The Top-percentile Traffic Routing Problem

Xinan Yang

Doctor of Philosophy

University of Edinburgh

December 2011

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

(*Xinan Yang*)

Abstract

Multi-homing is a technology used by Internet Service Provider (ISP) to connect to the Internet via multiple networks. This connectivity enhances the network reliability and service quality of the ISP. However, using multi-networks may imply multiple costs on the ISP. To make full use of the underlying networks with minimum cost, a routing strategy is requested by ISPs. Of course, this optimal routing strategy depends on the pricing regime used by network providers. In this study we investigate a relatively new pricing regime – top-percentile pricing. Under top-percentile pricing, network providers divide the charging period into several fixed length time intervals and calculate their cost according to the traffic volume that has been shipped during the θ-th highest time interval. Unlike traditional pricing regimes, the network design under top-percentile pricing has not been fully studied. This paper investigates the optimal routing strategy in case where network providers charge ISPs according to top-percentile pricing. We call this problem the Top-percentile Traffic Routing Problem (TpTRP). As the ISP cannot predict next time interval's traffic volume in real world application, in our setting up the TpTRP is a multi-stage stochastic optimisation problem. Routing decisions should be made at the beginning of every time period before knowing the amount of traffic that is to be sent. The stochastic nature of the TpTRP forms the critical difficulty of this study.

In this paper several approaches are investigated in either the modelling or solving steps of the problem. We begin by exploring several simplifications of the original TpTRP to get an insight of the features of the problem. Some of these allow analytical solutions which lead to bounds on the achievable optimal solution. We also establish bounds by investigating several "naive" routing policies. In the second part of this work, we build the multi-stage stochastic programming model of the TpTRP, which is hard to solve due to the integer variables introduced in the calculation of the top-percentile traffic. A lift-and-project based cutting plane

method is investigated in solving the SMIP for very small examples of TpTRP. Nevertheless it is too inefficient to be applicable on large sized instances. As an alternative, we explore the solution of the TpTRP as a Stochastic Dynamic Programming (SDP) problem by a discretization of the state space. This SDP model gives us achievable routing policies on small size instances of the TpTRP, which of course improve the naive routing policies. However, the solution approach based on SDP suffers from the curse of dimensionality which restricts its applicability. To overcome this we suggest using Approximate Dynamic Programming (ADP) which largely avoids the curse of dimensionality by exploiting the structure of the problem to construct parameterized approximations of the value function in SDP and train the model iteratively to get a converged set of parameters. The resulting ADP model with discrete parameter for every time interval works well for medium size instances of TpTRP, though it still requires too long to be trained for large size instances. To make the realistically sized TpTRP problem solvable, we improve on the ADP model by using Bézier Curves/Surfaces to do the aggregation over time. This modification accelerates the efficiency of parameter training in the solution of the ADP model, which makes the realistically sized TpTRP tractable.

Acknowledgement

First and foremost I offer my sincerest gratitude to my supervisor, Dr. Andreas Grothey, who has supported me throughout the course of my Ph.D. research with his patience and knowledge whilst allowing me the room to work in my own way. Without his invaluable guidance and assistance, I would not have completed this work. He is so nice, responsible and considerate. Working together with him is the most precious experience in my life.

I deeply thank my second supervisor Prof. Jacek Gondzio for his kind encouragement and support during these four years. He is a perfect gentleman who is always nice and polite. His advices on my character and career are of great importance for my future.

I am also grateful to Prof. Ken McKinnon for his valuable advice on the using of Approximate Dynamic Programming. This is definitely the crucial part of this work. I would not have completed my research so frequently without the suggestions and materials provided by him.

I am also grateful to the France Telecom Ltd. for offering me the opportunity to pursue this project and for their consistently support on my research. The technical feedback from them is so precious for improving this work.

My boyfriend and families have been the most important support for me. I could not have reached this important milestone without their encouragement and understanding, which helped me conquer difficulties and complete my work.

Last but definitely not the least, I would like to than all my friends and colleagues. There is so much fun during the four years of my stay in Edinburgh University. I will never forget this sweet memory.

List of Variables

Variable	Definitions
I	The set of network providers
i	The index of network providers, $i \in I$
n	The number of network providers, $n = \|I\|$
c_i	The per-unit cost of network provider i
Γ, Γ_i	The set of time intervals used by network provider i
τ	The index of time intervals, $\tau \in \Gamma$
$\hat{\tau}_{i,j}^k$	The control points (fixed) in the Bézier Curve model
q, q_i	The percentile parameter, e.g. $q = 5\%$ means we use top-5% pricing
θ, θ_i	The index of the top-percentile time interval, e.g. $\theta = 216$ means cost is charged on the 216-th highest volume of traffic
T^τ	The volume of traffic that required to be sent in time interval τ
$\hat{T}^\tau$	The realisation of T^τ
T_l, T_u	The lower and upper bounds for realisation of T^τ
T_i^τ	The volume of traffic that network provider i received in time interval τ
$\hat{T}_i^\tau$	The realisation of T_i^τ
$\hat{T}_i^{j,\tau}$	The j-th highest volume of traffic that has been sent to network provider i before time interval τ, i.e. j-th highest in $\{\hat{T}_i^1, ..., \hat{T}_i^{\tau-1}\}$
$\tilde{T}_i^\theta$	The θ-th highest volume of traffic that has been sent to network provider i
Ω^τ	The discrete set of traffic realisations in time interval τ
ω^τ	The random event of traffic realisation in τ, $\omega^\tau \in \Omega^\tau$
$\hat{\omega}^\tau$	The realisation of ω^τ
$\hat{\omega}_\tau$	The realisation history of ω^τ up to τ, $\hat{\omega}_\tau = (\hat{\omega}^1, \hat{\omega}^2, ..., \hat{\omega}^\tau)$
$\omega_{(m)}^\tau$	The realisation at time stage τ in the m-th sample path in ADP model

d^τ	A vector of the routing decisions for time interval τ, which has different definitions for different models
x_i^τ	The proportional decision on T^τ (in SMIP model) or on T_{add}^τ (in SDP/ADP models) for network provider i
x^τ	A vector of x_i^τ over network providers, $x^\tau = (x_1^\tau, x_2^\tau, ..., x_n^\tau)^T$
x_τ	The decision history of x^τ up to τ, $x_\tau = (x^1, x^2, ...x^\tau)$
χ^τ	The feasible decision set for time interval τ in SDP/ADP models
y_i^τ	The cut-off level for network i in time interval τ in SDP/ADP models
$y_i^{\hat{\omega}_{\|\Gamma\|}}$	The volume of the θ-th highest traffic for network provider i under realisation history $\hat{\omega}_{\|\Gamma\|}$ in SMIP model
$z_i^{\tau,\hat{\omega}_{\|\Gamma\|}}$	Binary variable for network provider i under realisation history $\hat{\omega}_{\|\Gamma\|}$ in SMIP model: $z_i^{\tau,\hat{\omega}_{\|\Gamma\|}} = \begin{cases} 1, & \text{if } \tau \text{ is in the top } q\text{-percentile time intervals} \\ 0, & \text{otherwise} \end{cases}$
S_i^τ	The state for network provider i at time stage τ in SDP/ADP models: $S_i^\tau = \{\hat{T}_i^{j,\tau} \| j = 1, ..., \theta\}$
S^τ	The state at time stage τ in SDP/ADP models: $S^\tau = \{S_i^\tau \| i \in I\}$
$\tilde{S}_x^\tau$	The post-decision state at time stage τ in ADP model
$S_{(m)}^\tau$	The state at time stage τ following the m-th sample path in ADP model
$T_{Add}(S^\tau)$	The additional traffic at state S^τ, $T_{Add}(S^\tau) = \max\{\hat{T}^\tau - \sum_{i \in I} \hat{T}_i^{\theta,\tau}, 0\}$
L	The discretization level of the state space in SDP model
m	The index of iterations in the ADP model
$\hat{v}_\tau^{(m)}$	The sample estimation of $\bar{V}_\tau^{(m-1)}$ in m-th iteration
$\bar{V}_\tau$	The value function approximation for interval τ in ADP model
$\bar{V}_{\tau,i}$	The component in $\bar{V}_\tau$ that only relate to network provider i
$\bar{V}_\tau^{(m)}$	The $\bar{V}_\tau$ after being trained through m iterations
$\tilde{V}_\tau^x$	The function value around the post-decision state $\tilde{S}_x^\tau$
$\bar{V}_x^\tau$	The value function approximation of $\tilde{V}_\tau^x$ around the post-decision state
$\beta_0^\tau, \beta_{i,j}^\tau$	The parameter in the linear regression model used in $\bar{V}_\tau$
K	The number of control points in τ-components in the Bézier Curve model
R	The number of control points in θ-components in the Bézier Surface model
$\beta_{i,j}^k$	The regression parameter in the Bézier Curve model
$\beta_i^{k,r}$	The regression parameter in the Bézier Surface model

Contents

Chapter 1

Introduction

Internet Service Providers (ISPs) do not generally have their own network infrastructure to route the incoming traffic of their customers, but instead use external network providers. Multi-homing is used by ISPs to connect to the Internet via more than one network provider (see Figure 1.0.1). This technique is currently widely adopted to provide fault tolerance and traffic engineering capabilities in ISPs as it improves the reliability and quality of service of the ISP. When failure occurs in one of the networks or its quality degrades, the ISP can use an alternative network to send data (Bagnulo et al. [2]).

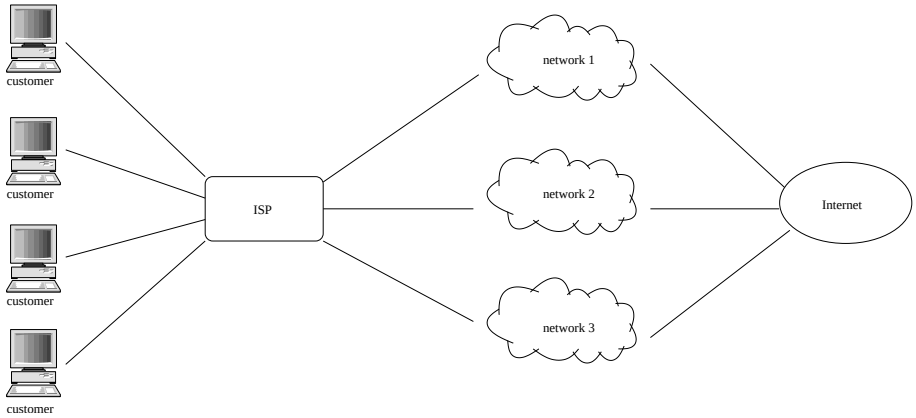


Figure 1.0.1: ISP connect customers to the Internet under multi-homing technique

However, an ISP that uses multi-homing is subject to extra charges due to the use of multiple networks. Important questions that are faced by such an ISP is how to assign traffic among network providers to minimize the inflicted cost of multi-homing. Of course, the answer to this question depends on the pricing

policy used by the network providers.

Traditionally, network providers have used pricing policies such as fixed-cost pricing and per-usage pricing, or a combination of them. Under per-usage pricing, the ISP pays for the actual amount of traffic shipped over the charging period, while fixed-cost pricing means the ISP pays a fixed price for the charging period regardless of the amount of traffic shipped. Optimal multi-homing routing strategies under these traditional pricing policies have been thoroughly investigated in past studies. For example, Altman et al. [1] investigate the competitive routing problem with polynomial link cost functions, show that these costs have appealing properties that lead to predictable and efficient network flows. These properties make the polynomial cost structure attractive for traffic regulation and link pricing in telecommunication networks. Other study such as Wang and Schulzrinne [58] propose a pricing scheme in a Differentiated Services framework environment based on the cost of providing different levels of quality of service to different classes. Efficient admission control is developed to provide a lower connection blocking rate at high loads. On the other hand, the fixed-cost pricing can be found mainly in the literature of network design, in which a fixed price for each edge is set beforehand. An example is Herzberg and Shleifer [30] in the context of designing reliable networks. Actually, in addition to these two extreme pricing policies, Shenker et al. [53] claim that there should be a continuum between them. Examples can be found in surveys e.g. Falkner et al. [25], Shenker et al. [53].

In this work however, we consider a relatively new pricing regime, top-percentile pricing. Transmitting information is a low (virtually zero) marginal cost activity with very high fixed costs, where required capacity is determined by peak demand. For this reason, network providers have been moving to a pricing strategy that depends in part on the traffic levels in the top-θth percentile. Nowadays, top-percentile pricing is the most widespread pricing mode used by network providers to charge service providers, that is quickly becoming established (see e.g. Odlyzko [43]). In this scheme, the network provider divides the charging period, say a month, into several time intervals with equal, fixed length. Then, it measures and evaluates the amount of data (traffic) sent in each of these time intervals. At the end of the charging period, the network provider selects the traffic volume of the top q-percentile interval as the basis for computing the cost (see Figure 1.0.2). For example, if the charging period (i.e. 30 days) is divided into 4320 time intervals with a length of 10 minutes each, and if top 5-percentile pricing is

used, the cost computed by top-percentile pricing is based on the traffic volume of the top 216th interval.

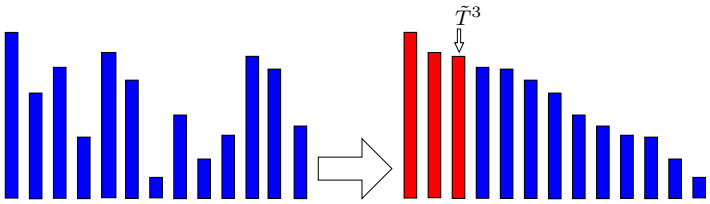


Figure 1.0.2: Random time intervals, reordered time intervals and the top-percentile traffic based on which the cost is computed

In practice, network providers might combine top-percentile pricing with other pricing policies. In this work however, we investigate mainly the optimal multi-homing routing policy in the context of pure top-percentile pricing, namely network providers charge the ISP according to and only according to the traffic volume in the top-percentile time interval. The ISP, then, faces the optimization problem of determining which network provider to use for an increment of traffic, while monitoring all network providers traffic levels that fall within this top percentile. Extensions to pure top-percentile (mixed pricing policies with fixed-cost pricing or per-usage pricing) are considered as well, most of which can be naturally covered by the model we developed in the end.

In contrast to the traditional pricing policies, very little work has been done for top-percentile pricing. The efficient network operation under top-percentile pricing is thus not well understood yet. The deterministic problem (in which we assume that we know all the traffic volumes in advance) has been analysed in Chardy et al. [19], where the authors investigate the traffic routing problem under a combined pricing policy – top-percentile pricing and fixed cost pricing. In the stochastic case, Levy et al. [36] develop a probabilistic model and provide analysis of the expected costs, demonstrating that multi-homing can be economical efficient under top-percentile pricing, although they did not derive an optimal routing policy. On the other hand, Goldenberg et al. [27] focus on the development of smart routing algorithms for optimising both cost and performance for multi-homing users under top-percentile pricing. However, in the case where traffic volumes are not available in advance (stochastic case), they apply the deterministic algorithm directly on the prediction of one later time interval's traffic and then accommodate the prediction error with some heuristics. As a conclusion, to the best of our knowledge there is no result dealing with the optimal multi-homing routing policy under top-percentile pricing as a stochastic optimization problem.

The purpose of this study is to investigate the optimal routing strategy under pure top-percentile pricing in a stochastic set up. Precisely, if all network providers charge the ISP based on the volume of the top q-percentile time interval's traffic, how to allocate all time intervals' traffic among those networks to minimise the total expected cost charged on the ISP. In the following parts of this paper we call this problem the Top-percentile Traffic Routing Problem (TpTRP). The TpTRP is a stochastic problem, where the ISP does not know the volume of future time intervals' traffic in advance. Instead, we assume that the probability distributions for all time intervals' traffic are known.

This study consists of three parts. In order to get an insight of the nature of Top-percentile pricing, we firstly investigate the deterministic counterpart of the original stochastic TpTRP, where all time periods' traffic volumes are assumed to be known beforehand. The optimal solution to this deterministic problem has been discussed in Chardy et al. [19], which fully exploits the advantage of pure top-percentile pricing's feature of charging is only based on the top-q percentile traffic. Starting from this optimal routing strategy, we suggest several simplifications of the original stochastic model which are analytically solvable. The naive routing policies derived from them provide us with lower and upper bounds on the optimal solution of the TpTRP.

In the second part of this study, we build a multi-stage stochastic programming model to solve the original stochastic TpTRP. As the randomness of traffic volume appears in every time stage, this stochastic programming model is extremely large in size. Unfortunately, the difficulty of solving this stochastic programming model is not only restricted to the large size, but also in the integer variables that are introduced in the modelling of the top-percentile cost. Unlike traditional pricing regimes, top-percentile pricing calculates the charge based on the volume of top-q percentile traffic, which is a non-convex function of the traffic input. In our stochastic programming model, we calculate the top-percentile traffic using integer variables. This makes the multi-stage stochastic programming model too difficult to be solved for any but extremely small instances.

In the third part of this study we suggest a Stochastic Dynamic Programming (SDP) model to solve the TpTRP. Traditional SDP works on discrete space, thus is not directly applicable on our problem. A discretization of the traffic space is required. Apart from this, several necessary simplifications and restrictions on either the decision space and the solution method are enforced on the SDP

model, which result in a fact that the SDP model forms an approximation of the original TpTRP problem. Nevertheless, the experiments on small size TpTRP instances show that the optimal routing polices derived from the solution of the SDP model outperform any naive routing polices for several different types of traffic distribution. Mean costs of the SDP routing policy on a large number of test samples are close to the lower bound given by their deterministic counterpart of the TpTRP. Of course, it is obvious that the SDP model is not directly applicable on large sized instances, due to the intractable size of the state space introduced by the discretization. This is the widely known 'curse of dimensionality' of dynamic programming and is the most often-cited reason why dynamic programming cannot be used.

It has been suggested by Powell [45] that Approximate Dynamic Programming (ADP) is a promising technique to avoid the curse of dimensionality. ADP has been applied in a wide range of applications, e.g. Jong and Jay [33], Nascimento and Powell [41, 44]. In the second section of the third part of this work, we follow the work on dynamic programming, modify the SDP model to build an ADP model to overcome the curse of dimensionality in the state space. Instead of discretizing the traffic region to make SDP applicable, ADP offers a way to allow dynamic programming steps to work on the continuous state space by exploiting the structure of the SDP model to construct continuous approximation of value functions. The resulting ADP model is trained iteratively with a large number of traffic samples until we obtain a set of converged parameters for the value function approximation, from which we can derive the optimal routing policy. Numerical experiments show that the ADP model can solve medium sized TpTRP instances within reasonable time, while giving as well performing routing policies as the SDP model does.

The realistically sized TpTRP consists of 4320 time intervals and charging based on the top-261th volume of traffic, while randomness is shown in every time interval. This is a hard problem because of the size, which makes the ADP model suffer from slow convergence and thus long training time. In order to make the realistically sized problem tractable, we improve on the ADP model by a further aggregation in modelling parameters. We focus on the investigation of the parameter structure in the original 'discrete' ADP model, develop a Bézier Curves/Surfaces aggregation of it to build a ADP-Bézier-Curve/Surface model. This significantly reduces the number of parameters in the ADP model, thus making realistically sized instances of the TpTRP solvable.

Chapter 2

The TpTRP and its Simplifications

This section gives a formal description of the TpTRP parameters and develops several simplified reformulations of the original TpTRP which can be solved analytically. These simplifications will provide lower and upper bounds for the optimal routing policy of the stochastic TpTRP.

2.1 Notations and assumptions

2.1.1 Problem parameters

Now we specify the symbol and the meaning of parameters which define the TpTRP problem.

- $I, |I| = n$: The set of network providers. There are n underlying network providers, which are ordered by increasing per-unit cost: $c_1 \leq c_2 \leq ... \leq c_n$.
- $\Gamma_i, i \in I$: The set of time intervals for network provider i. The modelling period is divided into $|\Gamma_i|$ time intervals of equal length by network provider i, for the purpose of top-percentile pricing calculation.
- $q_i, i \in I$: The percentile parameter for the top-percentile pricing calculation for network provider i

- $\theta_i = \lfloor |\Gamma_i| * q_i \rfloor$: The index of the top-percentile time interval: the cost inflicted on the ISP charged by network provider i is a function of the θ_i-th highest volume of traffic sent to network provider i.

- $c_i, i \in I$: The per unit cost charged by network provider i on the top-percentile traffic. Namely the cost charged by network provider i is equal to $c_i \tilde{T}_i^{\theta}$ if $\tilde{T}_i^{\theta}$ indicates the θ_i-th highest volume of traffic sent to provider i.

- $T^{\tau}, \tau \in \Gamma$: The volume of traffic required to be sent in time interval τ. T^{τ} is a stochastic parameter before the routing decision for period τ is made, whose randomness depending on the random event ω^{τ}. When the random event ω^{τ} becomes known as $\hat{\omega}^{\tau}$, we use $\hat{T}^{\tau} = T^{\tau}(\hat{\omega}^{\tau})$ to represent the realisation of T^{τ}. The probabilistic distribution of $\omega^{\tau}, \tau \in \Gamma$ is assumed to be available ahead of time.

2.1.2 Assumptions

This work focuses mainly on developing the optimal multi-homing routing policy under (pure) top-percentile pricing. In order to emphasise this goal, we impose the following simplifying assumptions on the TpTRP:

- **(a1)** There is no un-separable batch of data. Traffic for every time interval can be split into any sub-traffic sets.

- **(a2)** There is no upper bound on the volume of traffic that can be transferred by its network per time interval for any network provider.

- **(a3)** During the modelling period, there is no failure in any of the networks.

- **(a4)** The modelling period is divided into the same (finite) number of time intervals of the same length by all network providers. Namely the set $\Gamma_1 = \Gamma_2 = ... = \Gamma_n = \Gamma$.

Note that assumption (a1) is a simplification of the technical layer in telecommunication. As Internet traffic is naturally delivered in packets, where a packet is a formatted unit of data, in case where a packet is reasonably small (i.e. 1500 byte), the traffic can be assumed to be splittable. Assumptions (a2) and (a3) are imposed to allow us to ignore the influence of failure control (which is itself

a complicate network operation problem), providing us a perfect environment to focus on developing the optimal routing policy. Once the optimal routing policy is achieved, to deal with the situation of possible network failures it requires the introduction of failure events as an additional random variable on top of the observed traffic. The additional difficulty varies with model.

Although it might be unrealistic in real-world applications, (a4) is the most critical assumption we made to make the TpTRP tractable. The reason is, if network providers use different lengths of time intervals, there is no inherent one-to-one correspondence of time periods and stages. Therefore, the design of the appropriate stage and the decision process is not trivial. Note that in the rest of this work, we assume that all network providers use the same top-percentile parameters q (and thus θ) as well. Unlike (a4), this assumption is not critical in modelling but just a simplification.

With the above notation and assumptions, the specified TpTRP problem can be defined as follows. An ISP provides Internet service to a group of individual customers, while the total data generated by customers (T^τ) is random. To send data, the ISP has to rent network providers' network. All network providers charge the ISP according to pure top-percentile pricing regime. With the charging period been divided into $|\Gamma|$ time intervals, cost enforced on the ISP by network provider i is the multiplication of per unit cost c_i and the θ-th highest volume of traffic $\tilde{T}_i^\theta$. Then given a set of network providers I, assuming we know the probabilistic distribution for every time intervals' traffic volume ahead of time, the question is how to allocate the forthcoming traffic to network providers, which formed the TpTRP problem.

2.1.3 Decision variables

- $d^\tau, \tau \in \Gamma$: The routing decision for time interval τ.

As mentioned above, in real world applications the ISP cannot predict the amount of traffic to be sent during τ at the beginning of time interval τ. Thus the decision d^τ should be made under uncertainty. There are several possible choices for the framework of valid d^τ (e.g. d^τ can be percentage based: $d^\tau = (x_1^\tau, x_2^\tau, ..., x_n^\tau)^T$, $\sum_{i \in I} x_i^\tau = 1$, where x_i^τ indicates the proportion of T^τ to be sent to network provider i). At the moment, we just use d^τ to represent a generally implementable decision

and assume its implementation function is: $T_i^\tau(T^\tau, d^\tau)$, thus the traffic volume that network provider i received in τ is $T_i^\tau = T_i^\tau(T^\tau, d^\tau)$. We leave the discussion on how to design an appropriate decision space for a specific TpTRP model to later parts.

Nevertheless, it is worthwhile to point out a useful key property of TpTRP in defining decision rules. In the stochastic programming set up, we are assumed to know the realisation of T^τ right after the decision d^τ is made. However in the real world TpTRP the traffic is revealed continuously over time, since at any point during time interval τ, customers may generate new data. The traffic realisation $\hat{T}^\tau$ however, is the cumulative amount of all data generated within τ. This means, as shown in Figure 2.1.1, that we cannot see the complete amount of $\hat{T}^\tau$ before the end of time interval τ.

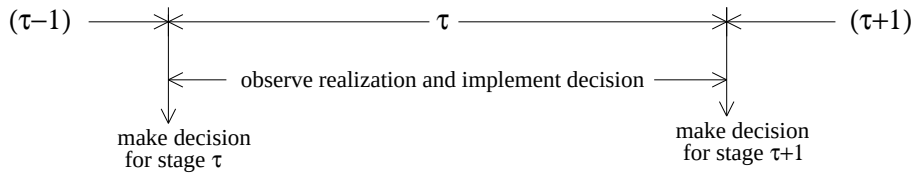


Figure 2.1.1: Process of data revelation and implementation of decisions

However, any fraction of data must be sent as soon as it is generated instead of waiting until the end of time interval τ when the whole traffic $\hat{T}^\tau$ has been revealed. Therefore in addition to being non-anticipative with respect to $\hat{T}^\tau$, another necessary condition for a feasible routing decision is that it is implementable without knowing the whole volume of $\hat{T}^\tau$. For example in the simplest case, we can decide at the start of every time period where to send the whole traffic for this period.

Remember that under assumption (a1), we can split the traffic into any packets, thus more sophisticated routing policies can be considered by operating a 'time slicing' scheme. For example, the percentage based routing policies (i.e., $d^\tau = (x_1^\tau, x_2^\tau, ..., x_n^\tau)^T, \sum_{i \in I} x_i^\tau = 1$, where x_i^τ represents the proportion of the whole traffic T^τ to be routed to network provider i) or cut-off based routing policies (i.e., $d^\tau = (r_1^\tau, y_1^\tau; r_2^\tau, y_2^\tau; ..., r_n^\tau, y_n^\tau)^T$, where r_i^τ indicates the order of network provider i in the implementation rule, namely the first $y_{i_1}^\tau$ unit of traffic is sent to provider i_1 if $r_{i_1}^\tau = 1$, the next $y_{i_2}^\tau$ to provider i_2 if $r_{i_2}^\tau = 2$ and so forth). In particular, since the revelation of traffic is gradual, we can also consider combined routing policies, e.g. $d^\tau = (r_1^\tau, y_1^\tau, x_1^\tau; r_2^\tau, y_2^\tau, x_2^\tau; ..., r_n^\tau, y_n^\tau, x_n^\tau)^T, \sum_{i \in I} x_i^\tau = 1$, where we firstly

allocate the cut-off level y^τ in turn and then split the remaining traffic (if there is) according to the percentage decision x^τ.

In this work we will consider either the percentage based decision rule or a combined routing policy, according to the model we build for TpTRP. The detailed description of decision space will be addressed in its own chapter for each model.

2.2 Stochastic nature and solution difficulties

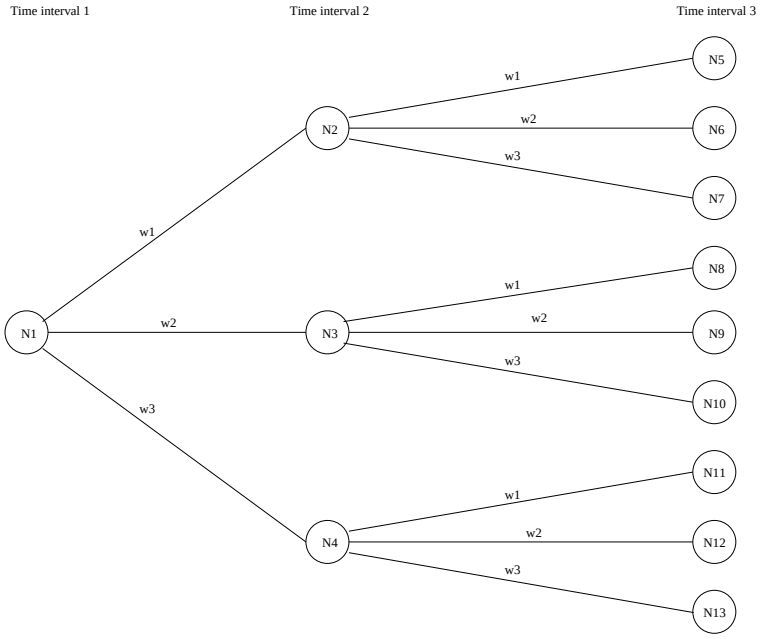


Figure 2.2.1: A sample stochastic tree

The TpTRP problem is a stochastic problem, where decisions should be made under the uncertainty of the volume of traffic. To illustrate the process of decision making and data revelation clearly, we show a sample stochastic tree in Figure 2.2.1. This is a three periods TpTRP instance, each of the first two periods has 3 potential realisations of uncertain traffic. At every node in the tree, we see the previous traffic realisation, while making a decision on how to allocate this period's traffic among network providers without knowing the volume of the forthcoming traffic. To obtain the optimal routing strategy for this TpTRP instance, we need to solve 4 decision problems (one for each on nodes $N1, N2, N3$ and $N4$) and record them for use.

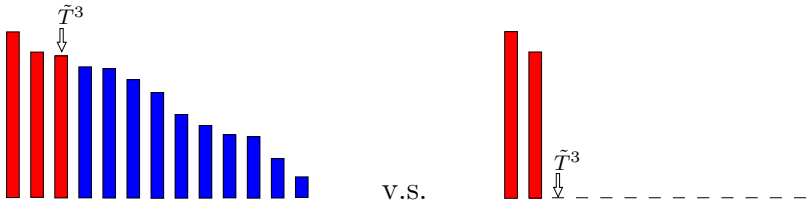
However, in reality the TpTRP is far larger than this 3-period instance. The

realistically sized TpTRP possesses 4320 periods (where a month is divided into time intervals of 10 minutes), and more seriously, a continuous probabilistic space for the random traffic volume. This introduces an incredible large stochastic model with millions of decision nodes even with a sparse discretization on the traffic space. To solve the problem it requires finding the optimal routing decision at every node, which altogether form the optimal routing policy we are seeking to find. In addition to the difficulty of getting the optimal decision for every single node, the size of the problem is already challenging enough.

2.3 Deterministic problem and naive routing policies

Instead of focusing on the complex stochastic model and trying to solve it, in this section we propose several simplifications of the original TpTRP we defined above. The purpose of this section is firstly to get a deeper insight of the features of TpTRP, and secondly derive bounds on the cost of the optimal routing policy, which can be used to evaluate the quality of the solution obtained by models we developed later. To obtain a lower bound we derive an analytic solution of the deterministic version of the TpTRP, where we assume that we know all future traffic volumes ahead of making the routing decisions. To obtain upper bounds we analyse several plausible (although not optimal) routing policies of increasing complexity.

Before we start, it is helpful to discuss how to make the full use of every network provider under pure top-percentile pricing policy.



(a) Instance (a) (b) Instance (b)

Figure 2.3.1: Top-percentile traffic

For example, Figure 2.3.1 shows a simplified top-percentile pricing procedure, in which we have 13 time intervals in total and are charged according to the 3rd highest volume of shipped traffic (denoted by $\tilde{T}^3$). In instance (a), every

time interval gets a nonzero amount of traffic shipped. If the network provider charges c per unit of traffic shipped during the top-3rd time interval, then the cost amounts to $c\tilde{T}^3$. However like what is shown in instance (b), if the ISP send no more than 2 time intervals' traffic to a network during the charging period, it will not be charged by this network provider as there is no traffic in the 3rd highest time interval. So that under pure top-percentile pricing policy, the ISP can ship up to $\theta - 1$ time intervals' traffic to a network without being charged by its provider.

2.3.1 Deterministic problem

Firstly, we consider the deterministic version of the TpTRP in which all the volumes of traffic $T^\tau, \forall \tau \in \Gamma$ are known (as $\hat{T}^\tau$) in advance. Without loss of generality, we can reorder these values into a non-increasing order $\hat{T}^1 \geq \hat{T}^2 \geq ... \geq \hat{T}^{|\Gamma|}$. The following lemma is translated from Matthieu Chardy's work [19], which gives the necessary conditions that the optimal routing policy should satisfy. Note that in real world applications there would be no more than 5 network providers, thus we assume $|\Gamma| >> n\theta$ always holds.

Lemma 2.3.1. *In the optimal routing policy for the deterministic case, the θ-th highest volume of traffic for all network providers apart from the cheapest one is zero, namely the ISP will incur a charge only from the cheapest network provider.*

Proof. We proof this lemma by contradiction.

Note that when we define the set of network providers in Section 2.1.1, we assume that they were ordered inherently by their cost: $c_1 \leq c_2 \leq ... \leq c_n$. Here we assume further that the network provider 1 is the unique cheapest network provider, namely $c_1 < c_2 \leq ... \leq c_n$. For a specific routing policy, let $\tilde{T}_i^\theta, i \in I$ be the volume of the θ-th highest volume of traffic shipped by network provider i. So that the cost charged on the ISP by this routing policy is $\sum_{i \in I} c_i \tilde{T}_i^\theta$.

Assume that there is an $i_0 \geq 2$ and $\tilde{T}_{i_0}^\theta > 0$, we show that we can reduce the objective value (cost) with a reallocation of traffic:

- reallocate $\tilde{T}_{i_0}^\theta$ and all the lower traffic which were previously sent by network provider i_0 to network provider 1;

- leave all the rest allocation intact.

With this reallocation, on one hand $\tilde{T}_{i_0}^{\theta,new} = 0$, that is we reduce the cost by $c_{i_0}\tilde{T}_{i_0}^{\theta}$. On the other hand, since all the reallocated traffic is no greater than $\tilde{T}_{i_0}^{\theta}$, therefore $\tilde{T}_1^{\theta,new} \leq c_1(\tilde{T}_{i_0}^{\theta} + \tilde{T}_1^{\theta})$, that is at worst the objective increases by $c_1\tilde{T}_{i_0}^{\theta}$. The total change in cost is thus $(c_1 - c_{i_0})\tilde{T}_{i_0}^{\theta}$, which is strictly negative since $c_{i_0} > c_1$. So, we have a better solution from the reallocation, and proved by contradiction that all the network providers apart from the cheapest one get a top-percentile traffic of zero.

□

From this lemma we can derive the optimal routing policy for the deterministic case (assuming provider 1 is the cheapest):

- Send (randomly) $\theta - 1$ time intervals' traffic out of $\hat{T}^1, ..., \hat{T}^{n(\theta-1)}$ to network provider 1;
- Send (randomly) $\theta - 1$ time intervals' traffic out of $\hat{T}^1, ..., \hat{T}^{n(\theta-1)}$ to network provider 2;
- ...
- Send (randomly) $\theta - 1$ time intervals' traffic out of $\hat{T}^1, ..., \hat{T}^{n(\theta-1)}$ to network provider n;
- Send all the remaining time intervals' traffic, i.e. $\hat{T}^{n(\theta-1)+1}, ..., \hat{T}^{|\Gamma|}$ to provider 1.

Under this routing policy, no traffic is split, i.e. every time interval's traffic is shipped by a single network provider. By implementing this optimal routing strategy, all network providers apart from the cheapest one (network provider 1) get $(\theta - 1)$ intervals filled with traffic, therefore the charge to the ISP is zero as the θ-th highest volume of traffic is left to be zero. On the other hand, the top θ-th time interval for the cheapest network provider (provider 1) has a volume of $\hat{T}^{n(\theta-1)+1}$, which introduces a cost of $c_1\hat{T}^{n(\theta-1)+1}$. As a result, the total cost charged to the ISP by involving the optimal routing policy is equal to $c_1\hat{T}^{n(\theta-1)+1}$.

As we assume that we have full knowledge of the traffic ahead of time, the optimal routing policy in the deterministic situation is not directly implementable on stochastic TpTRP. However, assuming we can predict the traffic volumes explicitly this principle can be used in designing heuristic rules. In addition to this, the optimal routing policy in the deterministic case also provides us with a lower bound on the optimal cost in the stochastic case. This lower bound can be used in practice to evaluate how good a stochastic routing policy is.

2.3.2 Naive routing policies

The aim of this section is to present three plausible routing policies of varying complexity for which we can analytically derive the theoretical expected cost. Throughout this section we need to assume:

(A1) All traffic volumes $T^\tau, \tau \in \Gamma$ are independent and identically-distributed and have compact support: $[T_l, T_u]$.

(A2) We do not consider splitting the traffic, i.e. the whole traffic T^τ for every time interval τ is routed to a single network provider.

Let $T^\tau, \tau \in \Gamma$ be i.i.d random variables with an absolutely continuous probability distribution function $F(x) = pr(T^\tau \leq x), x \in [T_l, T_u]$.

We start by deriving an expression for the expectation of the top-θth volume of traffic. Let $Y^\theta_{|\Gamma|}$ be the the volume of the θ-th highest out of $|\Gamma|$ traffic. The probability distribution function of $Y^\theta_{|\Gamma|}$ is:

$$F_{Y^\theta_{|\Gamma|}}(x) = Pr(Y^\theta_{|\Gamma|} \leq x) = \sum_{0 \leq j < \theta} B(j||\Gamma|, 1 - F(x)), \tag{2.3.1}$$

where $B(j||\Gamma|, 1 - F(x))$ is the probability mass function of the binomial distribution, which represents the probability that exactly j out of $|\Gamma|$ traffic are greater than x. As $F_{Y^\theta_{|\Gamma|}}(x)$ is a absolutely continuous distribution function, we define its density function as $f_{Y^\theta_{|\Gamma|}}(x)$ which satisfies $\int_{T_l}^{T_u} f_{Y^\theta_{|\Gamma|}}(x)dx = 1$ and

$F_{Y^{\theta}_{|\Gamma|}}(x) = \int_{T_l}^{x} f_{Y^{\theta}_{|\Gamma|}}(t)dt, \forall x \in [T_l, T_u]$. Accordingly, the expectation of the volume of the top-θth traffic is given by:

$$\mathbb{E}[Y^{\theta}_{|\Gamma|}] = \int_{T_l}^{T_u} x f_{Y^{\theta}_{|\Gamma|}}(x)d(x). \tag{2.3.2}$$

Single-homing Routing Policy (SRP)

The first routing policy we consider is single-homing, e.g. send everything to the cheapest network provider. The expected cost charged on the ISP will be:

$$Cost_{\mathbf{SRP}} = c_1 \mathbb{E}[Y^{\theta}_{|\Gamma|}] = c_1 \int_{T_l}^{T_u} x f_{Y^{\theta}_{|\Gamma|}}(x)d(x). \tag{2.3.3}$$

Trivial Multi-homing Routing Policy (TMRP)

Clearly the SRP is not good since we waste the $(\theta - 1)$ 'free' time intervals on all network providers apart from the cheapest one. Instead we consider the 'trivial' multi-homing routing policy that randomly routes $(\theta - 1)$ time intervals' traffic to every network provider and the rest to the cheapest one. In this way the ISP is only charged by the cheapest network provider, but uses the free time intervals of all network providers. The expected cost is accordingly:

$$Cost_{\mathbf{TMRP}} = c_1 \mathbb{E}[Y^{\theta}_{|\Gamma|-(n-1)(\theta-1)}] = c_1 \int_{T_l}^{T_u} x f_{Y^{\theta}_{|\Gamma|-(n-1)(\theta-1)}}(x)d(x). \tag{2.3.4}$$

Analytical Routing Policy (ARP)

It is trivial that the expected cost of TMRP is always lower than SRP. This means the routing policy derived from the single-homing architecture performs worse than the Trivial Multi-homing Routing Policy. Now the question is, whether we can improve on the TMRP.

The TMRP sends all the remaining time intervals' traffic to the cheapest network provider after filling the free time intervals of every network provider. However,

it is not obvious that the lowest expected future cost is incurred by choosing the cheapest network provider to send all the remaining traffic. As an alternative we consider sending all the remaining traffic to the network provider with the least expected future cost. Here we focus on the minimum actual cost we are going to pay instead of simply choosing the network provider with the least per unit cost. Following we give the Analytical Routing Policy (ARP) in detail:

- Firstly, fill all the network providers' free $(\theta - 1)$ time intervals with the first $n(\theta - 1)$ time intervals' traffic;
- Secondly, make decision on where to direct the $(n(\theta-1)+1)$st, and all the following time interval's traffic by comparing the future expected cost after time interval $n(\theta - 1)$. This means, we choose the decision which involves the least expected cost from n potential decisions: $D = \{d_1, d_2, ..., d_n\}$, where d_i represents 'allocate all the remaining traffic to network provider i'. The expected cost implied by decision $d = d_i$ is:

$$Cost_{d=d_i} = c_i \mathbb{E}[\theta\text{th highest out of } \hat{T}_i^1, ..., \hat{T}_i^{\theta-1}, T^{n(\theta-1)+1}, ..., T^{|\Gamma|}],$$

where $\hat{T}_i^j$ (known value) is the j-th highest volume of traffic that already been sent to network i, while $T^{n(\theta-1)+1}, ..., T^{|\Gamma|}$ are not yet revealed traffic. Comparing the expected cost implied by all decisions we get the routing policy and thus the future expected cost:

$$Cost_{\mathbf{ARP}} = \min_{d \in D} \quad Cost_{d=d_i}.$$

2.3.3 An example

In this section we give an example of TpTRP in which all time intervals' traffic follow a same uniform distribution and derive the theoretical expected costs of implementing the above routing policies on it. They will serve as benchmarks to judge the quality of the routing policy obtained by stochastic models we developed later.

Assume that we have 2 network providers with cost $c_1 < c_2$. Both of them divide the charging period into 10 time intervals and charge based on the 3rd highest

volume of traffic. Assume further that all time intervals' traffic follow a same uniform distribution with a cumulative distribution function:

$$F_U(x) = \begin{cases} 0, & 0 \leq x \leq 6000 \\ \frac{x-6000}{8000}, & 6000 < x < 14000 \\ 1, & x \geq 14000 \end{cases} \tag{2.3.5}$$

Let Y_m^k be the k-th highest traffic volume out of m random traffic. Then the distribution function of Y_m^k is

$$\begin{aligned} F_{Y_m^k}(x) = Pr(Y_m^k \leq x) &= \sum_{0 \leq j < k} B(j|m, 1 - F_U(x)) \\ &= \sum_{0 \leq j < k} \begin{pmatrix} m \\ j \end{pmatrix} (\tfrac{x-6000}{14000-6000})^{m-j} (\tfrac{14000-x}{14000-6000})^j \end{aligned} . \tag{2.3.6}$$

And its expectation is

$$\mathbb{E}[Y_m^k] = \int_{6000}^{14000} x d(F_{Y_m^k}(x)) = 14000 - \frac{k(14000-6000)}{m+1}. \tag{2.3.7}$$

According to formulae (2.3.3) and (2.3.4), the theoretical expected cost of implementing the SRP and TMRP are thus:

$$Cost_{\mathbf{SRP}} = c_1 \mathbb{E}[Y_{10}^3] = 10 * (14000 - \frac{3 * (14000-6000)}{10+1}) = 118181.82.$$

$$Cost_{\mathbf{TMRP}} = c_1 \mathbb{E}[Y_8^3] = 10 * (14000 - \frac{3 * (14000-6000)}{8+1}) = 113333.33. \tag{2.3.8}$$

Similarly, we can compute the theoretical expected cost of applying the Deterministic Routing Policy (DRP) as well if assuming we know all the traffic volumes in advance. This gives

$$Cost_{\mathbf{DRP}} = c_1\mathbb{E}[Y_{10}^5] = 10 * (14000 - \frac{5 * (14000 - 6000)}{10 + 1}) = 103636.36.$$

Now we show how to implement the ARP on the same example. It consists of two steps: Firstly allocate T^1, T^2 to network provider 1 and T^3, T^4 to network provider 2. Secondly, compare the expected cost of the following two choices to make the routing decision for all following time stages (here $\hat{T}^\tau$ represent the realisation of traffic T^τ):

1. send $T^5, ..., T^{10}$ to network 1

   $Cost_{d=d_1} = c_1\mathbb{E}[\text{top-3rd out of } \hat{T}^1, \hat{T}^2, T^5, ..., T^{10}]$,

2. send $T^5, ..., T^{10}$ to network 2

   $Cost_{d=d_2} = c_2\mathbb{E}[\text{top-3rd out of } \hat{T}^3, \hat{T}^4, T^5, ..., T^{10}]$,

Take $d = d_1$ for example, let A be the random variable representing the 3rd highest volume of traffic out of $\hat{T}^1, \hat{T}^2, T^5, ..., T^{10}$, and $F_A(x)$ be the probabilistic distribution function of A. Assume $\hat{T}^1 \geq \hat{T}^2$, then

$$F_A(x) = Pr(A \leq x) = \begin{cases} F_{Y_6^1}(x), & \text{if } \hat{T}^1 \geq x, \hat{T}^2 \geq x \\ F_{Y_6^2}(x), & \text{if } \hat{T}^1 \geq x, \hat{T}^2 \leq x \\ F_{Y_6^3}(x), & \text{if } \hat{T}^1 \leq x, \hat{T}^2 \leq x \end{cases} \tag{2.3.9}$$

So that $Cost_{d=d_1} = c_1\mathbb{E}[A] = \int_{6000}^{14000} x d(F_A(x))$.

For example, if $\hat{T}^1 = 13690, \hat{T}^2 = 13361$ and $\hat{T}^3 = 7730, \hat{T}^4 = 7238$, we can compute

$Cost_{d=d_1} = c_1\mathbb{E}[\text{top-3rd out of } 13690, 13361, T^5, ..., T^{10}] = 127659,$

$Cost_{d=d_2} = c_2\mathbb{E}[\text{top-3rd out of } 7730, 7238, T^5, ..., T^{10}] = 127295.$

Comparing this two expected cost we find the decision $d = 2$ is better and send

all the remaining traffic to network provider 2. In fact with provider costs $c_1 = 10$ and $c_2 = 12$, when $\hat{T}^1, \hat{T}^2$ are high enough (say greater than 13000) and $\hat{T}^3, \hat{T}^4$ are relatively low (say less than 8000), it might be better to allocate all the remaining traffic to the expensive network provider.

In theory, the ARP improves the TMRP by comparing the expected future cost to make routing decision thus should cost less than the TMRP. However, we cannot evaluate the cost of the ARP analytically. In Section 2.3.4 we evaluate the ARP on $1,000,000$ random traffic scenarios (Instance 2 with $c_2 = 12$) and find (in a format of '**Mean Cost±Standard Deviation**')

$$Cost_{\mathbf{ARP}} = 113351.84 \pm 12.08,$$

where on the same set of scenarios the TMRP gives a mean cost 113352.46 with the standard deviation $\sigma = 12.07$. Note that the mean cost for TMRP is slightly different from the analytical one we found in (2.3.8) (where $Cost_{\mathbf{TMRP}} = 113333.33$), because it is a sample estimation of the latter.

2.3.4 Numerical results

In last section we have presented several simple routing policies of TpTRP and also the theoretical expected cost of applying two of them on an example with uniform i.i.d. traffic (same as Instance 2 listed below). In this section however, we will test these plausible routing policies on a group of $1,000,000$ random scenarios, give the mean costs of implementing them in practice. As mentioned in Section 2.3.2, the ARP is only applicable in the case where traffic volumes for different time intervals are independent and identically distributed. Therefore we consider the following two instances listed in Table 2.1. Note that here we only picked out the instances with i.d.d. traffic distributions, while the full instances list can be found in Section 4.4.1.

Instance	**Parameters**			**Stochastic Information**	
	$\lvert\Gamma\rvert$	θ	**n**	**distribution**	**time dependency**
Instance 2	10	3	2	$U(6000, 14000)$	i.i.d.
Instance 4	10	3	2	truncated $N(10000, 10^6)$	i.i.d.

Table 2.1: List of TpTRP Instances

As shown in Table 2.1, in these two TpTRP instances we have 2 network providers to choose from, both of which divide the charging period into 10 time intervals and charge on the 3rd highest volume of traffic with per unit cost $c_1 = 10, c_2 = 11, 12$ or 15. These two instances differ by the assumptions made on the random traffic. In Instance 2 the traffic in every period follows the same uniform distribution ($U(6000, 14000)$) while in Instance 4 it follow the truncated normal distribution ($N(10000, 10^6)$) in which traffic outside the 99.7% ($\pm 3\sigma$) confidence region is projected onto the boundary of the region to avoid negative traffic volumes.

For each TpTRP instance, we generate $1,000,000$ random scenarios according to its traffic distribution and apply all the routing policies we discussed in Section 2.3 to them. We summarise the numerical test result in Table 2.2. The entries in this table is given in a format of '**Mean Cost±Standard Deviation**'.

Instance	c_2	**SRP**	**TMRP**	**ARP**	**DRP**
Instance 2	11	118193.04±10.32	113352.46±12.07	113138.24±12.27	103659.85±11.54
	12	118193.04±10.32	113352.46±12.07	113351.84±12.08	103659.85±11.54
	15	118193.04±10.32	113352.46±12.07	113352.46±12.07	103659.85±11.54
Instance 4	11	129490.72±13.99	122749.70±20.28	122645.30±20.72	107256.27±26.01
	12	129490.72±13.99	122749.70±20.28	122749.70±20.28	107256.27±26.01
	15	129490.72±13.99	122749.70±20.28	122749.70±20.28	107256.27±26.01

Table 2.2: Numerical result (mean cost and standard deviation) of implementing simple routing policies on $1,000,000$ scenarios

We considered three different choices of the per unit cost for network provider 2 ($c_2 = 11, 12$ or 15), where $c_1 = 10$. From Table 2.2 we can see, the inequality SRP $\geq$ TMRP $\geq$ ARP $\geq$ DRP holds for all choices of c_2. Apart from the ARP, the mean cost of all the other routing policies does not vary with the change of c_2, because they use no more than the free time intervals of network provider 2. Moreover, in case where $c_2 = 11$, the ARP performs definitely better than the TMRP by using network provider 2 to send several scenarios' remaining traffic. For example for Instance 2 ($c_2 = 11$), there are $75,932$ out of $1,000,000$ scenarios using network provider 2 to send the remaining traffic, which leads to a definitely lower mean cost for the ARP than TMRP. However, in case where $c_2 = 12$ the difference between these two routing policies is not significant as the standard deviation is greater than the difference in mean cost (where only $1,122$ out of $1,000,000$ scenarios use network provider 2 to send the remaining traffic). While when $c_2 = 15$, it makes no benefit to use provider 2 since it costs too much, thus the ARP works completely the same as TMRP.

In addition, comparing the mean cost for Instance 2 given in Table 2.2 with the theoretical expected cost of applying those simple routing policies (given in section 2.3.3), we see that in all cases, the theoretically obtained value is within $\pm 2\sigma$ of the result obtained by simulation, confirming our analysis.

Though we assumed identical traffic distribution throughout this section, we have to point out that this assumption is not necessary for the application of SRP and TMRP. Actually neither the implementation of SRP nor TMRP depends on the traffic distribution. The numerical results of implementing them on non-identical distributions are shown in Section 4.4.2. On the other hand, it is not trivial to extend the application of ARP to non-identical distributions. This is because in ARP we make routing decision by comparing the expected future cost (the expectation of the θ-th highest volume of traffic at the end) under the current usage of network providers, which is no longer a direct use of binomial distribution under non-identical distributions.

Chapter 3

Multi-stage Mixed-integer Stochastic Programming Model

3.1 Introduction to stochastic programming

Real world decision making problems almost invariably include some unknown parameters. A few of them can be simplified into deterministic problems, whereas many others require decisions to be made under uncertainty. Stochastic Programming (SP) is a framework for modelling optimization problem that involve uncertainty. Generally speaking, in an SP model the probabilistic distribution which governs the uncertain data is known or can be estimated beforehand. The goal is thus to find the optimal decision which maximize/minimize the expectation of the future profit/cost over this randomness.

As an example, consider a two-stage stochastic program (Schultz et al. [50]):

$$\min\{f_1(x) + E_\mu \varphi(x, \omega) : x \in X\}. \tag{3.1.1}$$

One interpretation of this formula is that it presents a decision with possible recourse action. At the first stage a decision x is to be selected from a set X of feasible solutions. This decision x gives rise directly to costs $f_1(x)$, $f_1 : X \mapsto \mathbb{R}$, and in the second stage to costs that are involved if a recourse action on the decision is taken. Besides x this recourse action depends on the realisation of

initially unknown data that are represented by the random object ω, element of some probability space $(\Omega, \mathcal{A}, \mu)$ where $\Omega, \mathcal{A}$ and μ, in turn, represent the sample space, the set of random events (each event is a a subset of the sample space) and the probability measure. For a realisation ω the recourse cost is denoted by the function $\varphi : X \times \Omega \mapsto \mathbb{R}$ of x and ω, and hence a random object itself. Since it becomes realised only after the decision x is made we include it in the total cost function by taking its expected value $E_\mu \varphi(x, \omega)$, which is a (non-random) function of x, depending on the probability measure μ.

The recourse action again asks for an optimal decision that is formulated as:

$$\varphi(x, \omega) = \min\{f_2(y, \omega) : y \in Y(x, \omega)\}. \tag{3.1.2}$$

Here, y represents the recourse action to be selected from the set of feasible recourse actions $Y(x, \omega)$. As indicated by the notation this set is dependent on x and ω. The cost of the action y may also depend on ω and is therefore denoted as $f_2(y, \omega)$, $f_2 : Y \times \Omega \mapsto \mathbb{R}$.

There are several types of recourse, in which a typical case is the fixed recourse model with complete recourse. We take the linear SP for example. Consider the following recourse problem:

$$\varphi(x, \omega) = \min_y\{q(\omega)y : W(\omega)y = h(\omega) - T(\omega)x, y \in \mathbb{R}_+\}. \tag{3.1.3}$$

In this model, all the matrices $W(\omega)$, $T(\omega)$ and $h(\omega)$ which defining the recourse model are random depending on ω. This model is a fixed recourse model if the recourse matrix $W(\omega)$ does not depend on ω, i.e. $W(\omega) = W$. While complete recourse means the columns of W (or $W(\omega)$) span $\mathbb{R}^m$, namely for $\forall z \in \mathbb{R}^m, \exists y \in \mathbb{R}^p$ satisfying $Wy = z$. In particular if the recourse problem is feasible for all scenarios and all choices of first stage decision x, we say we have a relatively complete recourse.

Random parameters in a stochastic programming model can be either discretely or continuously distributed. No matter in which case, the evaluation of the expected value function φ is proved to be NP-hard by Dyer and Stougie [23]. Nevertheless, a common practice is to approximate continuous distributions with

discrete ones, which allows the expected recourse function to be calculated as a simple weighted sum. Results in Schultz [49] show that, under mild conditions, it is possible to approximate solutions of stochastic programs with continuous distributions within any given accuracy by discrete distributions.

Stochastic programming has applications in a broad range of areas (see Wallace and Ziemba [57]) since its first appearance in Beale [8] and Dantzig [21]. An extension to the framework is Multi-stage Stochastic Programmes (MSP), which provide a framework within which to model problems which require decisions to be made regularly at various intervals over a particular time horizon using uncertain information. They capture both the dynamic and the uncertain nature of the planning problem. The use of MSP as a tool for decision-making has increased over the last decade. Not surprisingly, we can find its application in many varied fields, e.g. capacity planning [31], network revenue management [22] and [40], forest management [13], chemical engineering [26] and [37].

Generally speaking, as in MSP the scenario tree often grows exponentially with the number of periods, the problems are always very large and are often too large to solve by direct methods such as the simplex algorithm. This is due to either the limitations of the computer's memory, or because these problems take too long to solve on even relatively fast computers. However, they have a significant amount of exploitable structures which are critical in developing solution algorithms, e.g. based on decomposition.

For example, the L-shaped method of Van Slyke and Wets [55] is an iterative method for solving multi-stage stochastic linear programmes based on Benders Decomposition (Benders [10]), which takes advantage of the block-angular structure of the problem to decompose it into its natural sub-problems. Sub-problems are solved individually using a linear programing solver, e.g. the simplex method. This decomposition is not computationally expensive as sub-problems are relatively much smaller with only one scenario. In addition to this, the decomposition allows the application of parallel computing where sub-problems can be solved independently. Thus the Benders Decomposition allows otherwise intractable problems to be solved in reasonable time with less effort.

On another hand, Stochastic Mixed-Integer Programmes (SMIP) are problems in which integral constraints are imposed on (some of) the first-stage and/or the recourse. It has wide applications in industry, while the introduction of integer

variables brings another type of difficulty on top of the difficulties encountered in stochastic linear programming. Generally speaking, integer programming is NP-hard (see Dyer and Stougie [23]), where polynomial time algorithms for its solution do not exist. In particular, adding integral restrictions to the last stage problem significantly increases the complexity of the problem. The main difficulty in solving stochastic integer programs is that, when integral restrictions are present in the final stage, the final stage value function is not necessarily convex about previous stage variables but only lower semi-continuous (see Blair and Jeroslow [12] and Schultz [49]). Thus, most standard decomposition approaches that work by building a precise convex under-estimator of the value function break down when final stage integer variables are present. Unfortunately, in the following section we can see that the stochastic programming model for TpTRP is a SMIP, on which a general solution algorithm based on Benders Decomposition is not directly applicable.

3.2 SMIP model for TpTRP

Now we build the stochastic programming model for the TpTRP. Note that as a beginning, in this section we build a SP model which can approximate the stochastic TpTRP as precisely as possible, ignoring for the moment the question of solvability of the resulting SP model.

3.2.1 Stages

As mentioned previously in assumption (a4), in TpTRP all network providers use the same division of time intervals. Thus the stage of this problem can be defined naturally by the time intervals. At the beginning of every time interval, providing all the previous realisations of traffic and decisions we have made, a routing decision has to be made on how to allocate this time interval's traffic among network providers. Apart from these, an additional stage is added in the end to calculate the cost charged on the ISP under top-percentile pricing. Namely we have a SP model with $(|\Gamma| + 1)$ stages.

3.2.2 Decisions

As discussed in Section 2.1.3, feasible decision for TpTRP should be both independent of future traffic realisations and implementable without knowing the whole volume of current traffic. In the SP model we set the decision $d^\tau = x^\tau = \{x_i^\tau | i = 1, ..., n; \sum_{i \in I} x_i^\tau = 1\}$, where x_i^τ is a fractional decision, namely under decision x_i^τ the volume of traffic sent to network i is $x_i^\tau T^\tau$.

3.2.3 States

Let $\hat{\omega}_\tau = (\hat{\omega}^1, \hat{\omega}^2 ..., \hat{\omega}^\tau)$ be the information available at time stage τ, i.e. the realisations of the random events $(\omega^1, \omega^2 ..., \omega^\tau)$. Thus the traffic realisation is $(\hat{T}^1, \hat{T}^2 ..., \hat{T}^\tau) = (T^1(\hat{\omega}^1), T^2(\hat{\omega}^2) ..., T^\tau(\hat{\omega}^\tau))$. Further denote by $x_\tau = (x^1, x^2, ..., x^\tau)$ the history of routing decisions up to time stage τ. Thus the state which includes all the information available up to now, is given by $(x_\tau, \hat{\omega}_\tau)$. Let $Q^\tau(x_\tau, \hat{\omega}_\tau)$ denote the value function at time stage τ, i.e. the expected cost of being at time stage τ with decision history x_τ under random event $\hat{\omega}_\tau$.

3.2.4 Formula

Now we have established all the fundamental elements, the SP model can therefore be built. During the first $|\Gamma|$ time stages, we make routing decision on how to allocate every time interval's traffic among network providers. Thus for stages $\tau = 0, ..., |\Gamma| - 1$, the TpTRP can be stated in recourse form as:

$$\begin{aligned} Q^\tau(x_\tau, \hat{\omega}_\tau) = \min_{x^{\tau+1}} \quad & \mathbb{E}_{\omega^{\tau+1}}(Q^{\tau+1}(x_{\tau+1}, \hat{\omega}_{\tau+1})) \\ s.t.: \quad & \sum_{i \in I} x_i^{\tau+1} = 1, \\ & 0 \leq x_i^{\tau+1} \leq 1, \ \forall i \in I \end{aligned} \quad . \tag{3.2.1}$$

For terminal conditions, given all the decisions up to time stage $|\Gamma|$, an additional stage $|\Gamma|$ is added to calculate the cost implied by the decision set $x_{|\Gamma|} = (x^1, x^2, ..., x^{|\Gamma|})$ under top-percentile pricing policy. As the cost charged

on the ISP is based on the θ-th highest volume of traffic we send to it, in the final stage we need to introduce binary variables $z_i^{\tau,\hat{\omega}_{|\Gamma|}}$ to distinguish whether a particular time interval τ is in the top θ time intervals of network provider i, under the realisations $\hat{\omega}_{|\Gamma|} = (\hat{\omega}^1, \hat{\omega}^2, ..., \hat{\omega}^{|\Gamma|})$:

$$z_i^{\tau,\hat{\omega}_{|\Gamma|}} = \begin{cases} 1, & \text{if } \tau \text{ is in the top } q\text{-percentile time intervals} \\ 0, & \text{otherwise} \end{cases}$$

With the help of these binary variables we can find the volume of the top-percentile traffic $y_i^{\hat{\omega}_{|\Gamma|}}$ for network provider i, thus the cost charged on the ISP. For stage $\tau = |\Gamma|$,

$$\begin{aligned} Q^{|\Gamma|}(x_{|\Gamma|}, \hat{\omega}_{|\Gamma|}) = \min_{y,z} \quad & \sum_{i \in I} c_i y_i^{\hat{\omega}_{|\Gamma|}} \\ s.t.: \quad & x_i^\tau \hat{T}^\tau - y_i^{\hat{\omega}_{|\Gamma|}} \leq z_i^{\tau,\hat{\omega}_{|\Gamma|}} \hat{T}^\tau, && \forall i \in I, \forall \tau \in \Gamma \\ & \sum_{\tau \in \Gamma} z_i^{\tau,\hat{\omega}_{|\Gamma|}} < \theta, && \forall i \in I \\ & y_i^{\hat{\omega}_{|\Gamma|}} \geq 0, && \forall i \in I \\ & z_i^{\tau,\hat{\omega}_{|\Gamma|}} \in \{0,1\}, && \forall i \in I, \forall \tau \in \Gamma \end{aligned} \quad . \quad (3.2.2)$$

From the above discussion we can see that the final stage problem is just determining the volume of top q-percentile traffic for every network provider, i.e. the θ-th highest volume of $\{x_i^1 \hat{T}^1, x_i^2 \hat{T}^2, ..., x_i^{|\Gamma|} \hat{T}^{|\Gamma|}\}$ for network provider i. Although this step seems easy conceptually, we cannot find a better way than using integer variables to model it. Moreover, since the θ-th highest volume of $\{x_i^1 \hat{T}^1, x_i^2 \hat{T}^2, ..., x_i^{|\Gamma|} \hat{T}^{|\Gamma|}\}$ is non-convex in its arguments, no matter how the final stage problem is modelled we cannot avoid the difficulty of dealing with non-convex sub-problems in the solution step.

3.2.5 Deterministic equivalent

In conclusion, we have built a $(|\Gamma| + 1)$-stage mixed-integer stochastic programming model with binary requirement only on some of the final stage variables.

Given a discrete set of random realisations $\hat{\Omega}^\tau, \tau = 1, ..., |\Gamma|$, the deterministic equivalent of this multi-stage stochastic model is:

$$
\begin{aligned}
obj = \min_{x^1} \quad & \mathbb{E}_{\omega^1 \in \hat{\Omega}^1}[\min_{x^2} \quad \mathbb{E}_{\omega^2 \in \hat{\Omega}^2}[...[\min_{x^{|\Gamma|}} \quad \mathbb{E}_{\omega^{|\Gamma|} \in \hat{\Omega}^{|\Gamma|}}(\min_{y,z} \sum_{i \in I} c_i y_i^{\hat{\omega}_{|\Gamma|}})]...]] \\
s.t. : & \sum_{i \in I} x_i^{\tau, \hat{\omega}_\tau} = 1, && \forall \tau \in \Gamma, \forall \hat{\omega}_\tau \in \hat{\Omega}^1 \times ... \times \hat{\Omega}^\tau \\
& 0 \leq x_i^{\tau, \hat{\omega}_\tau} \leq 1, && \forall i \in I, \forall \tau \in \Gamma, \forall \hat{\omega}_\tau \in \hat{\Omega}^1 \times ... \times \hat{\Omega}^\tau \\
& x_i^{\tau, \hat{\omega}_\tau} \hat{T}^\tau - y_i^{\hat{\omega}_{|\Gamma|}} \leq z_i^{\tau, \hat{\omega}_{|\Gamma|}} \hat{T}^\tau, && \forall i \in I, \forall \tau \in \Gamma, \forall \hat{\omega}_{|\Gamma|} \in \hat{\Omega}^1 \times ... \times \hat{\Omega}^{|\Gamma|} \\
& \sum_{\tau \in \Gamma} z_i^{\tau, \hat{\omega}_{|\Gamma|}} < \theta, && \forall i \in I, \quad \forall \hat{\omega}_{|\Gamma|} \in \hat{\Omega}^1 \times ... \times \hat{\Omega}^{|\Gamma|} \\
& y_i^{\hat{\omega}_{|\Gamma|}} \geq 0, && \forall i \in I, \quad \forall \hat{\omega}_{|\Gamma|} \in \hat{\Omega}^1 \times ... \times \hat{\Omega}^{|\Gamma|} \\
& z_i^{\tau, \hat{\omega}_{|\Gamma|}} \in \{0, 1\}, && \forall i \in I, \forall \tau \in \Gamma, \forall \hat{\omega}_{|\Gamma|} \in \hat{\Omega}^1 \times ... \times \hat{\Omega}^{|\Gamma|}
\end{aligned}
\tag{3.2.3}
$$

This is a standard mixed-integer stochastic programming problem. If we assume there are K discrete realisations per stage ($K = |\hat{\Omega}^\tau|, \tau = 1, ..., |\Gamma|$), then the problem has $(\sum_{\tau=1}^{|\Gamma|} K^{\tau-1} + n \cdot K^{|\Gamma|} + n \cdot |\Gamma| \cdot K^{|\Gamma|})$ constraints and $n \cdot (\sum_{\tau=1}^{|\Gamma|} K^{\tau-1} + K^{|\Gamma|} + |\Gamma| \cdot K^{|\Gamma|})$ variables of which $n \cdot |\Gamma| \cdot K^{|\Gamma|}$ are integer.

3.3 Solving by CPLEX – Numerical results

The simplest approach to solve the problem is to consider its deterministic equivalent as a large scale monolithic mixed-integer linear program and to apply a commercial standard solver, e.g. CPLEX, to it.

3.3.1 Examples with time-dependent traffic distribution

Let us firstly have a look at two extremely small instances of the TpTRP, all of which use time-dependent structure for the traffic distributions. The basic parameters for these instances are shown in Table 3.1.

Instance	Parameters			Stochastic Information	
	$\|\Gamma\|$	θ	n	traffic realisations	probability
Instance 0(a)	3	2	2	{8000, 10000, 12000}	0.5
				{10000, 8000, 10000}	0.5
Instance 0(b)	5	2	2	{14000, 6000, 10000, 8000, 12000}	0.3333
				{10000, 10000, 6000, 10000,10000}	0.3333
				{6000, 14000, 14000, 12000, 8000}	0.3333

Table 3.1: Basic information for Instances 0 of TpTRP

In these instances we assume that we have 2 potential network providers to choose from, both of them divide the charging period into 3 time intervals in Instance 0(a) or 5 in Instance 0(b). Costs are computed based on the 2nd highest volume of traffic ($\theta = 2$) by both providers. We assume that there are only two (in Instance 0(a)) or three (in Instance 0(b)) potential scenarios, each with 1/2 or 1/3 of possibility to happen. Note that here we use a time-dependent traffic distribution structure in the definition of both instances, in which all the remaining realisations are known when the first traffic volume is revealed.

Due to the time-dependent structure of the stochastic information, we model this problem into a two-stage mixed-integer stochastic programming problem, in which we make routing decisions on all time intervals' traffic in the first stage and compute the cost in the second stage. As shown in Table 3.2, this gives rise to a SMIP model with 46 variables (30 of which are restricted to be integer) and 41 constraints in its deterministic equivalent for Instance 0(b).

Instance	No. of variables				No. of constraints	
	First stage		Second stage		First stage	Second stage
	Continuous	Integer	Continuous	Integer		
Instance 0(a)	6	0	4	12	3	16
Instance 0(b)	10	0	6	30	5	36

Table 3.2: Size information of the SMIP model for Instances 0 of TpTRP

Solving this problem by CPLEX gives routing policies shown in Table 3.3. For each instance we consider three different choices of the per unit cost for network provider 2, i.e. $c_2 = 11, 12$ or 15 (while $c_1 = 10$) as given in the second column. Optimal decisions obtained by solving the SMIP model are named SMIPRP, given in the form of $x_i = (x_i^1, x_i^2, ..., x_i^{|\Gamma|})^T$, where x_i^τ means we send $x_i^\tau T^\tau$ to network i. We can see that the optimal routing policy given by the SMIP model changes with the difference between per unit cost of network providers. The less the price

Instance	c_2	SMIPRP	CPLEX running information		
			no. of iterations	no. of nodes	time
Instance 0(a)	11	$x_1 = (1.00, 1.00, 0.6667)^T$ $x_2 = (0.00, 0.00, 0.3333)^T$	97	28	0.01s
	12	$x_1 = (1.00, 1.00, 0.6667)^T$ $x_2 = (0.00, 0.00, 0.3333)^T$	92	29	0.01s
	15	$x_1 = (1.00, 1.00, 0.00)^T$ $x_2 = (0.00, 0.00, 1.00)^T$	102	25	0.01s
Instance 0(b)	11	$x_1 = (0.44, 0.76, 0.62, 0.72, 0.72)^T$ $x_2 = (0.56, 0.24, 0.38, 0.28, 0.28)^T$	2044	322	0.12s
	12	$x_1 = (0.44, 1.00, 0.62, 0.72, 0.72)^T$ $x_2 = (0.56, 0.00, 0.38, 0.28, 0.28)^T$	3003	428	0.16s
	15	$x_1 = (1.00, 1.00, 0.00, 1.00, 1.00)^T$ $x_2 = (0.00, 0.00, 1.00, 0.00, 0.00)^T$	1939	279	0.11s

Table 3.3: Solutions given by CPLEX for Instances 0 of TpTRP

difference is, the more data we send to the expensive network provider. Columns 4-6 of Table 3.3 show the running information given by CPLEX, which in turn are the number of Branch-and-Cut iterations required, the number of branch nodes generated and the total running time of solving the SMIP model to optimality.

Instance	c_2	SRP	TMRP	SMIPRP	DRP
Instance 0(a)	11			80000.00	
	12	100000.00	80000.00	80000.00	80000.00
	15			80000.00	
Instance 0(b)	11			109979.20	
	12	120000.00	113333.33	113302.39	106666.67
	15			113333.33	

Table 3.4: Numerical result of implementing SMIPRP routing policies

Numerical results of comparing the SMIP routing policy with the TMRP (Trivial Multi-homing Routing Policy) we developed in Section 2.3 are summarised in Table 3.4. Instance 0(a) is an extreme case where the TMRP coincide with the DRP, thus the SMIPRP cannot perform better than TMRP with any c_2 value. However from Instance 0(b), we can see that the SMIP model provides us a better routing policy than the TMRP when $c_2 = 11$ and $c_2 = 12$. This justifies the fact that a better routing policy than TMRP exists in the stochastic case. Our goal is then seeking for it for larger sized TpTRP.

3.3.2 Examples with time-independent traffic distribution

On the other hand, in case where the realisation of traffic is independent among time intervals (which is actually the case in real world application as well), the TpTRP becomes too large to be tractable very easily. In the following we examine two fairly small such instances of the TpTRP with 5 periods. Unfortunately, the size and the integer nature of these problems prevent the efficient use of standard commercial software such as CPLEX.

Instance	Parameters			Stochastic Information	
	$\|\Gamma\|$	θ	n	distribution	probability
Instance 1(a)	5	2	2	$T^1, T^2, T^3 \in \{6000, 14000\}$	{0.5, 0.5}
				$T^4 \in \{7000, 13000\}$	{0.5, 0.5}
				$T^3 \in \{8000, 12500\}$	{0.5, 0.5}
Instance 1(b)	5	2	2	$T^1, ..., T^5 \in \{8000, 10000, 12000\}$	{0.3333, 0.3333, 0.3333}

Table 3.5: Basic information for Instances 1 of TpTRP

In Table 3.5, both instances have 2 network providers to choose from, who divide the charging period into 5 time intervals and charge based on the 2nd highest volume of traffic. Each time period has 2 (in Instance 1(a)) or 3 (in Instance 1(b)) potential realisations of traffic volume. Traffic distribution varies with time in Instance 1(a) while we use the same distribution for all time intervals in Instance 1(b).

Like what we did in last section, here we make a simplification on the decision structure as well. We combine all the $|\Gamma|$ stages together, making all time intervals' decisions within the first time stage, and then compute the cost in the second stage. Although this simplification does not reduce the number of scenarios to examine, it simplifies the structure of the stochastic tree, thus reduces both the number of variables and the number of constraints in the deterministic equivalent.

Instance	No. of variables				No. of constraints	
	First stage		Second stage		First stage	Second stage
	Continuous	Integer	Continuous	Integer		
Instance 1(a)	10	0	64	320	5	384
Instance 1(b)	10	0	486	2430	5	2916

Table 3.6: Size information of the SMIP model for Instances 1 of TpTRP

Table 3.6 shows the number of variables and constraints in detail, in the deterministic equivalent of the two-stage SIP model for Instance 1(a) and Instance 1(b). It is no doubt that the size of the SIP model is large, comparing with the size of the corresponding TpTRP instance. For example, in the deterministic equivalent of Instance 1(b), there are 2921 constraints, 10 first stage variables and 2916 second stage variables in which 2430 are restricted to be binary.

The running information and the numerical results are summarised below in Table 3.7 and Table 3.8. Note that here we use $c_2 = 12$ in these experiments.

Instance	**Best integer solution**		**Statistics when terminate**		
	Decisions	**Time to find**	**Running time**	**Tree size**	**Final gap**
Instance 1(a)	$x_1 = (1.00, 1.00, 1.00, 1.00, 0.00)^T$ $x_2 = (0.00, 0.00, 0.00, 0.00, 1.00)^T$	29.31s	5712.64s	276.63MB	24.58%
Instance 1(b)	$x_1 = (0.3637, 0.00, 0.00, 0.00, 0.00)^T$ $x_2 = (0.6363, 1.00, 1.00, 1.00, 1.00)^T$	32.88s	74793.46s	1543.50MB	37.56%

Table 3.7: Solutions given by CPLEX for Instances 1 of TpTRP

Instance	**SRP**	**TMRP**	**SMIPRP**	**DRP**
Instance 1(a)	124375.00	115000.00	115000.00	100937.50
Instance 1(b)	99876.54	95925.93	115400.76	90000.00

Table 3.8: Numerical result of implementing SMIPRP routing policies

Actually, neither of these two instances can be solved to optimality directly by CPLEX. We observe a very slow improvement on the optimal value of the current best linear relaxation. To overcome this, we terminate the solving process after numerically examining $1,000,000$ nodes for each instance, and treat the current best integer feasible solution as the SMIPRP provided by CPLEX. For example in Instance 1(a), although CPLEX can give the TMRP after 29.31 seconds (where the TMRP is not optimal, a better routing policy for this instance is given in Table 3.12), it has not improved on the TMRP by the time when the solving process terminated after 5712.64 seconds, when the Branch-and-Cut tree size is as large as $276.63MB$ but the gap between the current best integer solution and the linear relaxation is still 24.58%. The situation is even worse for Instance 1(b). The best feasible integer solution provided by CPLEX sends most of the time intervals' traffic to the expensive network provider, which leads to an objective value even higher than the SRP (Single-homing routing policy as stated in Section 2.3.2). This solution is not improved further in 74793.46 seconds, after numerically examining $1,000,000$ nodes.

Although Instance 1(a) and Instance 1(b) are fairly small instances of the TpTRP, their deterministic equivalents are too hard to be solved by commercial solver such as CPLEX as a MILP. In fact, SMIPs are well known for being challenging both from theoretical and computational points of view, because they can lead to very large scale problems with a large number of outcomes of the random parameters (Birge and Louveaux [11]). However, the difficulties we are facing are much more challenging than this, as our program is a stochastic program with integral constraints in the final stage. In the next section, we attempt to develop particular decomposition based solution method to solve the SMIP of the TpTRP.

3.4 Review on solution methods for SMIP

In this section we review solution methods which are developed to solve stochastic mixed-integer programming problems. The discussion mainly follows the investigation in surveys Haneveld and Van Der Vlerk [29], and Schultz et al. [50] about SMIP solution algorithms. Most of the algorithms heavily exploit the special structure of the SMIP (such as block-angular structure) and typically work only for problems with (fixed) complete recourse, and with stochastic parameters that have discrete distributions.

3.4.1 Benders Decomposition and L-shaped based methods

There are several types of stochastic mixed-integer programmes, whose difficulty varies. The easiest case is to impose the integer restriction only on the first-stage variables, as in this case we actually obtain a problem in which the recourse is convex. Given that the recourse problem is continuous and convex, all the advantageous properties which are held for the stochastic linear programming problem are maintained. Therefore one could connect any integer programming solution methods to any of the solution methods that were successful in dealing with the convex recourse functions from stochastic linear programming, i.e. use a IP solver to solve the Benders master problem, while Benders Decomposition is applied to deal with the uncertainty. This was first done by Wollmer [60] for

models with 0-1 first-stage variables and continuous second-stage problems. Such a method, as discussed in the survey Schultz et al. [50], though not efficient in a complexity theoretical sense, at least provides a conceptual approach to solving the problem.

For example, an algorithm which adapts the L-shaped method to deal with the recourse information and branch-and-bound to impose integer optimality is developed in Laporte and Louveaux [35], which extends Wollmer [60] to SMIPs having binary first-stage variables and arbitrary (but easily computed) second stage problems. This method introduces optimality and feasibility cuts on the value of the recourse function and combine this with Branch-and-Bound to develop a Branch-and-Cut method.

On the other hand, in cases when integral constraints are imposed on the recourse variables, the problem becomes much more unpleasant. The development of generally implementable solution algorithm becomes non-trivial due to the non-continuous and non-convex properties of the recourse function. Solution methods that were successful for stochastic linear programming cannot be simply adapted to solve this type of problems. Although some solution strategies have been developed for specific applications of SMIPs, relatively few technique have been developed to solve general SMIPs. Indeed, most of these methods are based on the duality theory for integer programming (Nemhauser and Wolsey [42]), which makes the algorithm work theoretically but be very limited practically. For example, a L-shaped method is generalised to solve the SMIP in Carøe and Tind [16], where nonlinear feasibility and optimality cuts are determined via general duality theory.

Disjunctive Programming and/or Lift-and-Project cutting plane (Balas et al. [4, 6]) are robust techniques for solving mixed-integer systems, which generate linear programming cuts iteratively to approximate the convexification of the feasible set. Carøe and Tind [15] developed a cutting plane method which takes advantage of the dual block-angular structure of stochastic systems to solve general two-stage SMIP. This approach was shown to be equivalent to computing a hull relaxation in the context of disjunctive programming, and solving this via the lift-and-project cutting plane technique of Balas et al. [4]. Cuts derived for one sub-problem can be lifted to derive valid inequalities for other sub-problems. However, in order to preserve facetial properties in this lifting process, a separate linear program needs to be solved. Apart from this, the sharply increasing number of cuts with

iterations prevents the large sized problem being solved solely by this approach.

In recent years, Benders Decomposition has been combined with cutting plane methods to solve SMIPs in which either the first-stage or the recourse problem possess integer variables. For example, Sherali and Fraticelli [54] develop a sequence of relaxations for 0-1 mixed-integer second stage problems, which were incorporated within a Benders Decomposition algorithm for solving two-stage SMIPs. Convergence of the method is guaranteed in case the first-stage variables are binary. Similarly, in Sen and Higle [51] disjunctive programming is used to develop set convexification in a sequential manner, which is theoretically equivalent to Sherali and Fraticelli [54] though their algorithmic schemes are different.

Branch-and-Cut algorithms provide one of the most successful approaches for deterministic mixed-integer programming. To extend it to SMIP, the main challenge is the need to combine decomposition-based methods (Ruszczyński [48]) that work well in the stochastic programming area with Branch-and-Cut methods that work well for mixed-integer programming. For example, Sen and Sherali [52] developed a Disjunctive Decomposition-based Branch-and-Cut ($D^2 - BAC$) algorithm to accommodate SMIP in which both first and second stages have integer variables. The advantage of this method is that it allows the SMIP to be solved by dividing the large problem into smaller MIP sub-problems which can be solved in parallel.

3.4.2 Scenario Decomposition based methods and Progressive Hedging

Beside Benders Decomposition, Scenario Decomposition is also popularly applied in solving large scale stochastic programming problems. Haneveld and Van Der Vlerk [29] argued that, Scenario Decomposition is in a sense "dual" to the L-shaped algorithms discussed above, as the L-shaped algorithms decompose the problem by time stages and operate by searching for increasingly better first-stage solutions, the scenario decomposition based algorithms consider sub-problems corresponding to scenarios and are governed by finding good dual multipliers.

For example, Carøe and Schultz [14] use scenario decomposition and Lagrangian relaxation within a branch-and-bound framework to solve two-stage stochastic IPs with mixed-integer variables in both stages. As indicated by the authors,

the method can readily be extended to multi-stage problems of the same type. With this method, scenario problems in which the non-anticipativity constraints are relaxed are solved individually to integer optimality, then possible first-stage solutions are constructed from these scenario solutions by taking the weighted average combined with a rounding heuristic in order to satisfy integer restrictions.

In addition to this, the Progressive Hedging algorithm (Rockafellar and Wets [47]) is another type of algorithm based on Scenario Decomposition, which is guaranteed to converge for convex non-integer problem. The application of Progressive Hedging to multi-stage mixed-integer 0-1 problems is first seen in Løkketangen and Woodruff [38], in which the authors propose using Progressive Hedging in combination with Tabu search to solve the quadratic mixed-integer 0-1 scenario problems. Although a formal justification on convergence is lacking for mixed-integer problems, they observe convergence to an implementable solution for several test problems.

3.5 Solving a simplification of TpTRP by cutting plane method

As addressed in the previous sections, the SMIP model of the TpTRP is hard to solve. Solution algorithms developed for stochastic programs based on decomposition are required to solve non-convex sub-problems. On the other hand, algorithms working well for MIP generally become inefficient in the stochastic case, due to the large size of the problem. To solve this problem, we need to investigate how to exploit the special structure of the stochastic system to make the solution process as efficient as possible.

As we discussed in the beginning of Section 3.4, the SMIP model with fixed and complete recourse is one type of problem on which more structural properties can be exploited, e.g. one can easily share cuts among scenarios with fixed recourse problem. Unfortunately, the current SMIP model given by formula (3.2.1) and (3.2.2) has a complete recourse but not fixed, where the recourse matrix $\tilde{W}(\omega)$ for one network provider is defined by

$$\tilde{W}(\omega) = \begin{pmatrix} -1 & -T^1(\omega^1) & & & \\ -1 & & -T^2(\omega^2) & & \\ \vdots & & & \ddots & \\ -1 & & & & -T^\tau(\omega^{|\Gamma|}) \\ & -1 & -1 & \cdots & -1 \end{pmatrix}.$$

Note that this is the case for one network provider. In the case of multiple network providers $W(\omega) = (\tilde{W}(\omega), ..., \tilde{W}(\omega))^T$, e.g. an example of $W(\omega)$ with 2 networks is given on page 45.

In this work we can easily modify $\tilde{W}(\omega)$ into fixed recourse, by replacing $T^1(\omega^1), ..., T^\tau(\omega^{|\Gamma|})$ by a term $T_{max} = \max\{T^1(\omega^1), ..., T^\tau(\omega^{|\Gamma|}\}$, which gives

$$\tilde{W} = \begin{pmatrix} -1 & -T_{\max} & & & \\ -1 & & -T_{\max} & & \\ \vdots & & & \ddots & \\ -1 & & & & -T_{\max} \\ & -1 & -1 & \cdots & -1 \end{pmatrix}.$$

As the constraints recognise which time intervals are higher than the current θ-th highest volume of traffic, this modification will not change the solution of the system. In the following part of this section, we work on the SMIP with fixed recourse, develop a cutting plane based solution method to solve the small sized TpTRP instances.

3.5.1 Introduction on the cutting plane method

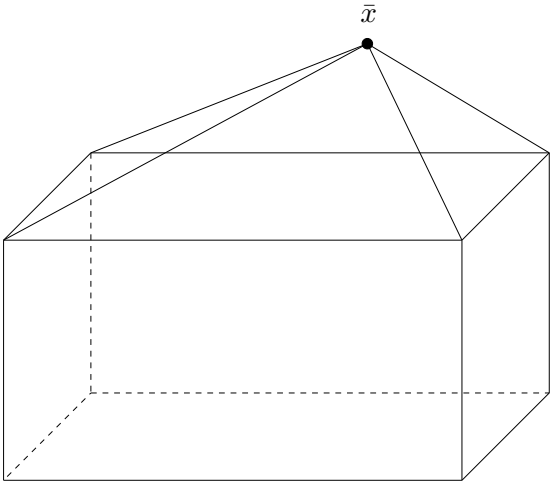
The cutting plane method is a mathematical optimization method which iteratively refine a feasible set or objective function by means of linear inequalities, termed cuts. Such procedures are popularly used to find integer solutions to mixed integer linear programming (MILP) problems, as well as to solve general, not necessarily differentiable convex optimization problems. Cutting plane methods for MILP work by solving the linear relaxation of the given integer program, where the optimum of the linear relaxation can always be found as an extreme point. If the optimum is not integer as required, we separate the optimum from

the convex hull of the integer feasible set with a linear inequality. Such a linear inequality forms a cut, which is to be added to the relaxed linear program. Then, the current non-integer solution is no longer feasible to the relaxation and we can begin the next iteration by solving the enlarged system. This process is repeated until an optimal integer solution is found.

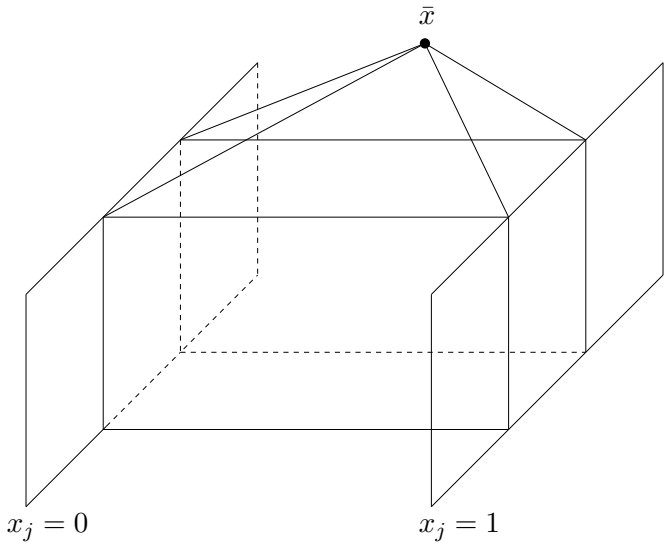
The use of cutting planes to solve MILP was introduced by Ralph E. Gomory [28], while the author's idea becomes practical many years later in mid-1990s, in combination with branch-and-cut and ways to overcome numerical instabilities. Another popular cutting planes for MILP are the Lift-and-Project cuts, based on the Disjunctive Programming approach which was first introduced by E. Balas [3]. Nowadays, Lift-and-Project cutting plane algorithms based on automatic cut generation have received increasing attention, e.g. Balas et al. [4, 5, 6] and Ceria et al. [18]. These algorithms are robust solvers for mixed integer programs and can handle a wide range of problems.

Disjunctive Programming is optimization over unions of polyhedral. While polyhedral are convex sets, their unions are not due to the presence of disjunctions. The basic idea of Disjunctive Programming and/or Lift-and-Project approach relies on the fact that, a large class of disjunctive sets, called facial, can be convexified sequentially, i.e. their convex hull can be derived by imposing the disjunctions one at a time, generating each time the convex hull of the current set. This process can be done by generating Lift-and-Project cuts iteratively.

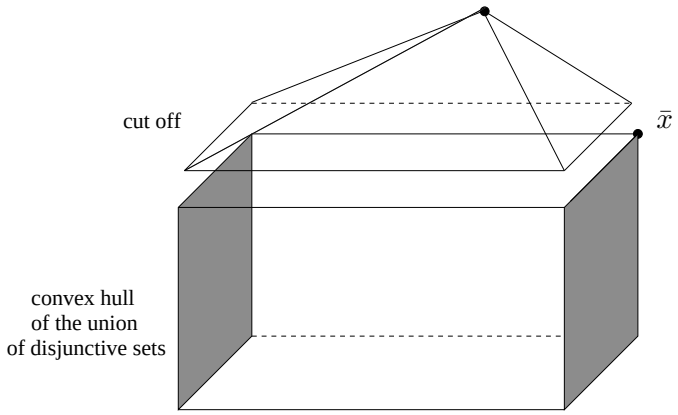
The procedure of generating the lift-and-project cuts are shown in Figure 3.5.1 [17]. For each $0-1$ variable x_j that is fractional at the linear programming optimum (Figure 3.5.1 (a)), we find its disjunction $x_j = 0 \vee x_j = 1$ (Figure 3.5.1 (b)) and then generate the convex hull of the union of disjunctive sets (Figure 3.5.1 (c)). This step cuts off the current linear programming optimum $\bar{x}$.



(a) The feasible set and the optimal 'fractional' solution $\bar{x}$



(b) The disjunction $x_j = 0 \vee x_j = 1$



(c) The convex hull of the union of the disjunctive sets and the new optimal 'fractional' solution $\bar{x}$

Figure 3.5.1: The lift-and-project cutting plane procedure

For example, consider a mixed integer system

$$\begin{array}{ll} \min & cx \\ s.t. & Ax \geq b, \\ & x \geq 0, \\ & x_i \in \{0,1\}, i = 1, ...p. \end{array} \tag{3.5.1}$$

where A is an $m \times n$ matrix and $b \in \mathbb{R}^m$. Solving its linear relaxation, we get a linear programming optimum $\bar{x}$.

If we define the polyhedra $P_{j0} := \{x \in \mathbb{R}^n_+ : Ax \geq b, x_j = 0\}$, $P_{j1} := \{x \in \mathbb{R}^n_+ : Ax \geq b, x_j = 1\}$, the convex hull of $P_j := P_{j0} \cup P_{j1}$ is the set of those $x \in \mathbb{R}^n_+$ for which there exist vectors $(y, y_0), (z, z_0) \in \mathbb{R}^{n+1}_+$ such that

$$\begin{array}{rcrcrcrcl} x & - & y & & & - & z & = & 0, \\ & & Ay & - & by_0 & & & \geq & 0, \\ & - & y_j & & & & & = & 0, \\ & & & & & & Az & - & bz_0 \geq 0, \\ & & & & & & z_j & - & z_0 = 0, \\ & & & & y_0 & & & + & z_0 = 1. \end{array} \tag{3.5.2}$$

Balas et al. [3] suggest that, a cut $\alpha x \geq \beta$ is valid for P_j, if and only if $\exists\, (\alpha, \beta) \in \mathbb{R}^{n+1}$ satisfying:

$$\begin{array}{ll} \alpha = uA + u_0 e_j, & \alpha = vA + v_0 e_j, \\ \beta \leq ub, & \beta \leq vb + v_0, \end{array}$$
$$(u, v) \in \mathbb{R}^m_+, (u_0, v_0) \in \mathbb{R}.$$

where $(u, u_0)^T, (v, v_0)^T \in \mathbb{R}^{m+1}$ can be seen as the dual variable of system (3.5.2), corresponding to constraints $Ay - by_0 \geq 0, -y_j = 0$ (describing P_{j0}) and $Az - bz_0 \geq 0, z_j - z_0 = 0$ (describing P_{j1}) respectively. The set (α, β) describes the polyhedral cone which is used to project (3.5.2) onto the x-space.

Thus for each $0-1$ variable x_j that is fractional at the linear programming optimum $\bar{x}$, the following linear system (3.5.3) is suggested by Balas et al. [4] to generate cutting planes, which maximize the (Euclidean) distance between the current point $\bar{x}$ and the hyperplane $\alpha x = \beta$.

$$
\begin{aligned}
\min_{\alpha,\beta,u,u_0,v,v_0} \quad & \alpha\bar{x} - \beta \\
s.t. \quad & \alpha - uA + u_0 e_j \geq 0, \\
& \alpha - vA - v_0 e_j \geq 0, \\
& -\beta + ub = 0, \\
& -\beta + vb + v_0 = 0, \\
& \sum_i u_i + u_0 + \sum_i v_i + v_0 = 1, \\
& u, v \geq 0,
\end{aligned}
\tag{3.5.3}
$$

Solving (3.5.3) we obtain a cut $\alpha x \geq \beta$, which maximizes the amount $\beta - \alpha\bar{x}$ by which $\bar{x}$ is cut off.

3.5.2 Carøe and Tind's cutting plane algorithm for SMIP

This part of the work applies a cutting plane based method developed by Carøe and Tind in [15] to the SMIP model of TpTRP. In [15], the authors consider a mixed 0-1 integer programming problem with dual block-angular structure arising from two-stage stochastic programs with mixed 0-1 recourse. A cutting-plane approach is developed to approximate the convex hull of feasible integer solutions by generating Lift-and-Project cuts iteratively. In case where the problem has fixed recourse, cuts generated for one sub-problem can be shared with others. Thus the general mixed 0-1 integer programming problem with fixed recourse can be solved efficiently.

It is worthwhile pointing out that, unlike most algorithms for stochastic programming, the approach in Carøe and Tind [15] avoids dealing with the expected recourse function but instead, considers the deterministic equivalent directly. This means, we generate the set-convexification in the complete decision space including both the first-stage decision x and the recourse decision y. By doing this, we can avoid the otherwise necessary cut lifting step, the derivation of which generally requires a discrete first-stage decision space. With this method, problems which have a mixed-integer recourse and a continuous first-stage space are solvable.

In the following parts of this section, we assume that ω follows a discrete distribution with support $\Omega = \{\omega^1, ..., \omega^r\}$ and corresponding probabilities $p^1, ..., p^r$.

Consider the deterministic equivalent:

$$\begin{array}{ll} \min\limits_{x,y^s} & cx + \sum\limits_{s=1}^{r} p^s q^s y^s \\ s.t.: & Ax \geq b, \\ & T^s x + W y^s \geq h^s, s = 1, ..., r, \\ & x \geq 0, y^s \in Y, s = 1, ..., r, \end{array} \tag{3.5.4}$$

where Y is a mixed integer set, $Y := \{0,1\}^h \times \mathbb{R}_+^{m-h}$, i.e. the first h components of y^j are 0-1 constrained. Note that this problem has a fixed recourse matrix. Then the cutting plane algorithm of Carøe and Tind [15] is outlined as follows:

Step 1. Set iteration $t := 1$, solve the linear relaxation of the deterministic equivalent (3.5.1) to obtain the optimal solution $(x(1), y^s(1)), s = 1, ..., r$.

Step 2. Check the stopping criterion. If $y_j^s(t) \in \{0,1\}$ for all $j = 1, ..., h$ and $s = 1, ..., r$, then stop. Alternatively, stop if $\sum\limits_{j=1}^{h} y_j^s(t)(1 - y_j^s(t)) \leq \epsilon, s = 1, ..., r$, for some level $\epsilon > 0$. Otherwise go to Step 3.

Step 3. For every $y_j^s(t), j = 1, ..., h, s = 1, ..., r$ such that $0 < y_j^s(t) < 1$, do:

Step 3a. Generate a Lift-and-Project cut $\alpha x + \beta y^s \geq \gamma$ by solving (3.5.5), cutting away the current non-integer point $(x(t), y^s(t))$.

Step 3b. Share cuts between scenarios, solve (3.5.6) to generate $\tilde{\alpha}$ and δ (while β is taken from Step 3a.), making the new cut $\tilde{\alpha} x + \beta y^k \geq \delta$ valid for scenarios $k \neq s$

Step 3c. Update the deterministic equivalent by adding all the valid cuts.

Step 4. Let $t := t + 1$, solve the linear relaxation of the current augmented deterministic equivalent to get the optimal solution $(x(t), y^s(t)), s = 1, ..., r$ and go back to Step 2.

The linear cut generating system used in Step 3a. (corresponding to variable y_j) is:

$$
\begin{array}{ll}
\min\limits_{\alpha,\beta,\gamma,u^0,u^1,u^2,v^0,v^1,v^2} & \alpha x(t) + \beta y^s(t) - \gamma \\
s.t. & \alpha - u^1 A - u^2 T^s \geq 0, \\
& \beta + u^0 e_j - u^2 W \geq 0, \\
& \alpha - v^1 A - v^2 T^s \geq 0, \\
& \beta - v^0 e_j - v^2 W \geq 0, \\
& u^1 b + u^2 h \geq \gamma, \\
& v^0 + v^1 b + v^2 h \geq \gamma, \\
& u^1, u^2, v^1, v^2 \geq 0.
\end{array}
\tag{3.5.5}
$$

This system maximizes the (Euclidean) distance between the current fractional point $(x(t), y^s(t))$ and the hyperplane $\alpha x + \beta y^s = \gamma$. The inequality $\alpha x + \beta y^s \geq \gamma$ obtained from an optimal solution of the linear system (3.5.5) is called a Lift-and-Project cut.

It is obvious that the Lift-and-Project cuts are computationally expensive, since they involve the solution of a linear program of roughly double size of the original system. It is therefore of interest to be able to find cutting planes for scenarios $k \neq s$, once a cut has been found for scenario s. Now we show the way of doing this. The following proposition is given by Carøe and Tind in [15].

Proposition 3.5.1. *Let the recourse matrix be fixed and suppose $\alpha x + \beta y^s \geq \gamma$ is a cutting plane for scenario s obtained from (3.5.5), denote the optimal solution of (3.5.5) by $(\alpha, \beta, \gamma, u^0, u^1, u^2, v^0, v^1, v^2)$. If the problem*

$$
\begin{array}{ll}
\min\limits_{\tilde{\alpha},\tilde{u}^1,\tilde{v}^1,\delta} & \tilde{\alpha} x(t) + \beta y^k(t) - \delta \\
s.t. & \tilde{\alpha} - \tilde{u}^1 A - u^2 T^k \geq 0, \\
& \tilde{\alpha} - \tilde{v}^1 A - v^2 T^k \geq 0, \\
& \tilde{u}^1 b + u^2 h^k - \delta \geq 0, \\
& v^0 + \tilde{v}^1 b + v^2 h^k - \delta \geq 0, \\
& \tilde{u}^1, \tilde{v}^1 \geq 0.
\end{array}
\tag{3.5.6}
$$

is feasible, then $\tilde{\alpha} x + \beta y^s \geq \delta$ is a valid inequality for scenario k. If the optimal value of (3.5.3) is negative, then $\tilde{\alpha} x + \beta y^s \geq \delta$ cuts off the point $(x(t), y^1(t), ..., y^r(t))$.

With this proposition, a smaller sized linear system (in which only coefficients corresponding to first-stage variables are unknown while all the ones correspond-

ing to second-stage variables are taken from the solution of (3.5.4)) need to be solved to obtain the cut for scenario k.

As said, this algorithm does not distinguish the first-stage variables from the second-stage ones, which generates the set-convexification on the complete decision space $(x, y^s), s = 1, ..., r$. Nevertheless, it requires repeated solution of the linear relaxation of the augmented deterministic equivalent for many many times, until an accurate enough set-convexification is built, which gives the optimal integer solution directly at one vertex. This is definitely expensive in computation and easily leads to numerical failure. To solve this problem, cut-aggregation and/or cut-deletion procedures can be considered. However in this work, we use a more effective way to avoid the numerical failure, where the Lift-and-Project cutting plane is only used to refine the feasible convex hull approximation, while a branch-and-Cut method is then applied to find the optimal integer solution.

3.5.3 An example

Now we take a small TpTRP instance as an example, to show how the algorithm works. Assume that we have 2 available network providers to send data (with $c_1 = 10$ and $c_2 = 12$), each divide the charging period into 3 time intervals and charge on the 2-highest volume of traffic. There are totally two possible traffic realisations, which are $(8000, 10000, 12000)$ and $(10000, 8000, 10000)$. Then we can construct the deterministic equivalent and define the matrices in formula (3.5.3):

$$A_{3\times 6} = \begin{pmatrix} 1 & & & 1 & & \\ & 1 & & & 1 & \\ & & 1 & & & 1 \end{pmatrix}, b_{3\times 1} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$

$$T^1_{8\times 3} = \begin{pmatrix} -8000 & & \\ & -10000 & \\ & & -12000 \\ 0 & 0 & 0 \\ -8000 & & \\ & -10000 & \\ & & -12000 \\ 0 & 0 & 0 \end{pmatrix}, T^2_{8\times 3} = \begin{pmatrix} -10000 & & \\ & -8000 & \\ & & -10000 \\ 0 & 0 & 0 \\ -10000 & & \\ & -8000 & \\ & & -10000 \\ 0 & 0 & 0 \end{pmatrix},$$

$$W_{8\times 8} = \begin{pmatrix} -1 & -12000 & & \\ -1 & & -12000 & \\ -1 & & & -12000 \\ & -1 & -1 & -1 \\ -1 & -12000 & & \\ -1 & & -12000 & \\ -1 & & & -12000 \\ & -1 & -1 & -1 \end{pmatrix}, h_{8\times 1} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

We begin the solution of this system by solving the linear relaxation of the deterministic equivalent. The optimal value of the linear relaxation is 30 with the optimal solution:

$$x(1) = \Big(0.4444,\ \ 0.4444,\ \ 1.0000,\ \ 0.5556,\ \ 0.5556,\ \ 0.0000\Big),$$

$$y^1(1) = \Big(3.3333,\ \ 0.0222,\ \ 0.1111,\ \ 0.8667,\ \ 0.0000,\ \ 0.4444,\ \ 0.5556,\ \ 0.0000\Big),$$

$$y^2(1) = \Big(2.6667,\ \ 0.1778,\ \ 0.0889,\ \ 0.7333,\ \ 0.0000,\ \ 0.5556,\ \ 0.4444,\ \ 0.0000\Big),$$

In this example, all y-variables except y^s_1 and y^s_4 are restricted to be integer. We generate cutting planes according to each one of those y-variables which does not satisfy the integrality constraint, to approximate the convex hull of the feasible integer solutions sequentially. For a disjunction on y^1_2 for example, the Lift-and-Project cut generated from (3.5.5) (with $j = 2$) is: $-8x_1 - 12x_3 + y^1_1 - 4y^1_2 \geq -12$, which cuts off the point $(x(1), y^1(1))$.

This cut is generated according to scenario 1. To obtain the Lift-and-Project cut valid for scenario 2, we solve a smaller cut sharing system (3.5.6), which gives: $-6x_1 + 19x_6 + y_1^2 - 4y_2^2 \geq 0$ and cuts off the point $(x(1), y^2(1))$. Note that in this instance, (3.5.6) has 41 variables and 23 constraints, while problem (3.5.5) has 79 variables and 47 constraints.

By doing these steps repeatedly for all non-integer y-variables, we can obtain an augmented deterministic equivalent with all the new cuts added. Solving it we get the new optimal solution and then continue to the next iteration. Although it seems computationally expensive, the lift-and-project cutting plane method is efficient in iteratively approximating the convex hull of the integer feasible region, due to the ability of sharing cuts among scenarios. In the following section we will show the numerical experiments on some specially designed instances.

3.5.4 Numerical results

In this section we show the numerical results of solving the SMIP model of TpTRP with the cutting plane method we introduced above.

Numerical results on TpTRP instances with time-dependent traffic distributions

Firstly we list the instances in Table 3.9. These instances are the same as the ones we examined in Section 3.3, where we solve their SMIP model directly by CPLEX. We will compare the running results of the cutting plane algorithm with CPLEX on these instances.

Instance	**Parameters**			**Stochastic Information**	
	$\lvert\Gamma\rvert$	θ	**n**	**traffic realisations**	**probability**
Instance 0(a)	3	2	2	{8000, 10000, 12000}	0.5
				{10000, 8000, 10000}	0.5
Instance 0(b)	5	2	2	{14000, 6000, 10000, 8000, 12000}	0.3333
				{10000, 10000, 6000, 10000, 10000}	0.3333
				{6000, 14000, 14000, 12000, 8000}	0.3333

Table 3.9: Basic information for Instances 0 of TpTRP

Instance 0(a) (with $c_2 = 12$) is the example we investigated in detail in Section

3.5.1. The optimal integer decision of this fairly small instance (as given in Section 3.3.1, which is the same as the TMRP) can be found by numerically going through 10 iterations, where 210 Lift-and-Project cuts are added onto the original deterministic equivalent. This takes 4.439 seconds. We can see that the number of cutting planes required to find the optimal integer solution is huge, comparing with the size of the original problem. This is definitely a challenge of applying this algorithm directly to solve a SMIP model to optimality.

Instance	c_2	**CPLEX (BAC)**		**C-BAC**				
				Cutting plane			**CPLEX (BAC)**	
		Nodes	**Time**	**Iterations**	**Cuts**	**Time**	**Nodes**	**Time**
Instance 0(a)	11	28	0.01s	11	242	4.752s	-	-
	12	29	0.01s	10	210	4.439s	-	-
	15	25	0.01s	11	252	4.990s	-	-
Instance 0(b)	11	322	0.12s	12	762	20.495s	40	0.11s
	12	428	0.16s	12	858	22.588s	83	0.26s
	15	279	0.11s	12	888	22.798s	89	0.30s

Table 3.10: Comparison between CPLEX and C-BAC method on running information for Instances 0 of TpTRP

On the other hand, Instance 0(b) is a relatively bigger example, with 5 time periods and a total of 3 realisations. Optimal solution of it has been investigated in Section 3.3.1 (given in Table 3.3), where we observed that the optimal solution for this instance is different from the trivial routing policy. Unfortunately, although this instance small in size and can be solved easily by CPLEX (taking only 0.16s), it cannot be solved to optimality solely with the cutting plane method. As said before, the cutting plane method developed by Carøe and Tind [15] generates a huge number of Lift-and-Project cuts to approximate the convex hull of the feasible integer solutions. This step may introduce numerical failure in later steps since the huge number of cuts makes the vertexes of the feasible region too close to each other. This is the reason which prevents the stated cutting plane method working on Instance 0(b).

Nevertheless, the cutting plane algorithm can still be beneficial in some senses, as it improves the optimal value of the linear relaxation quickly in the first iterations. For example to solve Instance 0(b), we consider to run the Lift-and-Project cutting plane algorithm for several iterations to generate a better convex hull approximation, and then conduct the well known Branch-and-Cut method over this shrunken feasible space to search for the optimal integer solution. We call this method, the C-BAC (Cutting then Branch-and-Cut) method.

We take the Instance 0(b) (with $c_2 = 12$) for example. Firstly we follow the cutting plane algorithm, generate Lift-and-Project cuts for 12 iterations. This step generates 858 cuts which requires 22.588s. Then we put the augmented deterministic equivalent into CPLEX and solve it by the inherent mixed-integer solver of CPLEX (Branch-and-Cut). This gives the same optimal routing policy as shown in Table 3.3, which takes 0.26s (generates 83 branch nodes). Although in this example the C-BAC method takes much more time than solving the deterministic equivalent directly by CPLEX, we can still see that after the cutting plane steps, CPLEX needs many fewer nodes to find the optimal mixed-integer solution than before. Thus for the larger sized instances which CPLEX cannot solve, we can still get something valuable with the C-BAC method.

Numerical results on TpTRP instances with time-independent traffic distributions

In this section we show the numerical results of running the proposed C-BAC method on a group of TpTRP instances with time-independent traffic distributions, as listed in Table 3.11. Again as what we did in Section 3.3.2, only the case $c_2 = 12$ is considered in this section.

Instance	Parameters			Stochastic Information	
	$\|\Gamma\|$	θ	n	distribution	probability
Instance 1(a)	5	2	2	$T^1, T^2, T^3 \in \{6000, 14000\}$	{0.5, 0.5}
				$T^4 \in \{7000, 13000\}$	{0.5, 0.5}
				$T^3 \in \{8000, 12500\}$	{0.5, 0.5}
Instance 1(b)	5	2	2	$T^1, ..., T^5 \in \{8000, 10000, 12000\}$	{0.3333, 0.3333, 0.3333}

Table 3.11: Basic information for Instances 1 of TpTRP

Instance 1(a) is a small TpTRP with time-independent traffic distribution. Every time interval has two possible traffic realisations, each has 50% possibilty to happen. The means for the time intervals' traffic are {10000, 10000, 10000, 10000, 10250 }. Thus the Trivial Multi-homing Routing Policy (discussed in Section 2.3.2) is:

$$d_{TMRP} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

which sends the time interval with highest mean to the expensive network provider,

resulting in an expected cost 115000. However, the TMRP is not optimal for this instance. Unfortunately, if solve this instance directly by CPLEX, TMRP is the best routing policy it can find after numerically examining $1,000,000$ nodes, which costs 5712.64s to run and the final gap between the best integer solution and the linear relaxation is 24.58% (the detailed data is given in Table 3.7). However, we can find a better routing policy (with expected cost 114687.50) quickly by the Cut-and-Branch method.

Instance	CPLEX (BAC)		C-BAC	
	Decisions	Mean cost	Decisions	Mean cost
Instance 1(a)	$x_1 = (1.00, 1.00, 1.00, 1.00, 0.00)^T$ $x_2 = (0.00, 0.00, 0.00, 0.00, 1.00)^T$	115000.00	$x_1 = (1.00, 0.00, 1.00, 1.00, 1.00)^T$ $x_2 = (0.00, 1.00, 0.00, 0.00, 0.00)^T$	114687.50
Instance 1(b)	$x_1 = (0.3637, 0.00, 0.00, 0.00, 0.00)^T$ $x_2 = (0.6363, 1.00, 1.00, 1.00, 1.00)^T$	115400.76	$x_1 = (0.00, 1.00, 1.00, 1.00, 1.00)^T$ $x_2 = (1.00, 0.00, 0.00, 0.00, 0.00)^T$	95925.93

Table 3.12: Comparison between CPLEX and C-BAC method on solutions for Instances 1 of TpTRP

Instance	CPLEX (BAC)			C-BAC				
	Nodes	Time	Gap	Cutting plane		CPLEX (BAC)		
				Cuts	Time	Nodes	Time	Gap
Instance 1(a)	1,000,000	5712.64s	24.58%	2451	189.895s	266,205	3911.27s	0.00%
Instance 1(b)	1,000,000	7493.46s	37.56%	7931	15944.800s	20,000	2892.44s	15.43%

Table 3.13: Comparison between CPLEX and C-BAC method on running information for Instances 1 of TpTRP

As shown in Table 3.12 and Table 3.13, the C-BAC method generates 2451 cuts in $189.895s$ and then takes only $50.56s$ to find a better solution by CPLEX:

$$d_{C\&B} = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix},$$

which sends T^1 to the expensive network provider instead of T^5. This routing policy can be proved to be optimal after examining $266,205$ nodes within $3911.27s$. As a result, the C-BAC method does provide a better routing policy than what CPLEX alone can find.

Similarly in Instance 1(b), the C-BAC algorithm can do better than CPLEX as well, which provides us a routing policy the same as the TMRP. Looking at Table 3.13 we can see, the cutting plane procedure for Instance 1(b) takes much longer time than before. This is due to the fact that the cut-generation step

needs to solve a linear system (3.5.4) for every single scenario and then a smaller system (3.5.5) to share cuts for all other scenarios. In case where there are totally $3^5 = 243$ different scenarios in Instance 1(b), it takes around $4000s$ to go through one iteration. Nevertheless, after generating 7931 cuts within 15944.800 seconds, solving the augmented deterministic equivalent of Instance 1(b) as a MILP by CPLEX, we can get the TMRP within only 92.64 seconds. Although this feasible integer solution cannot be proved to be optimal because it does not change until $20,000$ nodes are examined with takes $2892.44s$ by CPLEX, we did shrink the gap down to 15.43% with this method instead of 37.56% in the previous case.

No matter how, it is still hard for the current C-BAC algorithm to solve the general instances of TpTRP, e.g. Instance 1(b) in Table 3.11 to optimality. Due to the fact that the size of the SMIP model grows too quickly with the size of the TpTRP, thus gives arise to intractable large mixed-integer systems very easily.

3.6 Conclusions

Up to now, we have shown that solving the TpTRP as an SIP is intractable for all but the smallest instances, due to the fact that modelling of the top-percentile cost requires the introduction of integer variables within the final time stage.

Chapter 4

Stochastic Dynamic Programming Model

We have seen in Chapter 3 that the mixed-integer stochastic programming model of the TpTRP is very hard to solve. The reason is that the top-percentile pricing requires to be modelled with integer variables in the final stage. As an alternative, Stochastic Dynamic Programming (SDP) is a widely used optimization tool to solve stochastic problems in a recursive manner, which can avoid the need to model the top-percentile traffic by integer variables as it is shown directly in the state variable. In this section we show how, by a discretization of the state space, medium sized instances of TpTRP can be solved as a SDP model.

4.1 Introduction to dynamic programming

Dynamic Programming (DP) was invented in the 1950s by Richard Bellman [9]. It was introduced as, though not restricted to, a technique for making decisions over time or stages. The word dynamic was chosen by Bellman to capture the time-varying aspect of the problem.

The key idea behind dynamic programming is quite simple. In mathematics and computer science, dynamic programming is a method for solving complex problems by breaking them down into simpler sub-problems in a recursive manner. Namely DP divides a problem into sub-problems, which are themselves usually

divided into further sub-problems.

In dynamic programming, a problem is divided into stages with a decision required at each stage. Each stage has a number of states associated with it, which characterize sub-problems to capture history in a compact way. The decision at one stage transforms one state into a state in the next stage. All dynamic programming problems can be written in terms of a recursion that relates the value of being in a particular state at one point in time, i.e. $V_\tau(S_\tau)$, to the value of the states that we are moved into at the next point in time, i.e. $V_{\tau+1}(S_{\tau+1}|S_\tau, x_\tau))$ by taking the action x_τ. For deterministic problems, this equation can be written as:

$$V_\tau(S_\tau) = \min_{x_\tau \in X}(C_t(S_\tau, x_\tau) + V_{\tau+1}(S_{\tau+1}|S_\tau, x_\tau)), \tag{4.1.1}$$

where $(S_{\tau+1}|S_\tau, x_\tau)$ is the state we transition to if we are currently at state S_τ and take action x_τ, $C_t(S_\tau, x_\tau)$ is the immediate cost indicated by taking this action. This equation is known as Bellman's equation, or recurrence function.

It is important to point out that DP is not a method that is applicable on all multi-stage decision making problems. Actually from Bellman's equation we can see that at any stage τ the optimal decision x_τ does not depend on the previous states (i.e. $S_t, t = 1, 2, ..., \tau - 1$) or decisions (i.e. $x_t, t = 1, 2, ..., \tau - 1$). Instead, there exists a relationship that identifies the optimal decision for stage τ, given that stage $\tau + 1$ has already been solved. So that when we find the optimal solution of a DP problem, we have therefore found the optimal solutions for all its sub-problems.

To solve a DP problem, the final stage sub-problem is solved by itself. Starting from the final stage, taking one stage backward at a time, find the optimal decision for all possible states at this stage based on the previous calculated 'next time stage' value functions. Namely, the deepest level sub-problems are first solved, followed by those that depend immediately on them and so on up to the top level.

In deterministic dynamic programming, given a state and a decision, both the immediate payoff and next state are know. If we know either of these as a probability function, then we have a Stochastic Dynamic Program (SDP). Traditionally DP works only on discrete space, which requires the number of states to be fi-

nite for every time stage. Thus for SDP, in case where the probability space for uncertain data is continuous, we need discretize it before continuing the general DP steps.

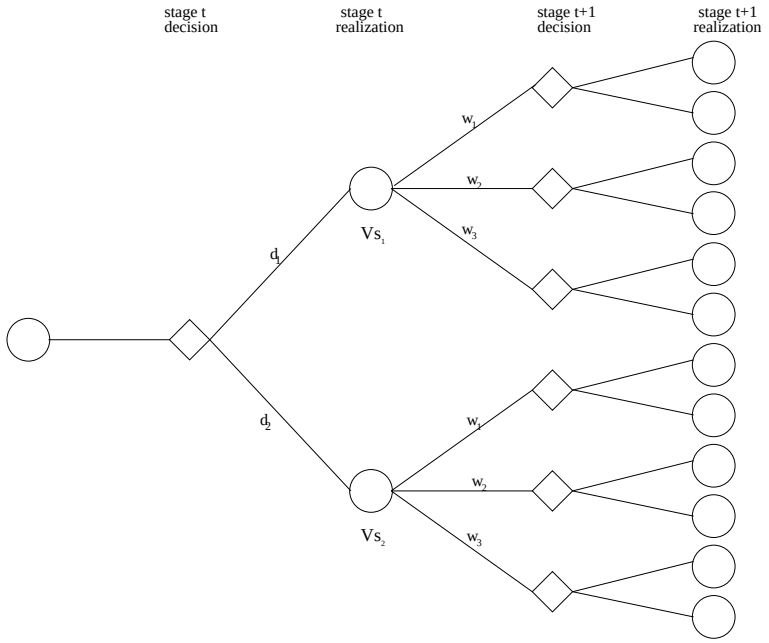


Figure 4.1.1: An example of stochastic dynamic tree

For example, Figure 4.1.1 shows a small SDP tree. In this figure, decision nodes are denoted by squares while state nodes (where we see the realisations) are denoted by circles. It is clear that in this example there are 3 time stages in total, each stage has 3 potential realisations of random data. At every decision node, two feasible decisions are available to choose from. To solve the problem, starting from the final stage we solve the Bellman's equation to determine the optimal decision at every decision node, and evaluate the expectation of future cost at every state node. These optimal decisions altogether, forms the optimal routing strategy of the SDP model.

4.2 SDP model for TpTRP

We now define the main modelling elements we need in the SDP model.

4.2.1 Stages

As in the SMIP model, stages in the SDP model can be defined naturally by time intervals. At the beginning of every time stage, we observe the current state and make routing decisions for this time interval's traffic.

4.2.2 States

At the beginning of time interval τ, all the previous realisations of traffic volumes $\hat{T}^t, t = 1, ..., \tau - 1$ and routing decisions $x^t, t = 1, ..., \tau - 1$ are known. Let $\hat{T}_i^t$ indicate the volume of traffic that has been sent to network i during time interval t. Assuming the decision implementation function is given by $T_i^t(.,.)$, $\hat{T}_i^t$ can be computed by

$$\hat{T}_i^t = T_i^t(\hat{T}^t, x^t), t = 1, ..., \tau - 1.$$

Then the list $\{\hat{T}_i^t | t = 1, ..., \tau - 1\}$ gives the current usage of network provider i. We use $\hat{T}_i^{j,\tau}$ to represent the j-th highest volume of traffic in $\{\hat{T}_i^t | t = 1, ..., \tau - 1\}$ and define the current state of the system as

$$\tilde{S}^\tau = \{\hat{T}_i^{j,\tau} | j = 1, ..., \tau - 1; i = 1, ..., n\}.$$

However, under pure top-percentile pricing the cost is solely determined by the θ-th highest volume of traffic shipped by every network provider, at the end of the charging period. We can see that at any point in time τ, any time intervals t whose traffic is greater than the current θ-th volume of traffic can be the θ-th highest in later stages, thus have an influence on the final cost. In contrast, any traffic which is lower than the current θ-th volume of traffic (namely, $\hat{T}_i^{j,\tau}, j = \theta + 1, ..., \tau - 1$ at time interval τ) can have no impact on the final cost. Thus for network provider i, $S_i^\tau = \{\hat{T}_i^{j,\tau} | j = 1, ..., \theta\}$ is sufficient to describe the current usage of this provider. Noting this, we delete the redundant information from $\tilde{S}^\tau$, which leads to the state at τ being redefined by

$$S^\tau = \{S_i^\tau | i = 1, ..., n\} = \{\hat{T}_i^{j,\tau} | j = 1, ..., \theta; i = 1, ..., n\}.$$

Namely the state is described by all traffic levels which are higher than the current θ-th highest for every network provider. Note that in the state variable, entries for every network provider are assumed to be sorted in non-increasing order of the traffic volume.

The value function $V_\tau(S^\tau)$ represents the expected cost for the ISP, given state S^τ at the beginning of time interval τ and optimal decisions in all future time intervals.

4.2.3 Decisions

In the stochastic programming model, we have used the proportional decision scheme, namely traffic sent to provider i during time interval τ is amounts to $\hat{T}_i^\tau = x_i^\tau \hat{T}^\tau$. In practice however, it is possible for us to split the traffic in a more complicated way, if only the resulting decision scheme obeys the necessary conditions we stated in Section 2.1.3.

In the dynamic programming model we consider a particular mixed routing policy. Recall that we defined a cut-off based decision scheme in Section 2.1.3, which has been justified to be implementable in practice. In this decision scheme the routing decision is divided into two parts, the cut-off based decision (r_i^τ, y_i^τ) and the fractional based decision x_i^τ. Namely the decision for time interval τ is described by

$$d^\tau = (r_1^\tau, y_1^\tau, x_1^\tau; r_2^\tau, y_2^\tau, x_2^\tau; ..., r_n^\tau, y_n^\tau, x_n^\tau)^T, \sum_{i \in I} x_i^\tau = 1,$$

where $r_i^\tau \in \{1, ..., n\}$ indicates the sequence of network provider i in the implementation rule. Namely when implementing this decision scheme, we send the first $y_{i_1}^\tau$ unit of traffic to provider i_1 if $r_{i_1}^\tau = 1$, the next $y_{i_2}^\tau$ to provider i_2 if $r_{i_2}^\tau = 2$ and so forth. Any remaining traffic $T^\tau - \sum_{i \in I} y_i^\tau$, if there is, is then allocated according to the proportional decision x_i^τ.

In the following section we motivate our choice of the cut-off level y^τ, which is instrumental in designing the feasible decision set for the following SDP model.

Revised decision space

Now we have built a decision space consisting of two types of decisions, where y_i^τ represents the cut-off decision and x_i^τ indicates the proportional decision applied on the 'additional traffic'. The aim of this section is to further simplify the current decision space by deriving near optimal decisions for the y-part. We begin with the following lemma which establishes a monotonicity property of the value function.

Lemma 4.2.1. *At any time stage $\tau \in \Gamma$, if there are two states ${}^1S^\tau = \{{}^1\hat{T}_i^{j,\tau}\}$, ${}^2S^\tau = \{{}^2\hat{T}_i^{j,\tau}\}$ which satisfy ${}^1S^\tau \leq {}^2S^\tau$ component-wise, i.e., ${}^1\hat{T}_i^{j,\tau} \leq {}^2\hat{T}_i^{j,\tau}, \forall i \in I, 1 \leq j \leq \theta$. Then we have $V_\tau({}^1S^\tau) \leq V_\tau({}^2S^\tau)$.*

Proof. We proof this assertion by induction over τ.

At $\tau = |\Gamma|$, we compute the cost based on the θ-th highest volume of traffic sent to every network provider. It is obvious that $V_{|\Gamma|}({}^1S^{|\Gamma|}) \leq V_{|\Gamma|}({}^2S^{|\Gamma|})$ holds.

Now we assume that for arbitrary ${}^1S^{\tau+1} \leq {}^2S^{\tau+1}$ we know $V_{\tau+1}({}^1S^{\tau+1}) \leq V_{\tau+1}({}^2S^{\tau+1})$. At time stage τ, assume $({}^2\hat{x}^\tau, {}^2\hat{y}^\tau)$ is the optimal routing decision we made on state ${}^2S^\tau = \{{}^2\hat{T}_i^{j,\tau}\}$. According to the implementation rule given in the beginning of this section, the amount of traffic ${}^{new}T_i^\tau$ sent to network provider i does not depend on the current state S^τ. This means if we apply the same decision set $({}^2\hat{x}^\tau, {}^2\hat{y}^\tau)$ on an arbitrary state ${}^1S^\tau$ (with ${}^1S^\tau \leq {}^2S^\tau$), every network provider receives the same traffic ${}^{new}T_i^\tau$ as when we were in state ${}^2S^\tau$. Thus for every single scenario $\hat{\omega}^\tau$, we will go to ${}^1\tilde{S}^{\tau+1} = S^{\tau+1}({}^1S^\tau; \hat{\omega}^\tau; {}^2\hat{x}^\tau, {}^2\hat{y}^\tau)$ which is no greater than ${}^2S^{\tau+1} = S^{\tau+1}({}^2S^\tau; \hat{\omega}^\tau; {}^2\hat{x}^\tau, {}^2\hat{y}^\tau)$ in all entries, i.e. ${}^1S^{\tau+1} \leq {}^2S^{\tau+1}$. From the induction we have $V_{\tau+1}({}^1\tilde{S}^{\tau+1}) \leq V_{\tau+1}({}^2S^{\tau+1})$. Taking the expectation over ω^τ we get

$$\tilde{V}_\tau({}^1S^\tau) = \mathbb{E}_{\omega^\tau}[V_{\tau+1}({}^1\tilde{S}^{\tau+1})] \leq \mathbb{E}_{\omega^\tau}[V_{\tau+1}({}^2S^{\tau+1})] = V_\tau({}^2S^\tau).$$

However, the decision set $({}^2\hat{x}^\tau, {}^2\hat{y}^\tau)$ we used might not be optimal in state ${}^1S^\tau$,

which means the best function value $V_\tau({}^1S^\tau) \leq \tilde{V}_\tau({}^1S^\tau)$. Combining these two inequalities, we have proved that $V_\tau({}^1S^\tau) \leq V_\tau({}^2S^\tau)$ holds for $\forall \tau \in \Gamma$.

□

From Lemma 4.2.1 we can see that the value function $V_\tau(S^\tau)$ is non-decreasing with every entry of the state S^τ. It is therefore our aim to make a routing decision that results in a minimal increase in components of the state from S^τ to $S^{\tau+1}$. To make more qualifying statements we would need to know how to trade-off increases in the state between its entries. However, this is difficult due to the non-convexitivity of the model. In this section we assume that the increase of the value function from S^τ to $S^{\tau+1}$ is accounted for by the total increase in the state, i.e. $||S^{\tau+1} - S^\tau||_1 = \sum_{i \in I} \sum_{1 \leq j \leq \theta} (\hat{T}_i^{j,\tau+1} - \hat{T}_i^{j,\tau})$. Then we can make statements about the shape of the optimal routing policy which minimize the expectation of this increase.

First note that for network provider i we have:

$$||S_i^{\tau+1} - S_i^\tau||_1 = \begin{cases} 0, & \text{if } {}^{new}T_i^\tau \leq \hat{T}_i^{\theta,\tau} \\ {}^{new}T_i^\tau - \hat{T}_i^{\theta,\tau}, & \text{otherwise} \end{cases}$$

Thus,

$$\begin{aligned} ||S^{\tau+1} - S^\tau||_1 = \sum_{i \in I} ||S_i^{\tau+1} - S_i^\tau||_1 &= \sum_{i \in I} \max\{{}^{new}T_i^\tau - \hat{T}_i^{\theta,\tau}, 0\} \\ &\geq \max\{\sum_{i \in I} ({}^{new}T_i^\tau - \hat{T}_i^{\theta,\tau}), 0\} \qquad (4.2.1) \\ &= \max\{\hat{T}^\tau - \sum_{i \in I} \hat{T}_i^{\theta,\tau}, 0\}. \end{aligned}$$

Let us define $T_{Add}(S^\tau) = \max\{\hat{T}^\tau - \sum_{i \in I} \hat{T}_i^{\theta,\tau}, 0\}$. We call $T_{Add}(S^\tau)$, the additional traffic, which represents the amount of traffic that cannot be sent without affecting the current θ-th highest volume of traffic of any network provider. Actually, any allocation of less than $\hat{T}_i^{\theta,\tau}$ in time interval τ will not change the price charged by network provider i since the cost is based on the θ-th highest volume of traffic. According to the inequality (4.2.1), $T_{Add}(S^\tau)$ is a lower bound on $||S^{\tau+1} - S^\tau||_1$.

Lemma 4.2.2. *Assume we are in state* $S^\tau = \{\hat{T}_i^{j,\tau} | j = 1, ..., \theta; i = 1, ..., n\}$ *at time stage* $\tau \in \Gamma$. *There is a routing policy that minimises* $\mathbb{E}_{\omega^\tau}[||S^{\tau+1} - S^\tau||_1]$, *in which* $y_i^\tau = \hat{T}_i^{\theta,\tau}, \forall i \in I$.

Proof. Firstly, it is obvious to see that with $y_i^\tau = \hat{T}_i^{\theta,\tau}, \forall i \in I$, for every single scenario $\omega^\tau \in \Omega^\tau$ we can guarantee $||S^{\tau+1} - S^\tau||_1 = T_{Add}(S^\tau)$.

Secondly, we prove that with any other choice of y_i^τ, we can always find scenarios for which $||S^{\tau+1} - S^\tau||_1 > T_{Add}(S^\tau)$.

- Assume $\exists i_0 \in I$, $\hat{y}_{i_0}^\tau < \hat{T}_{i_0}^{\theta,\tau}$. If the new traffic $\hat{T}^\tau$ satisfies $\hat{y}_{i_0}^\tau + \sum_{i \neq i_0} \hat{T}_i^{\theta,\tau} < \hat{T}^\tau < \sum_{i \in I} \hat{T}_i^{\theta,\tau}$, the amount of traffic sent to every network provider is given by

$$\begin{cases} {}^{new}T_{i_0}^\tau = \hat{y}_{i_0}^\tau + \hat{x}_{i_0}^\tau (\hat{T}^\tau - \hat{y}_{i_0}^\tau - \sum_{i \neq i_0} \hat{T}_i^{\theta,\tau}), \\ {}^{new}T_i^\tau = \hat{T}_i^{\theta,\tau} + \hat{x}_i^\tau (\hat{T}^\tau - \hat{y}_{i_0}^\tau - \sum_{i \neq i_0} \hat{T}_i^{\theta,\tau}), \quad \text{for} \quad i \neq i_0 \end{cases}$$

  As $\hat{T}^\tau < \sum_{i \in I} \hat{T}_i^{\theta,\tau}$, we have ${}^{new}T_{i_0}^\tau < \hat{y}_{i_0}^\tau + \hat{x}_{i_0}^\tau (\sum_{i \in I} \hat{T}_i^{\theta,\tau} - \hat{y}_{i_0}^\tau - \sum_{i \neq i_0} \hat{T}_i^{\theta,\tau}) = \hat{y}_{i_0}^\tau + \hat{x}_{i_0}^\tau (\hat{T}_{i_0}^{\theta,\tau} - \hat{y}_{i_0}^\tau) \leq \hat{T}_{i_0}^{\theta,\tau}$, and therefore $\sum_{i \neq i_0} {}^{new}T_i^\tau = \hat{T}^\tau - {}^{new}T_{i_0}^\tau > \hat{T}^\tau - \hat{T}_{i_0}^{\theta,\tau}$. Thus under this scenario,

$$||S^{\tau+1} - S^\tau||_1 = \sum_{i \neq i_0} ({}^{new}T_i^\tau - \hat{T}_i^{\theta,\tau}) > \hat{T}^\tau - \sum_{i \in I} \hat{T}_i^{\theta,\tau} = T_{Add}(S^\tau).$$

- Assume $\exists i_0 \in I$, $\hat{y}_{i_0}^\tau > \hat{T}_{i_0}^{\theta,\tau}$. If the new traffic $\hat{T}^\tau$ satisfies $\sum_{i=1}^{i_0} \hat{T}_i^{\theta,\tau} < \hat{T}^\tau < \sum_{i=1}^{i_0-1} \hat{T}_i^{\theta,\tau} + \hat{y}_{i_0}^\tau$, the amount of traffic sent to every network provider is given by

$$\begin{cases} {}^{new}T_i^\tau = \hat{T}_i^{\theta,\tau}, & \text{for} \quad i = 1, ..., i_0 - 1 \\ {}^{new}T_{i_0}^\tau = \hat{T}^\tau - \sum_{i=1}^{i_0-1} \hat{T}_i^{\theta,\tau}, & \\ {}^{new}T_i^\tau = 0, & \text{for} \quad i = i_0 + 1, ..., n \end{cases}$$

As $\hat{T}^\tau > \sum_{i=1}^{i_0} \hat{T}_i^{\theta,\tau}$, we have ${}^{new}T_{i_0}^\tau > \hat{T}_{i_0}^{\theta,\tau}$. For all other network providers, ${}^{new}T_i^\tau \leq \hat{T}_i^{\theta,\tau}$. Thus under this scenario,

$$||S^{\tau+1} - S^\tau||_1 =^{new} T_{i_0}^\tau - \hat{T}_{i_0}^{\theta,\tau} > 0 = T_{Add}(S^\tau).$$

Thus for all scenario we cannot do better than $T_{Add}(S^\tau)$, where $||S^{\tau+1} - S^\tau||_1 \geq T_{Add}(S^\tau)$ always holds according to (4.2.1). Taking the expectation we can deduce that in the routing policy that minimizes $\mathbb{E}_{\omega^\tau}[||S^{\tau+1} - S^\tau||_1]$, the cut-off level is given by $y_i^\tau = \hat{T}_i^{\theta,\tau}, \forall i \in I$.

□

In Lemma 4.2.2 we have proved $y_i^\tau = \hat{T}_i^{\theta,\tau}, \forall i \in I$, in the optimal policy under the assumption that the increase in value function from S^τ to $S^{\tau+1}$ only depends on $||S^{\tau+1} - S^\tau||_1$. While obviously simplifying, it seems therefore reasonable to limit our choice of routing policies to those for which $y_i^\tau = \hat{T}_i^{\theta,\tau}$. Note that once we fix $y_i^\tau = \hat{T}_i^{\theta,\tau}$, the order of allocating traffic to network providers (i.e. the r_i^τ) does not matter anymore. Thus when we drop out y_i^τ from the decision set, we do the same on r_i^τ as well. Therefore the decision set that we will consider is

$$\chi^\tau = \{x_1^\tau, x_2^\tau, ..., x_n^\tau | 0 \leq x_i^\tau \leq 1, \forall i \in I, \sum_{i \in I} x_i^\tau = 1\}, \tag{4.2.2}$$

with the understanding that decision x_i^τ means we send at most $T_{i,add}^\tau = \hat{T}_i^{\theta,\tau} + x_i^\tau T_{Add}(S^\tau)$ to provider i during τ. In the later part of this chapter, we fix $y_i^\tau = \hat{T}_i^{\theta,\tau}$ and use χ^τ as the feasible decision set for time interval τ in the SDP model. Namely when we implement a feasible decision x^τ, we allocate the random traffic T^τ according to the following **Implementation Rule**:

1. If $\sum_{j=1}^{\tilde{i}} \hat{T}_j^{\theta,\tau} \leq \hat{T}^\tau < \sum_{j=1}^{\tilde{i}+1} \hat{T}_j^{\theta,\tau}$ for some $\tilde{i} \in I$, we send:
   - ${}^{new}T_i^\tau = \hat{T}_i^{\theta,\tau}$ to network provider $1 \leq i \leq \tilde{i}$,
   - ${}^{new}T_{\tilde{i}+1}^\tau = \hat{T}^\tau - \sum_{j=1}^{\tilde{i}} \hat{T}_j^{\theta,\tau}$ to network provider $\tilde{i}+1$,
   - ${}^{new}T_i^\tau = 0$ to network provider $i > \tilde{i}+1$.

2. If $\hat{T}^\tau \geq \sum_{j \in I} \hat{T}_j^{\theta,\tau}$, we send:

   - ${}^{new}T_i^\tau = \hat{T}_i^{\theta,\tau} + x_i^\tau(\hat{T}^\tau - \sum_{j \in I} \hat{T}_j^{\theta,\tau})$ to provider $i \in I$.

Namely as time progresses, firstly we route traffic to every network provider in turn until reaching its cut-off level $\hat{T}_j^{\theta,\tau}$. If more traffic is to be routed, we split the additional amount of traffic $T^\tau - \sum_{i \in I} \hat{T}_j^{\theta,\tau}$ according to fractional decisions x_i^τ. This means to solve the SDP model, we need to find the best proportional amount x^τ of $T_{Add}(S^\tau)$, which leads to a minimum expected future cost.

4.2.4 Recurrence

We have introduced the Bellman's equation in Section 4.1. In this section, we write the recurrence for the SDP model of TpTRP. Firstly, as traffic distribution in TpTRP is continuous, we have to discretize the traffic region into discrete levels. Assume that the traffic region for time interval τ is discretized into a set Ω^τ of random realisations, where $pr(\omega_\tau)$ is the probability that $\hat{T}^\tau = T^\tau(\omega_\tau)$ occurs. Then we can write the recurrence function as:

$$V_\tau(S^\tau) = \min_{x^\tau \in \chi^\tau} (\sum_{\omega_\tau \in \Omega^\tau} pr(\omega_\tau) V_{\tau+1}(S^{\tau+1} | S^\tau, x^\tau, T^\tau(\omega_\tau))), \tag{4.2.3}$$

where χ^τ is the feasible decision set we defined in (4.2.2). Note that in TpTRP there is no immediate cost indicated by taking decision x^τ. Now $S^{\tau+1}$ is a function of the previous state S^τ, the decision x^τ and the realisation of traffic T^τ. The dynamic from state S^τ to $S^{\tau+1}$ given decision x^τ is as follows:

1. For every network provider i, implement the decision x_i^τ according to the **Implementation Rule** shown in Section 4.2.3, find the value of ${}^{new}T_i^\tau$. Define:

$$\tilde{S}_i^{\tau+1} = \{S_i^\tau, {}^{new}T_i^\tau\}. \tag{4.2.4}$$

2. Reorder entries in $\tilde{S}_i^{\tau+1}$ into a non-increasing order and delete the traffic that is lower than the current θ-th highest to obtain $S_i^{\tau+1}$.

3. Put them altogether to form $S^{\tau+1} = \{S_i^{\tau+1} | i = 1, ..., n\}$.

In the final stage $|\Gamma|+1$ we know all the routing decisions and outcomes of traffic, then we compute the cost charged on the ISP as follows:

$$V_{|\Gamma|+1}(S^{|\Gamma|+1}) = \sum_{i \in I} c_i T_i^{\theta,|\Gamma|}.$$

So that the complete recurrence function is:

- For $\tau = 1, ..., |\Gamma|$,

$$V_\tau(S^\tau) = \min_{x^\tau \in \chi^\tau} (\sum_{\omega_\tau \in \Omega^\tau} pr(\omega_\tau) V_{\tau+1}(S^{\tau+1} | S^\tau, x^\tau, T^\tau(\omega_\tau)).$$

- For the final stage $\tau = |\Gamma| + 1$,

$$V_{|\Gamma|+1}(S^{|\Gamma|+1}) = \sum_{i \in I} c_i T_i^{\theta,|\Gamma|}.$$

4.3 SMIP model v.s. SDP model

4.3.1 Problem size

The TpTRP possesses a continuous state space, in which the random volume of traffic can be any value within a non-negative interval. However, in order to build the deterministic equivalent of the mixed-integer stochastic programming (MISP) model or to conduct the dynamic programming calculation we have to discretize the traffic region. In this work we discretize the traffic region by allowing $T^\tau(\hat{\omega^\tau}), \tau \in \Gamma$ to be one of L possible values and use $pr(\hat{\omega}^\tau)$ to represent the probability that $\hat{\omega}^\tau$ occurs.

Thus in the MISP model we get $\sum_{\tau=0}^{|\Gamma|} L^\tau = \frac{1-L^{|\Gamma|+1}}{1-L}$ nodes, each corresponding to a single routing decision to be made. However in the SDP model, state S^τ is a combination of the θ highest volume of traffic sent to every network

provider up to now. Thus every entry $\hat{T}_i^{j,\tau}$ in the state variable can be one of L possible values. Since $\hat{T}_i^{1,\tau} \geq \hat{T}_i^{2,\tau} \geq ... \geq \hat{T}_i^{\theta,\tau}, \forall i \in I$, this gives a total of $C_{L+\theta-1}^{\theta} = \begin{pmatrix} L+\theta-1 \\ \theta \end{pmatrix}$ [1] different states S_i^{τ} for provider i and a total of $|\Gamma| \begin{pmatrix} L+\theta-1 \\ \theta \end{pmatrix}^n$ different values for the complete model.

We can illustrate this comparison for a small instance of the TpTRP.

Instance	Parameters			Stochastic Information	
	$\|\Gamma\|$	θ	n	Distribution	Time dependency
Instance 2	10	3	2	$U(6000, 12000)$	i.i.d.

Table 4.1: Parameters for TpTRP Instance 1

As shown in Table 4.1, we assume that in this small instance there are 2 network providers available to choose from, both of which divide the modelling period into 10 time intervals and charge based on the 3rd highest volume of shipped traffic. As all time intervals' traffic follow a same uniform distribution, with a simple discretization we set the the potential realisation set $\Omega^{\tau} = \{6000 + 1000k | k = 0, 1, ..., 6\}$ (with $L = 7$) and $pr(6000) = pr(7000) = ... = pr(12000) = 1/7$. Then with the MISP model we have potentially up to $\frac{1-7^{11}}{1-7} = 329554457$ nodes, each of which requires the solution of a (continuous) linear program, whereas with the SDP model we have to examine only $10 \begin{pmatrix} 9 \\ 3 \end{pmatrix}^2 = 70560$ nodes, each of which requires the evaluation of a simple recurrence. Of course an efficient implementation of a branch-and-bound scheme on the MISP search will be able to prune many of the nodes, however the principle argument of an exploding search space remains. Hence the SDP model is more preferable than the MISP model.

4.3.2 Solution difficulties

Stochastic programming problems are well known for being challenging both from a theoretical and a computational point of view, because they can lead to very large scale problems with a large number of outcomes of the random parameters,

[1] $\begin{pmatrix} L+\theta-1 \\ \theta \end{pmatrix}$ is the number of possibilities which satisfies $\hat{T}_i^{j,\tau} \in \Omega^{\tau}, j = 1, ..., \theta$ and $\hat{T}_i^{1,\tau} \geq \hat{T}_i^{2,\tau} \geq ... \geq \hat{T}_i^{\theta,\tau}$.

especially in the multi-stage models. However, the difficulties we are facing are much more challenging than this, as our program is a stochastic program with integer constraints on the last stage. Only few algorithms are applicable to this type of problems due to the non-convexity introduced by the integer restrictions and one is usually forced to apply a general purpose integer linear programming solver (such as CPLEX). Overall, the multi-stage mixed-integer stochastic programming model of TpTRP is hard to solve.

However, solving the SDP model is conceptually trivial. Taking one stage at a time, starting from the last stage we can in each stage, find the optimal decision for all possible states, thereby calculating the optimal accumulated cost from then until the end of the time horizon. Namely, the deepest level sub-problems are first solved, followed by those that depend immediately on them and so on up to the top level problem. Sub-problems with the same state will have identical optimal solutions and so are equivalent as far as the optimisation is concerned. However, this process still requires the processing of many nodes, leading to large time and computer memory requirements.

In conclusion, the solution to the SDP model seems more straightforward than the MISP model.

4.4 Numerical results

4.4.1 Test problems

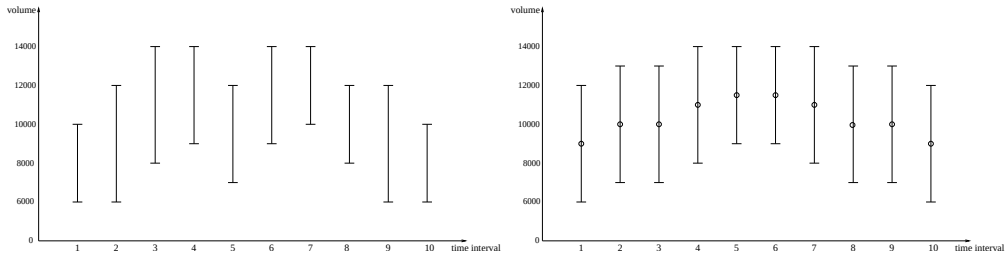
In this section we give some numerical results on several small instances of the TpTRP. For clarity, we firstly characterise and index these instances which are examined in the later part of this section.

Table 4.2 summarises the instances used. Instance 1(b) is the one that we failed to solve in the stochastic programming section, which is also the smallest one listed here. For the other instances, we assume that we divide the modelling region into 10 time intervals and costs are based on the time interval with the $\theta = \lfloor q * |\Gamma| \rfloor = 3\text{rd}$ ($q = 0.3$) highest volume of traffic. In all cases we use 2 network providers ($n = 2$) with costs $c_1 = 10, c_2 = 11, 12$ or 15.

Instance	Parameters			Stochastic Information	
	$\|\Gamma\|$	θ	n	distribution	time dependency
Instance 1(b)	5	2	2	$U(6000, 12000)$	i.i.d.
Instance 2	10	3	2	$U(6000, 14000)$	i.i.d.
Instance 3	10	3	2	uniform	see Fig. 4.4.1a
Instance 4	10	3	2	truncated $N(10000, 10^6)$	i.i.d.
Instance 5	10	3	2	truncated normal	see Fig. 4.4.1b
Instance 6	10	3	2	uniform	see Fig. 4.4.2a
Instance 7	10	3	2	truncated normal	see Fig. 4.4.2b

Table 4.2: List of TpTRP Instances

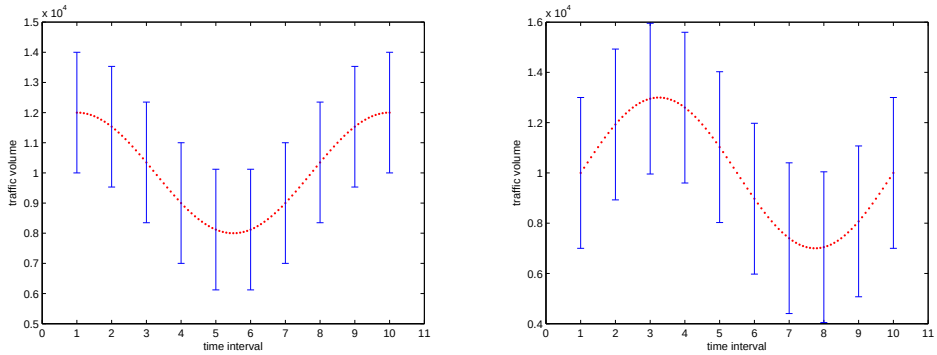
Apart from Instance 1(b), the other instances differ by the assumptions made on the random traffic. In Instances 2 and 4, the traffic in every period follows the same uniform ($U(6000, 14000)$ in Instance 2) or normal ($N(10000, 10^6)$ in Instance 4) distribution. Instances 3 and 5 on the other hand, use traffic distributed according to a time varying uniform or normal distribution, in which either the mean and the variance change with time. The parameter for each time interval are displayed in Figure 4.4.1a and Figure 4.4.1b. In addition to these, Instances 6 and 7 use a time varying uniform or normal distribution where the mean changes according to a cosine or sine function (see 4.4.2a and Figure 4.4.2b for detail). This type of traffic distribution is concluded from the analysis of data provided by a real-world ISP, which indicates the fact that time intervals for daytime are busy and for night are relatively free. Note that Instances 4, 5 and 7 use a truncated normal distribution in which traffic outside the 99.7% ($\pm 3\sigma$) confidence region is projected onto the boundary of the region.



(a) Upper and lower bounds for uniform distributions in Instance 3

(b) Mean and 99.7% ($\pm 3\sigma$) confidence region for normal distributions in Instance 5

Figure 4.4.1: Traffic distribution used in testing Instances 3 and 5



(a) Upper and lower bounds for uniform distributions in Instance 6

(b) Mean and 99.7% ($\pm 3\sigma$) confidence region for normal distributions in Instance 7

Figure 4.4.2: Traffic distribution used in testing Instances 6 and 7

4.4.2 Numerical results on TpTRP instances with 10 periods

To obtain the routing policy given by the SDP model, we discretize the possible traffic volumes into a specific number of levels with equal size, and approximate the continuous distribution functions of random traffic by discrete ones. For example, if we discretize the traffic region $[0, 14000]$ equally into 14 levels, then we use $\Omega = \{6000, 7000, ..., 13000\}$ as the set of potential realisations of traffic. In case of Instance 2 where uniform distribution is considered, we approximate it by the discrete distribution $pr(1000 * i) = 1/8, i \in \{6, 7, ..., 13\}$. Namely we always round the traffic down to the whole thousand.

With the discretization, we can solve the SDP model we built in Section 4.2.4 to get the SDP routing policy (SDPRP) of the TpTRP. By evaluating the minimisation problem (4.2.3), we get a decision on where to route the additional traffic at every time interval, proceeding backwards in time. Finally we end up with a routing table that gives for every stage and every state, the optimal routing decision for this period's additional traffic.

To evaluate the quality of this routing policy, we examine it in a simulation of $1,000,000$ random scenarios taken from the (original) continuous distribution. We obtain a routing decision by rounding the current (continuous) state down to the nearest tabulated discrete state.

Mean cost

For every problem instance, we experiment with different discretization levels, to see how the discretization reflects the optimal routing policy. We also give results of the numerical test on SRP, TMRP and DRP as a comparison. Numerical results on mean cost with different per unit cost for network provider 2 (where $c_1 = 10$ in all cases) are summarised in Table 4.3, 4.4, 4.5 and Figure 4.4.3.

Ins.	**SRP**	**TMRP**	L	**SDPRP**	**DRP**
Instance 1(b)	99385.72±11.64	96316.78±11.56	6	93242.73±12.40	90393.41±12.04
Instance 2	118178.53±10.28	113335.54±11.93	7	107219.29±12.69	103637.95±11.50
			14	106090.86±12.18	
			28	105734.36±12.03	
Instance 3	114340.25±7.52	104294.38±7.86	7	104728.46±8.15	102303.09±6.99
			14	103543.13±7.53	
			28	103140.84±7.42	
Instance 4	106564.38±4.18	104727.88±4.48	7	103564.95±5.25	101226.33±3.89
			14	102808.01±4.51	
			28	102255.24±4.11	
Instance 5	112379.96±4.53	105986.18±4.95	7	108078.29±5.89	105003.32±4.37
			14	106005.25±4.83	
			28	105536.23±4.64	
Instance 6	113697.78±6.60	103859.41±6.28	7	106554.03±7.65	102587.85±5.35
			14	103536.49±6.10	
			28	102996.71±5.54	
Instance 7	113404.28±5.48	103307.22±5.67	7	105041.39±7.35	102690.17±5.32
			14	103170.72±6.15	
			28	102953.15±6.34	

Table 4.3: Numerical result (mean cost and standard deviation) of implementing SDPRP on $1,000,000$ scenarios with $c_2 = 11$

Ins.	**SRP**	**TMRP**	L	**SDPRP**	**DRP**
Instance 1(b)	99385.72±11.64	96316.78±11.56	6	93650.80±12.51	90393.41±12.04
Instance 2	118178.53±10.28	113335.54±11.93	7	107811.61±12.90	103637.95±11.50
			14	106602.73±12.33	
			28	106203.10±12.17	
Instance 3	114340.25±7.52	104294.38±7.86	7	105256.80±8.44	102303.09±6.99
			14	103853.99±7.71	
			28	103375.22±7.54	
Instance 4	106564.38±4.18	104727.88±4.48	7	104097.79±5.54	101226.33±3.89
			14	103007.73±4.58	
			28	102424.21±4.18	
Instance 5	112379.96±4.53	105986.18±4.95	7	108541.85±6.09	105003.32±4.37
			14	106172.45±4.89	
			28	105677.18±4.73	
Instance 6	113697.78±6.60	103859.41±6.28	7	106803.13±7.64	102587.85±5.35
			14	103743.01±6.24	
			28	103285.52±5.73	
Instance 7	113404.28±5.48	103307.22±5.67	7	105290.33±7.34	102690.17±5.32
			14	103309.03±6.19	
			28	103081.45±6.18	

Table 4.4: Numerical result (mean cost and standard deviation) of implementing SDP routing policy (SDPRP) on $1,000,000$ scenarios with $c_2 = 12$

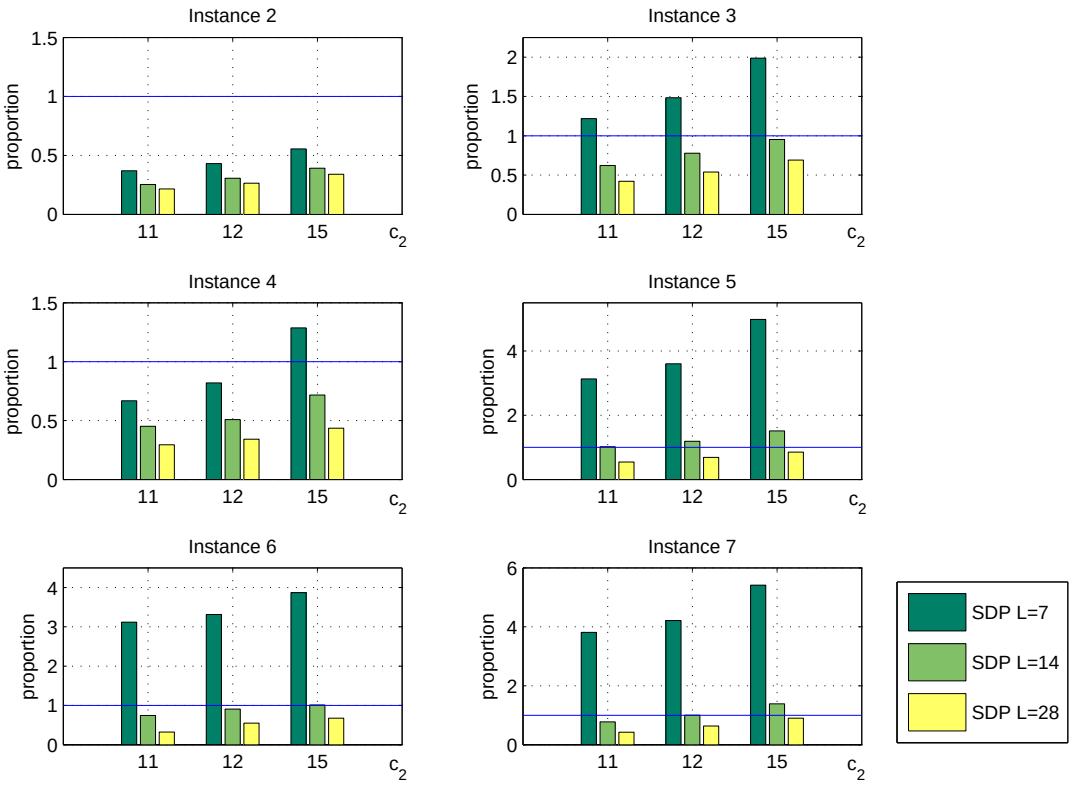


Figure 4.4.3: Comparison of mean costs of SDP routing policies with different discretization levels

Note that in Figure 4.4.3, each sub-figure is corresponding to one instance and consisting of 3 groups of columns, which in order give the relevant results for $c_2 = 11, c_2 = 12$ and $c_2 = 15$. Each group consists of 3 columns, which show the performance of SDP routing policy with 7, 14 or 28 discretization levels in turn. Performance in these figures are given in a proportional manner, where the value of a column corresponding to routing policy **X** is equal to (**X**-DRP)/(TMRP-DRP). Namely 0 means **X** works as well as DRP, while 1 means it works same as TMRP. Therefore, any routing policy that is above 1 (above the horizontal line) is obviously not desirable.

Looking at the numerical result we can see, generally speaking the SDP routing policy performs better (discretizing into 14 or 28 levels) than SRP and TMRP, and no more than 10% higher than the lower bound – the DRP. Mean costs of implementing SDP routing policies under different cost ratios c_2/c_1 follow almost the same pattern. And not surprisingly, the mean cost grows nearly linearly with per unit cost of c_2. Therefore, we have reason to believe that our SDP model is

Ins.	SRP	TMRP	L	SDPRP	DRP
Instance 1(b)	99385.72±11.64	96316.78±11.56	6	94023.69±12.42	90393.41±12.04
Instance 2	118178.53±10.28	113335.54±11.93	7	109022.21±13.42	103637.95±11.50
			14	107432.60±12.60	
			28	106932.48±12.38	
Instance 3	114340.25±7.52	104294.38±7.86	7	106260.86±9.04	102303.09±6.99
			14	104197.98±7.92	
			28	103679.88±7.67	
Instance 4	106564.38±4.18	104727.88±4.48	7	105733.64±6.65	101226.33±3.89
			14	103736.17±4.98	
			28	102750.73±4.37	
Instance 5	112379.96±4.53	105986.18±4.95	7	109898.90±6.98	105003.32±4.37
			14	106490.32±5.16	
			28	105837.72±4.89	
Instance 6	113697.78±6.60	103859.41±6.28	7	107506.27±7.64	102587.85±5.35
			14	103876.07±6.26	
			28	103444.19±5.58	
Instance 7	113404.28±5.48	103307.22±5.67	7	106030.33±7.48	102690.17±5.32
			14	103547.28±6.24	
			28	103245.73±6.14	

Table 4.5: Numerical result (mean cost and standard deviation) of implementing SDPRP on $1,000,000$ scenarios with $c_2 = 15$

a good approximation of the original TpTRP.

In addition to this, the average cost is getting lower as the discretization is getting finer. This means firstly, the optimal routing policy is changing as the number of level changes, and secondly, the better results might be achievable from an even finer discretization. Thus, with adequate number of discretization levels (e.g. $L = 28$), the SDP model can provide us with routing policies which perform better than the TMRP. However, both computation time and memory use increase drastically with the use of more discretization levels, therefore a finer discretization might not be computable.

Resource consumption

Recall that in Section 4.3.1, we have discussed the fact that in the case where there are n network providers and $|\Gamma|$ time intervals in the TpTRP instance, we have a state space with $|\Gamma|(C^{\theta}_{L+\theta-1})^n$ states in the underlying SDP model. Looking at the Bellman's equation (4.2.3) we can see that in order to get the routing decision for stage τ at all states S^{τ}, we need to compare the expected future cost $V_{\tau+1}(S^{\tau+1})$ for all states in the subsequent stage $\tau+1$. After all routing decisions for stage τ have been computed, the costs $V_{\tau+1}(S^{\tau+1})$ are not required anymore.

Therefore it is sufficient to store the $V_\tau(S^\tau)$ value for two subsequent stages τ and $\tau+1$. For the routing decision x_τ we have two choices. In the simplest case, all routing decisions are computed (and stored) ahead of time. Implementing a routing policy then amounts to simply looking up the corresponding x in a routing table. However, it calls for a great quantity of computer memory to keep the big decision matrix. For example in Instance 2 with 28 levels, the decision matrix has $10(C_{30}^3)^2 = 164836000$ entries,it needs at least 628.80MB to store the decision matrix only, and at least 251.52MB to store the value $V_\tau(S^\tau)$ for two stages.

Also, running time is another important issue that we need to be concerned about. In Table 4.6 we give a summary of the running time (to get the decision table only) and the memory consumption (theoretical) in solving the SDP model to optimality for Instance 2. All experiments are done with Java, on a computer with 1GB RAM under Redhat Enterprise Linux system, 3.20GHz Intel Dual-Core Pentium processor (one core is used).

Levels	**Problem Size**		**Resource Consumption**	
	Level Length	**No. of States**	**Running Time**	**Memory Consumption**
7	2000	7056	0.194s	0.38MB
14	1000	313600	15.628s	16.75MB
28	500	16483600	11487.095s	880.32MB

Table 4.6: Comparison of problem size and resource consumption - original model for Instance 2

From Table 4.6 we can see that the number of state variables and thus the memory consumption grows sharply with the number of discretization levels of the traffic region. Namely, as quality of the routing policy improves with the increasing of number of levels, we have to pay for this enhancement.

Alternatively, apart from storing all the decisions in a big matrix, we can regenerate the routing decisions x_τ for a given stage only when needed. As the SDP model is solved backwards, keeping one decision (say at τ) means dropping all decisions that we have made for time stages $\{\tau+1, ..., |\Gamma|\}$. Thus, when we move into time interval $\tau+1$ we need to resolve the system to generate $x_{\tau+1}$. Hence to implement a routing policy requires running the SDP model for $|\Gamma|$ times. This costs us a great deal of time. Table 4.7 gives a summary of running time and memory consumption in this case. Note that as we need to know the current state to decide which decision to generate at every time stage, we follow a single

scenario in this experiment. The running time shown in the table consists both the decision generating time and the implementing time on this scenario. (Although compared to the time to generate the decision, the implementation time is small.)

Levels	Problem Size		Resource Consumption	
	Level Length	No. of States	Running Time	Memory Consumption
7	2000	7056	0.676s	0.11MB
14	1000	313600	117.937s	4.79MB
28	500	16483600	37687.800s	251.52MB

Table 4.7: Comparison of problem size and resource consumption - alternative model for Instance 2

Looking at Table 4.7 we can see that although the memory consumption is reduced to about 1/4 of the previous amount, we have to spend much more time on getting the decision for a single run. Moreover, this alternative model maybe challenging to implement in reality. As shown in Figure 2.1.1, we know the cumulative value of $T^{\tau-1}$ at the end of time stage $\tau-1$, where we can run the SDP model to get the routing decision for time stage τ. However, the end of time stage $\tau-1$ is also the beginning of time stage τ, where we need to implement the decision x^τ at once. So that we have no time to wait for the decision x^τ been generated, although it needs a significant amount of time to do.

In general, there is a trade-off between running time and computer memory consumption. No matter which of these two is the critical issue, it does prevent the use of the SDP model for finer discretization levels or larger problem instances. This is the well known 'curse of dimensionality' in dynamic programming. In conclusion, the SDP model can provide us with routing policies which are superior to the TMRP and close to the lower bound DRP (optimal cost in the deterministic case). However, the huge number of states prevents the use of the SDP model on larger problem instances.

4.4.3 Decision rule by classification

As shown above, the optimal routing policy given by the SDP model is represented by a large discrete look-up table (7-dimensional with $n = 2, \theta = 3$). This is caused by the fact that, in order to apply the traditional SDP technique on the TpTRP,

we discretize the continuous traffic space into a discrete one. The discretization brings not only some errors within the discretization step, but also a large decision table which consumes a great amount of computer memory. Though in Section 4.4.2 we have seen that the SDP routing policy performs better than the TMRP, we did not actually know what the SDP routing policy looks like. In order to figure out what does the SDP decision rule look like and why it gives better routing policy than the TMRP, in this section we try to investigate the discrete decision table to identify the conditions under which the more expensive provider (provider 2) should be used from the routing table.

By looking through the decision table for Instance 2 with 7 levels, we find that generally the SDP routing decisions can be simplified in the following way (assuming we have 2 network providers with $c_1 = 10$ and $c_2 = 12$):

- For time intervals $\{1, 2, 3\}$, send everything to network provider 1;
- For time intervals $\{4, 5, 6, 7\}$ where $S^\tau = (\hat{T}_1^{1,\tau}, \hat{T}_1^{2,\tau}, \hat{T}_1^{3,\tau}; \hat{T}_2^{1,\tau}, \hat{T}_2^{2,\tau}, \hat{T}_2^{3,\tau})$,
  1. if $\hat{T}_1^{2,\tau} - \hat{T}_1^{3,\tau} \geq 1500$ and $\hat{T}_2^{1,\tau} \leq 2000$, send the additional traffic $T_{Add}(S_\tau)$ to network provider 2;
  2. otherwise send the additional traffic $T_{Add}(S_\tau)$ to network provider 1.
- For time intervals $\{8, 9, 10\}$,
  1. if $\hat{T}_2^{2,\tau} \leq 1500$, send the additional traffic $T_{Add}(S_\tau)$ to network provider 2;
  2. otherwise send the additional traffic $T_{Add}(S_\tau)$ to network provider 1.

This decision rule means, we prefer to send most time intervals' traffic by the cheapest network provider and no more than the additional amount of traffic by network provider 2. Only in case where the difference between $\hat{T}_1^{2,\tau}$ and $\hat{T}_1^{3,\tau}$ is high enough (say e.g. 1500 for Instance 2) and at the same time, the expected $\hat{T}_2^{3,|\Gamma|}$ (which might be $\hat{T}_2^{2,\tau}$ or $\hat{T}_2^{3,\tau}$ currently at time interval τ) is low (e.g. ≤ 1500 for Instance 2), then sending the additional traffic to network provider 2 is better.

Of course this classification rule depends on the setup of traffic distributions. We take Instance 2 only for example. Implementing this classified decision rule (CDR) on the instances, we get the following result:

Instance	SRP	TMRP	SDPRP - 7levels	CDR	DRP
Instance 2	118178.53 ±10.28	113335.54 ±11.93	107811.61 ±12.90	108805.82 ±13.03	103637.95 ±11.50

Table 4.8: Numerical result (mean cost and standard deviation) of implementing CDR routing policy on $1,000,000$ scenarios with $c_2 = 12$

From Table 4.8 we can see, the classified decision rule performs only a little worse than the SDP for Instance 2, but still works better than the TMRP. Observing this, we can develop an ad-hoc routing policy for instances with more time intervals. For example, in case where we extend the Instance 2 to 4320 time intervals and both network providers charge the ISP based on the 216th highest volume of traffic, the following decision rule was investigated:

- For time intervals $\{1, 2, ..., 216\}$, send everything to network provider 1;
- For time intervals $\{216, 217, ..., 4014\}$,
  1. if $\hat{T}_1^{215,\tau} - \hat{T}_1^{216,\tau} \geq 1500$ and $\hat{T}_2^{214,\tau} \leq 2000$, send the additional traffic $T_{Add}(S_\tau)$ to network provider 2;
  2. otherwise send the additional traffic $T_{Add}(S_\tau)$ to network provider 1.
- For time intervals $\{4105, 4106, ..., 4320\}$,
  1. if $\hat{T}_2^{215,\tau} \leq 1500$, send the additional traffic $T_{Add}(S_\tau)$ to network provider 2;
  2. otherwise send the additional traffic $T_{Add}(S_\tau)$ to network provider 1.

Index	SRP	TMRP	CDR	DRP
mean cost ± s.d	136001.02 ±2.63	135799.15±2.64	135792.52±2.76	132020.52±3.67

Table 4.9: Numerical result (mean cost and standard deviation) of implementing CDR routing policy on $1,000,000$ scenarios with $c_2 = 12$ for 4320 time intervals example

This routing decision was tested on $1,000,000$ random scenarios (where all time intervals' traffic follow the same uniform distribution $U(6000, 14000)$ as in Instance 2), the result is shown in Table 4.9. The extended routing policy performs slightly better than the TMRP, although it is still much higher than the lower bound.

Of course, the classified routing policy should depend on both the setup of traffic distribution and the cost of network providers. Though the one we give performs well in our case, it is not necessarily applicable on instances with arbitrary distributions. Here we just show the possibility of developing a crude decision rule for large size instance out of the investigation of its smaller counterpart's decision table, in case where the SDP model is definitely too large to be manageable.

4.5 Conclusions

The above experiments on the SDP routing policy show that, the TpTRP can be solved by our SDP model for small instances. We demonstrate a good routing policy which can obtain better results than the TMRP. For instances with uniform or normal traffic distributions, our corresponding SDP model gives routing polices whose numerical result on random data is just slightly greater than the lower bound (given by DRP). However, for the realistically sized problem (4320 time intervals and 5%-percentile pricing), modelling by the exact SDP yields too many states which consumes too much computer memory to run. As an alternation, we abstract a decision rule from the small size instances, which can be applied to realistically sized problems and the numerical test shows that it still outperforms the naive routing policies.

In conclusion, the DP model is a promising model for small scale TpTRP. However, more work should be done in order to make the SDP model available for realistically sized problems. For example the Approximate Dynamic Programming (ADP) as suggested by Powell [45] is a promising avenue to avoid the curse of dimensionality. In next chapter we will follow the basic SDP structure, develop an ADP model for the TpTRP.

Chapter 5

Approximate Dynamic Programming Model

In Chapter 3, we have shown that solving the TpTRP as an SMIP is intractable for all but the smallest instances, due to the fact that modelling of the top-percentile cost requires the introduction of integer variables within the final time stage. On the other hand, in Chapter 4 we suggested a Stochastic Dynamic Programming (SDP) model based on a discretization of the state space, which provides well performing routing policies for small sized instances. However due to the curse of dimensionality caused by discretization, the huge size of the state space prevents the use of this SDP model on larger problem instances.

It has been suggested by Powell [45] that Approximate Dynamic Programming (ADP) is a promising technique to avoid the curse of dimensionality. The focus of this part is on the application of ADP to solve the TpTRP.

5.1 Introduction to Approximate Dynamic Programming

5.1.1 Curse of dimensionality

All dynamic programs can be written in terms of a recursion that relates the value $V_\tau(S^\tau)$ of being in a particular state S^τ at τ to the value of the states that we are carried into at time stage $\tau+1$. In the discrete SDP model given in Chapter 4, we use a look-up table representation of $V_\tau(S^\tau)$. That is we discretize the state $S^\tau = \{\hat{T}_i^{j,\tau} | i = 1,...,n; j = 1,...,\theta\}$ by restricting $\hat{T}_i^{j,\tau}$ to be one of L possible values. Since $\hat{T}_i^{1,\tau} \geq \hat{T}_i^{2,\tau} \geq ... \geq \hat{T}_i^{\theta,\tau}, \forall i \in I$, this gives a total of $C^\theta_{L+\theta-1} = \begin{pmatrix} L+\theta-1 \\ \theta \end{pmatrix}$ [1] different states S_i^τ for provider i and a total of $\begin{pmatrix} L+\theta-1 \\ \theta \end{pmatrix}^n$ different values for S^τ.

Traditional SDP calculates and tabulates a value $V_\tau(S^\tau)$ for each possible state and time period, resulting in a total time and memory complexity of $|\Gamma| \begin{pmatrix} L+\theta-1 \\ \theta \end{pmatrix}^n$. The resulting exponential increase with θ and n is referred to in Powell [45] as the 'first curse of dimensionality' – the dimensionality in state space.

5.1.2 Main concepts in ADP

The SDP model in Chapter 4 is hit by the curse of dimensionality in two ways: first we need to evaluate $V_\tau(S^\tau)$ for an exponential (in θ and n) number of states and then we need to store these values. Approximate Dynamic Programming (ADP) avoids these by two modifications:

[1] $\begin{pmatrix} L+\theta-1 \\ \theta \end{pmatrix}$ is the number of possibilities which satisfies $\hat{T}_i^{j,\tau} \in \Omega^\tau, j = 1,...,\theta$ and $\hat{T}_i^{1,\tau} \geq \hat{T}_i^{2,\tau} \geq ... \geq \hat{T}_i^{\theta,\tau}$.

Value function approximation

Instead of a look-up table, ADP approximates the value function $V_\tau(S^\tau)$ by a continuous surrogate model with a small number of parameters that need to be estimated.

Step forward in time

The other important difference is that ADP is based on an algorithmic strategy that steps forward through time, rather than backward in SDP. Let $\bar{V}_\tau^{(m)}$ represent the value function approximation for time interval τ after being trained through (m) iterations. To solve ADP we choose a sample scenario for every iteration and step forward in time. At time interval τ, we solve the decision problem (see equation (5.1.1) for detail) based on $\bar{V}_{\tau+1}^{(m-1)}$. Let $\hat{v}_\tau^{(m)}$ denote the optimal solution of the decision problem (5.1.1). As it is a sample estimation of $\bar{V}_\tau^{(m-1)}$, we can then update $\bar{V}_\tau^{(m-1)}$ with the value of $\hat{v}_\tau^{(m)}$. By repeating these steps, parameters of the surrogate model converge to a stable estimation of the value function.

The stochastic gradient algorithm is an optimization method for estimating the mean of a random variable over a set of random observations, which is first brought into use by Robbins and Monro [46]. It works in an iterative way, such that every time one get a sample observation on the random event, the expected value is updated according to its gradient. Generally speaking, the stochastic gradient method is observed to converge to a local minimum even when the objective function is non-convex. In ADP, we use the stochastic gradient method to estimate the value of being in a state from random samples. The process can be interpreted as applying the stochastic gradient method to the problem of finding an optimal regression function $\bar{V}_\tau(S^\tau)$ for $V_\tau(S^\tau)$.

Note that with a continuous regression model representation of the value function, although we follow a single scenario during every iteration we effectively update the value function approximation for all states when changing its coefficients. This makes the process more efficient than traditional dynamic programming, in which by each computation we obtain the value for a single, discrete state. In addition, ADP focuses computational effort on the states which are more likely to be visited, rather than treating all possible states as equally important.

5.1.3 Main procedure of ADP

The basic approximate dynamic programming algorithm is summarised below: (Powell [45])

Step 0. Initialisation:

Step 0a. Build an initial value function approximation $\bar{V}_\tau^{(0)}(S^\tau)$ for all time intervals τ.

Step 0b. Choose an initial state $S_{(1)}^1$.

Step 0c. Set $m = 1$.

Step 1. Choose a sample path $\omega_{(m)} = (\omega_{(m)}^1, ..., \omega_{(m)}^{|\Gamma|})$.

Step 2. For $\tau = 0, 1, 2, ..., |\Gamma|$ do:

Step 2a. Solve

$$\hat{v}_\tau^{(m)} = \min_{x^\tau \in \chi^\tau} (\mathbb{E}_{\omega^\tau \in \Omega^\tau} \bar{V}_{\tau+1}^{(m-1)}(S^{\tau+1}|S_{(m)}^\tau, \omega^\tau, x^\tau)). \tag{5.1.1}$$

Step 2b. Update the value function approximation $\bar{V}_\tau^{(m-1)}(S^\tau)$ with the value of $\hat{v}_\tau^{(m)}$ by stochastic gradient method (see section 5.2.3 for detail).

Step 2c. Compute $S_{(m)}^{\tau+1}(S_{(m)}^\tau, \omega_{(m)}^\tau, \hat{x}^\tau)$, where $\hat{x}^\tau$ is the optimal solution of (5.1.1).

Step 3. If we have not met our stopping rule, let $m = m + 1$ and go to step 1.

5.2 ADP model for TpTRP

5.2.1 Value function approximation – regression model

As discussed in Section 5.1.1, the traditional look-up-table representation of the value function suffers from the curse of dimensionality. To estimate the value function with as few parameters as possible, in ADP we use a regression model on a continuous state space to approximate the value function. In order to get a good fit to the real value function, we need to use a form that is suited to the shape of the true value function if possible. Thus before discussing an appropriate regression model we examine several examples of value functions given by the SDP

model in Chapter 4, to investigate their structure.

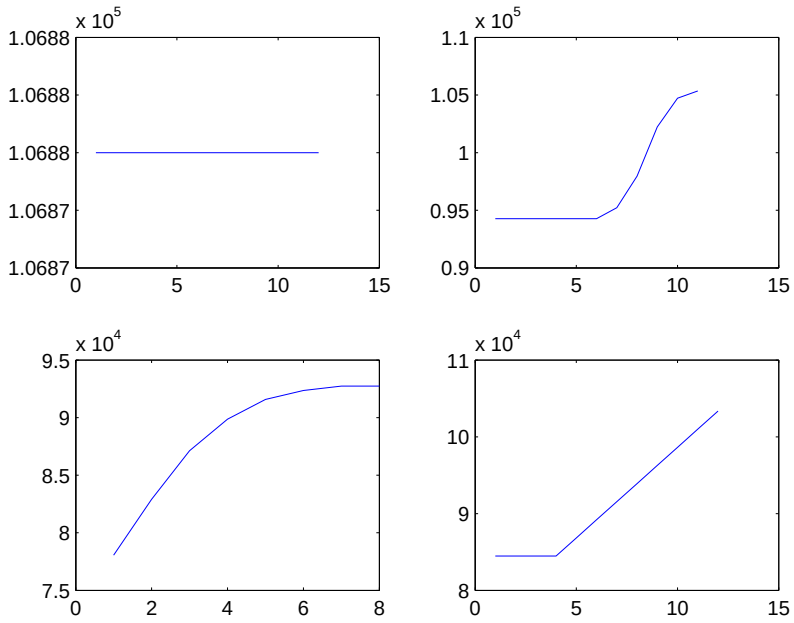


Figure 5.2.1: Examples of how function value $V_\tau(S^\tau)$ changes with a single entry of state variable (i.e. $\hat{T}_i^{j,\tau}$)

Figure 5.2.1 shows four examples of how the value function varies with a single entry of the state that are experimentally obtained from the SDP model. Although the value of $V_\tau(S^\tau)$ shown in these figures might not be exact since the SDP model itself is an approximation of the original problem (resulting from a discretization of the state space and restrictions on the decision space), we can still get some insight of the basic character of the value function. From these four specific examples we can see that the value function is in general neither a convex nor a concave function, sometimes it is not even smooth. Using the observation that some states (e.g. very extreme ones) in the SDP model are much less important than others, as they are rarely visited, we should focus more on the central part of the value functions.

In Lemma 4.2.1 we have proved that the value function is non-decreasing in every entry of the state. In this work we use the simplest model, linear regression to approximate the value function. Namely at state S^τ, we approximate $V_\tau(S^\tau)$ by:

$$\bar{V}_\tau(S^\tau) = \beta_0^\tau + \sum_{i\in I} \sum_{1\le j\le \theta} \beta_{i,j}^\tau \hat{T}_i^{j,\tau}. \tag{5.2.1}$$

Although there may be more sophisticated choices, e.g. quadratic regression models which include $(T_i^{j,\tau})^2$ and $(T_i^{j,\tau}T_{\tilde{i}}^{j,\tau}), \forall i \neq \tilde{i})$, it is non-obvious to decide which cross-terms are significant while which others are not. Simply including all cross-terms leads to far too many parameters. Therefore, we feel that the linear model will give a good trade-off between providing a satisfactory approximation, maintaining computability and avoiding the risk of over-fitting. It also provides robustness against possible spurious behaviour (locally decreasing approximations) exhibited by complex value function approximations. Actually, although it is very hard to numerically examine β values for all time intervals τ and indexes i, j, we can still expect that values of $\beta_{i,\theta}^{\tau}$ are comparatively higher (impact more) than their counterpart $\beta_{i,j}^{\tau}, 1 \leq j \leq \theta - 1$, particularly for time intervals toward the end of charging period. This property of β values can be justified with numerical results given in Section 5.3, which also justifies the choice of linear regression model.

5.2.2 Decision problem

Step 2a of the ADP algorithm requires the solution of the decision problem. In the m-th iteration, the decision problem at time interval τ is

$$\hat{v}_{\tau}^{(m)} = \min_{x^{\tau} \in \chi^{\tau}} (\mathbb{E}_{\omega^{\tau} \in \Omega^{\tau}} \bar{V}_{\tau+1}^{(m-1)}(S^{\tau+1} | S_{(m)}^{\tau}, \omega^{\tau}, x^{\tau})), \tag{5.2.2}$$

where $\bar{V}_{\tau+1}^{(m-1)}$, as given by (5.2.1), is the approximation of the value function $V_{\tau+1}$ built with the estimated coefficients after $m-1$ iterations. The decision problem (5.2.2) is a minimisation problem, whose objective is an expectation of the value function estimation in the next time stage. The feasible decision set χ^{τ}, as discussed in the end of Section 4.2.3, consists of only the proportional decision x^{τ} made on the routing of the additional traffic, while the cut-off level y^{τ} is determined by Lemma 4.2.2.

It is worth investigating the exact form of this objective function. As given in (5.2.1), $\bar{V}_{\tau+1}$ is a linear function of the state $S^{\tau+1} = \{\hat{T}_i^{j,\tau}\}$. Further $\bar{V}_{\tau}$ decomposes by network provider, that is if we define $\bar{V}_{\tau,i}(S_i^{\tau}) = \sum_{1 \leq j \leq \theta} \beta_{i,j}^{\tau} \hat{T}_i^{j,\tau}$, then we can write $\bar{V}_{\tau}(S^{\tau}) = \beta_0^{\tau} + \sum_{i \in I} \bar{V}_{\tau,i}(S_i^{\tau})$. On the other hand, the state $S_i^{\tau+1}$ is obtained from the state S_i^{τ}, the decision x^{τ} and the realisation of random

traffic $\hat{T}^\tau = T^\tau(\hat{\omega}^\tau)$ by applying the implementation rule given in Section 4.2.3 to obtain the new traffic for network provider i, ${}^{new}T_i^\tau$ and then reordering entries in non-increasing order. It is easy to see that for every given realisation $\hat{\omega}^\tau$, $\bar{V}_{\tau+1,i}(S_i^{\tau+1}|S_i^\tau, \hat{\omega}^\tau, x_i^\tau))$ is a piecewise linear function of x_i^τ. In principle it is possible to give an analytic expression for $\bar{V}_{\tau+1,i}$, as in

$$\bar{V}_{\tau+1,i}(S_i^{\tau+1}|S_i^\tau, \omega^\tau, x_i^\tau)) = \begin{cases} \sum\limits_{1\le j\le\theta} \beta_{i,j}^{\tau+1}\hat{T}_i^{j,\tau} \quad \text{for} \quad T^\tau(\hat{\omega}^\tau) \le \sum\limits_{i\in I}\hat{T}_i^{\theta,\tau}; \\ \sum\limits_{1\le j\le\theta-1} \beta_{i,j}^{\tau+1}\hat{T}_i^{j,\tau} + \beta_{i,\theta}^{\tau+1}(\hat{T}_i^{\theta,\tau} + x_i^\tau(T^\tau(\hat{\omega}^\tau) - \sum\limits_{i\in I}\hat{T}_i^{\theta,\tau})) \\ \qquad\qquad \text{for} \quad \sum\limits_{i\in I}\hat{T}_i^{\theta,\tau} < T^\tau(\hat{\omega}^\tau) \le \sum\limits_{i\in I}\hat{T}_i^{\theta,\tau} + \frac{\hat{T}_i^{\theta-1,\tau} - \hat{T}_i^{\theta,\tau}}{x_i^\tau}; \\ \sum\limits_{1\le j\le\theta-2} \beta_{i,j}^{\tau+1}\hat{T}_i^{j,\tau} + \beta_{i,\theta-1}^{\tau+1}(\hat{T}_i^{\theta,\tau} + x_i^\tau(T^\tau(\hat{\omega}^\tau) - \sum\limits_{i\in I}\hat{T}_i^{\theta,\tau})) + \beta_{i,\theta}^{\tau+1}\hat{T}_i^{\theta-1,\tau} \\ \qquad\qquad \text{for} \quad \sum\limits_{i\in I}\hat{T}_i^{\theta,\tau} + \frac{\hat{T}_i^{\theta-1,\tau} - \hat{T}_i^{\theta,\tau}}{x_i^\tau} < T^\tau(\hat{\omega}^\tau) \le \sum\limits_{i\in I}\hat{T}_i^{\theta,\tau} + \frac{\hat{T}_i^{\theta-2,\tau} - \hat{T}_i^{\theta,\tau}}{x_i^\tau}; \\ \vdots \\ \beta_{i,1}^{\tau+1}(\hat{T}_i^{\theta,\tau} + x_i^\tau(T^\tau(\hat{\omega}^\tau) - \sum\limits_{i\in I}\hat{T}_i^{\theta,\tau})) + \sum\limits_{2\le j\le\theta} \beta_{i,j}^{\tau+1}\hat{T}_i^{j-1,\tau} \\ \qquad\qquad \text{for} \quad \sum\limits_{i\in I}\hat{T}_i^{\theta,\tau} + \frac{\hat{T}_i^{1,\tau} - \hat{T}_i^{\theta,\tau}}{x_i^\tau} < T^\tau(\hat{\omega}^\tau). \end{cases}$$

With this the objective function of the decision problem becomes:

$$\mathbb{E}_{\omega^\tau}(\bar{V}_{\tau+1}(S^{\tau+1}|S^\tau, \omega^\tau, x^\tau)) = \int_{\omega^\tau} f(\omega^\tau)\beta_0^\tau \, d\omega^\tau + \sum_{i\in I}\int_{\omega^\tau} f(\omega^\tau)\bar{V}_{\tau+1,i}(S_i^{\tau+1}|S_i^\tau, \omega^\tau, x_i^\tau))d\omega^\tau, \tag{5.2.3}$$

which is difficult to simplify further due to the complex form of $\bar{V}_{\tau+1,i}(S_i^{\tau+1}|S_i^\tau, \omega^\tau, x_i^\tau))$. This means, we cannot generally write down an analytical expression (like what we did for $\bar{V}_{\tau+1,i}(S_i^{\tau+1}|S_i^\tau, \omega^\tau, x_i^\tau)))$ for the expectation (5.2.3), as a piecewise function of x^τ.

In the solution of the ADP model, decision problem (5.2.2) changes from iteration to iteration, according to the current estimation of regression parameters. This has been confirmed numerically by Monte Carlo simulation. In addition, it is obvious that $\bar{V}_{\tau+1,i}(S_i^{\tau+1}|S_i^\tau, \omega^\tau, x_i^\tau))$ is continuous and monotone in x_i^τ for every single network provider i. Nevertheless, combining them together (on the subspace given by $\sum\limits_{i\in I} x_i^\tau = 1$) and taking the expectation over random traffic, there is no reason to expect the resulting objective function of the decision problem (5.2.3) to be either monotone, or convex. Therefore an algorithm based on cutting planes as suggested by Powell [45] cannot be used. In addition, since the objective is given by an expectation (5.2.3) which we are unable to evaluate

analytically, any function or gradient evaluations are expensive and inexact.

As we need to solve the decision problem at every time stage for every iteration, we require an optimisation method that is efficient (solves the problem in reasonable time) and reliable (finds the optimal or near optimal solution). We have settled to solving this problem by a simple discretization of the decision space, i.e., generating several discrete decisions (for example $\chi_i^\tau = \{0.0, 0.1, 0.2...1.0\}$), calculating their objective value according to expression (5.2.3) and choosing the best one. Although we cannot find the exact optimal solution of the decision problem with this method, it gives a good compromise between speed and accuracy. We can see from the numerical results given in Section 5.3.2 that the practical advantage from solving the decision problem more accurately is minimal.

Computational complexity analysis

To train the ADP model, we need to solve the decision problem (5.2.2) for every time stage during every iteration (with different parameters). This forms a crucial part in the ADP scheme. It is worthwhile to discuss the computational complexity of it.

As said above, the decision problem is solved by simple enumeration. The number of discrete decisions to evaluate increases exponentially with n, the number of network providers available to use. With the same number of discretizations in every provider (where in this work we use $|\mathcal{D}| = 11$ in all experiments, namely we numerically examine $x_i^\tau \in \mathcal{D} = \{0.0, 0.1, 0.2...1.0\}$ and select the one with minimum objective as the final decision), instances with n network providers possess a decision pool with $|\mathcal{D}|^{n-1} = 11^{n-1}$ feasible decisions (as $\sum_{i \in I} x_i^\tau = 1$ restricts the freedom in one dimension of n). Nevertheless, the difficulty of searching for a solution does not increase exponentially with the number of network providers. This is caused by the fact that, the solution of the decision problem consists of two parts: evaluating expectations for all feasible decisions and comparing them to find the best solution. We discuss them separately.

1. Evaluating expectations for all feasible decisions:

   In our problem the state decomposes by provider: $S^\tau = \{S_i^\tau | i = 1, ..., n\}$, where $S_i^\tau = \{\hat{T}_i^{j,\tau} | j = 1, ..., \theta\}$. According to the state dynamic we discussed

in Section 4.2.4, we can see that Bellman's equation which connects $V_{\tau+1}$ to V_τ is separable by network providers as well. Thus for every $i \in I$ and all entries in $\mathcal{D}$, we can calculate

$$\mathbb{E}_{\omega^\tau}(\bar{V}_{\tau+1,i}(S_i^{\tau+1}|S_i^\tau, \omega^\tau, x_i^\tau)) = \int_{\omega^\tau} f(\omega^\tau)\bar{V}_{\tau+1,i}(S_i^{\tau+1}|S_i^\tau, \omega^\tau, x_i^\tau))d\omega^\tau. \tag{5.2.4}$$

The integrand $\bar{V}_{\tau+1,i}(S_i^{\tau+1}|S_i^\tau, \omega^\tau, x_i^\tau))$ is a piecewise linear function of x_i^τ (due to the reordering of traffic volumes in the enlarged state (4.2.4)) consisting of θ pieces, where the integral over each piece requires θ multiplications. Thus to evaluate the expectation in (5.2.3), it requires a computational complexity of $\mathcal{O}(\theta^2)$. For a problem with n network providers and $|\mathcal{D}|$ discrete decisions to evaluate, this step introduces a total complexity of $n|\mathcal{D}|\mathcal{O}(\theta^2)$.

2. Comparing the expected cost of all feasible decisions to find the best solution:

   Given values of $\mathbb{E}_{\omega^\tau}(\bar{V}_{\tau+1,i}(S_i^{\tau+1}|S_i^\tau, \omega^\tau, x_i^\tau))$ for all $i \in I$, finding the expectation corresponding to decision $x^\tau = \{x_i^\tau | \sum_{i\in I} x_i^\tau = 1, i \in I\} \in |\mathcal{D}|^n$ requires a summation of $\mathbb{E}_{\omega^\tau}(\bar{V}_{\tau+1,i}(S_i^{\tau+1}|S_i^\tau, \omega^\tau, x_i^\tau))$ over i. Comparing all feasible decisions we select the best one as the solution. While this step has a complexity of $\mathcal{O}(n|\mathcal{D}|^{n-1})$, the calculations involved are trivial, and for reasonable numbers of network providers ($n < 5$) computation time is dominated by step 1 with linear complexity. (Numerical experiments show that the expectation calculation step takes around 90% of the time in solving the decision problem for $n = 2$ instances and 89% for $n = 3$ instances). Numerical results on instances with 2 and 3 network providers in Section 5.3.3 confirm that indeed the solution time grows linearly with n.

Consideration of the post-decision state

It is strongly recommended by Powell [45] to use post-decision states, in order to avoid the difficulty of computing the expectation within the decision problem (5.2.2). Unlike the current state variable S^τ which we are using, the post-decision state variable $\tilde{S}_x^\tau$ is defined as the state immediately after we make a decision x. Let $\tilde{V}_\tau^x(\tilde{S}_x^\tau)$ be the value of being in state $\tilde{S}_x^\tau$, we have:

$$V_\tau(S^\tau) = \min_{x^\tau \in \chi^\tau} (C_\tau(S^\tau, x^\tau) + \tilde{V}_\tau^x(\tilde{S}_x^\tau)), \tag{5.2.5}$$

where $C_\tau(S^\tau, x^\tau)$ indicates the immediate cost by taking action x^τ at state S^τ. Note that in the TpTRP there is no such term. This formulation is for the general case of ADP. We also have:

$$\tilde{V}_\tau^x(\tilde{S}_x^\tau) = \mathbb{E}_{\omega^\tau}(V_{\tau+1}(S^{\tau+1})|\tilde{S}_x^\tau). \tag{5.2.6}$$

Substituting (5.2.4) into (5.2.5), we obtain the optimality equation around the post-decision state variable:

$$\tilde{V}_\tau^x(\tilde{S}_x^\tau) = \mathbb{E}_{\omega^\tau}\{\min_{x^{\tau+1} \in \chi^{\tau+1}} (C_{\tau+1}(S^{\tau+1}, x^{\tau+1}) + \tilde{V}_{\tau+1}^x(\tilde{S}_x^{\tau+1}))|\tilde{S}_x^\tau\}.$$

The expectation is now outside the minimization operator. Therefore, if we can find a nice approximation of $\tilde{V}_x^\tau(\tilde{S}_x^\tau)$, denoted by $\bar{V}_x^\tau$, the decision problem then becomes:

$$\hat{v}_\tau^{(m)} = \min_{x^\tau \in \chi^\tau} (C_\tau(S_{(m)}^\tau, x^\tau) + \bar{V}_{x,(m-1)}^\tau(\tilde{S}_{x,(m)}^\tau)). \tag{5.2.7}$$

Thus the decision problem is now a deterministic optimization problem, which should be computationally easier to solve. On the other hand, every time we solve (5.2.6) to optimality, the optimal objective $\hat{v}_\tau^{(m)}$ forms a sample of the value of being in state $\tilde{S}_{x,(m)}^{(\tau-1)}$. Thus we can update the post-decision value function approximation according to it.

Unfortunately, it seems that the use of post-decision states is not beneficial in the given setup of TpTRP. The reasons are as follows:

- We can find no better choice for the post-decision state than the augmented state (S^τ, x^τ), since the decision x^τ can provide no valuable information on the current state variable without knowing the traffic realisation.
- As we choose to define the post-decision state as the augmented state, the problem becomes a Q-learning problem (as discussed in Chapter 8 in

[45]). But the structure of the TpTRP (namely that only the final stage contributes to cost) means there is no immediate cost $C_\tau(S^\tau, x^\tau)$ for any stage $\tau = 1, ..., |\Gamma| - 1$. Thus, the Q-factor $Q_\tau^x = \bar{V}_x^\tau(S^\tau, x^\tau)$ which is minimised by the decision problem is solely described by the type of function used to approximate $\bar{V}_x^\tau$ around the post-decision state (S^τ, x^τ). The proper choice of regression model for $\bar{V}_x^\tau(S^\tau, x^\tau)$ is crucial now for success. In particular our linear model cannot be used anymore.

- Of course, we can still attempt to find a 'proper' type of function which can catch all the features of the value function around the post-decision state, though it will be very difficult and dangerous. Any choice for $\bar{V}_x^\tau(S^\tau, x^\tau)$ that is separable (i.e. $\bar{V}_x^\tau(S^\tau, x^\tau) = \bar{V}_x^\tau(S^\tau) + \bar{V}_x^\tau(x^\tau)$), in particular our original linear model, would result in optimal decisions x that are independent of the current state, which is clearly not desirable. The regression model would therefore need to include non-separable terms (such as $T_i^{j,\tau} x_{\tilde{i}}^\tau; \forall i, \tilde{i} \in I, \forall 1 \leq j \leq \theta$), the choice of which is non-obvious. Simply including all cross-terms leads to far too many parameters.

With the above arguments, we think the post-decision state is not making the problem easier to solve, thus we keep our current model with the pre-decision state.

5.2.3 Recursive methods for regression model – parameter estimation

We assume we are given an initial approximation $\bar{V}_\tau^{(0)}(S^\tau)$ of $V_\tau(S^\tau)$ for all τ. In iteration m, we update $\bar{V}_\tau^{(m)}$ from its previous estimate $\bar{V}_\tau^{(m-1)}$. As in the parametrised model (5.2.1), the value function approximation after m iterations can be written as $\bar{V}_\tau^{(m)}(S^\tau) = \bar{V}_\tau(S^\tau, \beta^{\tau,(m)})$, where $\beta^{\tau,(m)} = \{\beta_0^{\tau,(m)}, \beta_{i,j}^{\tau,(m)} | i \in I; j = 1, ..., \theta\}$. To final the optimal parameters for the regression model, for every time interval τ we want to find β^τ that solve:

$$\min_{\beta^\tau} \mathbb{E}_{\omega^\tau}[(\bar{V}_\tau(S_{\omega^\tau}^\tau, \beta^\tau) - \hat{v}_\tau^{(m)}(S_{\omega^\tau}^\tau))^2], \tag{5.2.8}$$

where $\hat{v}_\tau^{(m)}(S_{\omega^\tau}^\tau)$ is the sample estimate of the real function value $V_\tau(S^\tau)$ obtained

by solving (5.2.2). In ADP model however, we approximate the optimal solution of (5.2.8) using an iteratively updating scheme, which is known as the stochastic gradient algorithm by Robbins and Monro [46]:

$$\beta^{\tau,(m)} = \beta^{\tau,(m-1)} - \alpha_{m-1}[\bar{V}_\tau(S^\tau_{\omega^\tau_{(m)}}, \beta^{\tau,(m)}) - \hat{v}^{(m)}_\tau(S^\tau_{\omega^\tau_{(m)}})]\nabla_{\beta^{\tau,(m-1)}}\bar{V}_\tau(S^\tau_{\omega^\tau_{(m)}}, \beta^{\tau,(m)}). \tag{5.2.9}$$

The value α_m in formula (5.2.9) is the updating step-size from iteration m to $m+1$, which tells us how far we should go in the direction of $\nabla_{\beta^{\tau,(m-1)}}\bar{V}_\tau(S^\tau_{\omega^\tau_{(m)}}, \beta^{\tau,(m)})$. Finding the proper step-size α_m is one of the challenges in ADP. Theoretical analyses of the stochastic gradient algorithm (see Kushner and Yin [34], Wasan [59]) requires step-sizes satisfying $\sum \alpha_m = \infty, \sum \alpha_m^2 < \infty$ for convergence. However, in the given situation where the target $\hat{v}_\tau$ is changing with each iteration, which is the case in ADP as in ADP $\hat{v}^{(m)}_\tau$ denotes the sample estimation of value function which is also defined over time, the typical $1/n$ rule puts too large a weight to the earlier iterations, leading to the premature convergence (termed "apparent convergence" in Powell [45]). In this work we use McClain's formula [39]:

$$\alpha^{MC}_m = \frac{\alpha^{MC}_{m-1}}{(1 + \alpha^{MC}_{m-1} - \bar{\alpha})},$$

where $\bar{\alpha}$ is a specified parameter. This step-size is a typical deterministic step-size suggested by W.B. Powell in [45] Chapter 6, which avoids dropping to zero too quickly. Steps generated by McClain's formula satisfy $\alpha^{MC}_m > \alpha^{MC}_{m+1} > \bar{\alpha}$. McClain's rule combines the features of the $1/n$ rule which is ideal for stationary data and constant step-sizes for non-stationary data.

In addition to the 'smoothing factor' ($0 < \alpha_m \leq 1$), an important practical problem is the scaling of units of the left hand side and the right hand side in the updating equation (5.2.9). Since the value of $\beta^{\tau,(m-1)}$ and the gradient $\nabla_{\beta^{\tau,(m-1)}}\bar{V}_\tau(S^\tau_{\omega^\tau_{(m)}}, \beta^{\tau,(m)})$ (which is actually $\hat{T}^\tau_{i,j}$ according to (5.2.1)) may possess completely different scale, we follow the suggestion of Powell [45] to use an adaptively chosen α_0 to cover this difference. Thus our step-size consists of two components, which means $\alpha_m = \alpha_0\alpha^{MC}_m$. As we expect the $\beta^{\tau,(m)}$ to move monotonically at the beginning of the algorithm and start to alternate near convergence, we will increase α_0 as long as we observe monotonic behaviour in the

$\beta^{\tau,(m)}$ for the first few iterations and decrease it otherwise.

5.2.4 Stopping criterion

As in ADP we update the value function estimation iteratively, the question of when to stop becomes an important practical issue. Generally speaking, in ADP we expect to end up with a converged set of coefficients for our regression model. However, as we introduced many parameters in the value function estimation, it is hard to define a single guideline for convergence which works well for all coefficients. In addition, stochastic gradient algorithms typically converge rapidly at the beginning and then vibrate with noise. As we are interested primarily in a sensible routing policy rather than the exact optimal regression model, we use a heuristic that numerically evaluates the mean cost over every $10,000$ runs and stops once we observe alternating values rather than monotonic increase or decrease. For more detail please see Section 5.3.1.

5.2.5 ADP algorithm for TpTRP

Thus the complete ADP algorithm for our TpTRP is summarised below:

Step 0. Initialisation:

Step 0a. Build value function approximation

$$\bar{V}_\tau(S^\tau, \beta^\tau) = \beta_0^\tau + \sum_{i \in I} \sum_{1 \le j \le \theta} \beta_{i,j}^\tau \hat{T}_i^{j,\tau},$$

with initial choice $\beta_0^{\tau,(0)} = 0, \beta_{i,j}^{\tau,(0)} = 0, 0 \le j \le \theta, \forall i \in I, \forall \tau \in \Gamma$.

Step 0b. Choose an initial state $S_{(1)}^1$, set $m = 1$.

Step 1. Choose a sample path $\omega_{(m)} = (\omega_{(m)}^1, ..., \omega_{(m)}^\Gamma)$.

Step 2. For $\tau = 0, 1, 2, ..., \Gamma$ do:

Step 2a. Solve

$$\hat{v}_\tau^{(m)}(S_{(m)}^\tau) = \min_{x^\tau \in \chi^\tau} (\mathbb{E}_{\omega^\tau \in \Omega^\tau} \bar{V}_{\tau+1}^{(m-1)}(S^{\tau+1} | S_{(m)}^\tau, \omega^\tau, x^\tau)) \tag{5.2.10}$$

by selecting the best one from $\chi^\tau = \{x_i^\tau \in \{0.0, 0.1, 0.2...1.0\} | \sum_{i \in I} x_i^\tau = 1\}$.

Step 2b. Let $\beta^{\tau,(m)} = \{\beta_0^{\tau,(m)}, \beta_{i,j}^{\tau,(m)} | i \in I; j = 1, ..., \theta\}$ indicate the β values

after m iterations, update $\bar{V}_\tau^{(m-1)}$ by stochastic gradient algorithm:

$$\beta^{\tau,(m)} = \beta^{\tau,(m-1)} - \alpha_{m-1}[\bar{V}_\tau(S^\tau_{(m)}, \beta^{\tau,(m)}) - \hat{v}^{(m)}_\tau(S^\tau_{(m)})]\nabla_{\beta^{\tau,(m-1)}}\bar{V}_\tau(S^\tau_{(m)}, \beta^{\tau,(m)})$$

with $\alpha_m = \alpha_0 \frac{\alpha_{m-1}}{(1+\alpha_{m-1}-\bar{\alpha})}$.

Step 2c. Compute $S^{\tau+1}_{(m)}(S^\tau_{(m)}, \omega^\tau_{(m)}, \hat{x}^\tau)$, where $\hat{x}^\tau$ is the solution of (5.2.10).

Step 3. If we have not met our stopping rule described in Section 4.4, let $m = m + 1$ and go to step 1.

5.3 Numerical results

5.3.1 Numerical results on TpTRP instances with 10 periods

In this section we give numerical results on several small instances of the TpTRP which we used to test the SDP model in Chapter 4. We use the same instance index system as before.

Mean cost

To evaluate the quality of the routing policy obtained from the ADP model, we examine it in a simulation of $1,000,000$ random scenarios taken from the original distribution on all the instances shown in Section 4.4.1. As before we compare the results against the following benchmarks:

- SRP - Single-homing Routing Policy, i.e. send everything to the cheapest network provider – provider 1;
- TMRP - Trivial Multi-homing Routing Policy, i.e. send traffic volumes in the $\theta - 1$ periods with highest expected mean to the expensive provider and all the rest to the cheaper one. In this way the ISP is only charged by the cheapest network provider, but uses the free time intervals of all network providers;

- SDPRP - Stochastic Dynamic Programming Routing Policy given as a discrete look-up table by solving the SDP model in Chapter 4 , which requires discretization of the traffic region. We repeat the model with different number of discretization levels (L in Table 5.1) used;

- DRP - Deterministic Routing Policy, i.e. assuming we know all traffic volumes in advance. The optimal routing policy (as proved in Section 2.3.1) is to send the $\theta - 1$ highest traffic volumes to the expensive provider and the rest to the cheaper one. Note that as we assume that we have full knowledge of the traffic ahead in time, the DRP is not implementable. It provides us with lower bound on all the stochastic routing policies.

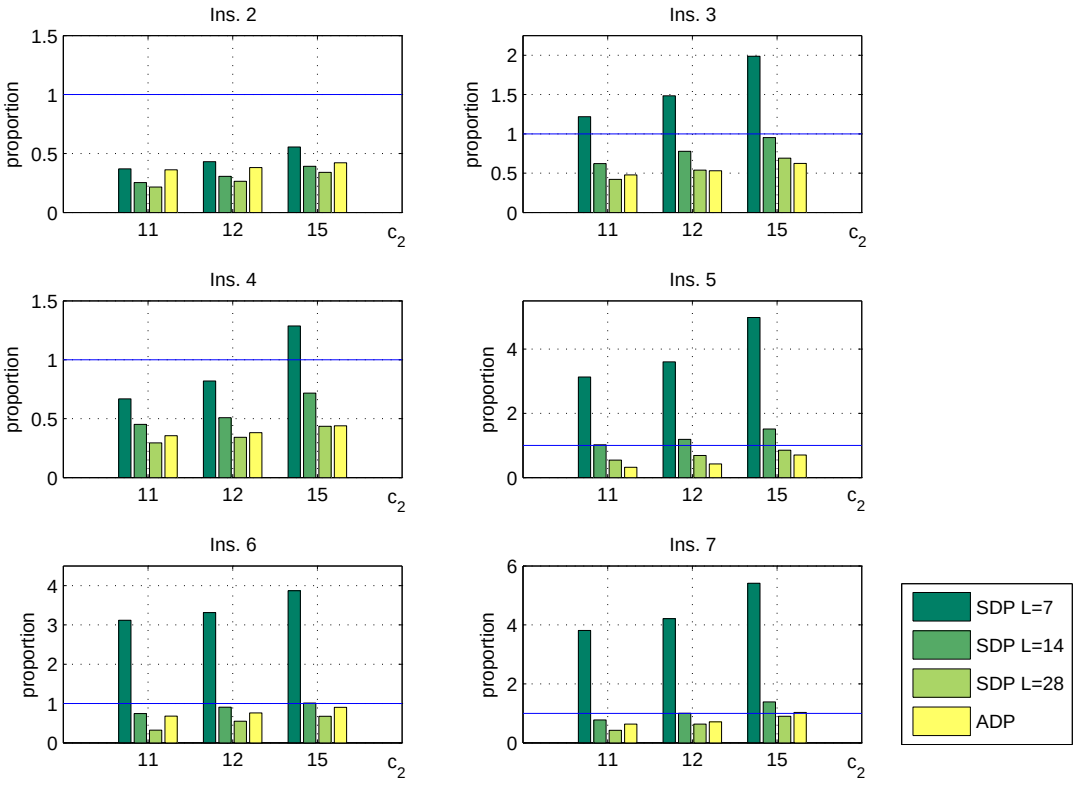


Figure 5.3.1: Comparison of mean cost between SDP and ADP routing policies

Numerical results on mean cost and standard deviation with different cost ratios c_2/c_1 are summarised in Table 5.1, 5.2, 5.3 and Figure 5.3.1. In Figure 5.3.1, each sub-figure is corresponding to one instance and consisting of 3 groups of columns, which in order give the relevant results for $c_2 = 11, c_2 = 12$ and $c_2 = 15$. Each group consists of 4 columns, 3 in green and 1 in yellow. Green columns

Instance	SRP	TMRP	L	SDPRP	ADPRP	DRP
Instance 2	118178.53±10.28	113335.54±11.93	7	107219.29±12.69	107140.45±12.35	103637.95±11.50
			14	106090.86±12.18		
			28	105734.36±12.03		
Instance 3	114340.25±7.52	104294.38±7.86	7	104728.46±8.15	103254.88±7.45	102303.09±6.99
			14	103543.13±7.53		
			28	103140.84±7.42		
Instance 4	106564.38±4.18	104727.88±4.48	7	103564.95±5.25	102470.25±4.30	101226.33±3.89
			14	102808.01±4.51		
			28	102255.24±4.11		
Instance 5	112379.96±4.53	105986.18±4.95	7	108078.29±5.89	105315.85±4.57	105003.32±4.37
			14	106005.25±4.83		
			28	105536.23±4.64		
Instance 6	113697.78±6.60	103859.41±6.28	7	106554.03±7.65	103450.85±6.15	102587.85±5.35
			14	103536.49±6.10		
			28	102996.71±5.54		
Instance 7	113404.28±5.48	103307.22±5.67	7	105041.39±7.35	103082.86±5.77	102690.17±5.32
			14	103170.72±6.15		
			28	102953.15±6.34		

Table 5.1: Numerical results (mean cost ± s.d.) of implementing routing policies on $1,000,000$ random scenarios with $c_2 = 11$

Instance	SRP	TMRP	L	SDPRP	ADPRP	DRP
Instance 2	118178.53±10.28	113335.54±11.93	7	107811.61±12.90	107335.60±12.35	103637.95±11.50
			14	106602.73±12.33		
			28	106203.10±12.17		
Instance 3	114340.25±7.52	104294.38±7.86	7	105256.80±8.44	103361.52±7.50	102303.09±6.99
			14	103853.99±7.71		
			28	103375.22±7.54		
Instance 4	106564.38±4.18	104727.88±4.48	7	104097.79±5.54	102561.69±4.31	101226.33±3.89
			14	103007.73±4.58		
			28	102424.21±4.18		
Instance 5	112379.96±4.53	105986.18±4.95	7	108541.85±6.09	105418.83±4.78	105003.32±4.37
			14	106172.45±4.89		
			28	105677.18±4.73		
Instance 6	113697.78±6.60	103859.41±6.28	7	106803.13±7.64	103553.42±5.97	102587.85±5.35
			14	103743.01±6.24		
			28	103285.52±5.73		
Instance 7	113404.28±5.48	103307.22±5.67	7	105290.33±7.34	103130.13±6.11	102690.17±5.32
			14	103309.03±6.19		
			28	103081.45±6.18		

Table 5.2: Numerical results (mean cost ± s.d.) of implementing routing policies on $1,000,000$ random scenarios with $c_2 = 12$

show the performance of SDP routing policy with 7, 14 or 28 discretization levels in turn, while the yellow column shows the performance of ADP routing policy. Performance in these figures are given in a proportional manner, where the value of a column corresponding to routing policy $\mathbf{X}$ is equal to ($\mathbf{X}$-DRP)/(TMRP-DRP). Namely 0 means $\mathbf{X}$ works as well as DRP, while 1 means it works same as TMRP. Therefore, any routing policy that is above 1 (above the horizontal line) is obviously not desirable.

Instance	SRP	TMRP	L	SDPRP	ADPRP	DRP
Instance 2	118178.53±10.28	113335.54±11.93	7	109022.21±13.42	107724.39±12.32	103637.95±11.50
			14	107432.60±12.60		
			28	106932.48±12.38		
Instance 3	114340.25±7.52	104294.38±7.86	7	106260.86±9.04	103544.28±7.58	102303.09±6.99
			14	104197.98±7.92		
			28	103679.88±7.67		
Instance 4	106564.38±4.18	104727.88±4.48	7	105733.64±6.65	102766.08±4.35	101226.33±3.89
			14	103736.17±4.98		
			28	102750.73±4.37		
Instance 5	112379.96±4.53	105986.18±4.95	7	109898.90±6.98	105694.71±5.13	105003.32±4.37
			14	106490.32±5.16		
			28	105837.72±4.89		
Instance 6	113697.78±6.60	103859.41±6.28	7	107506.27±7.64	103737.08±6.22	102587.85±5.35
			14	103876.07±6.26		
			28	103444.19±5.58		
Instance 7	113404.28±5.48	103307.22±5.67	7	106030.33±7.48	103327.54±6.10	102690.17±5.32
			14	103547.28±6.24		
			28	103245.73±6.14		

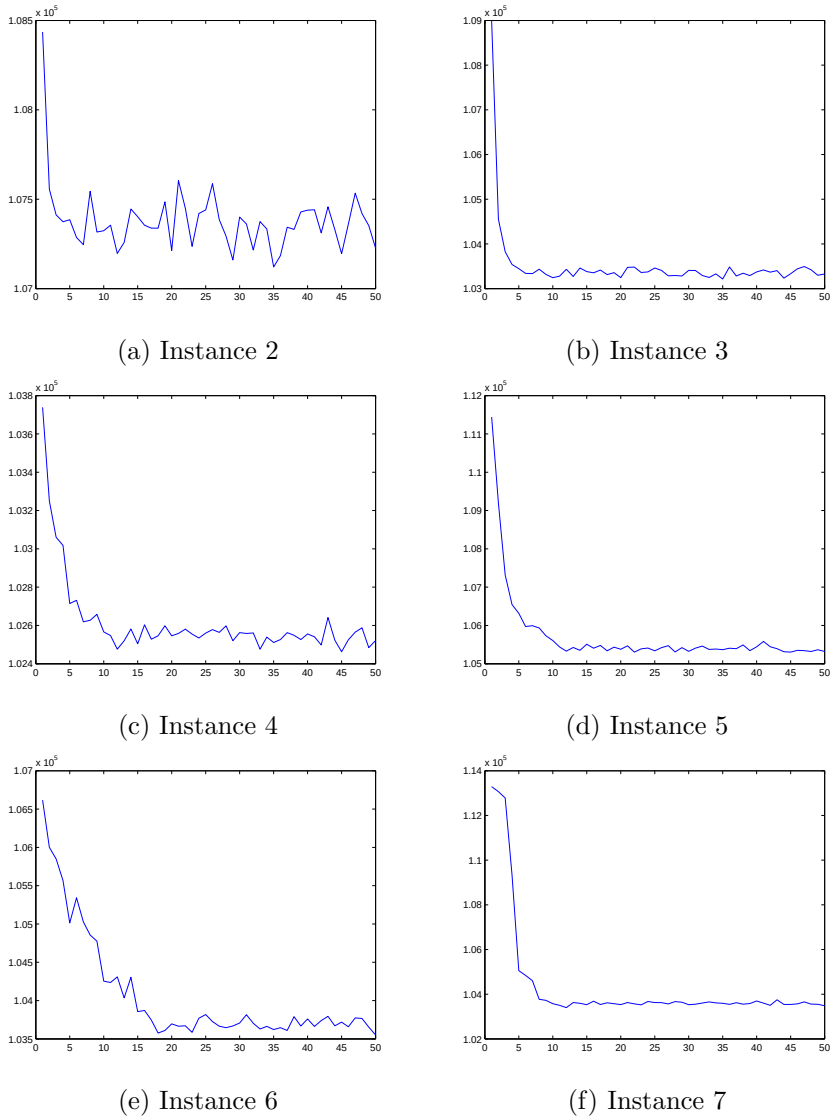
Table 5.3: Numerical results (mean cost ± s.d.) of implementing routing policies on $1,000,000$ random scenarios with $c_2 = 15$

We can see that the ADP routing policy outperforms the TMRP in most cases and works better than the SDP routing policy using a coarse discretization (e.g. $L = 7$ and 14). Sometimes, the ADP routing policy can be even better than SDP with $L = 28$ (e.g. in Instance 5), which is the finest model for which SDP could be solved. We think the reason for ADP outperforming SDP is due to the fact that, in SDP the new random traffic is rounded to the nearest tabulated value before taking a decision. However, the ADP model approximates the value function with a continuous linear regression model, thus the decision is made based on the real value of the state. Apart from this, ADP focuses attention on the states that are actually visited. As in normally distributed instances the traffic is more clustered around the mean, we get more updates, and thus coefficients that better represent the states that are more likely to occur. We think this is why for the normally distributed traffic (i.e. Instances 4, 5 and 7) the ADP routing policy seems to perform better compared to the uniformly distributed cases (Instances 2, 3 and 6). In conclusion, the ADP model gives very promising results with the linear regression model.

Convergence and resource usage

Apart from the mean cost, another important practical issue is running time of a model, specifically the convergence time in the ADP model. As notified in

Section 5.2.4, we justify the convergence of our model by evaluating the mean cost of implementing the routing policy derived from the current coefficients over every $10,000$ iterations.



(a) Instance 2

(b) Instance 3

(c) Instance 4

(d) Instance 5

(e) Instance 6

(f) Instance 7

Figure 5.3.2: Mean cost varies with iterations (x-axis unit is $10,000$ iterations) – convergence

Figure 5.3.2 shows how the mean cost varies with time (x-axis is an units on the $10,000$ iterations) for each instance. We see that initial convergence is fast (within $100,000$ iterations or so), and after which, it varies almost purely with noise. We try to identify by a heuristic the onset of noise and stop the algorithm then. The resulting numbers of iterations until convergence was determined are given in Table 5.4.

In addition to the running time, the 'curse of dimensionality' in the SDP model

also manifests itself in high memory use. In Table 5.4 we summarise the running time and the memory consumption (theoretical) in the solution of the ADP and SDP model.

Instance	**ADP model**			**SDP model**		
	Iterations	**Time**	**Memory**	L	**Time**	**Memory**
Instance 2	80,000	9.391s	5.34e-4MB	7	0.194s	0.38MB
				14	15.628s	16.75MB
				28	11487.095s	880.32MB
Instance 3	70,000	7.481s	5.34e-4MB	7	0.126s	0.38MB
				14	12.486s	16.75MB
				28	9732.705s	880.32MB
Instance 4	130,000	170.824s	5.34e-4MB	7	0.170s	0.38MB
				14	14.738s	16.75MB
				28	11185.036s	880.32MB
Instance 5	120,000	130.022s	5.34e-4MB	7	0.163s	0.38MB
				14	14.781s	16.75MB
				28	11243.042s	880.32MB
Instance 6	180,000	16.108s	5.34e-4MB	7	0.166s	0.38MB
				14	14.149s	16.75MB
				28	9002.732s	880.32MB
Instance 7	120,000	131.606s	5.34e-4MB	7	0.170s	0.38MB
				14	14.220s	16.75MB
				28	10114.663s	880.32MB

Table 5.4: Comparison of problem size and resource consumption

In Table 5.4, the first column shows the number of iterations needed to see the convergence of the ADP model, while the second column shows the running time. We can see that in Instances 2, 3 and 6 the solution times of ADP are comparable with the 14-level SDP model, while for Instances 4, 5 and 7 the latter runs faster. This is caused by the fact that, in ADP we need to calculate the expected function value over continuous region when solving the decision problem, for normal distribution it takes much more time than for the uniform distribution.

However, the most significant advantage of ADP model is that it does not require to discretize the traffic region, but works with a continuous state space. Thus instead of needing to record all decisions explicitly at every node in the DP tree, we need to store only the coefficients of the regression model, thus requiring much less memory. In fact the memory requirements of the ADP model are increasing linearly with the top-percentile parameter θ and the number of network providers n. This solves the 'curse of dimensionality' of the SDP model.

5.3.2 Solving the decision problem to higher accuracy

As stated above, the decision problem (5.2.2) is not convex, thus not easy to solve quickly to optimality. So far the decision problem has been solved by trying all decisions $\{0.0, 0.1, 0.2, ...1.0\}$ and choosing the one which leads to the best objective. In Table 5.5 we investigate the effect of solving the decision problem to a higher accuracy, by choosing from decisions $\{0.00, 0.01, 0.02, ...1.00\}$.

Instance	ADPRP_0.1		ADPRP_0.01	
	Mean Cost	Running Time	Mean Cost	Running Time
Instance 2	107335.60±12.35	9.391s	107335.43±12.35	84.473s
Instance 3	103361.52±7.50	7.481s	103361.51±7.50	68.245s
Instance 4	102561.69±4.31	170.824s	102561.69±4.31	1350.809s
Instance 5	105418.83±4.78	130.022s	105417.34±4.78	982.480s
Instance 6	103553.42±5.97	16.108s	103553.05±5.97	150.139s
Instance 7	103130.13±6.11	131.606s	103130.13±6.11	1084.454s

Table 5.5: Comparison of mean cost (± s.d.) and resource consumption of ADPRP_0.1 and ADPRP_0.01, $c_2 = 12$

We can see that this does not enhance the quality of ADP solution (differences in mean cost are not statistically significant), while of course increasing solution time. We therefore argue that our primitive but fast method to solve the decision problem is justified.

5.3.3 43-period TpTRP instances with more network providers

To justify the argument on computational analysis in Section 5.2.2, we examine the ADP model on several 43-period testing instances with 2 or 3 network providers, in which all network providers use top-5 percentile pricing (charge based on the 3rd highest volume of traffic). Every time interval's traffic follows uniform distribution in Instances 2, 3 and 6 (i.i.d for Instance 2 and time-varying for Instances 3 and 6), or normal distribution in Instances 4, 5 and 7 (i.i.d for Instance 4 and time-varying for Instances 5 and 7). The parameter for this time varying pattern (assuming every group of $\lfloor \Gamma/10 \rfloor$ time intervals follow a same distribution) is displayed in Figures 4.4.1 and 4.4.2.

For every one of these instances, we build an ADP model and then train it to get the converged value function estimations. Then we generate the routing

decision using the estimated value function when testing the ADP routing policy (ADPRP) on a simulation of $1,000,000$ random scenarios. Numerical results for instances with 2 and 3 network providers are given in Table 5.6 and 5.7. Note that here we only consider the case where $c_2 = 12$.

Instance	SRP	TMRP	ADPRP	DRP
Instance 2_43_n=2	134547.77±3.00	134289.64±3.14	132346.89±3.87	130914.21±3.78
Instance 2_43_n=3	134547.77±3.00	134003.29±3.29	130611.56±3.89	127277.11±4.36
Instance 3_43_n=2	131378.26±4.34	130284.84±4.75	127333.51±5.34	125797.11±4.88
Instance 3_43_n=3	131378.26±4.34	128895.05±5.21	124412.87±5.42	120829.20±4.48
Instance 4_43_n=2	115540.86±3.01	115301.25±3.03	113381.16±2.89	112457.28±2.54
Instance 4_43_n=3	115540.86±3.01	115040.92±3.06	111703.39±2.84	110246.16±2.31
Instance 5_43_n=2	122388.79±3.15	122115.87±3.59	120204.57±3.47	119283.57±3.00
Instance 5_43_n=3	122388.79±3.15	120424.92±3.36	117650.93±3.03	116255.73±2.51
Instance 6_43_n=2	133510.15±2.41	132967.80±2.50	131955.83±2.97	130639.33±2.56
Instance 6_43_n=3	133510.15±2.41	132350.85±2.59	130548.17±2.73	128161.59±2.62
Instance 7_43_n=2	130054.11±3.27	129985.05±3.29	127448.02±2.92	126584.66±2.77
Instance 7_43_n=3	130054.11±3.27	129913.40±3.31	127061.02±2.58	124060.68±2.51

Table 5.6: Numerical result (mean cost ± s.d.) of implementing routing policies on $1,000,000$ random scenarios

Instance	ADPRP_n=2		ADPRP_n=3	
	Iterations	Running Time	Iterations	Running Time
Instance 2_43	200,000	99.674s	200,000	135.308s
Instance 3_43	300,000	134.272s	400,000	228.562s
Instance 4_43	700,000	3697.362s	900,000	6879.312s
Instance 5_43	400,000	2187.375s	700,000	5351.445s
Instance 6_43	600,000	276.695s	800,000	477.028s
Instance 7_43	200,000	1264.290s	500,000	4441.088s

Table 5.7: Comparison of running time of ADPRP_n=2 and ADPRP_n=3

From Table 5.6 we can see that the ADP routing policy performs better than trivial routing policies (SRP and TMRP) and close to the lower bound given by the deterministic model (DRP). The data shown in Table 5.7 indicates the number of iterations required by the ADP model to converge and the corresponding running time. We can see that it needs around 1.5 times longer solution time (per iteration) than the model with 2 network providers, which justifies the argument that difficulty in solving the ADP model grows linearly with the number of network providers.

5.3.4 Solving realistically sized instances with aggregation method

Despite the improvement in terms of time and memory consumption of the ADP model over the SDP model, we are still not in a position to solve the realistically sized problem instances with thousands of time intervals directly. Rather we suggest to aggregate time periods, such that one model $V_\tau(S^\tau)$ is used for 100 time periods. Straightforward application of ADP to such a model would result in updating the parameters β_i^τ for one particular $V_\tau(S^\tau)$ for 100 consecutive times before moving on to $V_{\tau+1}(S^\tau)$, resulting in slow convergence. To speed up convergence we instead aggregate each scenario $\omega_m \in \mathbb{R}^{|\Gamma|}$ into a compact sample with $|\Gamma|/100$ components and use this to update in effect a $|\Gamma|/100$-time period model.

Table 5.8 gives the performance for this approach on a 4320-time period model (the information on running time is given in Table 5.7 for $n = 2$).

Instance	**SRP**	**TMRP**	**ADPRP**	**DRP**
Instance 2	136000.08±26.32	135789.57±27.54	134335.36±37.21	132008.21±36.92
Instance 3	133874.68±19.03	133022.12±24.19	129956.59±31.54	127791.87±21.00
Instance 4	116466.16±32.85	116216.75±33.28	114318.44±42.26	112840.00±25.99
Instance 5	124737.71±37.47	123666.23±38.71	121618.19±42.35	120235.63±30.27
Instance 6	134780.66±22.90	134302.58±23.59	133008.55±35.81	131646.65±25.19
Instance 7	130453.99±33.93	130375.86±34.34	128668.28±41.70	126368.03±27.88

Table 5.8: Numerical results (mean cost ± s.d.) of implementing the 43-periods aggregated ADP routing policy on a realistically sized instance using $1,000$ scenarios, $c_2 = 12$

Comparing with the numerical results for Instance 2 given by the 'classified decision rule' we developed in Section 4.4.3 (by implementing which the mean cost is 135792.52 ± 2.76 over $1,000,000$ scenarios), we can see that the stated ADP-time aggregation method works much better. Generally speaking, the ADP-time aggregation model performs well on a 4320-period problem, consistently closing the gap between the TMRP and DRP by $40\% - 60\%$.

5.4 Conclusions

In this chapter we have developed an ADP model to solve the TpTRP problem. Rather than using the discrete look-up table representation of the value function used in SDP, ADP approximates the value function by a proper regression model and updates its coefficients iteratively with fresh sample scenarios to get the final estimate. As all calculations are performed on a continuous state space, ADP overcomes the curse of dimensionality present in the SDP model which prevented larger instances to be solved.

ADP compares favourable to the SDP model in the solution of small instances (10-periods ones). Routing policies derived from ADP model are no worse than those generated from 14-levels SDP model and sometimes even outperform the SDP routing policy with 28-levels, while the running time is much smaller. By combining ADP with time aggregation we can solve realistically sized instances with thousands of time periods in a reasonable time. The routing policies obtained consistently outperform the TMRP on realistically sized problem instances.

Chapter 6

ADP Model with Bézier Curve/Surface Approximations

In the original ADP model given in Chapter 5, we approximate the value function for every time interval τ by a linear regression model:

$$\bar{V}_\tau(S^\tau) = \beta_0^\tau + \sum_{i \in I} \sum_{1 \le j \le \theta} \beta_{i,j}^\tau \hat{T}_i^{j,\tau}, \tag{6.0.1}$$

which means that we have separate value function estimation $\bar{V}_\tau$ to approximate V_τ for every time interval τ. With this 'discrete' ADP model (where the regression parameters are discrete in time), TpTRP instances up to 86 periods can be trained (achieving convergence of the β weights) within reasonable time (see Table 6.1).

Instance	**Parameters**			**No. of βs**	**Convergence**	
	$\|\Gamma\|$	θ	n	$\|\Gamma\|(n\theta+1)$	**Iterations**	**Time**
Instance 2_43	43	3	2	301	200,000	99.674s
Instance 2_86	86	5	2	946	500,000	515.743s
Instance 2_432	432	22	2	19440	-	-
Instance 2_4320	4320	216	2	1870560	-	-

Table 6.1: Size of the ADP model and its regression information

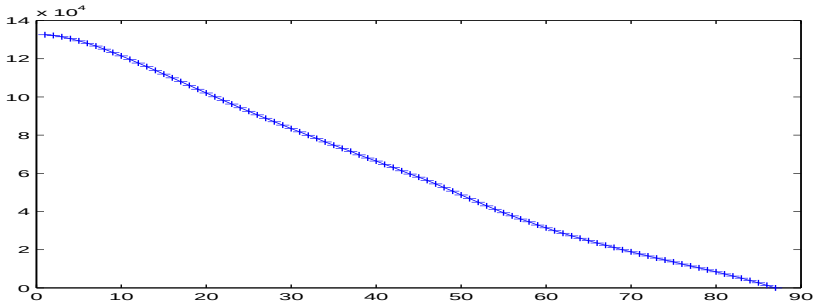
However, larger sized instances are still challenging. Though the curse of dimensionality is avoided in the discrete ADP model, the speed of achieving convergence depends on the number of parameters to estimate. From Table 6.1 we can see

that the number of regression parameters required in the discrete ADP model for Instance 2 (for detailed traffic distribution we assumed on this instance, please refer to Section 4.4.1) grows quadratically with the number of time intervals. This means it will take several hours to achieve convergence for the 432-period model.

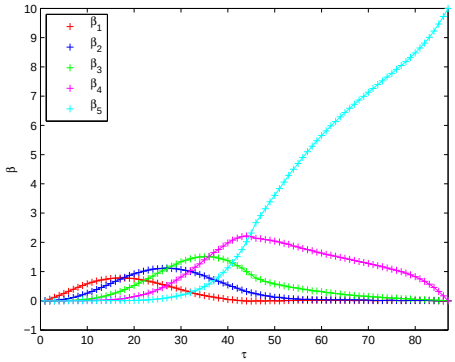
To solve this problem, we suggest aggregating the regression coefficients β_0^τ (which are currently discrete in τ) and $\beta_{i,j}^\tau$ (which are currently discrete in directions i, j and τ) over time intervals, namely to replace the β_0^τ, $\beta_{i,j}^\tau$ by functions $\beta_0(\tau)$, $\beta_{i,j}(\tau)$ to reduce the number of parameters to estimate.

6.1 Time-Aggregated ADP model

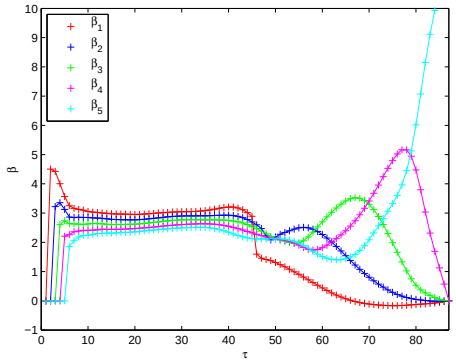
To guide the choice of good approximating functions $\beta_0(\tau)$, $\beta_{i,j}(\tau)$, firstly we have a look at the optimal β_0^τ, $\beta_{i,j}^\tau$ for an example of the discrete ADP model.



(a) Converged regression coefficients β_0^τ in 86-period Instance 2



(b) Converged regression coefficients $\beta_{1,j}^\tau$ in 86-period Instance 2



(c) Converged regression coefficients $\beta_{2,j}^\tau$ in 86-period Instance 2

Figure 6.1.1: Regression parameters in an example of the discrete ADP model

Figure 6.1.1 shows how the optimal β_0^τ, $\beta_{i,j}^\tau$ vary with time τ for a 86-period

instance. Every point corresponds an estimation of β_0, $\beta_{i,j}$ at some time point τ in the trained model. It is obvious that every single β_0^τ, $\beta_{i,j}^\tau$ is smooth (or near smooth) as a function of time, therefore it should be possible to approximate it with less parameters.

6.1.1 Bézier Curve

In this work we suggest using Bézier Curves approximating functions. Bézier curves were widely publicised in 1962 by the French engineer Pierre Bézier [20], who used them to design automobile bodies. A Bézier Curve is a parametric polynomial curve depending on a number of control points that is frequently used to produce curves which appear reasonably smooth. Given a large enough number of properly selected control points, any smooth function can be approximated by Bézier Curve to arbitrary accuracy (see Farin [24] for details).

The Bézier Curve of degree K can be generated as follows. Given control points $\mathbf{P_0}, \mathbf{P_1}, ..., \mathbf{P_K} \in \mathbb{R}^\mathbf{2}$, the Bézier Curve is the set of points:

$$\begin{aligned}\mathbf{B}(u) &= \sum_{k=1}^{K} \begin{pmatrix} K \\ k \end{pmatrix} (u)^k (1-u)^{K-k} \mathbf{P_k} \\ &= (1-u)^K \mathbf{P_0} + \begin{pmatrix} K \\ 1 \end{pmatrix} (1-u)^{K-1} u \mathbf{P_1} + ... \\ &\quad ... + \begin{pmatrix} K \\ K-1 \end{pmatrix} (1-u)(u)^{K-1} \mathbf{P_{K-1}} + u^K \mathbf{P_K},\end{aligned}$$

for $u \in [0,1]$, where $\begin{pmatrix} K \\ k \end{pmatrix}$ is the binomial coefficient.

6.1.2 An example

In our model, we use Bézier Curves in (τ, β)-space to estimate the regression parameters $\beta_{i,j}(\tau)$. Given a (fixed) set of τ-components of the control points

$\{\hat{\tau}_{i,j}^k, k = 1, ..., K\}$ and parameters $\beta_{i,j}^k$, the Bézier Curve model for $\beta_{i,j}(\tau)$ is

$$\begin{pmatrix} \tau \\ \bar{\beta}_{i,j}(\tau) \end{pmatrix} = \sum_{k=1}^{K} \begin{pmatrix} K \\ k \end{pmatrix} (u)^k (1-u)^{K-k} \begin{pmatrix} \hat{\tau}_{i,j}^k \\ \beta_{i,j}^k \end{pmatrix}. \tag{6.1.1}$$

With this model, in order to find $\bar{\beta}_{i,j}(\tau)$, for any given τ we need to solve a K-th degree polynomial equation to find its root $u_\tau \in [0, 1]$, then calculate the value of $\beta_{i,j}(\tau)$ with u_τ.

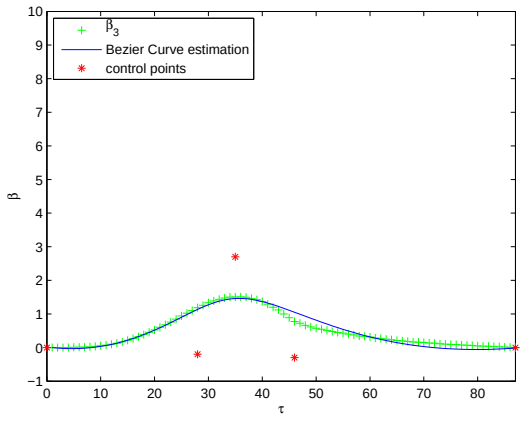


Figure 6.1.2: Comparison of original discrete values and the Bézier Curve approximation with $K = 5$

Figure 6.1.2 shows an example of the 5-degree Bézier Curve in the estimation of $\beta_{1,3}(\tau)$, taken from the instance shown in Figure 6.1.2. We can see that for the given choice of $\{\hat{\tau}_{i,j}^k, k = 1, ..., K\}$, the 5-degree Bézier Curve can approximate the discrete set of $\beta_{1,3}^\tau$ visibly well by a continuous curve with a maximum error 0.1809, which amounts to 21.71% of the original value. (Note that though the error for this example seems relevantly high, the real test on whether Bézier Curve is a good choice depends on the performing of routing policies generated by the ADP-Bézier-Curve model, which are given in Section 6.2.2.) With Bézier Curve, we can replace the original discrete regression model (with 86 coefficients to estimate: $\beta_{1,3}^\tau, \tau = 1, ..., 86$) by a continuous function with only 5 parameters $(\beta_{1,3}^k, k = 1, ..., 5)$. This reduces the size of the problem, thus speeding up the convergence of the ADP model.

6.1.3 ADP-Bézier-Curve model

We give a more detailed description on the aggregated ADP-Bézier-Curve algorithm. Specifically, we point out several modifications of the general ADP algorithm below in parameter structure and formulations.

The original ADP model

The basic Approximate Dynamic Programming algorithm (Powell [45]) is summarised below:

Step 0. Initialisation:

Step 0a. Build an initial value function approximation $\bar{V}_\tau^{(0)}(S^\tau)$ for all time intervals τ.

Step 0b. Choose an initial state $S_{(1)}^1$.

Step 0c. Set $m = 1$.

Step 1. Choose a sample path $\omega_{(m)} = (\omega_{(m)}^1, ..., \omega_{(m)}^\Gamma)$.

Step 2. For $\tau = 0, 1, 2, ..., \Gamma$ do:

Step 2a. Solve

$$\hat{v}_\tau^{(m)} = \min_{x^\tau \in \chi^\tau} (\mathbb{E}_{\omega^\tau \in \Omega^\tau} \bar{V}_{\tau+1}^{(m-1)}(S^{\tau+1}|S_{(m)}^\tau, x^\tau, \omega^\tau)). \tag{6.1.2}$$

Step 2b. Update the value function approximation $\bar{V}_\tau^{(m-1)}(S^\tau)$ with the value of $\hat{v}_\tau^{(m)}$.

Step 2c. Compute $S_{(m)}^{\tau+1}(S_{(m)}^\tau, \hat{x}^\tau, \omega_{(m)}^\tau)$, where $\hat{x}^\tau$ is the optimal solution of (6.1.2).

Step 3. If we have not met our stopping rule, let $m = m + 1$ and go to step 1.

Initialisation – Step 0a.

We use a Bézier Curve model with fixed values $\{\hat{\tau}^k, k = 1, ..., K\}$, while updating values $\{\beta_{i,j}^k, k = 1, ..., K\}$ iteration by iteration. Note that the set of $\{\beta_{i,j}^k, k = 1, ..., K\}$ is dependent on indexes i and j while $\{\hat{\tau}^k, k = 1, ..., K\}$ is not. Given K control points, as initialisation we set $\{\hat{\tau}^k, k = 1, ..., K\}$ equally among the

charging period $[0, |\Gamma|]$:

$$\hat{\tau}^k = \frac{k}{K}|\Gamma|, k = 1, ..., K,$$

and all the unknown $\beta_{i,j}^k$ are initialised to 0.

As the set $\{\hat{\tau}^k, k = 1, ..., K\}$ does not change with the iterations, we can calculate the solution u_τ of the polynomial equation:

$$\tau = \sum_{k=1}^{K} \begin{pmatrix} K \\ k \end{pmatrix} (u_\tau)^k (1 - u_\tau)^{K-k} \hat{\tau}^k$$

before the updating scheme, to find the roots $u_\tau \in [0, 1]$ for all time intervals τ.

Decision problem – Step 2a.

During the ADP algorithm, at time interval τ we need to solve the decision problem based on the current value function estimation to generate the optimal routing decision for this time interval's traffic. In the ADP-Bézier-Curve model, the value function estimation (6.0.1) is still a linear function of the state. The only difference from before is that the regression parameters $\beta_{i,j}^{\tau+1}$, are now approximated by Bézier Curves (6.1.1). Thus to evaluate the value function estimation as required in Step 2a of the ADP algorithm, we firstly need to calculate all the current estimates (after $m - 1$ iterations) of the $\beta_{i,j}^{\tau+1}$ using

$$\bar{\beta}_{i,j}^{(m-1)}(\tau + 1) = \sum_{k=1}^{K} \begin{pmatrix} K \\ k \end{pmatrix} (u_{\tau+1})^k (1 - u_{\tau+1})^{K-k} \beta_{i,j}^{k,(m-1)}. \tag{6.1.3}$$

Substituting these $\bar{\beta}_{i,j}^{(m-1)}(\tau + 1)$ into (6.0.1) we get the current value function estimation for time interval $\tau + 1$ and then solve the decision problem (6.1.2). This step introduces the computational complexity of $\mathcal{O}(n\theta)$ during every time interval to compute $\bar{\beta}_{i,j}^{(m-1)}$, in addition to the solution of the decision problem (discussed in Section 5.2.2).

Updating scheme – Step 2b.

The parameter update in Step 2b now becomes:

$$\begin{aligned}\beta_{i,j}^{k,(m)} &= \beta_{i,j}^{k,(m-1)} - \alpha_{m-1}[\bar{V}_\tau^{(m-1)}(S^\tau) - \hat{v}_\tau^{(m)}]\nabla_{\bar{\beta}_{i,j}^{(m-1)}}\bar{V}_\tau^{(m-1)}(S^\tau)\nabla_{\beta_{i,j}^{k,(m-1)}}\bar{\beta}_{i,j}^{(m-1)} \\ &= \beta_{i,j}^{k,(m-1)} - \alpha_{m-1}[\bar{V}_\tau^{(m-1)}(S^\tau) - \hat{v}_\tau^{(m)}] \cdot \hat{T}_i^{j,\tau} \cdot \begin{pmatrix} K \\ k \end{pmatrix} (u_\tau)^k (1-u_\tau)^{K-k}, \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{for all} \quad k = 1, ..., K.\end{aligned}$$

Thus in the aggregated ADP model every time we get an estimate of the value function $\hat{v}_\tau$ ($\forall \tau \in [0, |\Gamma|]$), we can update all $\beta_{i,j}^k, i \in I, j = 1, ..., \theta, k = 1, ..., K$ at once. Comparing with the original ADP model where $\hat{v}_\tau$ can update $\beta_{i,j}^\tau, i \in I, j = 1, ..., \theta$ only, the ADP-Bézier-Curve model accelerates the convergence speed significantly. Consequently, the ADP-Bézier-Curve model needs a much lower number of iterations to be trained, thus is more efficient than the original ADP model.

ADP-Bézier-Curve algorithm for TpTRP

Thus the complete ADP-Bézier-Curve algorithm for our TpTRP is summarised below:

Step 0. Initialisation:

Step 0a. Build value function approximation $\bar{V}_\tau(S^\tau) = \beta_0(\tau) + \sum_{i \in I} \sum_{1 \le j \le \theta} \beta_{i,j}(\tau) \hat{T}_i^{j,\tau}$, where $\beta_0(\tau)/\beta_{i,j}(\tau)$ is decided by

$$\begin{pmatrix} \tau \\ \beta_0(\tau)/\bar{\beta}_{i,j}(\tau) \end{pmatrix} = \sum_{k=1}^{K} \begin{pmatrix} K \\ k \end{pmatrix} (u)^k (1-u)^{K-k} \begin{pmatrix} \hat{\tau}_{i,j}^k \\ \beta_0^k/\beta_{i,j}^k \end{pmatrix}.$$

Step 0b. Set control points equally among the charging period $[0, |\Gamma|]$:

$$\hat{\tau}^k = \frac{k}{K}|\Gamma|, k = 1, ..., K,$$

and initialize $\beta_0^k = 0, \beta_{i,j}^k = 0, \forall i \in I, 1 \leq j \leq \theta, k = 1, ..., K$.

Step 0c. Compute the solution u_τ of the polynomial equation:

$$\tau = \sum_{k=1}^{K} \left(\begin{array}{c} K \\ k \end{array} \right) (u_\tau)^k (1 - u_\tau)^{K-k} \hat{\tau}^k$$

for all $\tau \in \Gamma$.

Step 0d. Choose an initial state $S_{(1)}^1$, set $m = 1$.

Step 1. Choose a sample path $\omega_{(m)} = (\omega_{(m)}^1, ..., \omega_{(m)}^\Gamma)$.

Step 2. For $\tau = 0, 1, 2, ..., \Gamma$ do:

Step 2a. Find the current estimation of β values using:

$$\bar{\beta}_{i,j}^{(m-1)}(\tau + 1) = \sum_{k=1}^{K} \left(\begin{array}{c} K \\ k \end{array} \right) (u_{\tau+1})^k (1 - u_{\tau+1})^{K-k} \beta_{i,j}^{k,(m-1)}.$$

Step 2b. Solve

$$\hat{v}_\tau^{(m)} = \min_{x^\tau \in \chi^\tau} (\mathbb{E}_{\omega^\tau \in \Omega^\tau} \bar{V}_{\tau+1}^{(m-1)}(S^{\tau+1} | S_{(m)}^\tau, \omega^\tau, x^\tau)) \tag{6.1.4}$$

by selecting the best one from $\chi^\tau = \{x_i^\tau \in \{0.0, 0.1, 0.2...1.0\} | \sum_{i \in I} x_i^\tau = 1\}$.

Step 2c. Update the weights $\beta_{i,j}^{k,(m)}$ by the stochastic gradient algorithm:

$$\begin{aligned} \beta_{i,j}^{k,(m)} &= \beta_{i,j}^{k,(m-1)} - \alpha_{m-1} [\bar{V}_\tau^{(m-1)}(S^\tau) - \hat{v}_\tau^{(m)}] \nabla_{\bar{\beta}_{i,j}^{(m-1)}} \bar{V}_\tau^{(m-1)}(S^\tau) \nabla_{\beta_{i,j}^{k,(m-1)}} \bar{\beta}_{i,j}^{(m-1)} \\ &= \beta_{i,j}^{k,(m-1)} - \alpha_{m-1} [\bar{V}_\tau^{(m-1)}(S^\tau) - \hat{v}_\tau^{(m)}] \cdot \hat{T}_i^{j,\tau} \cdot \left(\begin{array}{c} K \\ k \end{array} \right) (u_\tau)^k (1 - u_\tau)^{K-k}, \end{aligned}$$

$$\text{for all} \quad k = 1, ..., K.$$

Step 2c. Compute $S_{(m)}^{\tau+1}(S_{(m)}^\tau, \omega_{(m)}^\tau, \hat{x}^\tau)$, where $\hat{x}^\tau$ is the solution of (6.1.4).

Step 3. If we have not met our stopping rule described in Section 4.4, let $m = m + 1$ and go to step 1.

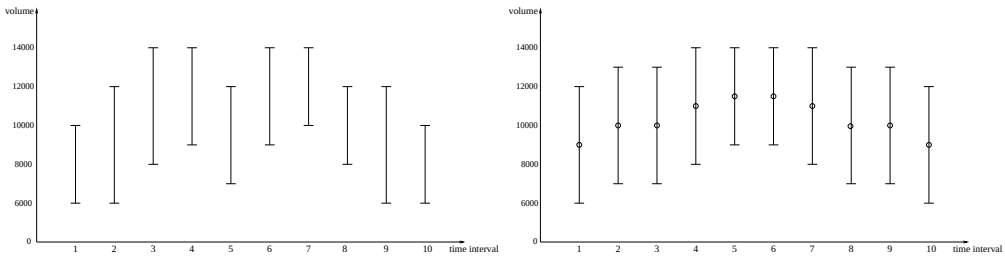
6.2 Numerical results for ADP-Bézier-Curve model

6.2.1 Test problems

In this section we discuss the numerical results of applying the ADP-Bézier-Curve algorithm.

Instance	Parameters			Stochastic Information	
	Γ	θ	n	distribution	time dependency
Instance 2_86	86	5	2	$U(6000, 14000)$	i.i.d.
Instance 2_216	216	11	2		
Instance 2_432	432	22	2		
Instance 3_86	86	5	2	uniform	see Fig. 6.2.1a
Instance 3_216	216	11	2		
Instance 3_432	432	22	2		
Instance 4_86	86	5	2	truncated $N(10000, 10^6)$	i.i.d.
Instance 4_216	216	11	2		
Instance 4_432	432	22	2		
Instance 5_86	86	5	2	truncated normal	see Fig. 6.2.1b
Instance 5_216	216	11	2		
Instance 5_432	432	22	2		
Instance 6_86	86	5	2	uniform	see Fig. 6.2.2a
Instance 6_216	216	11	2		
Instance 6_432	432	22	2		
Instance 7_86	86	5	2	truncated normal	see Fig. 6.2.2b
Instance 7_216	216	11	2		
Instance 7_432	432	22	2		

Table 6.2: List of TpTRP Instances

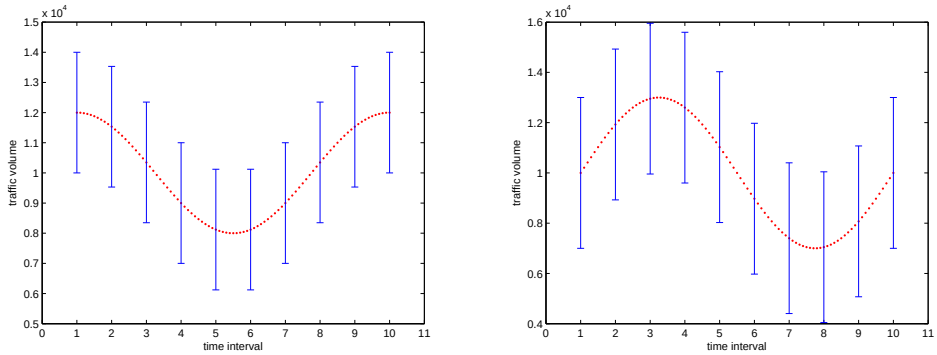


(a) Upper and lower bounds for uniform distributions in Instance 3

(b) Mean and 99.7% ($\pm 3\sigma$) confidence region for normal distributions in Instance 5

Figure 6.2.1: Traffic distribution used in testing instances

Table 6.2 summarises the instances used. In all instances, we assume that we have 2 network providers, where network provider 1 is cheaper than network provider 2 ($c_1 = 10, c_2 = 12$). All network providers use pure top-percentile pricing to charge



(a) Upper and lower bounds for uniform distributions in Instance 6

(b) Mean and 99.7% ($\pm 3\sigma$) confidence region for normal distributions in Instance 7

Figure 6.2.2: Traffic distribution used in testing Instances 6 and 7

the ISP, based on the top-5% time interval. Thus for example when the charge period is divided into 86 time intervals, we are charged on the time interval with the 4-th highest volume of traffic. The instances differ by the assumptions made on the random traffic. In Instances 2 and 4 the traffic in every period follows the same uniform ($U(6000, 14000)$ in Instance 2) or normal ($N(10000, 10^6)$ in Instance 4) distribution. Instances 3, 5, 6 and 7 on the other hand, use traffic distributed according to a time varying uniform or normal distribution. The parameter for this time varying pattern (assuming every group of $\lfloor |\Gamma|/10 \rfloor$ time intervals follow a same distribution) is displayed in Figures 6.2.1 and 6.2.2. Note that Instances 4, 5 and 7 use a truncated normal distribution in which traffic outside the 99.7% ($\pm 3\sigma$) confidence region is projected onto the boundary of the region to avoid negative traffic volumes.

6.2.2 Numerical results on 86-period TpTRP instances

For every instance we build an ADP-Bézier-Curve model, train this model with random scenarios until convergence of the β weights, then test the resulting routing policy on a simulation of $1,000,000$ random scenarios taken from the original distribution. The routing policy given by this model is indicated by ADPRP_BC in the following tables, where parameter K denotes the number of control points used in the ADP-Bézier-Curve model. All the results are compared with the original discrete ADP model developed in Chapter 5, and the SRP, TMRP and DRP developed in Section 2.3.2.

Table 6.3 and Figure 6.2.3 compare the mean cost of implementing several routing

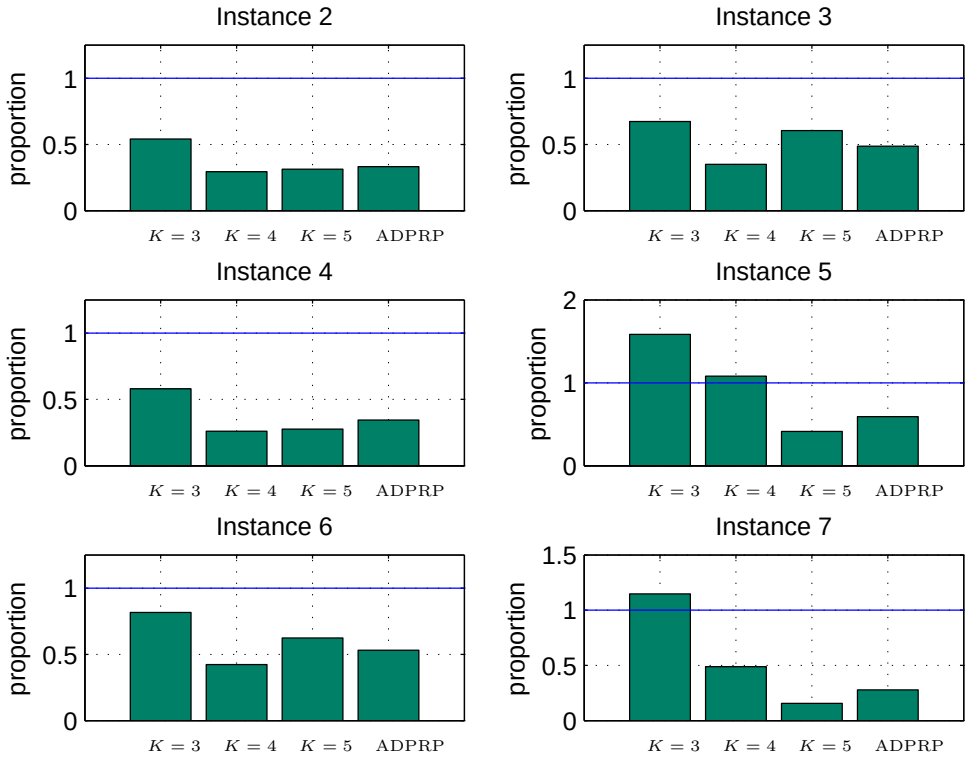


Figure 6.2.3: Comparison of mean cost between SDP and ADP routing policies

policies. Like before, each sub-figure in Figure 6.2.3 is corresponding to one instance. Each column shows the performance of ADPRP_BC with $K = 3, 4$ or 5 in turn. Performance in these figures are given in a proportional manner, where the value of a column corresponding to routing policy **X** is equal to (**X**-DRP)/(TMRP-DRP). Namely 0 means **X** works as well as DRP, while 1 means it works same as TMRP. Therefore, any routing policy that is above 1 (above the horizontal line) is obviously not desirable.

Generally speaking, the routing policies generated by the ADP-Bézier-Curve model perform well. In almost all cases the ADPRP_BC routing policy outperforms the trivial routing policies; sometimes it is even better than the ADPRP. Specifically, the ADPRP_BC with $K = 4$ works best for Instances 2, 3, 4 and 6, while $K = 5$ seems better for Instances 5 and 7.

It is worthwhile to point out that the performance of ADP-Bézier-Curve model is not necessarily improving monotonically with the number of control points K (For example $K = 5$ performs worse than $K = 4$ for Instances 2, 3, 4 and 6), due to our use of equidistant control points in τ-direction, which can result in

Instance	SRP	TMRP	ADPRP	K	ADPRP_BC	DRP
Instance 2_86	135404.34±1.98	135181.68±2.08	132902.35±2.71	3	133595.05±3.13	131727.00±2.60
				4	132739.81±2.77	
				5	132809.52±2.68	
Instance 2_216	135945.63±1.19	135749.17±1.25	-	4	133212.44±1.71	132258.12±1.60
Instance 2_432	135935.06±0.84	135727.89±0.88	-	4	132965.84±1.28	132054.59±1.15
Instance 3_86	132585.06±3.00	131588.78±3.35	129071.22±3.57	3	129980.79±4.65	126686.15±3.64
				4	128400.82±3.85	
				5	129645.63±3.67	
Instance 3_216	133663.71±1.77	132838.35±1.98	-	4	130575.08±1.98	127902.03±2.27
Instance 3_432	133770.80±1.24	132930.47±1.39	-	4	129602.03±1.88	127826.36±1.60
Instance 4_86	116104.12±2.22	115904.52±2.24	113892.97±2.46	3	114614.44±2.52	112833.05±1.84
				4	113631.76±2.16	
				5	113680.34±2.11	
Instance 4_216	116549.93±1.44	116319.57±1.46	-	4	113952.68±1.37	113091.56±1.18
Instance 4_432	116454.96±1.02	116212.32±1.03	-	4	113844.21±1.07	112898.46±0.83
Instance 5_86	123039.58±2.33	122175.78±2.40	121002.27±2.38	3	123850.12±2.95	119310.87±1.97
				4	122405.06±2.72	
				5	120497.46±2.22	
Instance 5_216	123705.80±1.50	122906.99±1.54	-	5	120918.74±1.40	119876.40±1.25
Instance 5_432	123720.58±1.05	122900.68±1.08	-	5	120906.23±1.32	119804.31±0.87
Instance 6_86	136163.34±1.12	135948.61±1.15	135127.54±1.20	3	135627.09±1.12	134197.79±1.30
				4	134938.52±1.48	
				5	135289.44±1.55	
Instance 6_216	137720.17±0.59	137609.83±0.62	-	4	136365.12±0.71	135876.84±0.80
Instance 6_432	137906.85±0.42	137800.15±0.44	-	5	136603.53±0.55	135966.52±0.57
Instance 7_86	124144.28±2.33	124032.45±2.35	121553.37±2.45	3	124537.89±2.32	120600.96±1.93
				4	122275.53±2.58	
				5	121132.94±2.63	
Instance 7_216	119456.24±1.46	119275.41±1.47	-	5	116574.50±1.70	115949.01±1.19
Instance 7_432	117824.85±1.02	117605.21±1.03	-	5	114811.45±1.13	114256.64±0.83

Table 6.3: Comparison of mean cost (± s.d.) of discrete ADPRP and ADP with Bézier Curve

worse approximation of the important features of the $\beta^{\tau}_{i,j}$ functions for small K. Nevertheless, generally speaking the performance of ADP-Bézier-Curve model is getting better with K, though with some noises due to the equidistant setting up of control points.

Table 6.4 compares the statistics on solution time of the ADP-Bézier-Curve model with the original discrete ADP model with four 86-period instances. The columns denoted by β**s** show the number of regression parameters to be estimated in either model. We can see that the ADP-Bézier-Curve model reduces this value by a factor of around 20 for the 86-period instances. In addition to this, in the ADP-Bézier-Curve model the number of βs increases linearly with the number of time periods (given the same number of control points used), as opposed to quadratically in the discrete ADP model. Consequentially, the ADP-Bézier-Curve model can be trained in a fraction of the time required for the discrete ADP model, despite the fact that a single iteration (given in column T/M) takes around twice

Instance	ADP_discrete				ADP_BC				
	βs	Iterations	Time	T/M	K	βs	Iterations	Time	T/M
Instance 2_86	946	500,000	515.743s	0.0010s	3	32	3,000	6.952s	0.0023s
					4	42	6,000	13.689s	0.0023s
					5	52	6,000	14.743s	0.0025s
Instance 2_216	4968	-	-	-	4	90	4,000	62.285s	0.0156s
Instance 2_432	19440	-	-	-	4	180	3,000	252.951s	0.0843s
Instance 3_86	946	800,000	748.245s	0.0009s	3	32	3,000	6.051s	0.0020s
					4	42	5,000	10.016s	0.0020s
					5	52	5,000	10.305s	0.0021s
Instance 3_216	4968	-	-	-	4	90	3,000	38.535s	0.0128s
Instance 3_432	19440	-	-	-	4	180	3,000	211.739s	0.0706s
Instance 4_86	946	800,000	13590.433s	0.0170s	3	32	4,000	96.594s	0.0241s
					4	42	6,000	158.715s	0.0265s
					5	52	7,000	187.663s	0.0268s
Instance 4_216	4968	-	-	-	4	90	7,000	836.776s	0.1195s
Instance 4_432	19440	-	-	-	4	180	4,000	2349.898s	0.5875s
Instance 5_86	946	1,000,000	14351.873s	0.0144s	3	32	4,000	70.995s	0.0177s
					4	42	4,000	73.869s	0.0185s
					5	52	5,000	93.914s	0.0188s
Instance 5_216	4968	-	-	-	5	112	6,000	732.712s	0.1221s
Instance 5_432	19440	-	-	-	5	225	3,000	1810.346s	0.6034s
Instance 6_86	946	1,200,000	1286.048s	0.0011s	3	32	4,000	8.601s	0.0022s
					4	42	4,000	8.857s	0.0022s
					5	52	5,000	12.043s	0.0024s
Instance 6_216	4968	-	-	-	4	90	6,000	169.733s	0.0283s
Instance 6_432	19440	-	-	-	5	225	6,000	606.538s	0.1011s
Instance 7_86	946	1,000,000	14332.636s	0.0143s	3	32	5,000	114.708s	0.0230s
					4	42	7,000	176.758s	0.0253s
					5	52	7,000	182.298s	0.0260s
Instance 7_216	4968	-	-	-	5	112	4,000	486.284s	0.1216s
Instance 7_432	19440	-	-	-	5	225	3,000	1645.930s	0.5486s

Table 6.4: Comparison of running time of ADPRP and ADPRP_BC

the time of the discrete ADP model.

Results of mean cost and running time on larger instances are summarised in Table 6.3 and 6.4 as well. We can see that TpTRP instances up to 216 periods can be solved within reasonable time (around 10 minutes in the longest case). However, for larger instances (e.g. 432-period) the running time is still long (though the routing policies generated performs equally well), which prevents the application of the ADP-Bézier-Curve model to larger problems.

6.3 Two dimensional approximation with Bézier Surface

From Table 6.4 we can see that even when the number of control points (K) stays the same with increasing problem size, the number of βs still increases linearly with θ. For the realistically sized instances which possesses n network providers, $\Gamma = 4320$ time intervals and $\theta = 215$, it requires $K \cdot n(\theta + 2) = 217nK$ regression parameters. Thus for larger instances, the current ADP-Bézier-Curve model is still not compact enough to be efficient. To reduce the problem size further, we extend the aggregation to two dimensions with Bézier Surfaces.

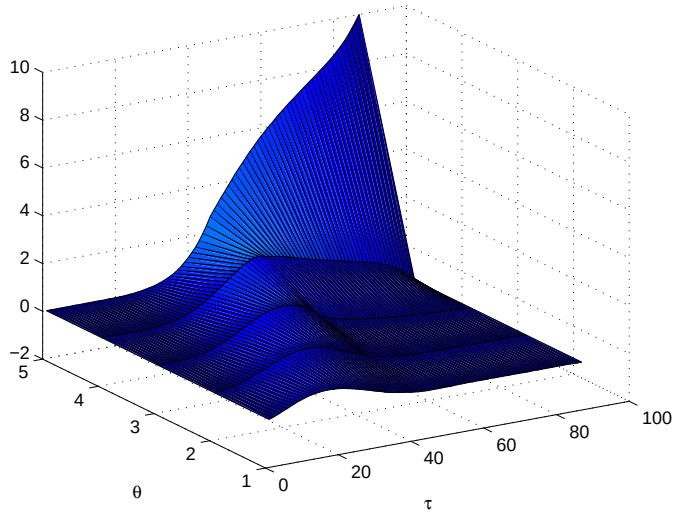


Figure 6.3.1: Converged regression coefficients $\beta^{\tau}_{1,j}$ in 86-period Instance 2

A higher dimensional Bézier Curve is called a Bézier Surface. Figure 6.3.1 gives a two dimensional view of Figure 6.1.1(b), which shows how the $\beta^{\tau}_{1,j}$ change in direction τ and j (j is the index of traffic while traffic volumes are in non-decreasing order). Comparing with Figure 6.1.1(b), we see that the surface is smooth in j-direction as well, thus should be well approximated with less than θ parameters.

In this part, we intend to approximate the surface shown in Figure 6.3.1 with a Bézier Surface, and then integrate it into the ADP model. We call this model, the ADP-Bézier-Surface model. The control points in the Bézier Surface model are now defined as (τ, j, β) and given by a (fixed) coordinate $(\hat{\tau}^k_i, \hat{j}^r_i)$ in (τ, j)-space and a corresponding parameter $\beta^{k,r}_i$. Assuming that we use control points

$\{\hat{\tau}_i^k, k = 1, ..., K\}$ for τ-direction and $\{\hat{j}_i^r, r = 1, ..., R\}$ for j-direction which are arranged in a grid layout, the Bézier Surface approximation to $\bar{\beta}_i(\tau, j)$ is given by

$$\begin{pmatrix} \tau \\ j \\ \bar{\beta}_i(\tau, j) \end{pmatrix} = \sum_{k=1}^{K} \sum_{r=1}^{R} \begin{pmatrix} K \\ k \end{pmatrix} (u_\tau)^k (1-u_\tau)^{K-k} \begin{pmatrix} R \\ r \end{pmatrix} (v_j)^r (1-v_j)^{R-r} \begin{pmatrix} \hat{\tau}_i^k \\ \hat{j}_i^r \\ \beta_i^{k,r} \end{pmatrix},$$

where $u_\tau \in [0, 1]$ is the root of equation $\sum_{k=1}^{K} \begin{pmatrix} K \\ k \end{pmatrix} (u_\tau)^k (1 - u_\tau)^{K-k} \hat{\tau}_i^k = \tau$ and $v_j \in [0, 1]$ is the root of $\sum_{r=1}^{R} \begin{pmatrix} R \\ r \end{pmatrix} (v_j)^r (1 - v_j)^{R-r} \hat{j}_i^r = j$.

Similarly to the ADP-Bézier-Curve model, we fix the values of $\{\hat{\tau}^k, k = 1, ..., K\}$ and $\{\hat{j}^r, r = 1, ..., R\}$ for all iterations:

$$\begin{cases} \hat{\tau}^k = \frac{k}{K}|\Gamma|, k = 1, ..., K, \\ \hat{j}^r = \frac{r}{R}\theta, r = 1, ..., R, \end{cases}$$

and iteratively update the values of $\beta_i^{k,r}$ (which are initialised to 0) to approximate $\beta_i(\tau, j)$. The updating formulation is thus:

$$\begin{aligned} &\beta_i^{k,r,(m)} \\ &= \beta_i^{k,r,(m-1)} - \alpha_{m-1}[\bar{V}_\tau^{(m-1)}(S^\tau) - \hat{v}_\tau^{(m)}][\sum_{j=1}^{\theta} (\nabla_{\bar{\beta}_{i,j}^{(m-1)}} \bar{V}_\tau^{(m-1)}(S^\tau) \nabla_{\beta_i^{k,r,(m-1)}} \bar{\beta}_{i,j}^{(m-1)})] \\ &= \beta_i^{k,r,(m-1)} - \alpha_{m-1}[\bar{V}_\tau^{(m-1)}(S^\tau) - \hat{v}_\tau^{(m)}] \\ &\qquad \cdot \begin{pmatrix} K \\ k \end{pmatrix} (u_\tau)^k (1 - u_\tau)^{K-k} \cdot [\sum_{j=1}^{\theta} (\hat{T}_i^{j,\tau} \cdot \begin{pmatrix} R \\ r \end{pmatrix} (v_j)^r (1 - v_j)^{R-r})], \\ &\qquad\qquad \text{for all} \quad k = 1, ..., K; r = 1, ..., R. \end{aligned}$$

6.4 Numerical results for ADP-Bézier-Surface model

6.4.1 Numerical results on 432-period TpTRP instances

Numerical results on instances with 432 periods are shown in Table 6.5 and 6.6.

Instance	SRP	TMRP	ADPRP_BC	ADPRP_BS	DRP
Instance 2_432	135935.06±0.84	135727.89±0.88	132965.84±1.28	132931.12±1.26	132054.59±1.15
Instance 3_432	133770.80±1.24	132930.47±1.39	129602.03±1.88	129964.61±1.75	127826.36±1.60
Instance 4_432	116454.96±1.02	116212.32±1.03	113844.21±1.07	113829.03±1.05	112898.46±0.83
Instance 5_432	123720.58±1.05	122900.68±1.08	120906.23±1.32	120934.34±1.20	119804.31±0.87
Instance 6_432	137906.85±0.42	137800.15±0.44	136603.53±0.55	136398.23±0.54	135966.52±0.57
Instance 7_432	117824.85±1.02	117605.21±1.03	114811.45±1.13	115012.38±1.10	114256.64±0.83

Table 6.5: Comparison of mean cost (± s.d.) of ADPRP_BC and ADPRP_BS on 432-period instances

Instance	ADP_BC					ADP_BS					
	K	βs	Iterations	Time	T/M	K	R	βs	Iterations	Time	T/M
Instance 2_432	4	180	3,000	252.951s	0.0843s	4	3	28	1,000	90.013s	0.0900s
Instance 3_432	4	180	3,000	211.739s	0.0706s	4	3	28	1,000	75.648s	0.0756s
Instance 4_432	4	180	4,000	2349.898s	0.5875s	4	3	28	1,000	684.169s	0.6842s
Instance 5_432	5	225	3,000	1810.346s	0.6034s	5	3	35	1,000	752.156s	0.7522s
Instance 6_432	5	225	6,000	606.538s	0.1011s	4	3	28	1,000	89.464s	0.0895s
Instance 7_432	5	225	3,000	1645.930s	0.5486s	5	3	35	1,500	934.727s	0.6232s

Table 6.6: Comparison of running time of ADPRP_BC and ADPRP_BS on 432-period instances

We can see that the routing policies generated with the ADP-Bézier-Surface model are comparable to their counterparts from the ADP-Bézier-Curve model, which are all better than the TMRP. However, Table 6.6 shows that the ADP-Bézier-Surface model saves around 2/3 of the training time of the model, thus making TpTRP instances with 432 periods solvable within reasonable time.

6.4.2 Realistically sized instances

In realistic sized instances, network providers divide the modelling period, i.e. a month, into 4320 time intervals with a length of 10 minutes each. If top 5-percentile pricing is used, the cost is based on the traffic volume of the top 216th interval. Although we perform the above aggregation to reduce the number of

regression parameters, it is still hard to solve the realistically sized problem with the current ADP-Bézier-Surface model. We can see from Table 6.6 that the number of iterations we need to train the model is significantly reduced from the original ADP model. The only problem left is the long running time it requires to go through every iteration, where we have to solve $|\Gamma|$ (which in realistically sized instances is 4320) decision problems and then update the value function estimation.

As an alternative, we can simplify the solution step by reducing the time to solve decision problems. Instead of solving the decision problem for every time interval, for the realistically sized problem we solve one decision problem for every 10 time intervals and use this decision for all these 10 time intervals. As the regression parameters change smoothly with time, this simplification will not introduce large errors.

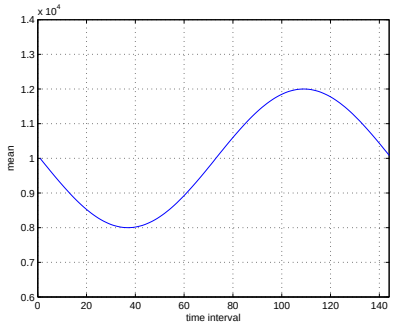


Figure 6.4.1: Mean of traffic distributions for realistically sized instance – one day long

Analysis data provided by a real-world ISP suggests that, generally speaking every time interval's traffic follows a normal distribution, where the mean changes according to a sine function day by day (where time intervals for daytime is busy and for night is respectively free). The detailed distribution we used for one day's time intervals (which amounts to 144 with the length of 10 minutes) is shown in Figure 6.4.1. Numerical results on this instance with either 2 ($c_1 = 10, c_2 = 12$) and 3 ($c_1 = 10, c_2 = 12$ and $c_3 = 13$) network providers (tested on 100 random scenarios) are shown in Table 6.7 and 6.8.

With the ADP-Bézier-Surface model and a simple decision aggregation step, realistically sized TpTRP instances are solvable, providing very good routing policies that improve significantly on the TMRP and close 54% (for $n = 2$) or 26% (for $n = 3$) of the gap between TMRP and the (unachievable) DRP. Due to the small number of control points we used in the ADP-Bézier-Surface model, it can be

Instance	SRP	TMRP	ADPRP_BS	DRP
Instance R_4320_n=2	127764.78±31.28	126194.29±34.65	124358.81±36.96	122822.98±26.30
Instance R_4320_n=3	127764.78±31.28	124387.26±36.89	123031.42±41.43	119228.71±27.81

Table 6.7: Comparison of mean cost (± s.d.) of ADPRP_BS on 4320-period instances

Instance	K	R	βs	Iterations	Time	T/M
Instance R_4320_n=2	5	6	65	400	3261.112s	8.1528s
Instance R_4320_n=3	5	6	95	650	7634.665s	11.7456s

Table 6.8: Comparison of running time of ADPRP_BS on 4320-period instances

trained after several hundred iterations taking around 1 hour for the instance with $n = 2$ and 2 hours for $n = 3$, while calculating the optimal routing policy from the trained model for a given set of observed traffic required $8 - 12$ seconds, comparable to one training iteration. Indeed, while applying the trained model as a routing oracle, the β update can be left in place at (virtually) no extra cost to continually improve the model.

6.5 Conclusions

In this chapter, we modified the original discrete ADP model for the TpTRP by aggregating regression coefficients β over both time interval τ and index of traffic order j using Bézier Surfaces. This reduces the number of parameters in the ADP model, thus drastically improves the model. The TpTRP instances up to 432-period are tractable with this ADP-Bézier-Surface model, giving routing policies which perform better than all naive routing policies. For realistically sized problem (with 4320 time intervals), we suggested aggregating decision problems, thus accelerating the iteration speed and make it solvable with the ADP-Bézier-Surface model.

Chapter 7

Conclusions and future work

7.1 On the TpTRP

In this work, we investigated a traffic routing problem (TpTRP) under multi-homing setup. Network providers charge the ISP using Top-percentile pricing, where the modelling period, say a month, is divided equally into $|\Gamma|$ time intervals on which the cumulative volumes of traffic are recorded. The cost is calculated at the end of the month, according to the volume of traffic in the θ-th highest time interval. The volume of traffic that is to be sent during every time interval is a random event for the ISP, for which the probabilistic distribution is assumed to be known beforehand. Thus the problem is a stochastic problem, where the routing decision for every time interval's traffic is required to be made under uncertainty.

We tried to solve the TpTRP using both stochastic programming and (stochastic) dynamic programming techniques. The stochastic programming model of TpTRP is a multi-stage, mixed-integer linear system. An effective solution method is hard to find due to the non-convexity (introduced by the top-percentile pricing) and large scale (introduced by the stochastic nature) of the model. Although for extremely small instances (e.g. Instance 1(a)) we can find the optimal solution using a stated C-BAC method, it is useless for the realistically sized TpTRP instances.

On the other hand, due to the recursive manner of the traffic routing problem,

stochastic dynamic programming is a better choice than multi-stage stochastic programming. We begin the investigation with a classic SDP model with a simple discretization of the state space, which solves the small sized TpTRP instances (e.g. Instances 2-7 with 10 periods) approximately but effectively. Then we exploit the well known Approximate Dynamic Programming algorithm (Powell [45]) to deal with the curse of dimensionality arising from the discretization in state space, developed an ADP model for the medium sized TpTRP (e.g. Instances 2-7 with up to 86 periods).

As the numerical results suggest that the ADP model is a good approximation of the original TpTRP, we follow the work on the ADP by an aggregation of its regression parameters, where the aggregation is done by Bézier Curves/Surfaces. This aggregation captures the smoothly varying feature of the regression parameters, and therefore forms a better approximation than simply combining several time intervals together and treat them indifferently in the aggregated model. At the same time, the Bézier Curves/Surfaces aggregation allows the ADP procedure to update the regression parameters more efficiently than in the discrete ADP model, thus makes the realistically sized TpTRP instances (with 4320 periods) solvable within reasonable time.

7.2 On the generalized TpTRP

The current work deals with traffic routing problems under pure top-percentile pricing policy, where if a provider is used for no more than $\theta - 1$ periods, no payment needs to be made. However in practice, network providers might combine top-percentile pricing with other pricing policies such as an additional start up cost. We can cover this extension by the presented ADP-Bézier-Curves/Surfaces model, with a trick in the value function approximation. Assuming that we use I_u^τ to indicate the set of network providers which have been used in previous time periods and I_0^τ to indicate those which has not until time interval τ. Then we have generally $I_u^\tau \cup I_0^\tau = I$, $I_u^\tau \cap I_0^\tau = \emptyset$. The value function approximation then consists of two parts, one is the same as before which approximates the pure top-percentile cost, while the other accounts for the start up cost c_i^0 if network provider i is used.

$$\tilde{V}_\tau(S^\tau) = \sum_{i \in I_u^\tau} c_i^0 + \bar{V}_\tau(S^\tau),$$

where $\bar{V}_\tau(S^\tau)$ accounts for the pure top-percentile pricing cost, as defined in (5.2.1). The Bellman's equation is thus

$$\hat{v}_\tau^{(m)} = \min_{x^\tau, z^\tau \in \chi^\tau} (\mathbb{E}_{\omega^\tau \in \Omega^\tau} \tilde{V}_{\tau+1}^{(m-1)}(S^{\tau+1} | S_{(m)}^\tau; \omega^\tau; x^\tau, z^\tau)), \tag{7.2.1}$$

where the feasible decision set χ^τ is now defined as:

$$\chi^\tau = \{(x_i^\tau, z_i^\tau) | z_i^\tau = 1, \forall i \in I_u^\tau; z_i^\tau \in \{0,1\}, \forall i \in I_0^\tau; 0 \leq x_i^\tau \leq z_i^\tau, \forall i \in I; \sum_{i \in I} x_i^\tau = 1\}.$$

Namely the adding of start up cost results in an additional discrete decision for every network provider, which indicates whether we start to use this network to send data or not. When the start up cost c_i^0 is not paid, network i is assumed to be unavailable in the model when we make further routing decisions. Note that this modification will not complicate the decision problem too much as the current decision problem is solved approximately by an enumeration on several discrete choices of x_i^τ. Once we get the (near) optimal value of decision problem (7.2.1), we can update the regression parameters as before with

$$\beta^{(m)} = \beta^{(m-1)} - \alpha_{m-1}[\bar{V}_\tau^{(m-1)}(S^\tau) - (\hat{v}_\tau^{(m)} - \sum_{i \in I_u^\tau} c_i^0)] \nabla_{\beta^{(m-1)}} \bar{V}_\tau^{(m-1)}(S^\tau). \tag{7.2.2}$$

Note that as the $\hat{v}_\tau^{(m)}$ given by (7.2.1) is a sample estimation of $\tilde{V}_\tau(S^\tau)$, we need to take the start up cost $(\sum_{i \in I_u^\tau} c_i^0)$ off from it to obtain a sample estimation of $\bar{V}_\tau$. Therefore, the extension of TpTRP with start up cost for using one network provider can be covered by the ADP-Bézier-Curves/Surfaces model we stated in this work.

To deal with the situation of possible network failures, the presented framework could be adapted by introducing failure events as an additional random variable on top of the observed traffic. The interpretation of the value functions and the ADP algorithm would remain valid in this case. If the ADP model is trained using

the random events with failures considered, the resulting parameters should be able to account for the affect of failures.

Another interesting extension to the current TpTRP is if network providers use different lengths of time intervals. In this case there is no inherent one-to-one correspondence of time periods and DP stages, which forms another difficulty of the problem. For example, if network provider 2 divide a month into 8640 time intervals while network provider 1 divide it into 4320, then the period for network provider 1 is twice the length as it for network provider 2. Thus if we define the DP stage the same as time intervals for network provider 2, we need to revise the state dynamic and the decision making problem for network provider 1 to fix for the enlarged stage space. Fortunately, this can be achieved by adding artificial state variables for stage τ, if τ is not an end of a time interval for network provider 1. In detail, if Γ denotes the set of time intervals used by network provider 2 ($|\Gamma| = 8640$), for an arbitrary $\tau \in \Gamma$:

- If the beginning of τ is at the beginning of a time interval for network provider 1 (i.e. τ is even), then the artificial state $\tilde{S}^\tau$ and the state dynamic are defined as

$$\tilde{S}^\tau(S^{\tau-1}, \omega^\tau, x^\tau) = \{S^{\tau-1}, {}^{new}T_1^\tau, {}^{new}T_2^\tau\},$$

  where ${}^{new}T_i^\tau$ is defined as

$${}^{new}T_1^\tau = \begin{cases} \hat{T}^\tau & \text{if} \quad \hat{T}^\tau \le \frac{1}{2}\hat{T}_1^{\theta,\tau}, \\ \frac{1}{2}\hat{T}_1^{\theta,\tau} & \text{if} \quad \frac{1}{2}\hat{T}_1^{\theta,\tau} < \hat{T}^\tau \le \frac{1}{2}\hat{T}_1^{\theta,\tau} + \hat{T}_2^{\theta,\tau}, \\ \frac{1}{2}\hat{T}_1^{\theta,\tau} + x_1^\tau(\hat{T}^\tau - \frac{1}{2}\hat{T}_1^{\theta,\tau} - \hat{T}_2^{\theta,\tau}) & \text{if} \quad \hat{T}^\tau > \frac{1}{2}\hat{T}_1^{\theta,\tau} + \hat{T}_2^{\theta,\tau} \end{cases}$$

  and

$${}^{new}T_2^\tau = \begin{cases} 0 & \text{if} \quad \hat{T}^\tau \le \frac{1}{2}\hat{T}_1^{\theta,\tau}, \\ \hat{T}^\tau - \frac{1}{2}\hat{T}_1^{\theta,\tau} & \text{if} \quad \frac{1}{2}\hat{T}_1^{\theta,\tau} < \hat{T}^\tau \le \frac{1}{2}\hat{T}_1^{\theta,\tau} + \hat{T}_2^{\theta,\tau}, \\ \hat{T}_2^{\theta,\tau} + x_2^\tau(\hat{T}^\tau - \frac{1}{2}\hat{T}_1^{\theta,\tau} - \hat{T}_2^{\theta,\tau}) & \text{if} \quad \hat{T}^\tau > \frac{1}{2}\hat{T}_1^{\theta,\tau} + \hat{T}_2^{\theta,\tau} \end{cases}$$

- If the end of τ is at the end of a time interval for network provider 1 (i.e. τ is odd), then firstly we find the artificial state $\tilde{S}^\tau(S^{\tau-1}, \omega^\tau, x^\tau) = \{S^{\tau-1}, {}^{new}T_1^\tau, {}^{new}T_2^\tau\}$, where ${}^{new}T_i^\tau$ is defined as

$$
{}^{new}T_1^{\tau} = \begin{cases} {}^{new}T_1^{\tau-1} + \hat{T}^{\tau} & \text{if} \quad \hat{T}^{\tau} \leq \frac{1}{2}\hat{T}_1^{\theta,\tau}, \\ {}^{new}T_1^{\tau-1} + \frac{1}{2}\hat{T}_1^{\theta,\tau} & \text{if} \quad \frac{1}{2}\hat{T}_1^{\theta,\tau} < \hat{T}^{\tau} \leq \frac{1}{2}\hat{T}_1^{\theta,\tau} + \hat{T}_2^{\theta,\tau}, \\ {}^{new}T_1^{\tau-1} + \frac{1}{2}\hat{T}_1^{\theta,\tau} + x_1^{\tau}(\hat{T}^{\tau} - \frac{1}{2}\hat{T}_1^{\theta,\tau} - \hat{T}_2^{\theta,\tau}) & \text{if} \quad \hat{T}^{\tau} > \frac{1}{2}\hat{T}_1^{\theta,\tau} + \hat{T}_2^{\theta,\tau} \end{cases}
$$

and

$$
{}^{new}T_2^{\tau} = \begin{cases} 0 & \text{if} \quad \hat{T}^{\tau} \leq \frac{1}{2}\hat{T}_1^{\theta,\tau}, \\ \hat{T}^{\tau} - \frac{1}{2}\hat{T}_1^{\theta,\tau} & \text{if} \quad \frac{1}{2}\hat{T}_1^{\theta,\tau} < \hat{T}^{\tau} \leq \frac{1}{2}\hat{T}_1^{\theta,\tau} + \hat{T}_2^{\theta,\tau}, \\ \hat{T}_2^{\theta,\tau} + x_2^{\tau}(\hat{T}^{\tau} - \frac{1}{2}\hat{T}_1^{\theta,\tau} - \hat{T}_2^{\theta,\tau}) & \text{if} \quad \hat{T}^{\tau} > \frac{1}{2}\hat{T}_1^{\theta,\tau} + \hat{T}_2^{\theta,\tau} \end{cases}
$$

Then for both network providers, reorder entries in $\{S_i^{\tau-1}, {}^{new}T_i^{\tau}\}$ into a non-increasing order and delete the current $(\theta+1)$-th highest volume of traffic to obtain the real state S_i^{τ}. This is the same as if, we divide one time interval for network provider 1 into two to fix for the stage defined by network provider 2, and set the cut-off decision $y_1^{\tau} = \frac{1}{2}\hat{T}_1^{\theta,\tau}$. Routing decisions for the additional traffic at one time interval for network provider 1 however, is split into x_1^{τ} and $x_1^{\tau+1}$. With this structure, if working on the enlarged state space $\tilde{S}^{\tau} = \{S^{\tau-1}, {}^{new}T_1^{\tau}, {}^{new}T_2^{\tau}\}$, the presented ADP-Bézier-Curves/Surfaces model should be applicable as well. Although this might not be optimal, it provides an easy and reasonable way to deal with the different setup of time intervals.

7.3 On the ADP-Bézier-Curves/Surfaces solution method

To the best of our knowledge, the use of Bézier Curves/Surfaces to approximate the time-varying pattern of regression parameters in the ADP model is introduced in this work for the first time. It has been proved to be effective and efficient in solving the TpTRP, but the applicability of this technique is not restricted to this particular problem. Indeed, the aggregation using Bézier Curves/Surfaces within ADP should be applicable to the (vast) majority of problems in which the number of modelling parameters increases sharply with a/several particular dimension(s) while the value of those parameters changes smoothly over time. For example, in a vehicle assignment problem, as an often cited example in Powell [45] which can be solved by ADP, where state is represented by the 3-digit zip code of the location. As the zip code is naturally assigned according to the geographical

location, we have reason to believe that the value function can be assumed to be smooth and thus can be approximated by a 2-dimensional Bézier Surface.

Apart from this, multi-stage stochastic problems (integer or not) in which decisions should be made repeatedly for a number of periods can generally be solved by the presented ADP based approximation algorithm. For instance, consider a dynamic portfolio management problem, in which the investor can re-balance the portfolio at finite dates over a finite horizon (For a review of discrete time dynamic portfolio management models, see [32]). The uncertainty is forced on the price of risky assets. At the beginning of each period, the decision maker observes the current market prices for the risky assets and revises the portfolio composition, while the corresponding transaction costs have to be paid. Key features of the dynamic portfolio model are the non-linear risk averse utility function, and the rapidly increasing size of the problem resulting from the description of the uncertainty.

Although in Barro and Canestrelli [7] the authors have proved that the dynamic portfolio management problem can be solved efficiently by a clever application of stochastic programming approaches, the multi-period and clear dynamic features of the resulting stochastic programming model make it looks very similar to the TpTRP. The only difference is that, the stochastic programming model of TpTRP is hard to solve due to the integer variables used to count for the top-percentile time interval, while the dynamic portfolio management problem is hard due to the non-linear risk averse utility function in the objective. Therefore, it is reasonable to expect that the dynamic portfolio management problem can be solved efficiently using the presented ADP-Bézier-Curves/Surfaces model, if stage is naturally defined by the time period, and state is a combination of asset amounts that we currently hold in the portfolio. The value function approximation, although it might not be linear due to the non-linear risk averse utility function, should be well approximated by a quadratic regression model. In case where the number of time period is high enough, we can expect that the regression parameters change smoothly over time thus can be nicely approximated by Bézier Curves.

Appendix A

Publication List

A.1 Journal

1. A. Grothey and X. Yang, Top-percentile traffic routing problem by dynamic programming, *Optimization and Engineering, vol.12, Iuuse 4 (2011)*, DOI:10.1007/s11081-010-9130-2.

2. A. Grothey and X. Yang, Solving the top-percentile traffic routing problem by Approximate Dynamic Programming, submitted to *IMA Journal of Management Mathematics*, Feb 2010.

3. A. Grothey and X. Yang, Approximate Dynamic Programming with Bézier Curves/Surfaces for top-percentile traffic routing, accepted for *European Journal of Operational Research*, DOI:10.1016/j.ejor.2011.11.041, Nov 2011.

A.2 Conference Proceeding

1. A. Grothey and X. Yang, Top-percentile traffic routing problem and model, *2nd Student Conference on Operational Research*, Nottingham, Apr 2010.

Bibliography

[1] E. Altman, T. Basar, T. Jimnez, and N. Shimkin, *Competitive routing in networks with polynomial costs*, IEEE Transactions on Automatic Control, 47 (2002), pp. 92–96.

[2] M. Bagnulo, A. Garcia-Martinez, J. Rodriguez, and A. Azcorra, *The case for source address dependent routing in multihoming*, Lecture Notes In Computer Science, 3266 (2004), pp. 237–246.

[3] E. Balas, *Disjunctive programming*, Annals of Discrete Mathematics, 5 (1979), pp. 3–51.

[4] E. Balas, S. Ceria, and G. Cornuéjols, *A lift-and-project cutting plane algorithm for mixed 0-1 programs*, Mathematical Programming, 58 (1993), pp. 295–324.

[5] ——, *Mixed 0-1 programming by lift-and-project in a branch-and-cut framework*, Management Science, 42/9 (1996), pp. 1229–1246.

[6] E. Balas and M. Perregaard, *Lift-and-project for mixed 0-1 programming: recent progress*, Discrete Applied Mathematics, vol. 123, Issue 1-3 (2002), pp. 129–154.

[7] D. Barro and E. Canestrelli, *Dynamic portfolio optimization: Time decomposition using the maximum principle with a scenario approach*, European Journal of Operational Research, 163 (2005), pp. 217–229.

[8] E. Beale, *On minimizing a convex function subject to linear inequalities*, Journal of the Royal Statistical Society, 17B (1955), pp. 173–184.

[9] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, New Jersey, 1957.

[10] J. Benders, *Partitioning procedures for solving mixed-variables programming problems*, Numerische Mathematik, 4 (1962), pp. 238–252.

[11] J. Birge and F. Louveaux, *Introduction to Stochastic Programming*, Springer, New York, 1997.

[12] C. Blair and R. Jerslow, *The value function of a mixed-integer program ii*, Discrete Mathematics, 25 (1979), pp. 7–19.

[13] D. Boychuk and D. Martell, *A multistage stochastic programming model for sustainable forest-level timber supply under risk of fire*, Forest Science, Vol. 42, No. 1 (1996), pp. 10–26(17).

[14] C. Carøe and R. Schultz, *Dual decomposition in stochastic integer programming*, Operations Research Letters, 24(1) (1999), pp. 37–45.

[15] C. Carøe and J. Tind, *A cutting-plane approach to mixed 0-1 stochastic integer programs*, European Journal of Operational Research, 101 (1997), pp. 306–316.

[16] ——, *L-shaped decomposition of two-stage stochastic programs with integer recourse*, Mathematical Programming, 83 (1998), pp. 451–464.

[17] S. Ceria, *Lift-and-project cuts: an efficient solution method for mixed-integer programs*, technical report, Columbia University, 1999.

[18] S. Ceria, C. Cordier, H. Marchand, and L. Wolsey, *Cutting planes for integer programs with general integer variables*, Mathematical Programming, vol. 81, No. 2 (1998), pp. 201–214.

[19] M. Chardy, A. Ouorou, and T. VanDonselaar, *Optimization of interconnection strategy in top-percentile pricing framework*, technical report, Orange Labs, France Telecom, 38-40 rue du général Leclerc, BP 92130, Issy-les-Moulineaux, 2009.

[20] J. Choi and G. Elkaim, *Bézier curves for trajectory guidance*, World Congress on Engineering and Computer Science, WCECS 2008, San Francisco, CA, Oct.22-24 (2008).

[21] G. Dantzig, *Linear programming under uncertainty*, Management Science, 1 (1955), pp. 197–206.

[22] V. Demiguel and N. Mishra, *What multistage stochastic programming can do for network revenue management*, technical report, London Business School, 6 Sussex Place, Regent's Park, London NW1 4SA, UK, July 2008.

[23] M. Dyer and L. Stougie, *Computational complexity of stochastic programming problems*, Mathematical Programming, 106 (2006), pp. 423–432.

[24] G. Farin, *Curves and surfaces for computer aided geometric design (3rd ed.):a practical guide*, Academic Press Professional, San Diego, CA, USA, 1993.

[25] M. Flakner, M. devetsikiotis, and I. Lambadaris, *An overview of pricing concepts for broadband ip networks*, IEEE Communications Surveys and Tutorials, 3(2) (2000), pp. 2–13.

[26] V. Goel and I. Grossmann, *A stochastic programming approach to planning of offshore gas field developments under uncertainty in reserves*, Computer & Chemical Engineering, Vol. 28, Issue 8 (2004), pp. 1409–1429.

[27] D. Goldenberg, L. Qiu, H. Xie, Y. Yang, and Y. Zhang, *Optimizing cost and performance for multihoming*, ACM SIGCOMM Computer Communication Review, 34 (2004), pp. 79–92.

[28] R. Gomory, *Outline of an algorithm for integer solutions to linear programs*, Bulletin of the American Mathematical Society, 64 (1958), pp. 275–278.

[29] W. Haneveld and M. V. D. Vlerk, *Stochastic integer programming: general models and algorithms*, Annals of Operations Research, 85 (1999), pp. 39–57.

[30] M. Herzberg and F.Shleifer, *Optimization models for the design of bi-directional self-healing ring based networks*, Teletraffic Science and Engineering, 3 (1999), pp. 183–194.

[31] K. Huang and S. Ahmed, *The value of multistage stochastic programming in capacity planning under uncertainty*, Operations Research, Vol. 57, No. 4 (2009), pp. 893–904.

[32] R. Jarrow, V. Maksimovic, and W. Ziemba(Eds.), *Finance – Handbooks in Operations Research and Management Science*, North-Holland, Amsterdam, 1995.

[33] M. Jong and H. Jay, *Approximate dynamic programming strategies and their applicability for process control: A review and future directions*, International Journal of Control, Automation and Systems, vol.2 no.3 (2004), pp. 263–278.

[34] H. Kushner and G. Yin, *Stochastic approximation algorithms and applications*, Springer-Verlag, New York, 1997.

[35] G. Laporte and F. Louveaux, *The integer l-shaped method for stochastic integer programs with complete recourse*, Operations Research Letters, 13 (1993), pp. 133–142.

[36] J. Levy, H. Levy, and Y. Kahana, *Top percentile network pricing and the economics of multi-homing*, Annals of Operations Research, 146 (2006), pp. 153–167.

[37] X. Li, E. Armagan, A. Tomasgard, and P. Barton, *Long-term planning of natural gas production systems via a stochastic pooling problem*, Proceedings of the 2010 American Control Conference, ACC 2010, 5531053 (2010), pp. 429–435.

[38] A. Løkketangen and D. Woodruff, *Progressive hedging and tabu search applied to mixed integer (0, 1) multistage stochastic programming*, Journal of Heuristics, 2 (1996), pp. 111–128.

[39] J. McClain, *Dynamics of exponential smoothing with trend and seasonal terms*, Management Science, 20 (1974), pp. 1300–1304.

[40] A. Möller, W. Römisch, and K. Weber, *Airline network revenue management by multistage stochastic programming*, Computational Management Science, Vol. 5, No. 4 (2008), pp. 355–377.

[41] J. Nascimento and W. Powell, *An optimal approximate dynamic programming algorithm for the lagged asset acquisition problem*, Mathematics of Operations Research, vol. 34, no. 1 (2009), pp. 210–237.

[42] G. Nemhauser and L. Wolsey, *Integer and Combinatorial Optimization*, Wiley Interscience Series in Discrete Mathematics and Optimization, Wiley, New York, 1988.

[43] A. Odlyzko, *Internet pricing and the history of communications*, Computer Networks, 36 (2001), pp. 493–517.

[44] W. Powell, *The optimizing-simulator: Merging simulation and optimization using approximate dynamic programming*, Proceedings of the 2005 Winter Simulation Conference, (2005).

[45] ——, *Approximate Dynamic Programming - Solving the Curses of Dimensionality*, John Wiley & Suns, New Jersey, 2007.

[46] H. Robbins and S. Monro, *A stochastic approximation method*, The Annals of Mathematical Statistics, 22 (1951), pp. 400–407.

[47] R. Rockafellar and R. Wets, *Scenarios and policy aggregation in optimization under uncertainty*, Mathematics of Operations Research, 16 (1991), pp. 119–147.

[48] A. Ruszczyński, *Decomposition methods in stochastic programming*, Mathematical Programming B, 79 (1997), pp. 333–353.

[49] R. Schultz, *On structure and stability in stochastic programs with random technology matrix and complete integer recourse*, Mathematical Programming, 70/1 (1995), pp. 73–90.

[50] R. Schultz, L. Stougie, and M. van der Vlerk, *Two-stage stochastic integer programming: a survey*, Statistica Neerlandica, Vol. 50, nr. 3 (1996), pp. 404–416.

[51] S. Sen and J. Higle, *The c^3 theorem and a d^2 algorithm for large scale stochastic optimization: Set convexification*, Mathematical Programming, vol.104, Issue 1 (2005), pp. 1–20.

[52] S. Sen and H. Sherali, *Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming*, Mathematical Programming, vol.106, no.2 (2006), pp. 203–223.

[53] S. shenker, D. Clark, and S. H. D. Estrin, *Pricing in computer networks: Reshaping the research agenda*, Telecommunications Policy, 20(3) (1996), pp. 183–201.

[54] H. Sherali and B. Fraticelli, *A modification of benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse*, Journal of Global Optimization, 22 (2002), pp. 319–342.

[55] R. V. Slyke and R. Wets, *L-shaped linear program with application to optimal control and stochastic linear programming*, SIAM Journal and Applied Mathematics, 17 (1969), pp. 638–663.

[56] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 1995.

[57] S. Wallace and W. Ziemba, *Applications of stochastic programming*, MPS-SIAM Book Series on Optimization 5, 3600 University City Science Centre Philadelphia, PA 19104-2688 USA, 2005.

[58] X. Wang and H. Schulzrinne, *Pricing network resources for adaptive applications*, IEEE/ACM Transactions on Networking (TON), 14 (2006), pp. 506–519.

[59] M. Wasan, *Stochastic Approximation*, Cambridge University Press, Cambridge, 1969.

[60] R. Wollmer, *Two-stage linear programming under uncertainty with 0-1 first stage variables*, Mathematical Programming, 19 (1980), pp. 279–288.