SOLVING THE TOP-PERCENTILE TRAFFIC ROUTING PROBLEM BY APPROXIMATE DYNAMIC PROGRAMMING

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). The TpTRP is a multi-stage stochastic optimisation problem in which routing decision should be made before knowing the amount of traffic that is to be shipped in the following time period. The stochastic nature of the problem forms the critical difficulty of the problem.

Solution approaches based on Stochastic Integer Programming (SIP) techniques or Stochastic Dynamic Programming (SDP) suffer from the curse of dimensionality which restricts their applicability. To overcome this we suggest to use Approximate Dynamic Programming (ADP) which exploit the structure of the problem to construct approximations of the value function in SDP. Thus the curse of dimensionality is largely avoided.

Keywords: top-percentile pricing, multi-homing, stochastic, routing policy, approximate dynamic programming

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 [6]. 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 [6]) what the optimal multi-homing routing strategies looks 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 traffics in advance) has been analysed in [2], where the authors build a mixed-integer linear programming model to evaluate if multi-homing is economically viable and develop an efficient B&B algorithm to solve it with combined top-percentile pricing and fixed cost. In the stochastic case, Levy et al. in [5] develops a probabilistic model and provides an analysis of the expected costs, thus demonstrate that multi-homing can be economical efficient under top-percentile pricing though they did not give the optimal routing policy. On the other hand, Goldenberg et al. [3] focus on the development of smart routing algorithms for optimising both cost and performance for multi-homing users under top-percentile pricing, but not in the stochastic case. To the best of our knowledge however, there is no result dealing with the optimal multi-homing routing policy under top-percentile pricing in the stochastic case.

The purpose of this 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). Under pure top-percentile pricing, 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. 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 anticipate 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 [4], 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 last time stage. 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 any naive routing policy and whose mean cost is close to the lower bound given by the deterministic case for medium sized instances. However due to the curse of dimensionality derived from the discretization in dynamic programming, the huge number of states prevents the use of the SDP model on larger problem instances.

It has been suggested in [7] that Approximate Dynamic Programming (ADP) is a promising technique to avoid the curse of dimensionality. The focus of this work is on the application of ADP to the TpTRP. In the remainder of this report, we present the decision space and justify its appropriateness in Section 2. In Section 3, we introduce the curses of dimensionality involved in our SDP model and show how to deal with them with ADP techniques. Then we give details of our implementation of ADP on the TpTRP problem in Section 4. Section 5 gives the numerical results and shows how the aggregation with ADP can be used to solve real-world sized instances. Finally we make conclusions in Section 6.

2. Top-percentile Traffic Routing Problem

Instead of building the ADP model directly, in this section we would like to investigate the important features of the TpTRP problem first. This section gives a formal description of the TpTRP model and highlights a few of its key prospectives which will be useful in defining our set of considered decision rules.

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.

We 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. All network providers divide the charge period into the same $|\Gamma|$ time intervals of equal length and use top-percentile pricing with a same q. At the end of the charge period, cost charged by provider i is $Cost_i = c_i y_i$, if y_i is the θ-th highest volume of traffic shipped to network provider i.

- $T^\tau, \tau \in \Gamma$: 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.

Decision variables.

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

Note that x^τ should be made and implemented before knowing the whole value of the random traffic T^τ (see Section 2.3 for detail).

2.2. **State variable 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, 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 these redundant information from the state space, which makes the state variable at τ described by

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

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.

2.3. **Implementable Routing Policies.** As mentioned above, in our TpTRP problem traffic T^τ is a random variable, of which the distribution is known beforehand. In reality, the traffic is revealed continuously over the time period. This means, as shown in Figure 2.1, we cannot see the complete amount of $\hat{T}^\tau$ before the end of time interval τ.

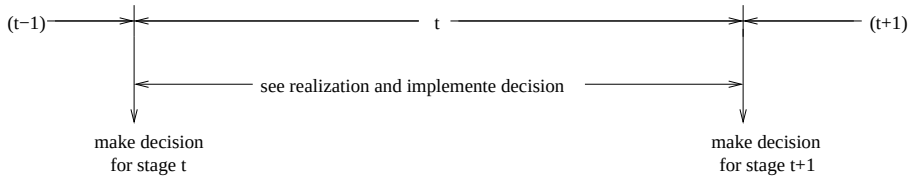


Figure 2.1: Process of data revelation and implementation of decisions

However, any bit 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 the whole volume $\hat{T}^\tau$, another necessary condition for a feasible routing decision is that it is implementable without knowing $\hat{T}^\tau$. In the simplest case, we can decide at the start of every time period where to send the whole traffic for this period. However, since the revelation of traffic is gradual, more sophisticated routing policies can be considered. We are, of course, limited by what is technically implementable. In particular, we assume that percentage based routing policies (i.e., $x^\tau = (x_1^\tau, x_2^\tau, ..., x_n^\tau)^T, \sum_{i\in I} x_i^\tau = 1$, 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., $x^\tau = (y_1^\tau, y_2^\tau, ..., y_n^\tau)^T$, where the first y_1^τ unit of traffic are sent to provider 1, the next y_2^τ to provider 2 and so forth) or a combination of these are implementable by operating a 'time slicing' scheme.

In this work we consider a particular mixed routing policy. Firstly, we set up a cut-off y_i^τ for network provider i, any remaining traffic $\hat{T}^\tau - \sum_{i\in I} y_i^\tau$ (if there is) is routed according to the proportional decision x_i^τ. Thus our feasible decision set is

$$F = \{(x_i^\tau, y_i^\tau) \geq 0 | i = 1, ..., n; \sum_{i\in I} x_i^\tau = 1\}.$$

Note that any decision $(x^\tau, y^\tau) \in F$ gives an implementable routing decision. When implementing it, we allocate the random traffic T^τ according to the following rule:

- If $\sum_{i=1}^{\tilde{i}} y_i^\tau \leq \hat{T}^\tau < \sum_{i=1}^{\tilde{i}+1} y_i^\tau$ for some $\tilde{i} \in I$, we send:
  - ${}^{new}T_i^\tau = y_i^\tau$ to network provider $1 \leq i \leq \tilde{i}$,
  - ${}^{new}T_i^\tau = \hat{T}^\tau - \sum_{i=1}^{\tilde{i}} y_i^\tau$ to network provider $\tilde{i}+1$,

- $^{new}T_i^\tau = 0$ to network provider $i > \tilde{i} + 1$.

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

In the following section we investigate the proper choice of cut-off level y^τ, which gives some idea about the optimal routing policy.

2.4. **Revised decision space.**

Lemma 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 $^1S^\tau \leq {}^2S^\tau$, 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 charged on the ISP 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 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 for state $^2S^\tau = \{^2\hat{T}_i^{j,\tau}\}$. According to the implementation rule given in Section 2.3, the amount of traffic $^{new}T_i^\tau$ sent to network provider i does not depend on the current state. This means if we apply the same decision set $(^2\hat{x}^\tau, ^2\hat{y}^\tau)$ on an arbitrary state $^1S^\tau \leq {}^2S^\tau$, every network provider gets the same amount of $^{new}T_i^\tau$ as when we were on 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 $^2S^{\tau+1} = S^{\tau+1}(^2S^\tau; \hat{\omega}^\tau; ^2\hat{x}^\tau, ^2\hat{y}^\tau)$ on all entries. From the induction we have $V_{\tau+1}(^1\tilde{S}^{\tau+1}) \leq V_{\tau+1}(^2S^{\tau+1})$. Take 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 on state $^1S^\tau$, which means the best function value $V_\tau(^1S^\tau) \leq \tilde{V}_\tau(^1S^\tau)$. Combine these two inequalities together, we have proved that $V_\tau(^1S^\tau) \leq V_\tau(^2S^\tau)$ holds for $\forall \tau \in \Gamma$. □

From Lemma 2.1 we can see, value function $V_\tau(S^\tau)$ is non-decreasing with every entry of the state S^τ. Notifying this, when we make routing decision at every time interval we hope the increase on state components can be as small as possible, namely minimise the difference between $S^{\tau+1}$ and S^τ. Assume this minimisation can be represented by $|S^{\tau+1} - S^\tau|_1 = \sum_{i \in I} \sum_{1 \leq j \leq \theta} (\hat{T}_i^{j,\tau+1} - \hat{T}_i^{j,\tau})$, then for network provider i we have:

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

Thus,

$$|S^{\tau+1} - S^\tau|_1 = \sum_{i \in I} (|S^{\tau+1} - S^\tau|_1)_i = \sum_{i \in I} \max\{{}^{new}T_i^\tau - \hat{T}_i^{\theta,\tau}, 0\}$$

$$\geq \max\{\sum_{i\in I} {}^{new}T_i^\tau - \hat{T}_i^{\theta,\tau}, 0\} \qquad (2.1)$$

$$= \max\{\hat{T}^\tau - \sum_{i\in I} \hat{T}_i^{\theta,\tau}, 0\}.$$

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. According to the inequality (2.1), $T_{Add}(S^\tau)$ is the lower bound of $|S^{\tau+1} - S^\tau|_1$.

Lemma 2.2. *Assume we are on state $S^\tau = \{\hat{T}_i^{j,\tau} | i = 1, ..., n; j = 1, ..., \theta\}$ at time stage $\tau \in \Gamma$. In the optimal routing policy which minimising $\mathbb{E}_{\omega^\tau}[|S^{\tau+1} - S^\tau|_1]$, we have $y_i^\tau = \hat{T}_i^{\theta,\tau}, \forall i \in I$.*

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

- Assume $\exists i_0 \in I$, $\hat{y}_{i_0}^\tau < \hat{T}_{i_0}^{\theta,\tau}$. Then if we get a new traffic $\hat{T}^\tau$ which 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 satisfies
  - ${}^{new}T_{i_0}^\tau = \hat{y}_{i_0}^\tau + \hat{x}_{i_0}^\tau(\hat{T}^\tau - \hat{y}_{i_0}^\tau - \sum_{i\neq i_0} \hat{T}_i^{\theta,\tau})$;
  - ${}^{new}T_i^\tau = \hat{T}_i^{\theta,\tau} + \hat{x}_i^\tau(\hat{T}^\tau - \hat{y}_{i_0}^\tau - \sum_{i\neq i_0} \hat{T}_i^{\theta,\tau}), \forall i \neq i_0$.

  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

$$\sum_{i\neq i_0} {}^{new}T_i^\tau = \hat{T}^\tau - {}^{new}T_{i_0}^\tau > \hat{T}^\tau - \hat{T}_{i_0}^{\theta,\tau}.$$

  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 $\exists i_0 \in I$, $\hat{y}_{i_0}^\tau > \hat{T}_{i_0}^{\theta,\tau}$. Then if we get a new traffic $\hat{T}^\tau$ which 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 satisfies
  - ${}^{new}T_i^\tau = \hat{T}_i^{\theta,\tau}, i = 1, ..., i_0 - 1$;
  - ${}^{new}T_{i_0}^\tau = \hat{T}^\tau - \sum_{i=1}^{i_0-1} \hat{T}_i^{\theta,\tau}$;
  - ${}^{new}T_i^\tau = 0, i = i_0 + 1, ..., n$.

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

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

As for all scenario we cannot do better than $T_{Add}(S^\tau)$, thus taking the expectation we can prove $y_i^\tau = \hat{T}_i^{\theta,\tau}, \forall i \in I$ is optimal.

□

In Lemma 2.2 we have proved $y_i^\tau = \hat{T}_i^{\theta,\tau}, \forall i \in I$ in the optimal decision. This means in this work, our decision should be made on the additional traffic only. Thus the feasible decision set is

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

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.

3. Introduction to Approximate Dynamic Programming

3.1. **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 [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 allowing $\hat{T}_i^{j,\tau}$ 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 $C_{L+\theta-1}^\theta = \begin{pmatrix} L+\theta-1 \\ \theta \end{pmatrix}$ [1] different states S_i^τ for provider i and a total of $\begin{pmatrix} L+\theta-1 \\ \theta \end{pmatrix}^n$ 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 L, θ and n is referred to in [7] as the 'first curse of dimensionality' – the dimensionality in state space.

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

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

[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 $\hat{T}_i^{1,\tau} \geq \hat{T}_i^{2,\tau} \geq ... \geq \hat{T}_i^{\theta,\tau}$.

Step forward in time. Another important difference is that ADP is based on an algorithmic strategy that steps forward through time, rather than backward in SDP. In ADP we choose a sample scenario and step forward in time. At each time step τ, we solve the decision problem based on the current estimation of the value function approximation $\bar{V}_{\tau+1}^{(m-1)}$ at time interval τ for the given sample scenario, and then update $\bar{V}_{\tau}^{(m-1)}$ with the optimal sample value $\hat{v}_{\tau}^{(m)}$. By repeating these steps the process converges the parameters to a stable estimation of the value function. 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 continuous regression model representation of value function approximation, 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 the dynamic programming, in which by each computation we get the value for a single, discrete state variable. In ADP it focuses more on the states which are more likely to be visited, rather than treat all the possible states as equally important.

3.3. **Main procedure of ADP.** A basic approximate dynamic programming algorithm is summarised below: [7]

Step 0. Initialisation:

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

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

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 $S_{(m)}^{\tau+1}(S_{(m)}^{\tau}, \omega_{(m)}^{\tau}, \hat{x}^{\tau})$, 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.

---

4. ADP model

4.1. **Value function approximation – regression model.** As discussed in Section 3.2, the traditional look-up-table representation of value function suffers from the first curse of dimensionality. To estimate the value function with as few parameters as possible, in ADP we use regression model to approximate the value function. To get a good fit to the real value function, it requires us to exploit the special structure of it. Thus before the definition of a proper regression model let us go through several examples of value functions given by the SDP model [4], to see what structure we can take to use.

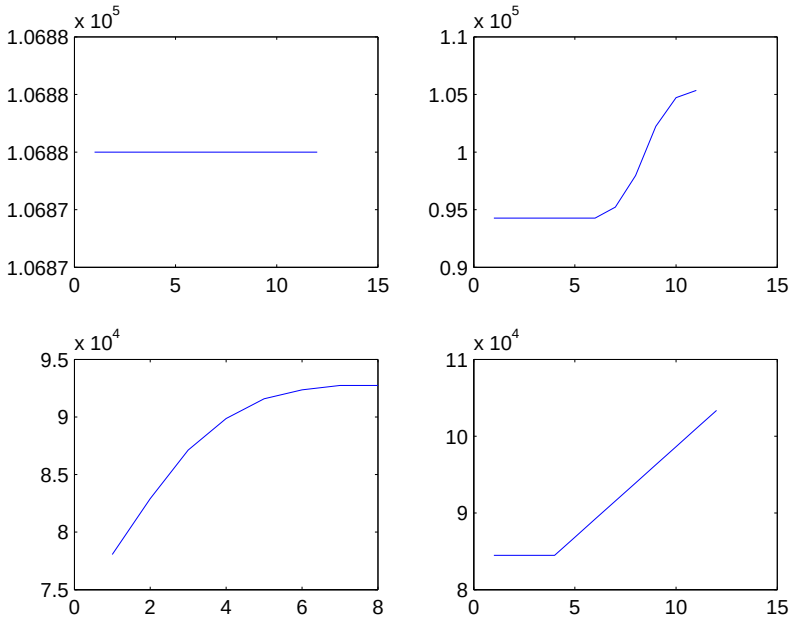


Figure 4.1: Examples of how function value changes with a single entry of state variable

Figure 4.1 shows four examples of how the function value varies with a single entry of the state variable. Although the value of $V_\tau(S^\tau)$ shown in these figures might not be exact since SDP itself is an approximation of the problem (with discretization of state space and restrictions on the decision space), we can still get some insight of the basic character of value functions. From these three specific examples we can see that the value function is neither a convex or a concave function of its variables, sometimes it is not even smooth. Apart from this, actually some states (e.g. very extreme ones) in SDP model are much less important than others as they are rarely visited. This requires us to focus more on the centre part of value functions.

In Lemma 2.1 we have proved that the function value is non-decreasing with every entry of the state variable. In this work we use the simplest model, linear regression to approximate the value function for every time period. Namely at state S^τ, we estimate $V_\tau(S^\tau)$ by:

$$\bar{V}_\tau(S^\tau) = \beta_0^\tau + \sum_{i\in I} \sum_{1\le j\le \theta} \beta_{i,j}^\tau \hat{T}_i^{j,\tau} \quad (4.1).$$

Although a linear model is certainly not exact, we feel that it will give the best trade-off between providing good approximation and maintaining computability and avoiding the risk of over-fitting. It also provides robustness against possible spurious behaviour (locally decreasing approximations) of more complex value

function approximations. We feel this choice is justified by the numerical results in Section 5.2.

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

$$\hat{v}_\tau^{(m)} = \min_{x^\tau \in \chi^\tau} (\mathbb{E}_{\omega^\tau \in \Omega^\tau} \bar{V}_{\tau+1}^{(m-1)}(S^{\tau+1}|S_{(m)}^\tau, \omega^\tau, x^\tau)) \quad (4.2),$$

where $\bar{V}_{\tau+1}^{(m-1)}$ as given by (4.1) is the approximation of the value function $V_{\tau+1}$ build with the estimated coefficients after $m-1$ iterations. Decision problem (4.2) is a minimisation problem, whose objective is an expectation of the value function estimation in next time stage.

It is worth investigating the exact form of this objective function. As given in (4.1), $\bar{V}_{\tau+1}$ 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 $\bar{V}_{\tau,i}(S_i^\tau) = \sum_{1 \le j \le \theta} \beta_{i,j}^\tau \hat{T}_i^{j,\tau}$, 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 rules given in Section 2.3 to obtain the new traffic for network provider i, ${}^{new}\hat{T}_i^\tau$ 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

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

Thus the objective function of the decision problem becomes:

$$\mathbb{E}_{\omega^\tau}(\bar{V}_{\tau+1}^{m-1}(S^{\tau+1})) = \int_{\omega^\tau} f(\omega^\tau) \beta_0^\tau \, d\omega^\tau + \sum_{i \in I} \int_{\omega^\tau} f(\omega^\tau) \bar{V}_{\tau+1,i}^{m-1}(S^{\tau+1}) \, d\omega^\tau \quad (4.3),$$

which is difficult to simplify further due to the complex form of $\bar{V}_{\tau+1,i}(S^{\tau+1})$.

Actually, numerically examining some instances we observed that the objective function of the decision making problem (4.2) might not be convex, even for the linear regression model (4.1). Therefore an algorithm based on cutting planes as suggested by [7] cannot be used. In addition, since the objective is given by an

expectation (4.3) which we are unable to evaluate analytically, any function or gradient evaluations are expensive and inexact.

In ADP, as we need to solve the decision problem at every time stage for every iteration, an optimisation method that is efficient (solve the problem in reasonable time) in addition to reliable (find the optimal or near optimal solution) is required. We have settled to solving this problem by a simple discretization of the decision space, i.e., generating several discrete decisions (for example $x^\tau = 0.0, 0.1, 0.2...1.0$), calculating their objective value and choosing the best one. Although with the simple discretization way we cannot find the optimal solution of the decision problem, it gives a good compromise between speed and accuracy. We can see from the numerical results given in Section 5.2 that the practical advantage from solving the decision problem more accurately is minimal.

4.3. **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 estimation $\bar{V}_\tau^{(m-1)}$. As in this work we use the linear regression model to estimate the value function, our value function approximation after m iterations can be written as $\bar{V}_\tau^{(m)} = \bar{V}_\tau(\beta^{(m)})$. To update parameters for the regression model, we want to find $\beta^{(m)}$ that solves:

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

where $\hat{v}_\tau$ is the sample estimate of the real function value $V_\tau(S^\tau)$ obtained by solving (4.2). Applying stochastic gradient algorithm, we obtain the updating scheme [7]:

$$\beta^{(m)} = \beta^{(m-1)} - \alpha_{m-1}[\bar{V}_\tau^{(m-1)}(S^\tau) - \hat{v}_\tau^{(m)}]\nabla_{\beta^{(m-1)}}\bar{V}_\tau^{(m-1)}(S^\tau) \quad (4.4).$$

The value α_m in formula (4.4) is the updating stepsize from iteration m to $m+1$, which tells us how far we should go in the direction of $\nabla_{\beta^{(m-1)}}\bar{V}_\tau^{(m-1)}(S^\tau)$. Finding the proper stepsize α_m is one of the challenges in stochastic gradient methods. Poor choice of stepsize may cause the provably convergent algorithm not to work. However, in most applications of stochastic problem it is impossible to find an optimal stepsize due to the intractable calculation of expectations. Thus for the seek of easy in implementing, in this work we use one of the typical deterministic stepsize – McClain's formula:

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

where $\bar{\alpha}$ is a specified parameter. Steps generated by this formula satisfy $\alpha_m^{MC} > \alpha_{m+1}^{MC} > \bar{\alpha}$. McClain's rule combines the features of the $1/n$ rule which is ideal for stationary data (when values to estimate are mainly decreasing) and constant stepsizes for non-stationary data (when noise in the observations is dominating). Moreover, as in the limit $\alpha_m^{MC} \to \bar{\alpha}$, the stepsize avoids going to zero. This makes the rule work well in non-stationary environments, and also effective when we are not sure how many iterations are required to start converging.

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 (4.3). Since the value of $\beta^{(m-1)}$ and $[\bar{V}_\tau^{(m-1)}(S^\tau) - \hat{v}_\tau^{(m)}]\nabla_{\beta^{(m-1)}}\bar{V}_\tau^{(m-1)}(S^\tau)$ may possess completely different scale, we need an adaptively chosen α_0 to cover this difference. Thus our stepsize consists of two components, which means $\alpha_m = \alpha_0\alpha_m^{MC}$. As we expect the $\beta^{(m)}$ to move monotonically at the beginning of the algorithm and start alterating near convergence, we will increase α_0 if we observe monotonic behaviour in the $\beta^{(m)}$ for the first few iterations and decrease otherwise.

4.4. **Stopping criterion.** As in ADP we update the value function estimation iteratively, 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 algorithm typically converge rapidly at the beginning and then vibrate with noise. As in our problem, what we are seeking for is whether a routing policy gives us a mean costs that is low enough in a long run, instead of the exact expression of the regression model (4.1). Therefore in our ADP model, we numerically evaluate the mean cost over every $10,000$ runs and once we observe the mean cost changes mainly with noise instead of decreasing/increasing rapidly, we stop and treat the current coefficients as the converged parameters for (4.1). For more detail please see Section 5.2.2.

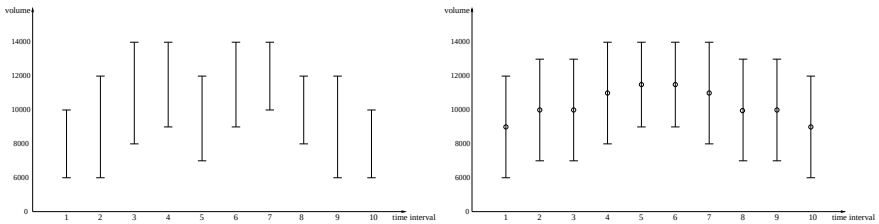
5. Numerical Results

5.1. **Test Problems with 10 periods.** In this section we give some numerical results on several small instances of the TpTRP taken from [4]. For clarity, we firstly characterise and index these instances which are examined in the later part of this section.

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

Table 1: List of TpTRP Instances

Table 1 summarises the instances used. In all instances, we assume that we divide the modelling region into 10 time intervals and cost are based on the time interval with the $\theta = \lfloor q * |\Gamma| \rfloor = 3$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. 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 ($N(10000, 10^6)$ 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 each time interval are displayed in Figures 5.1. Note that Instance 4 and 5 uses 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.



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

(b) Mean and 99.7% confidence region for normal distributions in Ins.5

Figure 5.1: Traffic distribution used in testing instances

5.2. **Numerical results on TpTRP instances with 10 periods.**

5.2.1. *Mean cost.* To evaluate the quality of this routing policy we get 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 above. We compare results with 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 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;
- SDPRP - Stochastic Dynamic Programming Routing Policy given as a discrete look-up table by solving the SDP model in [4], which requires discretization of the traffic region. We repeat the model with different number of discretization levels (L in Table 2) used;
- DRP - Deterministic Routing Policy, i.e. assuming we know all traffics in advance. The optimal routing policy (as proved in [4]) 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.

Results with different cost ratios c_2/c_1 are summarised in Table 2. We can see that the ADP routing policy outperforms trivial routing policies and works better than the SDP routing policy in most cases for coarse discretizations (e.g. $L = 7$ and 14). Sometimes, the ADP routing policy can be even better than SDP with $L = 28$, which is the finest model for which SDP is tractable (due to the 'curse of dimensionality' in state space). 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 approximate 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, forward dynamic programming focuses attention on the states that we actually visit. As in normal distribution instances the traffic is more clustered around the mean, we can get better coefficients for those more likely happen states. We think this is why in

Ins.	SRP	TMRP	L	SDPRP	ADPRP	DRP
Ins.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		
Ins.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		
Ins.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		
Ins.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		

(a) $c_2 = 11$

Ins.	SRP	TMRP	L	SDPRP	ADPRP	DRP
Ins.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		
Ins.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		
Ins.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		
Ins.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		

(b) $c_2 = 12$

Ins.	SRP	TMRP	L	SDPRP	ADPRP	DRP
Ins.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		
Ins.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		
Ins.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		
Ins.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		

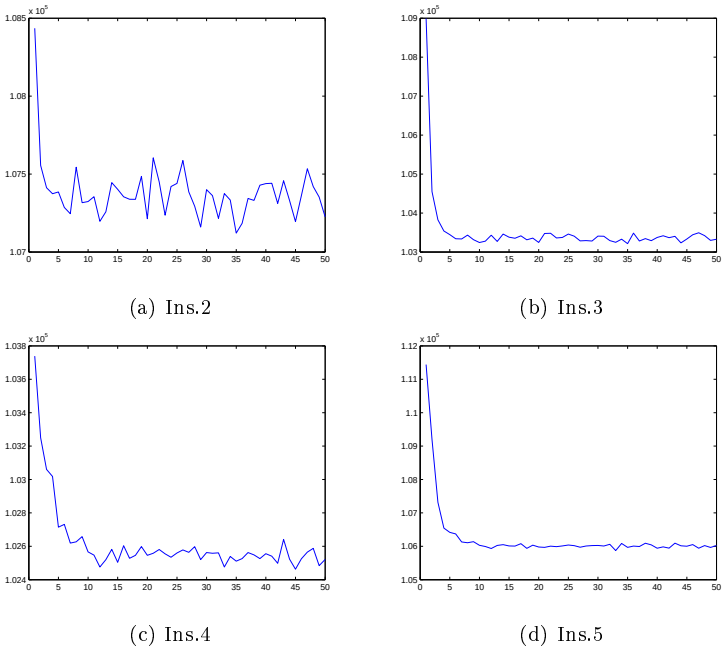
(c) $c_2 = 15$

Table 2: Numerical result (mean cost ± s.d.) of implementing routing policies on $1,000,000$ random scenarios

case where the traffic volumes follow the normal distribution (i.e. Ins.4 and Ins.5), the ADP routing policy seems performing better than where uniform distribution (Ins.2 and Ins.3) is applied. In conclusion, the ADP model gives very promising results with the linear regression model.

5.2.2. *Convergence and resource consumption.* Apart from performance, another important practical issue is running time of a model, specifically the convergence time in the ADP model. As notified in Section 4.4, 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.

Figure 5.2 shows how the mean cost varies with time (x-axis represents the number of $10,000$ 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



(a) Ins.2 (b) Ins.3

(c) Ins.4 (d) Ins.5

Figure 5.2: Mean cost over 10,000 samples varies with number of iterations – convergence

then. The resulting number of iterations until convergence was determined are given in Table 3.

In addition to the running time, the 'curse of dimensionality' in the SDP model also manifests itself in high memory use. In Table 3 we summarise the running time and the memory consumption (theoretical) in the solution of the ADP and SDP model.

Ins.	ADP model			SDP model		
	Iterations	Time	Memory	L	Time	Memory
Ins.2	80,000	9.391s	5.34e-4MB	7	0.194s	0.38MB
				14	15.628s	16.75MB
				28	11487.095s	880.32MB
Ins.3	70,000	7.481s	5.34e-4MB	7	0.126s	0.38MB
				14	12.486s	16.75MB
				28	9732.705s	880.32MB
Ins.4	130,000	170.824s	5.34e-4MB	7	0.316s	0.38MB
				14	27.074s	16.75MB
				28	21185.037s	880.32MB
Ins.5	120,000	130.022s	5.34e-4MB	7	0.390s	0.38MB
				14	27.211s	16.75MB
				28	22643.042s	880.32MB

Table 3: Comparison of problem size and resource consumption

In Table 3, the first column shows the number of iterations needed to see the convergence in their own ADP model for all instances, while the second column shows the running time these iterations consumes. From this table we can see that in Ins.2 and Ins.3 the solution times of ADP model are comparable with the 14-level SDP model, while in Ins.4 and Ins.5 it seems the latter runs quicker. 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, thus the computer memory it consumes is constant for a predetermined instance. Also, as we are working on continuous state space in ADP, there is no need to record all decisions explicitly at every node in the dynamic tree. What we need to keep are only the coefficients according to state variables, decision and value function are all implicit in these coefficients. In fact the computer memory which ADP model consumes is 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.

5.2.3. *Solving the decision problem to higher accuracy.* As stated above, the decision problem 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 4 we investigate the effect of solving the decision problem to a higher accuracy, by choosing from decisions $\{0.00, 0.01, 0.02, ...1.00\}$.

Ins.	ADPRP_0.1		ADPRP_0.01	
	Mean Cost	Running Time	Mean Cost	Running Time
Ins.2	107335.60±12.35	9.391s	107335.43±12.35	84.473s
Ins.3	103361.52±7.50	7.481s	103361.51±7.50	68.245s
Ins.4	102561.69±4.31	170.824s	102561.69±4.31	1350.809s
Ins.5	105418.83±4.78	130.022s	105417.34±4.78	982.480s

Table 4: 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. **Solving real-world 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 real 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. Applying ADP to such a model would result in updating the parameters β_i^τ for one particular $V_\tau(S^\tau)$ 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 5/6 give running time and performance for this approach on a 4320-time period model with traffic distribution according to Table 1.

Ins.	Iterations	Running Time
Ins.2	200,000	99.674s
Ins.3	400,000	179.029s
Ins.4	1,000,000	5281.945s
Ins.5	1,000,000	5468.437s

Table 5: Resource consumption of solving the 43-periods ADP model

Ins.	SRP	TMRP	ADPRP	DRP
Ins.2	136000.08±2.63	135789.57±2.75	134335.36±3.72	132008.21±2.69
Ins.3	133874.68±1.90	133022.12±2.42	129956.59±3.15	127791.87±2.10
Ins.4	116466.16±3.28	116216.75±3.33	114318.44±4.23	112840.00±2.60
Ins.5	124737.71±3.75	123666.23±3.87	121618.19±4.24	120235.63±3.03

Table 6: Numerical result (mean cost ± s.d.) of implementing 43-periods ADP routing policy on real 4320-periods instance over $1,000$ scenarios, $c_2 = 12$

From the numerical results we can see that the combined ADP-time aggregation method work well on a 4320-period problem, consistently outperforming the trivial routing policy in all instances.

6. Conclusions and Future Work

6.1. **Conclusions.** In this work we have developed an ADP model to solve the TpTRP problem. Rather than using the discrete look-up table representation of value function in SDP, in ADP we approximate the value function by a proper regression model and train its coefficients iteratively with fresh sample scenarios to get the final estimation. As all works are done in a continuous state space, ADP overcomes the curse of dimensionality we met in the SDP model which prevented larger instances (more than 10-periods and 2 network providers and $q = 5\%$) 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 outer-performs the SDP routing policy with 28-levels, while the running time is much smaller. By combining ADP with time aggregation we can solve real sized instances with thousands of time periods in a reasonable time. The routing policies obtained consistently outperform naive routing policies on real sized problem instance.

6.2. **Future works.** Currently we work on traffic routing problems under pure top-percentile pricing policy, where if we send no more than $\theta - 1$ traffics to a provider, we are not going to pay anything to this provider. However in practise, network provider might combine top-percentile pricing with other pricing policies such as an additional start up cost. What we need to do is examine whether ADP model with linear regression value function approximation can manage these changes as well. If not try to find proper regression models for this problem.

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] 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.

[4] 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.

[5] 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.

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

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