APPROXIMATE DYNAMIC PROGRAMMING WITH BÉZIER CURVES/SURFACES FOR TOP-PERCENTILE TRAFFIC ROUTING

ANDREAS GROTHEY, XINAN YANG
SCHOOL OF MATHEMATICS
COLLEGE OF SCIENCE AND ENGINEERING
THE UNIVERSITY OF EDINBURGH

Abstract. Multi-homing is used by Internet Service Provider (ISP) to connect to the Internet via different network providers. This study investigates the optimal routing strategy under multi-homing in the case where network providers charge ISPs according to top-percentile pricing (i.e. based on the θ-th highest volume of traffic shipped). We call this problem the Top-percentile Traffic Routing Problem (TpTRP).

Solution approaches based on Stochastic Dynamic Programming require discretization in state space, which introduces a large number of state variables. This is known as the curse of dimensionality. To overcome this we suggested to use Approximate Dynamic Programming (ADP) to construct approximations of the value function in previous work, which works nicely for medium size instances of TpTRP. In this work we keep working on the ADP model, use 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 real-sized TpTRP tractable.

Keywords: top-percentile pricing, multi-homing, stochastic, routing policy, approximate dynamic programming, Bézier Curves/Surfaces

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. This technique is currently widely adopted to provide fault tolerance and traffic engineering capabilities [1].

Traditionally network providers charge ISPs based on a combination of fixed cost and per usage pricing. Top-percentile pricing is a relatively new and increasingly popular pricing regime used by network providers to charge service providers (although it usually appears as part of a mixed pricing strategy), that is quickly becoming established [8]. In this scheme, the network provider divides the charge period, say a month, into several time intervals with equal, fixed length. Then, it measures and evaluates the amount of data (traffic) sent in these time intervals. At the end of the charge period, the network provider selects the traffic volume of the top q-percentile interval as the basis for computing the cost. For example, if the charge period (i.e. 30 days) is divided into 4320 time intervals with the length of 10 mins, 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.

It has been discussed (e.g. in [8]) what the optimal multi-homing routing strategies look like under traditional pricing regimes and whether they are economically viable. In contrast, very little work has been done on network operation under top-percentile pricing. The deterministic problem (in which we assume that we know all the traffic volumes in advance) has been analysed in [2], 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. in [7] develop a probabilistic model and provide analysis of the expected costs, thus demonstrate that multi-homing can be economical efficient under top-percentile pricing though they did not investigate the optimal routing policy. On the other hand, Goldenberg et al. [4] focus on the development of smart routing algorithms for optimising both cost and performance for multi-homing users under top-percentile pricing. However, in case where traffic volumes are not available in advance (stochastic case), the algorithm only depends on the prediction of one later time interval's traffic but the expectation of the future cost. 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 in the stochastic case.

The purpose of our study is to find the optimal routing strategy in order to allow the ISP to make full use of the underlying networks with minimum cost, when all network providers charge the ISP based on the volume of the top q-percentile time interval's traffic (pure top-percentile pricing). 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 can not predict the volume of future time intervals' traffic. Instead, we assume that the ISP knows the probabilistic distributions of every time intervals' traffic ahead of time.

In [5], the authors have shown that solving the TpTRP as an Stochastic Mixed-integer Programming (SMIP) problem is intractable for all but extremely small instances, due to the fact that modelling of the top-percentile cost requires the introduction of integer variables within the final time stage, which make the problem non-convex thus hard to solve. On the other hand, we suggested a Stochastic Dynamic Programming (SDP) model based on a discretization of the state space, which gives routing policies that outperform all available naive routing policies for small sized instances. However due to the huge number of states arising from the discretization of traffic volumes, an effect well known as the curse of dimensionality prevents the use of the SDP model on larger problem instances. As a modification, in [6] the authors applied the Approximate Dynamic Programming (ADP) technique to solve the TpTRP, which allows to work on the continuous state space thus overcoming the curse of dimensionality introduced by the discretization. With the ADP model, medium sized TpTRP instances can be solved within reasonable time.

This work follows the study in [6], where we intend to develop an ADP model based aggregation algorithm to make the real sized TpTRP problem tractable. The focus of this work is on the investigation of the parameter structure in the original discrete ADP model given by [6], and the resulting Bézier Curves/Surfaces aggregation of the original ADP model. In the remainder of this paper, we firstly give the parameters of the TpTRP problem and its basic SDP modelling elements in Section 2. In Section 3, we give a brief introduction to the ADP technique and build the ADP model. Then we analysis the problem of the current ADP model and show how to exploit its special structure of data to improve it with Bézier Curves

in Section 4. Section 5 gives the numerical results and provides a stronger – Bézier Surfaces aggregation model, which makes the real-world sized instances tractable. Finally we give conclusions in Section 6.

2. The Top-percentile Traffic Routing Problem

This section gives a formal description of the TpTRP parameters and defines the main modelling elements in the dynamic programming model formulation.

2.1. Notations and Assumptions.

Problem parameters.

- $I, |I| = n$: The set of network providers.
- Γ : The set of time intervals.
- q : The percentile parameter.
- $\theta = \lfloor |\Gamma| * q \rfloor$: The index of the top-percentile time interval.
- $c_i, i \in I$: The per unit cost charged by network provider i on the top-percentile traffic.

In this work, we assume that all network providers divide the charge period equally into $|\Gamma|$ time intervals. Network providers use pure top-percentile pricing with parameter q, namely the cost charged on the ISP depends solely on the θ-th highest volume of traffic that has been sent to network provider i. It is worthwhile to point out that, under this assumption, the ISP can ship several time intervals' traffic via a network without being charged, provided traffic shipped during the top-percentile time interval is zero. We also assume that there is no upper bound on the volume of traffic that can be shipped to each network provider, and no failure occurring in any network during the charge period.

- $T^\tau, \tau \in \Gamma$: The volume of traffic in time interval τ.

We assume that before the routing decision for period τ is made, $T^\tau(\omega^\tau)$ is a random variable depending on the random event ω^τ. When the random event $\hat{\omega}^\tau$ becomes known, we use $\hat{T}^\tau = T^\tau(\hat{\omega}^\tau)$ to represent the realisation of T^τ.

State variables and value function. In our problem, at the beginning of time interval τ, we know all the previous realisations of traffic volumes $\hat{T}^t, t = 1, ..., \tau - 1$ and routing decisions $x^t, t = 1, ..., \tau - 1$. The implied usage $\hat{T}_i^t = T_i^t(\hat{T}^t, x^t), t = 1, ..., \tau - 1$ of network i can be computed. Then a combination of $\{\hat{T}_i^t | t = 1, ..., \tau - 1; i = 1, ..., n\}$ defines the current state S^τ of the system. 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 rewrite $S^\tau = \{\hat{T}_i^{j,\tau} | i = 1, ..., n; j = 1, ..., \tau - 1\}$.

However, under pure top-percentile pricing policy 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 time interval τ, only traffics which are 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. Instead, any traffic which is no higher than the current θ-th volume of traffic (namely, traffics $\hat{T}_i^{j,\tau}, j = \theta + 1, ..., \tau - 1$ at time interval τ) has no impact on the final cost. Noting this, we delete this redundant information from the state space, which leads to the state at τ being described by

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

Namely the dimension of the state space is equal to $n\theta$.

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.

Decision variables.

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

In our model, x^τ is the decision made on the proportional routing of the 'additional traffic' [1]. Given a state $S^\tau = \{\hat{T}_i^{j,\tau}\}$, it is obvious that if we send no more than $\hat{T}_i^{\theta,\tau}$ to provider i then the system will remain in this state for time interval $\tau + 1$. The additional traffic represents the amount of traffic exceeding $\hat{T}_i^{\theta,\tau}$ that cannot be sent without affecting the current θ-th highest volume of traffic of any network provider. Making decision on the additional traffic allows us to use most of every network provider, thus is appropriate. A detailed justification of this argument is given in [6]. The feasible decision set for time interval τ is thus,

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

When implementing a decision x^τ, we allocate the random traffic T^τ according to the following rule:

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

(2) If $\hat{T}^\tau \geq \sum_{i \in I} \hat{T}_i^{\theta,\tau}$, we send:
   - ${}^{new}T_i^\tau = \hat{T}_i^{\theta,\tau} + x_i^\tau (\hat{T}^\tau - \sum_{i \in I} \hat{T}_i^{\theta,\tau})$ to provider $i \in I$.

Namely 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.

3. Approximate Dynamic Programming Model

Given the definitions of state representation and feasible decision set, the Tp-TRP can be solved by dynamic programming. Starting from the final time stage, the expected future cost and optimal routing decision for all possible states can be computed by stepping backwards though time. However, traditional dynamic programming is only applicable for discrete state spaces. The discretization of the state space combined with the large dimension of the state space will result in a large number of states, which prevents large sized instances being solved.

To avoid this problem, we have suggested an ADP model in [6]. It replaces the look-up table representation of the value function by a continuous regression model, thus reduces the number of parameters required to be estimated. During every iteration, we follow a new sample path and make routing decisions according to

[1] The additional traffic is defined as: $T_{Add}(S^\tau, T^\tau) = \max\{\hat{T}^\tau - \sum_{i \in I} \hat{T}_i^{\theta,\tau}, 0\}$.

the current value function estimation, then update the regression model iteratively with a stochastic gradient algorithm until the value function estimation converges. This makes the process more efficient as it focuses on the states which are more likely to be visited as well as significantly reducing the number of parameters in the model.

3.1. **ADP model.** The basic Approximate Dynamic Programming algorithm is summarised below [9]:

Step 0. Initialisation:

Step 0a. Build a 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 $\omega_{(m)} = (\omega^1_{(m)}, ..., \omega^{|\Gamma|}_{(m)})$.

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

Step 2a. Solve

$$\hat{v}_\tau^{(m)} = \min_{x^\tau \in \chi^\tau} (\mathbb{E}_{\omega^\tau \in \Omega^\tau} \bar{V}_{\tau+1}^{(m-1)}(S^{\tau+1} | S^\tau_{(m)}, \omega^\tau, x^\tau)). \tag{3.1}$$

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 $S^{\tau+1}_{(m)}(S^\tau_{(m)}, \omega^\tau_{(m)}, \hat{x}^\tau)$, where $\hat{x}^\tau$ is the optimal solution of (3.1).

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

Specifically, we approximate the value function by linear regression model in Step 0a:

$$\bar{V}_\tau(S^\tau) = \beta_0^\tau + \sum_{i \in I} \sum_{1 \le j \le \theta} \beta^\tau_{i,j} \hat{T}_i^{j,\tau}, \tag{3.2}$$

which means that we suppose the value function changes linearly with every entry of the state variable. The update used in Step 2b is derived from the stochastic gradient algorithm and given by:

$$\beta^{(m)} = \beta^{(m-1)} - \alpha_{m-1}[\bar{V}_\tau^{(m-1)}(S^\tau) - \hat{v}_\tau^{(m)}] \nabla_{\beta^{(m-1)}} \bar{V}_\tau^{(m-1)}(S^\tau), \tag{3.3}$$

where the updating stepsize α_m is defined by the McClain's formula ($\bar{\alpha}$ is a specified parameter):

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

The decision problem given in Step 2a is built with the 'current' estimation of parameters $\beta^{(m-1)}$, namely parameters updated with the previous $m-1$ iterations. Due to the requirement of reordering entries in state variable after a new traffic is allocated, the dynamic step from one time interval to the next (in Step 2a) is

non-trivial. As a result, the decision problem in Step 2a is non-convex, which makes it difficult to solve to global optimality. (a detailed discussion on this issue is given in [6]). Thus in the ADP model we suggested to solve the decision problem by a simple discretization of the decision space, i.e., generating several discrete decisions (for example $x^\tau = 0.0, 0.1, 0.2...1.0$), and finding the best one by enumeration.

As the stochastic gradient algorithm typically converges rapidly at the beginning and then vibrates with noise, in this work we check for convergence of the ADP model by evaluating the mean cost over blocks of iterations (e.g. blocks of every 100 iterations). Once we observe the mean cost changes mainly with noise instead of decreasing / increasing rapidly, we stop and treat the current model as converged model. This forms the stopping criterion in Step 3.

3.2. **Problem size.** As shown in formula (3.2), for every time interval τ we set a single value function estimation $\bar{V}_\tau$ to approximate V_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 1). However, for larger sized instances it is still hard. 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 1 we can see that the number of regression parameters required in the discrete ADP model grows quadratically with the number of time intervals. This means it will take several hours to achieve convergence for the 432-period model.

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

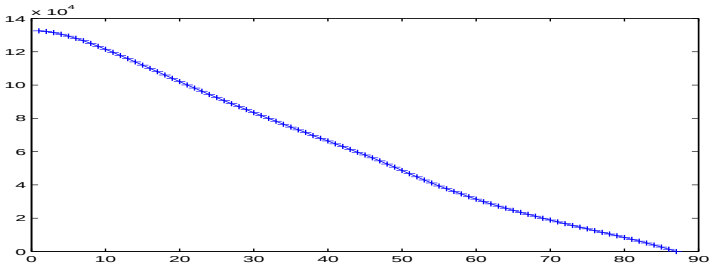
Table 1: Size of the ADP model and its regression information

To solve this problem, we suggest to aggregate 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.

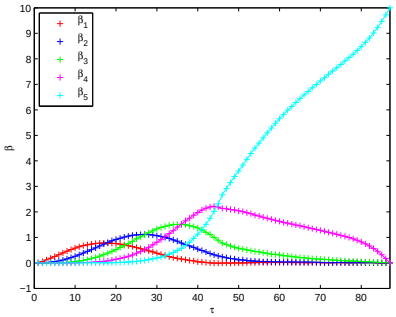
4. Time-Aggregated ADP Model

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 (for the detail of this ADP model please refer to [6]).

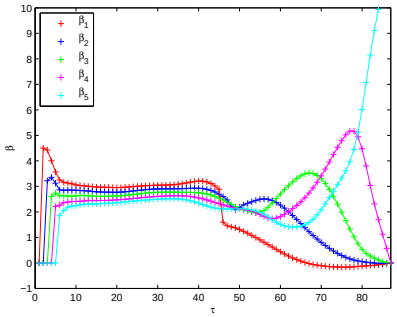
Figure 4.1 shows how the optimal β_0^τ, $\beta_{i,j}^\tau$ vary with time τ for the 86-period Instance 2 (for instance parameters see Table 1). Every point shows an estimation of β_0, $\beta_{i,j}$ at some time point τ in the trained model. It is obvious that every single β_0, $\beta_{i,j}$ has its time varying pattern, which is smooth (or near smooth) thus can be approximated with less parameters. The purpose of this study is to replace the discrete values β_0^τ, $\beta_{i,j}^\tau$ by approximating function $\beta_0(\tau)$, $\beta_{i,j}(\tau)$ with a small number of parameters, which are updated by the normal ADP iteration.



(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 4.1: Traffic distribution used in testing instances

4.1. **Bézier Curve.** In this work we suggest to use Bézier Curves approximating functions. Bézier curves were widely publicised in 1962 by the French engineer Pierre Bézier, who used them to design automobile bodies [3]. A Bézier Curve is a parametric curve that is frequently used to produce curves which appear reasonably smooth. Mathematically, Bézier Curves approximate polynomials depend on certain control points. Given a large enough number of properly selected control points, any smooth function can be approximated by Bézier Curve to arbitrary accuracy.

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

$$\begin{aligned}\mathbf{B}(u) &= \sum_{k=1}^{K} \begin{pmatrix} K \\ k \end{pmatrix} (u)^k (1-u)^{K-k} \mathbf{P_k} \\ &= (1-u)^K \mathbf{P_0} + \begin{pmatrix} K \\ 1 \end{pmatrix} (1-u)^{K-1} u \mathbf{P_1} + ... \\ &\quad ... + \begin{pmatrix} K \\ K-1 \end{pmatrix} (1-u)(u)^{K-1} \mathbf{P_{K-1}} + u^K \mathbf{P_K}, u \in [0,1],\end{aligned}$$

where $\begin{pmatrix} K \\ k \end{pmatrix}$ is the binomial coefficient.

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

$$\begin{pmatrix} \tau \\ \bar{\beta}_{i,j}(\tau) \end{pmatrix} = \sum_{k=1}^{K} \begin{pmatrix} K \\ k \end{pmatrix} (u)^k (1-u)^{K-k} \begin{pmatrix} \hat{\tau}^k_{i,j} \\ \beta^k_{i,j} \end{pmatrix}. \tag{4.1}$$

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

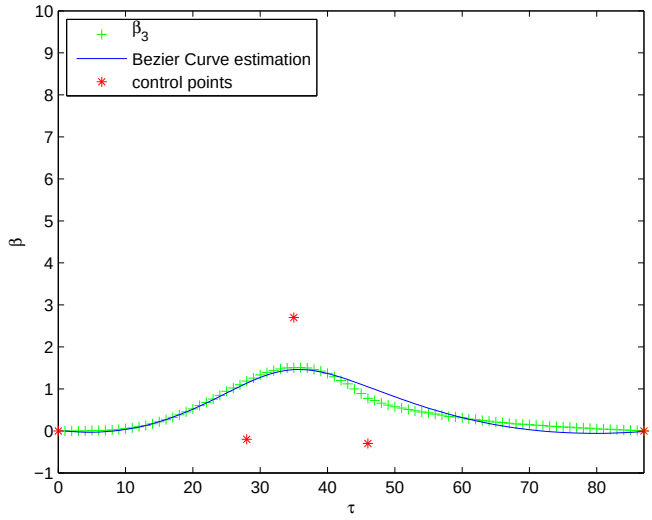


Figure 4.2: Comparison of original discrete values and the Bézier Curve approximation with $K = 5$

Figure 4.2 shows an example of how the 5-degree Bézier Curve works in the estimation of $\beta_{1,3}(\tau)$, taken from the instance shown in Figure 4.1. We can see that for the given choice of $\{\hat{\tau}^k_{i,j}, k = 1, ..., K\}$, the 5-degree Bézier Curve can approximate the discrete set of $\beta^\tau_{1,3}$ reasonably well by a continuous curve. This means we can replace the original discrete regression model (with 86 coefficients to estimate: $\beta^\tau_{1,3}, \tau = 1, ..., 86$) by a continuous function with only 5 parameters $(\beta^k_{1,3}, k = 1, ..., 5)$ to estimate. This reduces the size of the problem, thus speeding up the convergence of the ADP model.

4.3. **ADP-Bézier-Curve model.** Now we describe the aggregated ADP-Bézier-Curve algorithm we use in this work.

Initialisation – Step 0a. For the simplicity and generality of the model, in this work we use the Bézier Curve model with fixed values $\{\hat{\tau}^k, k = 1, ..., K\}$, while updating values $\{\beta^k_{i,j}, k = 1, ..., K\}$ 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 solutions u_τ of polynomial equations:

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

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

Decision problem – Step 2a. 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 (3.2) is still assumed to be a linear regression function of state variables. The only difference from before is that the regression parameters $\beta_{i,j}^\tau$s, are now approximated by Bézier Curves (4.1). Thus to get the current value function estimation (3.2), we firstly need to calculate all the current estimation (estimation after $m - 1$ iterations) of the $\beta_{i,j}^\tau$ values using:

$$\bar{\beta}_{i,j}^{(m-1)}(\tau) = \sum_{k=1}^{K} \left(\begin{array}{c} K \\ k \end{array} \right) (u_\tau)^k (1 - u_\tau)^{K-k} \beta_{i,j}^{k,(m-1)}. \tag{4.2}$$

Updating scheme – Step 2b. The parameter update in Step 2b. now becomes:

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

Thus in the aggregated ADP model every time we get a sample estimation of the value function $\hat{v}_\tau$ ($\forall \tau \in [0, |\Gamma|]$), we can update all $\beta_{i,j}^k, k = 1, ..., K$ at once, which accelerate the convergence speed significantly.

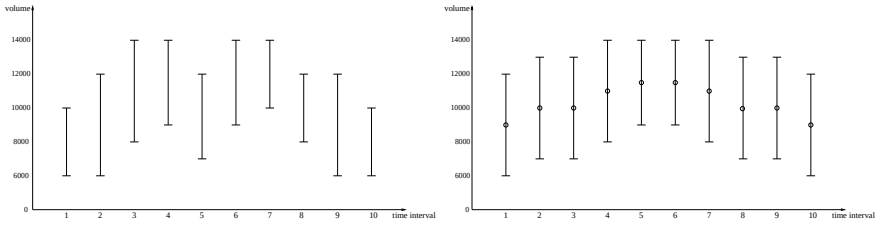
5. Numerical results

5.1. **Test Problems.** In this section we discuss the numerical results of applying the ADP-Bézier-Curve algorithm. We start by the TpTRP instances with 86-period from [6]. For clarity, we firstly characterise and index these instances which are examined in the later part of this section.

Table 2 summarises the instances used. In all instances, we assume that we divide the modelling region into 86 time intervals and charges are based on the time interval with the 5th ($q = 5\%$) highest volume of traffic. In all cases we use 2 network providers ($n = 2$) with costs $c_1 = 10, c_2 = 12$. The instances differ by the assumptions made on the random traffic. In instance 2 and 4 the traffic in every period follows the same uniform ($U(6000, 14000)$ in Instance 2) or normal

Index	Parameters			Stochastic Information	
	$\|\Gamma\|$	θ	n	distribution	time dependency
Ins.2	86	5	2	$U(6000, 14000)$	i.i.d.
Ins.3	86	5	2	uniform	see Fig. 5
Ins.4	86	5	2	truncated $N(10000, 10^6)$	i.i.d.
Ins.5	86	5	2	truncated normal	see Fig. 6

Table 2: List of TpTRP Instances



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

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

Figure 5.1: Traffic distribution used in testing instances

($N(10000, 10^6)$ in Instance 4) distribution. Instance 3 and 5 on the other hand, use traffic distributed according to a time varying uniform or normal distribution. The parameter for this time varying pattern is displayed in Figures 5.1. Note that Instances 4 and 5 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.

5.2. **Numerical results on 86-period TpTRP instances.** In this section we evaluate the ADP-Bézier-Curve model by testing it on several instances with 86 periods. For every testing instance we build its own ADP-Bézier-Curve model, train this model with random scenarios until it converges, 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. All the results are compared with the original discrete ADP model developed in [6], and three naive routing policies summarised below:

- 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 randomly $\theta - 1$ traffics 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;
- DRP - Deterministic Routing Policy, i.e. assuming we know all traffics in advance. The optimal routing policy (as proved in [5]) is to send the $\theta - 1$ highest traffics 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.

Table 3 shows the comparison of mean cost of implementing several routing policies, where K indicates the number of control points in the ADP-Bézier-Curve model.

Ind.	SRP	TMRP	ADPRP	K	ADPRP_BC	DRP
Ins.2_86	135404.34±1.98	135181.68±2.08	132902.35±2.71	3 4 5	133595.05±3.13 132739.81±2.77 132809.52±2.68	131727.00±2.60
Ins.2_216	135945.63±1.19	135749.17±1.25	-	4	133212.44±1.71	132258.12±1.60
Ins.2_432	135935.06±0.84	135727.89±0.88	-	4	132965.84±1.28	132054.59±1.15
Ins.3_86	132585.06±3.00	131588.78±3.35	129071.22±3.57	3 4 5	129980.79±4.65 128400.82±3.85 129645.63±3.67	126686.15±3.64
Ins.3_216	133663.71±1.77	132838.35±1.98	-	4	130575.08±1.98	127902.03±2.27
Ins.3_432	133770.80±1.24	132930.47±1.39	-	4	129602.03±1.88	127826.36±1.60
Ins.4_86	116104.12±2.22	115904.52±2.24	113892.97±2.46	3 4 5	114614.44±2.52 113631.76±2.16 113680.34±2.11	112833.05±1.84
Ins.4_216	116549.93±1.44	116319.57±1.46	-	4	113952.68±1.37	113091.56±1.18
Ins.4_432	116454.96±1.02	116212.32±1.03	-	4	113844.21±1.07	112898.46±0.83
Ins.5_86	123039.58±2.33	122175.78±2.40	121002.27±2.38	3 4 5	123850.12±2.95 122405.06±2.72 120497.46±2.22	119310.87±1.97
Ins.5_216	123705.80±1.50	122906.99±1.54	-	5	120918.74±1.40	119876.40±1.25
Ins.5_432	123720.58±1.05	122900.68±1.08	-	5	120906.23±1.32	119804.31±0.87

Table 3: Comparison of mean cost (± s.d.) of discrete ADPRP and ADP with Bézier Curve

Generally speaking, the routing policy generated by the ADP-Bézier-Curve model performs well. In almost all cases the ADPRP_BC routing policy outperforming the trivial routing policies, sometimes even better than the ADPRP. Specifically, the ADPRP_BC with $K = 4$ works fine for Instance 2, 3 and 4, while $K = 5$ seems better for Instance 5. With the best selection of K, mean costs given by ADPRP_BC can be (in most cases) even less than the ADPRP generated with the discrete ADP model.

In addition to the comparison with other routing policies, it is also worthwhile to point out that the performance of ADP-Bézier-Curve model is not necessarily getting better with the number of control points K, though it should be true in our expectation. Look at the results for the 86-period instances, we can see that the ADPRP_BC with $K = 5$ might be a little worse than the one with $K = 4$. This is because no matter how many control point we use in the Bézier Curve model, we always set their τ-components equidistant within the charging period. This might make the position of control points in the larger model worse than the ones in the smaller model in the approximation of function shape, especially in cases when K is small. 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 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 βs 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 instance size (given the

Ind.	ADP_discrete				ADP_BC				
	βs	Iterations	Time	T/I	K	βs	Iterations	Time	T/I
Ins.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
Ins.2_216	4968	-	-	-	4	90	4,000	62.285s	0.0156s
Ins.2_432	19440	-	-	-	4	180	3,000	252.951s	0.0843s
Ins.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
Ins.3_216	4968	-	-	-	4	90	3,000	38.535s	0.0128s
Ins.3_432	19440	-	-	-	4	180	3,000	211.739s	0.0706s
Ins.4_86	946	1,800,000	13590.433s	0.0076s	3	32	4,000	96.594s	0.0241s
					4	42	6,000	158.715s	0.0265s
					5	52	7,000	187.663s	0.0268s
Ins.4_216	4968	-	-	-	4	90	7,000	836.776s	0.1195s
Ins.4_432	19440	-	-	-	4	180	4,000	2349.898s	0.5875s
Ins.5_86	946	2,000,000	14351.873s	0.0072s	3	32	4,000	70.995s	0.0177s
					4	42	4,000	73.869s	0.0185s
					5	52	5,000	93.914s	0.0188s
Ins.5_216	4968	-	-	-	5	112	6,000	732.712s	0.1221s
Ins.5_432	19440	-	-	-	5	225	3,000	1810.346s	0.6034s

Table 4: Comparison of running time of ADPRP and ADPRP_BC

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/I) takes around twice the time longer than the discrete ADP model.

Results of mean cost and running time on larger instances are summarised in Table 3 and 4 as well. We can see with the current ADP-Bézier-Curve model, TpTRP instances up to 216 periods can be solved within reasonable time (around 10 mins). 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.

5.3. **Two dimensional approximation with Bézier Surface.** From Table 4 we can see that although the number of control points (K) stays the same with increasing problem size, the number of βs still increases linearly with θ. For real sized instances which possesses $n = 2$ network providers, $|\Gamma| = 4320$ time intervals and $\theta = 215$, it requires $K \cdot n(\theta + 2) = 434K$ 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, in this section we extend the aggregation to two dimensions with Bézier Surfaces.

The higher dimensional Bézier Curve is called a Bézier Surface. Figure 5.2 gives a two dimensional view of Figure 4.1(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 4.1(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 5.2 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

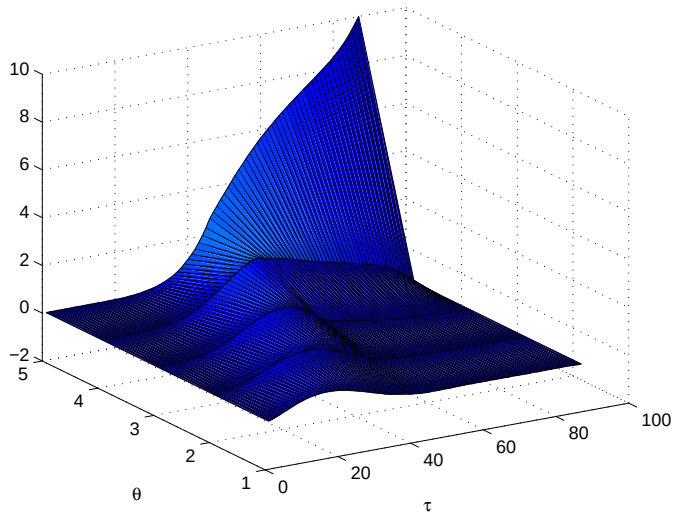


Figure 5.2: Converged regression coefficients $\beta^{\tau}_{1,j}$ in 86-period Instance 2

corresponding parameter $\beta_i^{k,r}$. Assuming that we set K τ-components of control points $\{\hat{\tau}_i^k, k = 1, ..., K\}$ and R j-components of control points $\{\hat{j}_i^r, r = 1, ..., R\}$, 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:

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

Numerical results on instances with 432 periods are shown in Table 5 and 6.

Ind.	SRP	TMRP	ADPRP_BC	ADPRP_BS	DRP
Ins.2_432	135935.06±0.84	135727.89±0.88	132965.84±1.28	132931.12±1.26	132054.59±1.15
Ins.3_432	133770.80±1.24	132930.47±1.39	129602.03±1.88	129964.61±1.75	127826.36±1.60
Ins.4_432	116454.96±1.02	116212.32±1.03	113844.21±1.07	113829.03±1.05	112898.46±0.83
Ins.5_432	123720.58±1.05	122900.68±1.08	120906.23±1.32	120934.34±1.20	119804.31±0.87

Table 5: Comparison of mean cost (± s.d.) of ADPRP_BC and ADPRP_BS on 432-period instances

Ind.	K	βs	Iterations	Time	T/I	K	R	βs	Iterations	Time	T/I
Ins.2_432	4	180	3,000	252.951s	0.0843s	4	3	28	1,000	90.013s	0.0900s
Ins.3_432	4	180	3,000	211.739s	0.0706s	4	3	28	1,000	75.648s	0.0756s
Ins.4_432	4	180	4,000	2349.898s	0.5875s	4	3	28	1,000	684.169s	0.6842s
Ins.5_432	5	225	3,000	1810.346s	0.6034s	5	3	35	1,000	752.156s	0.7522s

Table 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 can be compared with their counterparts of ADP-Bézier-Curve model, which are all better than any naive routing policy. However, Table 6 shows the ADP-Bézier-Surface model saves about 2/3 of the training time of the model, thus making the TpTRP instances with 432 periods solvable within reasonable time.

5.4. **Real-sized Instances.** Though we make the above aggregation to reduce the number of regression parameters, it is still hard to solve the real-sized problem with the current ADP-Bézier-Surface model. We can see from Table 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 $|\Gamma|$ (in real-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 real-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. Numerical results (tested on 100 random scenarios) are shown in Table 7 and 8.

Ind.	SRP	TMRP	ADPRP_BS	DRP
Ins.2_4320	136012.25±27.86	135812.07±29.09	133550.22±34.52	132029.70±39.71
Ins.3_4320	133900.08±38.82	133075.17±42.18	130901.60±41.23	127857.37±46.14
Ins.4_4320	116477.31±32.59	116228.23±33.36	114042.22±34.89	112820.29±24.56
Ins.5_4320	123733.71±28.14	122892.62±33.26	120994.18±32.10	119768.49±24.52

Table 7: Comparison of mean cost (± s.d.) of ADPRP_BS on 4320-period instances

Ind.	K	R	βs	Iterations	Time	T/I
Ins.2_4320	4	6	52	300	2036.554s	6.7885s
Ins.3_4320	4	6	52	300	1527.347s	5.0912s
Ins.4_4320	4	6	52	350	3801.735s	10.8621s
Ins.5_4320	5	6	65	400	3741.024s	9.3526

Table 8: Comparison of running time of ADPRP_BS on 4320-period instances

With the ADP-Bézier-Surface model and a simple decision aggregation step, real-sized TpTRP instances are solvable, providing very good routing policy for all four instances with different distributions. Due to the small number of control points we used in the ADP-Bézier-Surface model, it can be trained after several hundred of iterations within around 1 hour, while calculating the optimal routing policy from the trained model for a given set of observed traffic required $5 - 10$ 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. Conclusions and Future Works

In this work, we achieved to modify the original discrete ADP model for the TpTRP by aggregating regression coefficients β over both time interval τ and index of traffic order j with 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 real-size problem (which possesses 4320 time intervals, network providers charge the ISP based on the 216th highest volume of shipped traffic), we developed a way to aggregate decision problems, thus accelerate the speed of going through every single iteration and make it solvable with the ADP-Bézier-Surface model.

References

[1] 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.

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

[3] 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).

[4] 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.

[5] A. Grothey and X. Yang, *Top-percentile traffic routing problem by dynamic programming*, Technical Report ERGO-09-006, School of Mathematics, University of Edinburgh, Edinburgh EH9 3JZ, Scotland, UK, March 2009.
[6] A. Grothey and X. Yang, *Solving the top-percentile traffic routing problem by approximate dynamic programming*, Technical Report ERGO-10-003, School of Mathematics, University of Edinburgh, Edinburgh EH9 3JZ, Scotland, UK, Feb 2010.
[7] 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.
[8] A. Odlyzko, *Internet pricing and the history of communications*, Computer Networks, 36 (2001), pp. 493–517.
[9] W. Powell, *Approximate Dynamic Programming - Solving the Curses of Dimensionality*, John Wiley & Suns, New Jersey, 2007.