(t_e)$, which incurs the same fuel cost, i.e., $\tilde{c}_e(t_e) = \tilde{c}_e(\tilde{t}_e)$. Clearly, we also have $\tilde{c}(p^*, \tilde{\boldsymbol{t}}_{p^*}) \leq N$ and $\tilde{t}_e \leq t_e$ where $\tilde{\boldsymbol{t}}_{p^*} \triangleq \{\tilde{t}_e : e \in p^*\}$.

Therefore path p^* and travel time $\tilde{\boldsymbol{t}}_{p^*}$ must be examined by Algorithm 2, which completes the proof of the first part, i.e., Algorithm 2 must return a feasible path p and travel time $\boldsymbol{t}_p$. Moreover, we have

$$\tilde{c}(p, \boldsymbol{t}_p) \leq \tilde{c}(p^*, \tilde{\boldsymbol{t}}_{p^*}) = \tilde{c}(p^*, \boldsymbol{t}_{p^*}). \tag{17.15}$$

From (17.14), we first note that

$$\tilde{c}(p^*, \boldsymbol{t}_{p^*}) \leq \frac{\mathsf{OPT}}{V} + n. \tag{17.16}$$

Second, since Algorithm 2 returns in line 13, we must have

$$\tilde{c}(p, \boldsymbol{t}_p) \triangleq \sum_{e\in p} \tilde{c}_e(t_e) = \sum_{e\in p} \min\{\lfloor \frac{c_e(t_e)}{V} \rfloor + 1, N+1\} \leq N,$$

which clearly implies that

$$\tilde{c}_e(t_e) = \lfloor \frac{c_e(t_e)}{V} \rfloor + 1, \forall e \in p.$$

We then note that

$$\begin{aligned}\tilde{c}(p, \boldsymbol{t}_p) &= \sum_{e\in p} c_e(t_e) = \sum_{e\in p} \min\{\lfloor \frac{c_e(t_e)}{V} \rfloor + 1, N+1\} \\ &= \sum_{e\in p} \left(\lfloor \frac{c_e(t_e)}{V} \rfloor + 1 \right) \geq \sum_{e\in p} \left(\frac{c_e(t_e)}{V} \right) = \frac{c(p, \boldsymbol{t}_p)}{V}. \end{aligned} \tag{17.17}$$

Inserting inequalities (17.16) and (17.17) into (17.15), we obtain

$$\frac{c(p, \boldsymbol{t}_p)}{V} \leq \frac{\mathsf{OPT}}{V} + n,$$

which means

$$c(p, \boldsymbol{t}_p) \leq \mathsf{OPT} + nV \leq \mathsf{OPT} + L\delta.$$

The proof is competed.

17.6 Proof of Theorem 12.1

The first part of this theorem directly follow the analysis of *Steps 1-3* in Sec. 12.3. Namely, Algorithm 3 returns a $(1+\epsilon)$-approximate solution for PASO in time

$$O((mn \log \xi + mn^2) \log\log \frac{\mathsf{UB}}{\mathsf{LB}} + \frac{mn \log \xi}{\epsilon} + \frac{mn^2}{\epsilon^2}). \tag{17.18}$$

Now we prove the second part of this theorem. Namely, if we use $\mathsf{LB} = \mathsf{C}^{\mathsf{lb}}$ and $\mathsf{UB} = n\mathsf{C}^{\mathsf{ub}}$ where $\mathsf{C}^{\mathsf{lb}} \triangleq \min_{e\in\mathcal{E}} c_e(t_e^{\mathsf{ub}})$ and $\mathsf{C}^{\mathsf{ub}} \triangleq \max_{e\in\mathcal{E}} c_e(t_e^{\mathsf{lb}})$, Algorithm 3 has time complexity polynomial in the input size of the problem PASO and therefore is an FPTAS. According to (17.18), we only need to show $\log\log \frac{\mathsf{UB}}{\mathsf{LB}} = \log\log \frac{n\mathsf{C}^{\mathsf{ub}}}{\mathsf{C}^{\mathsf{lb}}}$ is polynomial in the input size.

Suppose that $\mathsf{C}^{\mathsf{ub}} \triangleq \max_{e\in\mathcal{E}} c_e(t_e^{\mathsf{lb}}) = c_{e_1}(t_{e_1}^{\mathsf{lb}})$. For edge e_1, we should input all its properties, i.e., $\{D_{e_1}, R_{e_1}^{\mathsf{lb}}, R_{e_1}^{\mathsf{ub}}, f_{e_1}\}$ where f_{e_1} is a polynomial function. Suppose that

$$f_{e_1}(x) = a_1 x^{k_1} + a_2 x^{k_2} + \cdots + a_q x^{k_q}.$$

Then to input fuel-rate-speed function f_{e_1}, we only need to input $a_1, k_1, a_2, k_2, \cdots, a_q, k_q$. Therefore, for edge e_1, we should input the following real numbers,

$$\{D_{e_1}, R_{e_1}^{\mathsf{lb}}, R_{e_1}^{\mathsf{ub}}, a_1, k_1, a_2, k_2, \cdots, a_q, k_q\}.$$

The input size for edge e_1 is

$$I_{e_1} \geq \log\left(\frac{D_{e_1} + R_{e_1}^{\mathsf{lb}} + R_{e_1}^{\mathsf{ub}} + a_1 + k_1 + a_2 + k_2 + \cdots + a_q + k_q}{\mathsf{eps}}\right),$$

where $\mathsf{eps} \ll 1$ is the machine epsilon, i.e., the maximum relative error of for rounding a real number to the nearest floating point number that can be represented by a digital machine. Now let us show that $\log\log \frac{\mathsf{C}^{\mathsf{ub}}}{\mathsf{eps}}$ is polynomial in I_{e_1}.

According to the definition of the fuel-time function $c_{e_1}(\cdot)$ in (11.2), we get

$$\begin{aligned}
&\log\log\left(\frac{\mathsf{C}^{\mathsf{ub}}}{\mathsf{eps}}\right) = \log\log\left(\frac{c_{e_1}(t_{e_1}^{\mathsf{lb}})}{\mathsf{eps}}\right) \\
&= \log\log\left(\frac{t_{e_1}^{\mathsf{lb}} \cdot f_{e_1}(\frac{D_{e_1}}{t_{e_1}^{\mathsf{lb}}})}{\mathsf{eps}}\right) = \log\log\left(\frac{\frac{D_{e_1}}{R_{e_1}^{\mathsf{ub}}} \cdot f_{e_1}(R_{e_1}^{\mathsf{ub}})}{\mathsf{eps}}\right) \\
&= \log\left[\log\left(\frac{D_{e_1}}{R_{e_1}^{\mathsf{ub}}}\right) + \log\left(\frac{f_{e_1}(R_{e_1}^{\mathsf{ub}})}{\mathsf{eps}}\right)\right] \\
&= \log\left[\log\left(\frac{D_{e_1}}{\mathsf{eps}}\right) - \log\left(\frac{R_{e_1}^{\mathsf{ub}}}{\mathsf{eps}}\right) + \log\left(\frac{f_{e_1}(R_{e_1}^{\mathsf{ub}})}{\mathsf{eps}}\right)\right] \\
&\leq \log\left[I_{e_1} + \log\left(\frac{f_{e_1}(R_{e_1}^{\mathsf{ub}})}{\mathsf{eps}}\right)\right] \\
&\qquad \text{(Since } R_{e_1}^{\mathsf{ub}} > 0 \text{ and thus } R_{e_1}^{\mathsf{ub}} \geq \mathsf{eps}\text{)} \\
&= \log\left[I_{e_1} + \log\left(\frac{a_1(R_{e_1}^{\mathsf{ub}})^{k_1} + \cdots + a_q(R_{e_1}^{\mathsf{ub}})^{k_q}}{\mathsf{eps}}\right)\right] \\
&\leq \log\left[I_{e_1} + \log\left(\frac{q a_i (R_{e_1}^{\mathsf{ub}})^{k_i}}{\mathsf{eps}}\right)\right] \\
&\qquad \left(\text{Define } i \in \arg\max_{j\in[1,q]} a_j (R_{e_1}^{\mathsf{ub}})^{k_j}\right) \\
&= \log\left[I_{e_1} + \log q + \log\left(\frac{a_i}{\mathsf{eps}}\right) + \log\left(\frac{(R_{e_1}^{\mathsf{ub}})^{k_i}}{\mathsf{eps}^{k_i}} \cdot \mathsf{eps}^{k_i}\right)\right] \\
&\leq \log\left[I_{e_1} + I_{e_1} + I_{e_1} + k_i \log\left(\frac{R_{e_1}^{\mathsf{ub}}}{\mathsf{eps}}\right)\right] \\
&\qquad \text{(Since } \log \mathsf{eps} < 0\text{)} \\
&\leq \log\left[I_{e_1} + I_{e_1} + I_{e_1} + \frac{k_i}{\mathsf{eps}} \log\left(\frac{R_{e_1}^{\mathsf{ub}}}{\mathsf{eps}}\right)\right] \\
&\qquad \text{(Since } \mathsf{eps} < 1\text{)} \\
&\leq \log\left[I_{e_1} + I_{e_1} + I_{e_1} + \frac{k_i}{\mathsf{eps}} \cdot I_{e_1}\right] \\
&\leq \log\left[I_{e_1} + I_{e_1} + I_{e_1} + 2^{I_{e_1}} \cdot I_{e_1}\right] \\
&= \log I_{e_1} + \log(3 + 2^{I_{e_1}}) \\
&\leq \log I_{e_1} + \log(3 \cdot 2^{I_{e_1}} + 2^{I_{e_1}}) \\
&= \log I_{e_1} + I_{e_1} + 2 = O(I_{e_1}),
\end{aligned}$$

which is thus polynomial in I_{e_1}.

Then

$$\log\log\frac{n\mathsf{C}^{\mathsf{ub}}}{\mathsf{C}^{\mathsf{lb}}} = \log\log\frac{\frac{n\mathsf{C}^{\mathsf{ub}}}{\mathsf{eps}}}{\frac{\mathsf{C}^{\mathsf{lb}}}{\mathsf{eps}}} \le \log\log\frac{n\mathsf{C}^{\mathsf{ub}}}{\mathsf{eps}} = \log\left(\log n + \log\frac{\mathsf{C}^{\mathsf{ub}}}{\mathsf{eps}}\right)$$
$$\le 2\max\left\{\log\log n, \log\log\frac{\mathsf{C}^{\mathsf{ub}}}{\mathsf{eps}}\right\} = \max\{O(\log\log n), O(I_{e_1})\}, \qquad (17.19)$$

which is polynomial in the input size of PASO because both $O(\log\log n)$ and $O(I_{e_1})$ are polynomial in the input size of PASO. We thus prove the second part of this theorem.

The proof is completed.

17.7 Proof of Theorem 12.2

The time complexity has been shown in the three-step analysis above Theorem 12.2. Next we show the approximate ratio.

After quantizing in *Step 1 (lines 1-6)*, each edge $e \in \mathcal{E}$ is associated with a quantized fuel-time function $\tilde{c}_e(t_e)$. According to (12.1), the quantized fuel-time function is

$$\tilde{c}_e(t_e) = \min\left\{\left\lfloor\frac{c_e(t_e)}{V}\right\rfloor + 1, N\right\}. \qquad (17.20)$$

Since $\mathsf{C}^{\mathsf{ub}} \triangleq \max_{e\in\mathcal{E}} c_e(t_e^{\mathsf{lb}})$, we have

$$\left\lfloor\frac{c_e(t_e)}{V}\right\rfloor + 1 \le \left\lfloor\frac{\mathsf{C}^{\mathsf{ub}}}{V}\right\rfloor + 1 = N, \quad \forall t_e \in [t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]. \qquad (17.21)$$

Thus, the quantized fuel-time function is

$$\tilde{c}_e(t_e) = \min\left\{\left\lfloor\frac{c_e(t_e)}{V}\right\rfloor + 1, N\right\} = \left\lfloor\frac{c_e(t_e)}{V}\right\rfloor + 1. \qquad (17.22)$$

Considering problem PASO for graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with original fuel-time function $c_e(t_e)$, we denote $(p^*, \boldsymbol{t}_{p^*})$ as an optimal solution. Namely, p^* is an optimal path and $\boldsymbol{t}_{p^*} = \{t_e : e \in p^*\}$ is the corresponding optimal travel time set. Clearly we have $c(p^*, \boldsymbol{t}_{p^*}) \triangleq \sum_{e\in p^*} c_e(t_e) = \mathsf{OPT}$.

Considering problem PASO for graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with quantized fuel-time function $\tilde{c}_e(t_e)$, we define $\tilde{\mathsf{OPT}}$ as the optimal value.

Since $(p^*, \boldsymbol{t}_{p^*})$ is also a feasible solution to problem PASO for graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with quantized fuel-time function $\tilde{c}_e(t_e)$, we have

$$\tilde{\mathsf{OPT}} \leq \sum_{e \in p^*} \tilde{c}_e(t_e) = \sum_{e \in p^*} \left\{ \left\lfloor \frac{c_e(t_e)}{V} \right\rfloor + 1 \right\} \leq \sum_{e \in p^*} \left\{ \frac{c_e(t_e)}{V} + 1 \right\} \leq \frac{\mathsf{OPT}}{V} + n. \tag{17.23}$$

Considering problem PASO for graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with quantized fuel-time function $\tilde{c}_e(t_e)$, since $\tilde{c}_e(t_e)$ is a staircase function as discussed in Sec. 12.1, we can always find an optimal solution where the travel time of any edge e over the optimal path belongs to one of the representative points. Therefore, solving problem PASO for graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with quantized fuel-time function $\tilde{c}_e(t_e)$ is equivalent to solving problem PASO for the new constructed graph $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{E}})$ in *Step 2 (lines 7-14)*. Note that in the new constructed graph $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{E}})$, every edge has a fixed travel time and a fixed travel cost, thus problem PASO is exactly problem RSP. Therefore the optimal value of problem RSP for graph $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{E}})$ is also $\tilde{\mathsf{OPT}}$.

We denote $(\tilde{p}, \tilde{\boldsymbol{t}}_{p_1})$ as the output $(1+\epsilon_2)$-approximate solution to problem RSP for graph $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{E}})$ in *Step 3 (lines 15-16)*. Namely $\tilde{p}$ is the output path and $\tilde{\boldsymbol{t}}_{\tilde{p}} = \{\tilde{t}_e : e \in \tilde{p}\}$ is the corresponding travel time set. Since $(\tilde{p}, \tilde{\boldsymbol{t}}_{p_1})$ is $(1+\epsilon_2)$-approximate, we have

$$\sum_{e \in \tilde{p}} \tilde{c}_e(\tilde{t}_e) \leq (1+\epsilon_2)\tilde{\mathsf{OPT}}. \tag{17.24}$$

In addition, we have

$$\sum_{e \in \tilde{p}} \tilde{c}_e(\tilde{t}_e) = \sum_{e \in \tilde{p}} \left\{ \left\lfloor \frac{c_e(\tilde{t}_e)}{V} \right\rfloor + 1 \right\} \geq \sum_{e \in \tilde{p}} \frac{c_e(\tilde{t}_e)}{V} = \frac{\sum_{e \in \tilde{p}} c_e(\tilde{t}_e)}{V}. \tag{17.25}$$

Combining (17.23), (17.24), and (17.25), we obtain

$$\begin{aligned} \sum_{e \in \tilde{p}} c_e(\tilde{t}_e) &\leq (1+\epsilon_2)(\mathsf{OPT} + nV) = (1+\epsilon_2)(\mathsf{OPT} + n \cdot \frac{\mathsf{LB} \cdot \epsilon_1}{n}) \\ &= (1+\epsilon_2)(\mathsf{OPT} + \mathsf{LB} \cdot \epsilon_1) \leq (1+\epsilon_2)(\mathsf{OPT} + \mathsf{OPT} \cdot \epsilon_1) \\ &= (1+\epsilon_1)(1+\epsilon_2)\mathsf{OPT} = (1+\frac{\epsilon}{2})(1+\frac{\epsilon}{2+\epsilon})\mathsf{OPT} \\ &= (1+\epsilon)\mathsf{OPT}. \end{aligned} \tag{17.26}$$

Therefore, $(\tilde{p}, \tilde{\boldsymbol{t}}_{p_1})$ is a $(1+\epsilon)$-approximate solution to our original problem PASO for graph $\mathcal{G}$ with original fuel-time function $c_e(t_e)$.

17.8 Proof of Lemma 13.1

Define function $h(t_e) = c_e(t_e) + \lambda t_e$. Then we can get the first derivative as

$$h'(t_e) = c'_e(t_e) + \lambda. \tag{17.27}$$

Since $c_e(t_e)$ is a strictly convex and strict decreasing function, we know that $c'_e(t_e)$ (and also $h'(t_e)$) is a strictly increasing function and $c'_e(t_e) < 0$ at interval $[t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]$. We then consider the following three cases.

Case 1: If $0 \leq \lambda < -c'_e(t_e^{\mathsf{ub}})$, we get that $c'_e(t_e^{\mathsf{ub}}) + \lambda < 0$ and thus

$$h'(t_e) \leq h'(t_e^{\mathsf{ub}}) < 0, \forall, t \in [t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]. \tag{17.28}$$

This shows that $h(t_e)$ is strictly decreasing at $[t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]$ and the minimal value is attained at $t_e^*(\lambda) = t_e^{\mathsf{ub}}$.

Case 2: If $-c'_e(t_e^{\mathsf{ub}}) \leq \lambda \leq -c'_e(t_e^{\mathsf{lb}})$, then we can get that $c_e'^{-1}(-\lambda) \in [t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]$. Clearly, the monotonic increasing property of $h'(t_e)$ implies that $h'(t_e) < 0$ at $[t_e^{\mathsf{lb}}, c_e'^{-1}(-\lambda))$ and $h'(t_e) > 0$ at $(c_e'^{-1}(-\lambda), t_e^{\mathsf{lb}}]$. This means that the minimal value is attained at $t_e^*(\lambda) = c_e'^{-1}(-\lambda)$.

Case 3: If $\lambda > -c'_e(t_e^{\mathsf{lb}})$, we get that $c'_e(t_e^{\mathsf{lb}}) + \lambda > 0$ and thus

$$h'(t_e) \geq h'(t_e^{\mathsf{lb}}) > 0, \forall, t \in [t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]. \tag{17.29}$$

This shows that $h(t_e)$ is strictly increasing at $[t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]$ and the minimal value is attained at $t_e^*(\lambda) = t_e^{\mathsf{lb}}$.

The proof is completed.

17.9 Proof of Theorem 13.1

Let us consider any two λ_1, λ_2 with $0 \leq \lambda_1 < \lambda_2$. We need to prove $\delta(\lambda_1) \geq \delta(\lambda_2)$. Suppose that the optimal path at λ_1 is $p^*(\lambda_1) = p_1$ and the optimal path at λ_2 is $p^*(\lambda_2) = p_2$[2].

For any path p and any $\lambda \geq 0$, we denote its (optimal) generalized path cost as

$$W_p(\lambda) \triangleq \sum_{e \in p} w_e(\lambda) = \sum_{e \in p} \left[c_e(t_e^*(\lambda)) + \lambda t_e^*(\lambda) \right], \tag{17.30}$$

[2]Paths p_1 and p_2 could be the same.

and denote its corresponding path fuel cost as

$$C_p(\lambda) \triangleq \sum_{e \in p} c_e(t_e^*(\lambda)). \tag{17.31}$$

and denote its corresponding path delay

$$T_p(\lambda) \triangleq \sum_{e \in p} t_e^*(\lambda). \tag{17.32}$$

Clearly, we have $W_p(\lambda) = C_p(\lambda) + \lambda T_p(\lambda)$.

Based on such notations, we have $\delta(\lambda_1) = T_{p_1}(\lambda_1)$ and $\delta(\lambda_2) = T_{p_2}(\lambda_2)$, and we need to prove $T_{p_1}(\lambda_1) \geq T_{p_2}(\lambda_2)$.

When $\lambda = \lambda_1$, the optimal path is p_1, which means that

$$W_{p_1}(\lambda_1) = C_{p_1}(\lambda_1) + \lambda_1 T_{p_1}(\lambda_1) \leq W_{p_2}(\lambda_1) = C_{p_2}(\lambda_1) + \lambda_1 T_{p_2}(\lambda_1) \tag{17.33}$$

Similarly, when $\lambda = \lambda_2$, the optimal path is p_2, which means that

$$W_{p_2}(\lambda_2) = C_{p_2}(\lambda_2) + \lambda_2 T_{p_2}(\lambda_2) \leq W_{p_1}(\lambda_2) = C_{p_1}(\lambda_2) + \lambda_2 T_{p_1}(\lambda_2) \tag{17.34}$$

Now we will use the fact that $t_e^*(\lambda)$ minimizes $w_e(\lambda)$, as defined in (13.3). Since both $t_e^*(\lambda_1)$ and $t_e^*(\lambda_2)$ are feasible, i.e., in the interval $[t_e^{\mathsf{lb}}, t_e^{\mathsf{ub}}]$, we get that

$$\begin{aligned} W_{p_2}(\lambda_1) = C_{p_2}(\lambda_1) + \lambda_1 T_{p_2}(\lambda_1) &= \sum_{e \in p_2} \left(c_e(t_e^*(\lambda_1)) + \lambda_1 t_e^*(\lambda_1)\right) \\ &= \sum_{e \in p_2} \min_{t_e^{\mathsf{lb}} \leq t_e \leq t_e^{\mathsf{ub}}} \left(c_e(t_e) + \lambda_1 t_e\right) \leq \sum_{e \in p_2} \left(c_e(t_e^*(\lambda_2)) + \lambda_1 t_e^*(\lambda_2)\right) \\ &= C_{p_2}(\lambda_2) + \lambda_1 T_{p_2}(\lambda_2). \end{aligned} \tag{17.35}$$

Similarly, we have

$$W_{p_1}(\lambda_2) = C_{p_1}(\lambda_2) + \lambda_2 T_{p_1}(\lambda_2) \leq C_{p_1}(\lambda_1) + \lambda_2 T_{p_1}(\lambda_1). \tag{17.36}$$

Inserting (17.35) into (17.33), we get that

$$C_{p_1}(\lambda_1) + \lambda_1 T_{p_1}(\lambda_1) \leq C_{p_2}(\lambda_2) + \lambda_1 T_{p_2}(\lambda_2),$$

which implies that

$$\lambda_1 \left[T_{p_1}(\lambda_1) - T_{p_2}(\lambda_2)\right] \leq C_{p_2}(\lambda_2) - C_{p_1}(\lambda_1). \tag{17.37}$$

Similarly, inserting (17.36) into (17.34), we get that

$$C_{p_2}(\lambda_2) + \lambda_2 T_{p_2}(\lambda_2) \leq C_{p_1}(\lambda_1) + \lambda_2 T_{p_1}(\lambda_1),$$

which implies that

$$-\lambda_2 \left[T_{p_1}(\lambda_1) - T_{p_2}(\lambda_2)\right] \leq C_{p_1}(\lambda_1) - C_{p_2}(\lambda_2). \tag{17.38}$$

Summing (17.37) and (17.38), we get that

$$(\lambda_1 - \lambda_2)\left[T_{p_1}(\lambda_1) - T_{p_2}(\lambda_2)\right] \leq 0. \tag{17.39}$$

Since we assume that $\lambda_1 < \lambda_2$, we must have

$$T_{p_1}(\lambda_1) \geq T_{p_2}(\lambda_2). \tag{17.40}$$

The proof is completed.

17.10 Proof of Theorem 13.2

At the point λ_0, the dual function has value

$$\begin{aligned}
D(\lambda_0) &= -\lambda_0 T + \min_{x \in \mathcal{X}} \sum_{e \in \mathcal{E}} x_e \cdot \min_{t_e^{\mathsf{lb}} \leq t_e \leq t_e^{\mathsf{ub}}} (c_e(t_e) + \lambda_0 t_e) \\
&= -\lambda_0 T + \min_{x \in \mathcal{X}} \sum_{e \in \mathcal{E}} x_e \cdot (c_e(t_e^*(\lambda_0)) + \lambda_0 t_e^*(\lambda_0)) \\
&= -\lambda_0 T + \sum_{e \in p^*(\lambda_0)} [c_e(t_e^*(\lambda_0)) + \lambda_0 t_e^*(\lambda_0)] \\
&= -\lambda_0 T + \sum_{e \in p^*(\lambda_0)} c_e(t_e^*(\lambda_0)) + \lambda_0 \sum_{e \in p^*(\lambda_0)} t_e^*(\lambda_0) \\
&= -\lambda_0 T + \lambda_0 \delta(\lambda_0) + \sum_{e \in p^*(\lambda_0)} c_e(t_e^*(\lambda_0)) \\
&= -\lambda_0 T + \lambda_0 T + \sum_{e \in p^*(\lambda_0)} c_e(t_e^*(\lambda_0)) \\
&= \sum_{e \in p^*(\lambda_0)} c_e(t_e^*(\lambda_0)).
\end{aligned} \tag{17.41}$$

On one hand, we know that any dual function value will be a lower bound of OPT according to the weak duality. Thus,

$$D(\lambda_0) \leq \mathsf{OPT}. \tag{17.42}$$

On the other hand, we know that $p^*(\lambda_0)$ is a feasible path and $\{t_e^*(\lambda_0), e \in p^*(\lambda_0)\}$ satisfies

$$\sum_{e \in p^*(\lambda_0)} t_e^*(\lambda_0) = T. \tag{17.43}$$

Here $p^*(\lambda_0)$ and $\{t_e^*(\lambda_0), e \in p^*(\lambda_0)\}$ is a feasible solution to PASO with the objective value $\sum_{e \in p^*(\lambda_0)} c_e(t_e^*(\lambda_0)) = D(\lambda_0)$, which is an upper bound of OPT, i.e.,

$$D(\lambda_0) \geq \mathsf{OPT}. \tag{17.44}$$

Eq. (17.42) and (17.44) conclude that $D(\lambda_0) = \mathsf{OPT}$, and $p^*(\lambda_0)$ and $\{t_e^*(\lambda_0), e \in p^*(\lambda_0)\}$ is an optimal solution to PASO.

The proof is completed.

17.11 Proof of Theorem 13.3

First, if we let total travel delay be $T' = \sum_{e \in p^*(\lambda_L)} t_e^*(\lambda_L) > T$, we get a relaxed version of PASO. According to Theorem 13.2, we know that $\mathsf{LB} = \sum_{e \in p^*(\lambda_L)} c_e(t_e^*(\lambda_L))$ is the optimal solution of the relaxed version, and thus we have $\mathsf{LB} \leq \mathsf{OPT}$.

Second, since $\sum_{e \in p^*(\lambda_U)} t_e^*(\lambda_U) < T$, we know that $p^*(\lambda_U)$ and $\{t_e^*(\lambda_U) : e \in p^*(\lambda_U)\}$ is a feasible solution to PASO. Thus, $\mathsf{UB} = \sum_{e \in p^*(\lambda_L)} c_e(t_e^*(\lambda_U)) \geq \mathsf{OPT}$.

The proof is completed.

17.12 Proof of Lemma 13.2

Based on Lemma 13.1 and definition of λ_1 in (13.10), for any edge $e \in \mathcal{E}$, when $\lambda \geq \bar{\lambda} \geq \lambda_1 \geq -c_e'(t_e^{\mathsf{lb}})$, we have

$$t_e^*(\lambda) = t_e^{\mathsf{lb}}, \tag{17.45}$$

which means that the truck will run at the highest speed over edge e and the resulting fuel cost is $c_e^{\mathsf{ub}} = c_e(t_e^{\mathsf{lb}})$. Then when $\lambda \geq \bar{\lambda}$, any edge e has a generalized cost $w_e(\lambda) = c_e(t_e^*(\lambda)) + \lambda t_e^*(\lambda) = c_e^{\mathsf{ub}} + \lambda t_e^{\mathsf{lb}}$ and any $s-d$ path $p \in \mathcal{P}$ has a generalized cost $\sum_{e \in p} w_e(\lambda) = C_p + \lambda T_p$.

To obtain $\delta(\lambda) = \sum_{e \in p^*(\lambda)} t_e^*(\lambda)$ (see (13.6)), we now need to get a shortest-generalized-cost path $p^*(\lambda)$, i.e,

$$p^*(\lambda) \in \arg\min_{p \in \mathcal{P}} [C_p + \lambda T_p]. \tag{17.46}$$

Next we show that p^* defined in (13.9) is such a shortest-generalized-cost path. Since p^* has the minimal travel cost among all fastest path in $\mathcal{P}_F$, we have

$$C_{p^*} + \lambda T_{p^*} = C_{p^*} + \lambda T^{\mathsf{lb}} \leq C_p + \lambda T^{\mathsf{lb}} = C_p + \lambda T_p, \quad \forall p \in \mathcal{P}_F. \tag{17.47}$$

In addition, based on the definition of λ_2 in (13.11), when $\lambda \geq \bar{\lambda}$, we have

$$\lambda \geq \bar{\lambda} \geq \lambda_2 \geq \frac{C_{p^*} - C_p}{T_p - T_{p^*}}, \quad \forall p \in \mathcal{P} \backslash \mathcal{P}_F, \tag{17.48}$$

implying that

$$C_{p^*} + \lambda T_{p^*} \leq C_p + \lambda T_p, \quad \forall p \in \mathcal{P} \backslash \mathcal{P}_F. \tag{17.49}$$

From (17.47) and (17.49), when $\lambda \geq \bar{\lambda}$, we obtain

$$C_{p^*} + \lambda T_{p^*} \leq C_p + \lambda T_p \leq C_p + \lambda T_p, \quad \forall p \in \mathcal{P}, \tag{17.50}$$

Therefore p^* is a shortest-generalized-cost path. We then let $p^*(\lambda) = p^*$ when $\lambda \geq \bar{\lambda}$.[3] Therefore,

$$\delta(\lambda) = \sum_{e \in p^*(\lambda)} t_e^*(\lambda) = \sum_{e \in p^*} t_e^{\mathsf{lb}} = T_{p^*} = T^{\mathsf{lb}}, \quad \forall \lambda \geq \bar{\lambda}. \tag{17.51}$$

The proof is completed.

17.13 Proof of Theorem 13.4

First of all, according to the week duality, we have

$$\mathsf{O\tilde{P}T}(\lambda) \leq \mathsf{OPT}, \quad \forall \lambda \geq 0, \tag{17.52}$$

Then for λ_0, since $\lambda_0 \cdot g(\boldsymbol{x}^*(\lambda_0), \boldsymbol{t}^*(\lambda_0)) = 0$, we have

$$\begin{aligned} f(\boldsymbol{x}^*(\lambda_0), \boldsymbol{t}^*(\lambda_0)) &= f(\boldsymbol{x}^*(\lambda_0), \boldsymbol{t}^*(\lambda_0)) + \lambda_0 g(\boldsymbol{x}^*(\lambda_0), \boldsymbol{t}^*(\lambda_0)) \\ &\leq \alpha \mathsf{O\tilde{P}T}(\lambda_0) \quad \text{(see (13.26))} \\ &\leq \alpha \mathsf{OPT}. \quad \text{(see (17.52))} \end{aligned} \tag{17.53}$$

[3] It is possible that there exists multiple shortest-generalized-cost paths, but we can easily show that all of them are fastest paths, i.e., in $\mathcal{P}_F$. Therefore, any $p^*(\lambda)$ has the same path travel time T^{lb} and thus $\delta(\lambda) = T^{\mathsf{lb}}$. Therefore, we can safely set $p^*(\lambda) = p^*$.

In addition, since $(\boldsymbol{x}^*(\lambda_0), \boldsymbol{t}^*(\lambda_0))$ is a feasible solution to problem $(\tilde{\mathbf{P}}(\lambda_0))$ and $g(\boldsymbol{x}^*(\lambda_0), \boldsymbol{t}^*(\lambda_0)) \leq 0$, $(\boldsymbol{x}^*(\lambda_0), \boldsymbol{t}^*(\lambda_0))$ is also a feasible solution to problem $(\mathbf{P})$. Therefore, $(\boldsymbol{x}^*(\lambda_0), \boldsymbol{t}^*(\lambda_0))$ is an α-approximate solution to problem $(\mathbf{P})$.

The proof is completed.

Part IV

Summary

Motivated by the importance of provisioning delay-constrained services in the communication and transportation systems, in this thesis, we study two delay-constrained problems: one is the timely wireless flow problem and the other is the energy-efficient timely transportation problem.

For the first timely wireless flow problem, we develop an MDP-based framework to characterize the timely capacity region and design scheduling policies to maximize the network utility for general traffic patterns. Our characterization for the timely capacity region generalizes the LP formulation of stationary MDPs to cyclostationary MDPs. We believe that our MDP-based framework can solve other problems in the timely wireless flow research area, because it is a systematic way to explore the full design space. The major drawback of our MDP-based framework is the curse of dimensionality rooted in the MDP itself, which prevents it from being applied in large-scale wireless networks. Although we take a first step to address the curse of dimensionality in this thesis, it still has a long way to go to obtain performance-guaranteed efficient solutions.

For the second energy-efficient timely transportation problem, we design an FPTAS to solve this NP-complete problem, which actually generalizes the FPTAS design for the classical Restricted-Shortest-Path (RSP) problem [50] with an extra design space of speed planning. Moreover, by leveraging elegant insights from studying the dual problem, we design a heuristic algorithm with much lower complexity. Though such a heuristic generally does not provide performance guarantee, interestingly, we characterize a condition under which our heuristic generates an optimal solution. Such a condition holds on most of practical instances in our simulations based on real-world dataset. We believe that our dual-based heuristic design is not limited to our energy-efficient timely transportation problem, but can be applied to more mixed discrete-continuous problems, as we discussed in Sec. 13.4.3. It is thus an interesting future direction to generalize our approach to solve other concrete problems with a similar structure, or even develop a general dual-based framework to solve a class of mixed discrete-continuous problems.

In summary, we observe that both the problem structure and the problem-solving methodology of delay-constrained problems in the communication and transportation systems are completely different from those of the well-understood delay-unconstrained ones. Many other questions that were asked for delay-unconstrained systems can also be asked

for delay-constrained systems, such as how to design distributed solutions in a communication network and how to do traffic assignment in a transportation network. Therefore, there are a lot of opportunities in the delay-constrained research area and we call for investigation participation.

Bibliography

[1] ADVISOR documentation. http://adv-vehicle-sim.sourceforge.net/advisor_doc.html.

[2] CHM U.S. national highway systems. http://courses.teresco.org/chm/graphs/usa-national.gra.

[3] Energy consumption estimates by end-use sector, ranked by state, 2013. http://www.eia.gov/state/seds/data.cfm?incfile=/state/seds/sep_sum/html/rank_use.html&sid=US.

[4] Fuel economy at various driving speeds. http://www.afdc.energy.gov/data/10312.

[5] Hours of Service. https://en.wikipedia.org/wiki/Hours_of_service.

[6] Improving the fuel efficiency of American trucks, 2014. https://www.whitehouse.gov/the-press-office/2014/02/18/fact-sheet-opportunity-all-improving-fuel-efficiency-american-trucks-bol.

[7] Keeping your vehicle in shape. https://www.fueleconomy.gov/feg/maintain.jsp.

[8] Kenworth T800 vehicle. http://www.kenworth.com/trucks/t800.

[9] National heavy vehicle regulator fatigue management. https://www.nhvr.gov.au/safety-accreditation-compliance/fatigue-management/about-fatigue-management.

[10] Paths for one instance. https://www.dropbox.com/s/a870bc0zboaswzz/show_path_9_22_40.html?dl=0.

[11] Place an order with guaranteed delivery. https://www.amazon.com/gp/help/customer/display.html/ref=hp_left_v4_sib?ie=UTF8&nodeId=201117390.

[12] Smarter trucking saves fuel over the long haul, 2011. http://news.nationalgeographic.com/news/energy/2011/09/110923-fuel-economy-for-trucks/.

[13] Special Route Restrictions, California Department of Transportation, `http://www.dot.ca.gov/hq/traffops/trucks/routes/restrict-list.htm`.

[14] Supertruck team achieves 115% freight efficiency improvement in Class 8 long-haul truck, 2015. `http://energy.gov/eere/vehicles/articles/supertruck-team-achieves-115-freight-efficiency-improvement-class-8-long-haul`.

[15] Traffic flow using corridor in HERE maps. `https://developer.here.com/api-explorer/rest/traffic/flow-using-corridor`.

[16] Transportation logistics enhances your business efficiency, 2014. `http://www.readytrucking.com/transportation-logistics-business-efficiency/`.

[17] Transportation overview. `http://www.c2es.org/energy/use/transportation`.

[18] USGS elevation point query service. `http://nationalmap.gov/epqs/`.

[19] K. Ahn. Microscopic fuel consumption and emission modeling. Master's thesis, Virginia Polytechnic Institute and State University, 1998.

[20] A. A. Alam, A. Gattami, and K. H. Johansson. An experimental study on the fuel reduction potential of heavy duty vehicle platooning. In *Prof. IEEE ITSC*, 2010.

[21] F. An and M. Ross. Model of fuel economy with applications to driving cycles and traffic management. *Transportation Research Record*, 1993.

[22] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The traveling salesman problem: a computational study.* Princeton university press, 2011.

[23] S. Ardekani, E. Hauer, and B. Jamei. Traffic impact models. *Chapter 7 in Traffic Flow Theory, Oak Bridge National Laboratory Report*, 1992.

[24] B. H. Ashby. *Protecting Perishable Foods during Transport by Truck.* U.S. Department of Agriculture, 2006.

[25] F. Bai, T. Elbatt, G. Hollan, H. Krishnan, and V. Sadekar. Towards characterizing and classifying communication-based automotive applications from a wireless networking perspective. In *Proc. IEEE AutoNet*, 2006.

[26] S. K. Baruah, L. E. Rosier, and R. R. Howell. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2(4):301–324, 1990.

[27] I. M. Berry. The effects of driving style and vehicle performance on the real-world fuel consumption of U.S. light-duty vehicles. Master's thesis, Massachusetts Institute of Technology, 2010.

[28] D. P. Bertsekas. *Nonlinear Programming.* Athena scientific, 1999.

[29] S. Bodas, S. Shakkottai, L. Ying, and R. Srikant. Scheduling for small delay in multi-rate multi-channel wireless networks. In *Proc. IEEE INFOCOM*, 2011.

[30] T. Breugem, T. Dollevoet, and W. van den Heuvel. Analysis of FPTASes for the multi-objective shortest path problem. *Computers & Operations Research*, 78:44–58, 2017.

[31] K. Chatterjee. Markov decision processes with multiple long-run average objectives. In *Proc. Springer FSTTCS*. 2007.

[32] M. Chen, S. C. Liew, Z. Shao, and C. Kai. Markov approximation for combinatorial network optimization. *IEEE Transactions on Information Theory*, 59(10):6301–6327, 2013.

[33] Cisco visual networking index: global mobile data traffic forecast update, 2016–2021. Cisco, 2017.

[34] J.-F. Cordeau, G. Laporte, and A. Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society*, 52(8):928–936, 2001.

[35] S. C. Davis, S. W. Diegel, and R. G. Boundy. *Transportation Energy Data Book: Edition 34*. U.S. Department of Energy, 2015.

[36] D. P. de Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations research*, 51(6):850–865, 2003.

[37] E. Demir, T. Bektaş, and G. Laporte. A comparative analysis of several vehicle emission models for road freight transportation. *Transportation Research Part D: Transport and Environment*, 16(5):347–357, 2011.

[38] E. Demir, T. Bektaş, and G. Laporte. A review of recent research on green road freight transportation. *European Journal of Operational Research*, 237(3):775–793, 2014.

[39] L. Deng, M. H. Hajiesmaili, M. Chen, and H. Zeng. Energy-efficient timely transportation of long-haul heavy-duty trucks. In *Proc. ACM e-Energy*, 2016.

[40] L. Deng, C.-C. Wang, M. Chen, and S. Zhao. Timely wireless flows with arbitrary traffic patterns: Capacity region and scheduling algorithms. In *Proc. IEEE INFOCOM*, 2016.

[41] D. Eppstein. Finding the k shortest paths. *SIAM Journal on computing*, 28(2):652–673, 1998.

[42] W. Ford Torrey and D. Murray. An analysis of the operational costs of trucking: A 2015 update. 2015.

[43] R. G. Gallager. *Stochastic Processes: Theory for Applications*. Cambridge University Press, 2013.

[44] L. M. Gambardella, É. Taillard, and G. Agazzi. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. 1999.

[45] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. 1979.

[46] K. Ghoseiri and B. Nadjari. An ant colony optimization algorithm for the bi-objective shortest path problem. *Applied Soft Computing*, 10(4):1237–1246, 2010.

[47] B. L. Golden, S. Raghavan, and E. A. Wasil. *The vehicle routing problem: latest advances and new challenges*. Springer Science & Business Media, 2008.

[48] G. J. Gordon. *Approximate Solutions to Markov Decision Processes*. PhD thesis, Carnegie Mellon University Pittsburgh, PA, 1999.

[49] W. Harrington and A. Krupnick. Improving fuel economy in heavy-duty vehicles. *Resources for the Future DP*, 2012.

[50] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations research*, 17(1):36–42, 1992.

[51] E. Hellström, J. Åslund, and L. Nielsen. Design of an efficient algorithm for fuel-optimal look-ahead control. *Control Engineering Practice*, 18(11):1318–1327, 2010.

[52] E. Hellström, M. Ivarsson, J. Åslund, and L. Nielsen. Look-ahead control for heavy trucks to minimize trip time and fuel consumption. *Control Engineering Practice*, 17(2):245–254, 2009.

[53] I. Hou, V. Borkar, and P. Kumar. A theory of QoS for wireless. In *Proc. IEEE INFOCOM*, 2009.

[54] I. Hou and P. Kumar. Admission control and scheduling for QoS guarantees for variable-bit-rate applications on wireless channels. In *Proc. ACM MobiHoc*, 2009.

[55] I. Hou and P. Kumar. Scheduling heterogeneous real-time traffic over fading wireless channels. In *Proc. IEEE INFOCOM*, 2010.

[56] I. Hou and P. Kumar. Utility maximization for delay constrained QoS in wireless. In *Proc. IEEE INFOCOM*, 2010.

[57] I. Hou and P. Kumar. Utility-optimal scheduling in time-varying wireless networks with delay constraints. In *Proc. ACM MobiHoc*, 2010.

[58] I. Hou and R. Singh. Scheduling of access points for multiple live video streams. In *Proc. ACM MobiHoc*, 2013.

[59] J. J. Jaramillo and R. Srikant. Optimal scheduling for fair resource allocation in ad hoc networks with elastic and inelastic traffic. In *Proc. IEEE INFOCOM*, 2010.

[60] J. J. Jaramillo, R. Srikant, and L. Ying. Scheduling for optimal rate allocation in ad hoc networks with heterogeneous delay constraints. *IEEE Journal on Selected Areas in Communications*, 29(5):979–987, 2011.

[61] A. Jeffrey and D. Zwillinger. *Table of Integrals, Series, and Products, Seventh Edition.* Academic Press, 2007.

[62] L. Jiang and J. Walrand. A distributed CSMA algorithm for throughput and utility maximization in wireless networks. *IEEE/ACM Transactions on Networking*, 18(3):960–972, 2010.

[63] A. Jüttner, B. Szviatovski, I. Mécs, and Z. Rajkó. Lagrange relaxation based method for the QoS routing problem. In *Proc. IEEE INFOCOM*, 2001.

[64] X. Kang, I. Hou, and L. Ying. On the capacity requirement of largest-deficit-first for scheduling real-time traffic in wireless networks. In *Proc. ACM MobiHoc*, 2015.

[65] X. Kang, W. Wang, J. J. Jaramillo, and L. Ying. On the performance of largest-deficit-first for scheduling real-time traffic in wireless networks. In *Proc. ACM MobiHoc*, 2013.

[66] Ł. Kruk, J. Lehoczky, and S. Shreve. Accuracy of state space collapse for earliest-deadline-first queues. *The Annals of Applied Probability*, 16(2):516–561, 2006.

[67] J. Larson, K.-Y. Liang, and K. H. Johansson. A distributed framework for coordinated heavy-duty vehicle platooning. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):419–429, 2015.

[68] S. Lashgari and A. S. Avestimehr. Timely throughput of heterogeneous wireless networks: Fundamental limits and algorithms. *IEEE Trans. Information Theory*, 59(12):8414–8433, 2013.

[69] J. Leskovec and R. Sosič. SNAP: A general purpose network analysis and graph mining library in C++, 2014. `http://snap.stanford.edu/snap`.

[70] D. H. Lorenz and D. Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Operations Research Letters*, 28(5):213–219, 2001.

[71] W. Mallett. *Freight Performance Measurement: Travel Time in Freight-Significant Corridors.* U.S. Federal Highway Administration, 2006.

[72] F. Mannering, W. Kilareski, and S. Washburn. *Principles of Highway Engineering and Traffic Analysis*. John Wiley & Sons, 2007.

[73] T. Markel, A. Brooker, T. Hendricks, V. Johnson, K. Kelly, B. Kramer, M. O'Keefe, S. Sprik, and K. Wipke. ADVISOR: a systems analysis tool for advanced vehicle modeling. *Journal of Power Sources*, 110(2):255–266, 2002.

[74] Z. Mohamed-Kassim and A. Filippone. Fuel savings on a heavy vehicle via aerodynamic drag reduction. *Transportation Research Part D: Transport and Environment*, 15(5):275–284, 2010.

[75] E. K. Nam and R. Giannelli. Fuel consumption modeling of conventional and advanced technology vehicles in the physical emission rate estimator (PERE). *U.S. Environmental Protection Agency*, 2005.

[76] J. Ni, B. Tan, and R. Srikant. Q-CSMA: Queue-length-based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks. *IEEE/ACM Transactions on Networking*, 20(3):825–836, 2012.

[77] I. Norstad, K. Fagerholt, and G. Laporte. Tramp ship routing and scheduling with speed optimization. *Transportation Research Part C: Emerging Technologies*, 19(5):853–865, 2011.

[78] W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[79] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[80] Global Internet phenomena report. Sandvine, 2015.

[81] M. W. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations research*, 4(1):285–305, 1985.

[82] P. J. Schweitzer and A. Seidmann. Generalized polynomial approximations in Markovian decision processes. *Journal of mathematical analysis and applications*, 110(2):568–582, 1985.

[83] G. Scora, K. Boriboonsomsin, and M. Barth. Value of eco-friendly route choice for heavy-duty trucks. *Research in Transportation Economics*, 52:3–14, 2015.

[84] P. Serafini. Some considerations about computational complexity for multi objective combinatorial problems. In *Recent advances and historical development of vector optimization*, pages 222–232. Springer, 1987.

[85] F. Stodolsky, L. Gaines, and A. Vyas. Analysis of technology options to reduce the fuel consumption of idling trucks. Technical report, Argonne National Lab, 2000.

[86] Y. Suzuki. A new truck-routing approach for reducing fuel consumption and pollutants emission. *Transportation Research Part D: Transport and Environment*, 16(1):73–77, 2011.

[87] T. Szigeti and C. Hattingh. Quality of service design overview. Cisco, 2004.

[88] K. C. Tan, L. H. Lee, Q. Zhu, and K. Ou. Heuristic methods for vehicle routing problem with time windows. *Artificial intelligence in Engineering*, 15(3):281–295, 2001.

[89] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1948, 1992.

[90] J. D. Teresco. The Clinched Highway Mapping (CHM) project. `http://cmap.m-plex.com/`.

[91] G. Tsaggouris and C. Zaroliagis. Multiobjective optimization: Improved FPTAS for shortest paths and non-linear objectives with applications. *Theory of Computing Systems*, 45(1):162–186, 2009.

[92] M. Tunnell. Estimating truck-related fuel consumption and emissions in maine: A comparative analysis for six-axle, 100,000 pound vehicle configuration. In *Proc. TRB Annual Meeting*, 2011.

[93] C.-C. Wang. Delay-constrained capacity for broadcast erasure channels: A linear-coding-based study. In *Proc. IEEE ISIT*, 2016.

[94] Y. Xu, C. Yu, J. Li, and Y. Liu. Video telephony for end-consumers: measurement study of Google+, iChat, and Skype. In *Proc. ACM IMC*, 2012.