Open Shortest Path First Routing Under Random Early Detection

Jiaxin Liu[*] Stanko Dimitrov[†]

November 13, 2017

Abstract

In this paper we consider a variant of Open Shortest Path First (OSPF) routing that accounts for Random Early Detection (RED), an Active Queue Management method for backbone networks. In the version of OSPF we consider in this paper we only require a single network path be available between each origin and destination, a simplification of the OSPF protocol. We formulate a mixed integer non-linear program to determine the data paths, referred to as a routing policy. We prove that determining an optimal OSPF routing policy that accounts for RED is NP-Hard. Furthermore, in order for the generated routing policies to be real-world implementable, referred to as realizable, we must determine weights for all arcs in the network such that solving the all-pairs shortest path problem using these weights reproduces the routing policies. We show that determining if a set of all-pairs routes is realizable is also NP-Hard. Fortunately, using traffic data from three real-world backbone networks, we are able to find realizable routing policies for these networks that account for RED, using an off-the-shelf solver, and policies found perform better than those used in each network at the time the data was collected.

Keywords: OSPF, RED, all-pairs routing, complexity, application, back-

[*]j69liu@uwaterloo.ca. Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON N2L 3G1

[†]sdimitro@uwaterloo.ca. Department of Management Sciences, University of Waterloo, Waterloo, ON N2L 3G1; contact author

bone networks

1 Introduction

Today's world is witnessing the fast growth of the Internet while embracing the benefits it brings. Over the past decade, Internet usage has increased by nearly 40% year by year [5], and according to an estimate by Cisco [6], this upward trend is likely to continue over the next few years at a consistent pace. As much as people enjoy the convenience offered by the vast development of the Internet, they have to occasionally contend with slow speeds, especially when networks fail to accommodate large demands, resulting in network congestion [26, 36]. To resolve the slow speeds resulting from congestion, network service providers may use network provisioning, usually associated with hardware upgrades. However, network provisioning may require investment in network infrastructure, and this process could be very costly. Alternatively, network operators may reconfigure the protocols used by the existing infrastructure to obtain a better traffic distribution, thereby reducing the degree of congestion without further network investment. In this paper, we show one method that can indeed deliver more traffic in a network simply by reconfiguring existing network protocols. Specifically, we discuss a mathematical model that maximizes the amount of traffic delivered in a network under a congestion control mechanism while accounting for the behavior of standard network protocols for backbone networks. In our research the backbone networks we found tend to be small in size, approximately fifteen routers and each router is connected to approximately two to three other routers, and there is traffic between each pair of routers in the network.

We model a computer backbone network as a simple graph that contains nodes to represent routers and arcs to represent links. Based on this graphical model, we develop a mathematical optimization program, inspired by the classical multi-commodity flow problem [24], to model the network traffic flow process. This model finds routing policies, i.e., the paths that the data must take from its source to destination, that maximize the total weighted data received at all destinations, while accounting for Random Early Detection (RED), to deal with congestion. RED is a type of Active Queue

Management protocol that is implemented on each router. A router with RED enabled increasingly drops incoming data as the flow of data across that router increases. In conjunction with other network protocols, RED is able to control the data transmission rate and hence reduces network congestion.

Our model determines the routing policies that take into account the data dropping mechanism of RED, and indeed, our model minimizes data loss from such dropping mechanisms by maximizing the total data delivered. We show that the developed model is computationally intractable when determining simplified OSPF routing policies. However, using an off-the-shelf solver we can obtain good solutions to three *real-world* backbone network instances [1]. The results of our experiments show that our generated policies perform better than the policies used by the real-world networks. We also develop a revised formulation that provides robust routing policies that account for different demand realizations, and the results of the corresponding experiments show that the generated robust routing policies outperform the policies used in real-world networks. In addition, we examine the realizability problem for generated routing policies; that is, we would like to determine whether routing policies generated by our mathematical programs can be configured using standard routing protocols, so that these policies are real-world applicable. In particular, we consider the Open Shortest Path First (OSPF) routing protocol [35], where OSPF requires paths be the same as those found by solving an all-pairs shortest path problem, or simply put, the paths must be the shortest with respect to a set of pre-determined arc weights. We show that we can find OSPF arc weights for the corresponding network graph for each of our instances in the experiments even though, in general, this problem is computationally intractable, one thing that is also shown in this paper.

To summarize the contributions of this work are as follows:

- Define the mathematical model to find the all-pairs routing policies that account for RED.
- Prove that the model above is NP-Hard.
- Formulate the all-pairs inverse shortest path problem, used to determine OSPF arc weights that realize a prescribed set of paths as shortest paths.

- Prove that the all-pairs inverse shortest path problem is NP-Hard.
- Show, using the models above, we can generate realizable all-pairs OSPF routing policies for three real-world networks, and the ones found are better than those currently used.

This paper is organized as follows. Section 2 introduces the background and context of our research, as well as the related work. Section 3 formulates the mathematical model for network traffic flow under RED and shows the properties associated with the model. Proofs of the two main theorems in Section 3 are omitted due to length (47 pages total) but short proof outlines are provided; we refer interested readers to the work of Liu [34, Chapters 5 and 6] for the complete proofs. Sections 4 and 5 discuss the practical implications of the model in Section 3, through a series of practical and algorithmic considerations and experiments. Finally, Section 6 concludes the paper and discusses future research directions.

2 Background

In this section, we introduce the background of our work, focusing on the structure of the network protocol stack and review related work in intra-domain traffic engineering and Random Early Detection. In general, configurations of routing protocols tend to be stable. For example, according to Cisco's suggestion [4], OSPF arc weights can be set as the reciprocals of arc capacities. Arc capacities are unlikely to change, unless hardware upgrades are made; therefore, Cisco's standard arc weights are likely to remain constant. This stability may seem reliable in normal situations, but it does not guarantee good network performance, which is important especially as network usage increases. Typical network performance measures include average delay, packet loss percentage, and total throughput. As traffic increases due to increased network usage, it may be preferable for network operators to reconfigure routing protocols so that some performance measure(s) is (are) optimized. This idea motivates work in Traffic Engineering. Traffic Engineering (TE) [8, 9] is the process that provides decisions in the construction and administration of a network by using the measurement, modeling, application, and control of Internet traffic in order to optimize network performance measures. In particular, the intra-domain TE problem deals with intra-domain routing protocols, routing protocols for a set of routers that are administered by one entity. Many researchers studied the intra-domain TE problem and have focused specifically on variants where shortest path routing protocols are used [7, 12, 28]. In order to restrict the model under the context of the shortest path routing protocols, the inverse shortest path problem [19] is incorporated as part of the problem. Even though most of these TE problems are proven to be hard to solve, they are still able to perform well for certain real-world situations, findings that echo those in our paper. Some national research networks have used relevant results from solving their TE problems in designing their routing policies for a few years [13].

Intra-domain TE problem under shortest path routing has been studied extensively in the past decade [12, 28]. A survey conducted in 2009 [7] by a group of outstanding researchers in this field has a thorough breakdown of this field. Fortz and Thorup present a formulation based on the multicommodity flow problem [24] and prove the NP-Hardness of the TE problem with Equal Cost Multipath rule [28], and discuss local-search heuristics. Bley focuses on the unique shortest path rule [12], and proves the NP-Hardness of the inverse shortest path problem for the unique shortest path rule as well as the routing problem. The path realizability complexity result we derive is motivated by Bley's proof [12].

Some Internet traffic carries Transmission Control Protocol (TCP) segments, while others carry User Datagram Protocol (UDP). As network utilization increases, congestion may occur and lead to datagrams being dropped from the network. In the case of a drop event TCP exponentially reduces the sending rate in an effort to alleviate network congestion. However, traffic that is not congestion or flow sensitive, such as UDP, will not reduce the rate of sending, potentially leading to a disproportionate amount of UDP traffic being delivered. Further, in the case of a burst of traffic, TCP may be too slow to react [15]. In a response to such situations, Active Queue Management (AQM) was developed and standardized in the early 2000s [15]. One of the earliest AQM disciplines, Random Early Detection (RED), is widely considered and extended during the past two decades [27]. Since its first introduction variations of RED have been proposed, but for simplicity we only consider RED in this paper. RED is defined with respect to an interface queue length, in which the probability that an incoming packet is enqueued into the interface queue decreases proportionally with the length. As discussed in the next section, we model RED similarly to Dimitrov [22].

3 Model

In this section, we formally introduce the main routing problem considered, the OSPF routing problem under RED. We reexamine RED from a mathematical perspective, and develop functions that approximate the RED process. To solve the routing problem, we propose and discuss a mathematical program that incorporates the RED functions. We discuss the inverse shortest path problem used to determine arc metrics based on the derived routing policy from the math program. Finally, we show several properties associated with the models.

3.1 RED functions

In this section we present RED's active queue management algorithm and the associated mathematical functional model. Before we present the mathematical model we make two modeling abstractions for a link connecting two routers, A and B: 1) We assume that there is only a single buffer where queuing may occur between two interfaces that are physically connected to one another; 2) Queue lengths may be approximated as the rate of data arriving on the queue, more on this assumption later in this section. The first assumption is an abstraction of the fact that each interface, outgoing or incoming, has its own buffer on which RED may be implemented. Consider data being sent from router A to router B. For example if A's switching fabric is faster than the line speed connecting A to B, then queuing may occur on the output interface of A. Conversely, if the line speed connecting A to B is faster than the switching fabric of B, then queuing may occur on the input interface of B. To alleviate this we assume that queuing only occurs either at A or B, not both, and we call the buffer where queuing occurs as the line buffer.

In order to implement RED with respect to this single buffer, we need to specify two parameters, β and u, of buffer length, the starting and ending points of the interface queue where incoming packets are dropped probabilistically and all future packets are dropped, respectively. According to RED, all incoming packets will be dropped when the queue length exceeds u, so we refer to u as the effective capacity of the queue. Note that in a traditional

Drop-Tail mechanism, all incoming packets are dropped when the buffer is full, i.e., the queue length reaches the capacity of the buffer, or u in our case. Thus β is the effective starting point of data dropping: if the queue length is less than β then all incoming packets will be successfully enqueued. For any routing to take place in a network the following must hold: $\beta \leq u$. Let t_{in} be the variable representing the current queue length in the buffer. Then the surviving probability of the data packet is 1 if $t_{\text{in}} \leq \beta$ and 0 if $t_{\text{in}} \geq u$. We also highlight that in our research most network operators use the values of u and β that are recommended by the manufacturer, and in this paper we assume that these values are exogenously fixed. A direction for future research would be to determine the values of u and β while achieving some sort of quality-of-service guarantee. We note that if the objective to maximize the total flow received, and all traffic is equally important, then $\beta = u =$ queue capacity and that is why we do not consider this variation in this paper.

When $\beta \leq t_{\text{in}} \leq u$, any incoming data packet is assigned a probability, whose value is determined by the queue length. RED assumes that the relationship between the dropping probability and the queue length is linear. With the notations and assumptions above, we define a function $g : \mathbb{Z}^+ \to [0, 1]$ of queue length, to represent the surviving probability as follows:

$$g(t_{\text{in}}) = \begin{cases} 1 & 0 \leq t_{\text{in}} \leq \beta \\ 1 - \frac{t_{\text{in}} - \beta}{u - \beta} & \beta \leq t_{\text{in}} \leq u \\ 0 & u \leq t_{\text{in}} \end{cases} .$$

In the definition of $g(t_{textin})$ let $\beta = 3$, and $u = 10$. If there are less than 3 packets in the queue, then an incoming packed will be enqueued into the queue. However, if there are say 4 packets in the queue, then an incoming packet will be enqueued with probability 6/7, if there were 5 packets, then an incoming packet will be enqueued with probability 5/7, etc. When there are 10 packets in the queue no incoming packet will be enqueued.

Note that the domain of the surviving probability function is all positive integers, since we are counting the number of packets in the queue. It is much more convenient to extend the domain into all the real numbers, so we have to tweak some concepts here and make our second assumption. As each packet is of variable size, we consider the rate of flow through a queue to determine the dropping probability. Therefore, we redefine $g : \mathbb{R}^+ \to [0, 1]$.

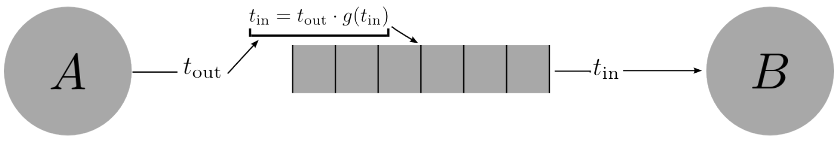


Figure 1: A visual presentation of the relationship between t_{in} and t_{out}.

Note that the function g determines the retained data according to the flow received t_{in} (see Figure 1, i.e., enqueued packets are those that were not dropped and it follows that the probability of a future packet not being dropped is a function of the number of packets that have been successfully enqueued). However, it is more convenient to express the retained data in terms of the flow sent over the arc, t_{out}. We introduce a function to quantify the percentage of data retained over an arc in terms of t_{out}. Consider an arc (A, B). Specifically, suppose we send t_{out} units of traffic flow over (A, B) from A. Note that the buffer at B is not accepting all t_{out} units of flow, due to probabilistic dropping. We define a function $f : \mathbb{R}^+ \rightarrow [0, 1]$ of flow sent from the head node, A, to represent the proportion of flow received at the tail node, B. With this function, we see that $t_{\text{out}} \cdot f(t_{\text{out}})$ represent the amount of flow that is enqueued at node B, i.e., t_{in}. By definition, $g(t_{\text{in}}) = g(t_{\text{out}} \cdot f(t_{\text{out}}))$ is the proportion of flow received at node j, which equals to $f(t_{\text{out}})$. Thus we have:

$$\begin{aligned} f(t_{\text{out}}) &= g(t_{\text{out}} f(t_{\text{out}})) \\ &= \begin{cases} 1 & 0 \leq t_{\text{out}} f(t_{\text{out}}) \leq \beta \\ 1 - \frac{t_{\text{out}} f(t_{\text{out}}) - \beta}{u - \beta} & \beta \leq t_{\text{out}} f(t_{\text{out}}) \leq u \\ 0 & u \leq t_{\text{out}} f(t_{\text{out}}) \end{cases}. \end{aligned} \tag{1}$$

Note that since $f(t_{\text{out}}) \leq 1$, $t_{\text{out}} f(t_{\text{out}}) \leq \beta$ if $t_{\text{out}} \leq \beta$, so $f(t_{\text{out}}) = 1$ for all $0 \leq t_{\text{out}} \leq \beta$. Also when $t_{\text{out}} \geq \beta$, we have$f(t_{\text{out}}) = 1 - \frac{t_{\text{out}} f(t_{\text{out}}) - \beta}{u - \beta}$, and solving the equation for $f(t_{\text{out}})$ gives $f(t_{\text{out}}) = \frac{u}{u - \beta + t_{\text{out}}}$. Moreover, as t_{out} grows, $f(t_{\text{out}})$ decreases, and $t_{\text{out}} f(t_{\text{out}})$ will never be greater than u, therefore, $f(t_{\text{out}})$ will never be zero. Thus, we have the following closed-form formula for f:

$$f(t_{\text{out}}) = \begin{cases} 1 & \text{if } 0 \leq t_{\text{out}} \leq \beta \\ \frac{u}{u - \beta + t_{\text{out}}} & \text{if } \beta \leq t_{\text{out}} \end{cases}. \tag{2}$$

3.2 Problem Definition

We define the All-Pairs routing problem under RED as follows.

Given: Let $G = (N, A)$ be a directed graph, T be the set of all-pair commodities such that $|T| = |N| \cdot (|N| - 1)$. For each Commodity $k \in T$, let c^k be the weight, $s^k > 0$ be the units of flow at the source node $o^k \in N$ to be sent to the destination node $d^k \in N$.

Find: A unique simple path $\{a^k\} \subseteq A$, set of arcs that are a subset of all arcs in the directed graph G, for each $k \in T$.

Objective: Maximize the total weighted flow delivered under RED.

Considering an Internet backbone network, we assume that there is positive demand across all pairs of nodes in the network. Therefore in the problem definition, there are $|N| \cdot (|N| - 1)$ commodities to indicate that every ordered pair induces a commodity, and $s^k > 0, \forall k \in T$ indicates that all the commodities have positive demands.

We show that this problem is NP-Hard, deducing that the original OSPF Routing Problem under RED is NP-Hard.
Theorem 1. *All-Pairs OSPF Routing Problem under RED is NP-Hard.*

The proof is shown in Liu [34, Chapter 5]. The theorem follows from a reduction from Set Cover, a known NP-Hard problem [30]. We create a network instance with a set of four commodities each with different weights. We then create a feasible solution and show that it is optimal and satisfies OSPF constraints that may be used to determine a solution to the Set Cover problem. The key step in the proof is to set the commodity demands, s^k, and weights, c^k, such that a solution of the set cover problem may be determined using a solution to the all-pairs routing problem under RED, mathematical program (3). One may point out that the input consists of all pairs of commodities and does not necessarily consider OSPF constraints. As we find in the complete reduction, we may find single unique paths for all commodities in the network, i.e., those that satisfy OSPF constraints. As such, the reduction will hold even if we consider OSPF constraints.

3.3 The Mathematical Program

We define the following variables and notations:

- x_i^k: the amount of flow of commodity $k \in T$ present at node $i \in N$
- α_{ij}^k: binary variables indicating whether commodity $k \in T$ is sent through arc (i,j), for each arc $(i,j) \in A$
- f_{ij}: the RED function f as in equation (2), for arc $(i,j) \in A$

We propose a model to solve the All-Pairs routing problem under RED as follows:

$$
\begin{aligned}
\max_{\alpha,x} \quad & \sum_{k=1}^{K} c^k x_{d^k}^k && && \text{(3a)}\\
\text{s.t.} \quad & x_{o^k}^k = s^k && k \in T && \text{(3b)}\\
& \sum_{j\in\delta^+(o^k)} \alpha_{o^k j}^k = 1 && k \in T && \text{(3c)}\\
& \sum_{j\in\delta^-(d^k)} \alpha_{j d^k}^k = 1 && k \in T && \text{(3d)}\\
& \sum_{j\in\delta^-(o^k)} \alpha_{j o^k}^k = 0 && k \in T && \text{(3e)}\\
& \sum_{j\in\delta^+(d^k)} \alpha_{d^k j}^k = 0 && k \in T && \text{(3f)}\\
& \sum_{i\in\delta^-(j)} \alpha_{ij}^k = \sum_{i\in\delta^+(j)} \alpha_{ji}^k && k \in T, j \neq o^k, d^k && \text{(3g)}\\
& \sum_{i\in\delta^-(j)} \alpha_{ij}^k \leq 1 && k \in T, j \neq o^k, d^k && \text{(3h)}\\
& \sum_{i\in\delta^-(j)} \alpha_{ij}^k x_i^k f_{ij}\Big(\sum_{l=1}^{K} \alpha_{ij}^l x_i^l\Big) - x_j^k = 0 && k \in T, j \in N : j \neq o^k && \text{(3i)}\\
& \alpha_{ij}^k \in \{0,1\}^n && (i,j) \in A, k \in T && \text{(3j)}\\
& x_i^k \in \mathbb{R}. && && \text{(3k)}
\end{aligned}
$$

Liu [34] shows that mathematical program (3) is non-convex. The objective, equation (3a), is the weighted total flow received of all commodities k at their respective destinations d^k. The weight of commodity k is c^k that is determined exogenously of the model. The Source Constraints, equation (3b), represent the value of the flow units at the origin node for each commodity. The four constraints (3c), (3d), (3e), (3f) ensure that there will be exactly one arc out of the origin selected and one arc into the destination selected for a fixed commodity, and a commodity does not return to its source or leave its destination. The Balance Constraints, equation (3g), guarantees the outflow and inflow of each commodity is the same for all non-source and non-destination nodes. The next constraint, inequality (3h), ensures there is at most one arc used to send each commodity out of each non-source and non-destination node in the network, and guarantees that there will not be any cycles (notice that α may induce some node disjoint cycles, but these cycles are disconnected with either the origin or destination of any commodity and therefore are ignored). Finally, equation (3i) is the arc flow balance constraint, however now it is augmented by the behavior of RED. Recall that for RED we must define the total flow on an arc, this is $\sum_{l=1}^{K} \alpha_{ij}^l x_i^l$; $f(\sum_{l=1}^{K} \alpha_{ij}^l x_i^l)$ is the fraction of the total flow that will successfully be received at the destination, node j. For commodity k, the total amount sent on arc (i,j) is the fraction of commodity k sent from i to j, denoted as $\alpha_{ij}^k x_i^k$, meaning that the total amount of commodity k available at node j is $x_j^k = \sum_{i \in \delta^-(j)} \alpha_{ij}^k x_i^k f_{ij}(\sum_{l=1}^{K} \alpha_{ij}^l x_i^l)$.

In summary, the constraints ensure there is one simple path between each origin-destination pair as the routing path and data flow is sent through these paths subject to RED, and the total weighted flow is maximized.

3.4 OSPF Path Realizability

Since our ultimate goal is to find an improved OSPF routing policy, we need to verify the computed routing policy from mathematical program (3) is indeed OSPF-configurable, or OSPF-realizable. In general, this is a decision problem in which given a set of simple paths, we decide if there is a set of arc metrics with which the paths can be realizable. This problem is called the Inverse Shortest Path (ISP) problem. ISP has been studied since the early 1990s [19]. Multiple researchers [12, 18, 20] show how to apply ISP to network applications, OSPF in particular. A detailed review paper discusses various developments in not only OSPF routing but also ISP [14].

For each node $v \in N$, we define a vector $y^u \in \mathbb{R}^{|N|}$, and formulate the Inverse Shortest Path problem as follows:

$$y_w^u - y_v^u \leq \lambda_{(v,w)}, \forall u \in N, \forall (v,w) \in A \tag{4a}$$
$$y_w^u - y_v^u = \lambda_{(v,w)}, \forall u \in N, u = o^k, (v,w) \in P_k, \forall k \in T \tag{4b}$$
$$\lambda \geq 1, \lambda \in \mathbb{N}^A, \tag{4c}$$

where P_k in equation (4b) is the prescribed path for commodity k. One may interpret $\lambda_{(v,w)}$ as the weight of arc (v,w), and y_v^u may be interpreted as the weight of the shortest path from u to v. Note that mathematical program (4) is a feasibility problem and thus has no explicit objective function; when implementing mathematical program (4) in practice, a constant may be used as an objective if one is required.

Note that if we allow λ to take values from all non-negative real numbers, we can just set $\lambda = 0$ to be the trivial solution and the problem is solved. Therefore, we want to set the weight to be positive and for practical purposes, we let $\lambda \geq 1$.

Note that mathematical program (4) is polynomially solvable when all the parameters are rational, because we can solve it as a linear program and round the solution to integer values. However, in the context of network routing, as arc weights used by shortest path routing protocols, such as OSPF and IS-IS, are bounded above, so we need to add an additional constraint to accommodate this feature, obtaining the following formulation:

$$(4a)(4b) \tag{5a}$$
$$\lambda \geq 1, \lambda \in \mathbb{N}^A, \lambda \leq D, \tag{5b}$$

where D is the upper bound on the arc weights.

Most of the inverse shortest path formulations in the literature are based on the one above. Many researchers [12, 29] in the field of network design have studied this problem to help better understand the OSPF path realizability problem. For example, Ben-Ameur and Gourdin [11] and Ben-Ameur [10] consider the realizability problem using a MIP and LP formulations. Similar to Ben-Ameur and Gourdin [11], Broström and Holmberg [16, 17] derives necessary conditions for a set of routing policies to be realized. In the presented computational work we do not enforce the determined necessary conditions. Finally, we like to highlight that computing weights is not solely restricted to OSPF networks; work has been done to find link weights for the Private Network-to-Network Interface routing protocol in Asynchronous Transfer Mode networks [25, 33]. Bley [12] proved an inapproximability result for the inverse shortest path problem with unique shortest paths. We consider a revised version of this problem such that a path between any pair of nodes is prescribed as the shortest path while the paths do not have to be unique.

In our research we consider backbone networks where routing policies for all pairs of nodes need to be found, we also need to consider the All-Pairs variant of the Inverse Shortest Path Problem, which to our knowledge has yet to be addressed in the literature.

3.4.1 All-Pairs Inverse Shortest Path Problem

Similar to the All-Pairs OSPF Routing under RED, we also study the All-Pairs variant of the Inverse Shortest Path Problem where the inputs are paths between every pair of nodes. We show that this variant of the problem, where the paths are not necessarily unique shortest paths, is NP-Hard. Note that this is a generalization of the work we have done so far. In our work in finding OSPF routing policies we restricted attention to unique routing policies, see mathematical program (3). However, when determining if the resulting policies are realizable, we solve a slightly more general instance where the paths between each origin and destination need not be unique. Work by Bley [12] looks at the ISP variants for directed networks and where the commodities are a subset of all pairs of nodes. We note that the approach by Bley cannot be trivially extended to the scenario we consider as the directed graphs generated in Bley's work are not strongly connected, while the graphs we construct in our proof are strongly connected. Similarly, work by Call and Holmberg [20] consider the complexity of the ISP in which the inputs are a set of paths that are on the shortest path and a set of paths that are not on the shortest path. We consider a slightly relaxed version of this ISP variant, in that we want all given paths to be shortest paths, but other paths may or may not be shortest paths as well, i.e., we do not enforce uniqueness. We relax the uniqueness requirement in our proof as variants of OSPF allow for multiple paths between an origin and destination, though we do not consider it in the mathematical models presented in Section 3.3.

Given: a graph $G = (V, E)$, where V is the node set and E is the arc set, and a set of paths of G: $\{P_{ij} \subset E : i, j \in V, i \neq j\}$, a positive number D.

Find: arc weights $\lambda_e \in \mathbb{N}_+, e \in E$, s.t. $\lambda_e \leq D$, and P_{ij} is a shortest path between i and j. (The shortest paths are not necessarily unique.).

Theorem 2. *The All-Pairs Non-Unique Inverse Shortest Path Problem is NP-Hard.*

The proof is shown in Liu [34, Chapter 6]. Our proof is inspired by Bley's proof. We show our result by presenting a reduction from a variant of the set partitioning problem. In this variant each set has exactly three elements and each element is contained in exactly three sets, and is a known NP-Hard problem [31]. Motivated by the work of Bley [12], we construct, in a polynomial number of steps, an instance of the ISP described above. The construction involves a set of graphs that represent an element and a set of the set partitioning problem, arcs connect elements that are common amongst sets in our construction. Due to the repetitive nature of the graph we build, we characterize the arc weights between each type of node we introduce in the graph through exhaustive enumeration, leading to a lengthy and lucid proof. We show how the all-pairs inverse shortest path instance we create may be used to generate a solution to the three-element, three-set set partitioning problem and vice versa. This key step in the reduction is addressing the paths used by all pairs of nodes in an exhaustive manner.

4 Algorithmic Considerations

Before we present the results of our numerical experiments, we discuss some algorithmic considerations we made to reduce the computation time for each network instance.

4.1 Possibility of Merging the Routing Problem and the Realizability Problem

We study the All-Pairs Routing Problem under RED and the All-Pairs Inverse Shortest Path Problem separately and examine properties associated with the individual problems. We use the paths generated by mathematical program (3) as input into mathematical program (5) to see if the paths are realizable. It should be noted that it is not necessary to separate the All-Pairs Routing Problem under RED and the All-Pairs Inverse Shortest Path Problem. We could incorporate mathematical program (5) constraints into mathematical program (3) and solve the combined problem. However, combining the formulations introduces additional integer variables of the order of $|A|$ and therefore can potentially increase the runtime. In fact, when we merged the two models in our experimental networks, for the smallest network instances, a single instance could not be solved for over three months.[1] Furthermore, we find that it might not be necessary to restrict α in mathematical program (3) with the ISP constraints, as our experiments show that all the computed routing policies from the All-Pairs Routing Problem are realizable. The details of the experiments are shown in Section 5.

4.2 Modification of the formulation

We illustrate a few approaches in modifying mathematical program (3) in this section.

4.2.1 Approximating the RED function

Recall that the RED function defined in equation (2) is a continuous, piecewise function. Since $f(t) \leq 1$ for all $t > \beta$, we can rewrite the function as a capped function, i.e.,

$$f(t) = \min\left\{1, \frac{u}{u-\beta+t}\right\}, t \geq 0.$$

[1] In all of the instances all other simplifications, discussed in Section 5, were carried out, and only the models were merged.

In general, the original RED function defined in equation (2), being a non-smooth function, brings computational difficulty to the problem. In fact, the problem was not solvable with all of the non-linear solvers we attempted using the original RED function. Our understanding of the issue is that the non-smoothness of the original RED function made approximating gradients impossible or extremely difficult for the solvers we tried; note that we tried the non-linear solvers available on the NEOS servers at the time of this study [21, 23, 32]. Preliminary results suggest that the problem scales exponentially, and for a medium sized network with 11 nodes and 14 arcs, most solvers cannot stop iterating after weeks of computation. To address this computational issue, we propose approximating the RED function with a smooth function, with which the modified model can still produce a feasible routing policy, α. Even with this approximation, some instances took multiple hours to solve for the largest network instance.

The approximated RED function, denoted by $f^{\star}(t)$, should satisfy the following three properties:

1. $f^{\star}$ is a smooth function with closed-form formula,
2. $f^{\star}$ should be less than or equal to 1 to preserve its probabilistic definition,
3. $f^{\star}$ should be a strictly decreasing function and converges to 0 when $t \to \infty$.

We define

$$f^{\star}(t) = \frac{u-\beta}{u-\beta+t}, \tag{6}$$

which satisfies all three properties above.

Thus, we can formulate the approximated model as follows:

$$\max_{\alpha,x} \quad \sum_{k=1}^{K} c^k x^k_{d^k} \tag{7a}$$

$$\text{s.t.} \quad \sum_{i\in\delta^-(j)} \alpha^k_{ij} x^k_i f^{\star}_{ij}\Big(\sum_{l=1}^{K} \alpha^l_{ij} x^l_i\Big) - x^k_j = 0 \qquad k \in T, j \in N : j \neq o^k \tag{7b}$$

(3b)(3c)(3d)(3e)(3f)(3g)(3h)(3j)(3k) (7c)

Note that mathematical program (7) only differs from mathematical program (3) by the RED functions. In order to better understand how the two mathematical programs relate, we briefly compare the two functions. We first rewrite equation (2):

$$f(t_{\text{out}}) = \begin{cases} 1 & \text{if } 0 \leq t_{\text{out}} \leq \beta \\ \frac{u}{u-\beta+t_{\text{out}}} & \text{if } \beta \leq t_{\text{out}} \end{cases} .$$

We make the following observations with respect to equations (2) and (6): 1) $f(t) \geq f^{\star}(t)$ for $t \leq \beta$, this follows from the first condition. 2) $f(t) \geq f^{\star}(t)$ for $t > \beta$, as $\frac{u}{u-\beta+t} \geq \frac{u-\beta}{u-\beta+t}$. It immediately follows that for the same set of α values the objective function value of mathematical program (7) is at most the objective function value of mathematical program (3). With the approximated RED function defined above, the runtime of most solvers is reduced significantly, from weeks to hours, and thus we are able to obtain feasible solutions for many larger instances. We use mathematical program (7) only to determine α. Treating the determined α as a parameter, we use mathematical program (3) to determine the corresponding x and use that x to evaluate the performance of mathematical program (7). In the numerical results that we present later in this paper we present the objective function value of mathematical program (3) and not of mathematical program (7) or its derivative, mathematical program (8).

4.2.2 Tree Constraints

In addition to addressing the computational issues of the RED functions of mathematical program (3), we must help with the realizability, finding corresponding arc weights, given α. We add tree constraints into mathematical program (7) in order to determine arc weights. To accommodate the new tree constraints, we define the new variables

$$\gamma_{ij}^{u} = \begin{cases} 1 & \text{if some path rooted from } u \text{ uses arc } (i,j) \\ 0 & \text{o.w.} \end{cases}, u \in N, (i,j) \in A.$$

We add the tree constraints to mathematical program (7) to have the formulation used in our experiments:

$$
\begin{aligned}
\max_{\alpha,\gamma,x} \quad & \sum_{k=1}^{K} c^k x^k_{d^k} && && \text{(8a)}\\
\text{s.t.} \quad & \alpha^k_{ij} \leq \gamma^u_{ij} && u \in N, o^k = u && \text{(8b)}\\
& \sum_{(i,j)\in A} \gamma^u_{ij} = |N| - 1 && u \in N && \text{(8c)}\\
& \text{(7b)(7c)} && && \text{(8d)}\\
& \gamma^u_{ij} \in \{0,1\} && (i,j) \in A, u \in N. && \text{(8e)}
\end{aligned}
$$

The constraints (8b) describe the relationship between the α and γ variables, and together with the tree constraints (8c), they ensure that every subgraph induced by shortest paths rooted from each node is a spanning tree. We highlight the fact that the paths constructed in the proof of Theorem 1 satisfy the tree constraints and thus even model (8) is still NP-Hard.

4.2.3 Robust Model

In practice, network arc weights are updated very infrequently. To find a good, long term policy, we formulate and solve the robust formulation of mathematical program (8). The robust formulation maximizes the minimum weighted flow delivered across a set of realizations.

$$
\begin{aligned}
\max_{\alpha,\gamma,x} \quad & z && && \text{(9a)}\\
& z \leq \sum_{k=1}^{K} c^k x^{kq}_{d^k}, q \in Q && && \text{(9b)}\\
\text{s.t.} \quad & x^{kq}_{o^k} = s^{kq} && k \in T, q \in Q && \text{(9c)}\\
& \sum_{j\in\delta^+(o^k)} \alpha^k_{o^k j} = 1 && k \in T && \text{(9d)}\\
& \sum_{j\in\delta^-(d^k)} \alpha^k_{j d^k} = 1 && k \in T && \text{(9e)}
\end{aligned}
$$

$$\sum_{j \in \delta^-(o^k)} \alpha^k_{jo^k} = 0 \quad k \in T \tag{9f}$$

$$\sum_{j \in \delta^+(d^k)} \alpha^k_{d^k j} = 0 \quad k \in T \tag{9g}$$

$$\sum_{i \in \delta^-(j)} \alpha^k_{ij} = \sum_{i \in \delta^+(j)} \alpha^k_{ji} \quad k \in T, j \neq o^k, d^k \tag{9h}$$

$$\sum_{i \in \delta^-(j)} \alpha^k_{ij} \leq 1 \quad k \in T, j \neq o^k, d^k \tag{9i}$$

$$\sum_{i \in \delta^-(j)} \alpha^k_{ij} x^{kq}_i f_{ij}(\sum_{l=1}^{K} \alpha^l_{ij} x^{lq}_i) - x^{kq}_j = 0 \quad k \in T, j \in N : j \neq o^k, q \in Q \tag{9j}$$

$$\alpha^k_{ij} \leq \gamma^u_{ij} \quad u \in N, o^k = u \tag{9k}$$

$$\sum_{(i,j) \in A} \gamma^u_{ij} = |N| - 1 \quad u \in N \tag{9l}$$

$$\alpha^k_{ij} \in \{0,1\} \quad (i,j) \in A, k \in T \tag{9m}$$

$$\gamma^u_{ij} \in \{0,1\} \quad (i,j) \in A, u \in N, \tag{9n}$$

where Q represents the collection of traffic demands scenarios. In the numerical experiments presented in Section 5 we have $|Q| = 8$. In particular, we considered a week as our basis and determined the average daily demand for each 3 hour period, i.e., we had 8 demand instances in total. For example demand instance 1 is the average demand for all commodities Monday through Sunday for the hours of 1 AM through 3 AM, instance 2 is the average demand for all commodities Monday through Sunday for 4 AM through 6 AM, etc. We experimented with larger uncertainty sets, for example we tried every hour, but we found there was too much variability for hourly observations and the resulting routing policies were too conservative. In addition, with fewer demand instances, we were able to have faster computational times for our robust instances, relative to larger robust instances.

5 Numerical Experiments

Even though the All-Pairs Routing Problem under RED is NP-hard, in this section, we show how we can solve the problem for three real-world networks. We use mathematical programs (8) and (9) to determine the routing policies for non-robust and robust instances, respectively. Note that we solve the robust model once using the first week of data available. Given a routing policy, we use mathematical program (3) to determine its objective function value. We highlight the fact that our work is focused on finding OSPF routing policies for backbone networks, those with demand from all origin and destination pairs. As such, the network instances, all of which are real-world networks, may be considered small, but backbone networks tend to be rather small networks relative to tier 3 or access networks, though those networks do not have demand between all node pairs. We also would like to note that computation times for the largest network are rather long, taking over a day to complete. Thus we do not consider larger networks.

5.1 Experiments and Results

For the experiments, we use Bonmin [1] as the solver, as it is one of the few solvers available that are able to handle non-linear, non-convex math programs and obtain a feasible solution in a relatively short amount of time. Like many other off-the-shelf solvers, Bonmin ensures optimality for convex optimization but does not guarantee optimality for non-convex problems; however, as we will see, it produces a relatively "good" solution for these problems.

5.1.1 Networks

Our experiments use data from three real-world networks, all of which are National Research and Education Networks (NREN). NREN is a backbone network dedicated to research and education institutes; the routers in the network are typically located at major cities and universities. Abilene Network is the United States' NREN prior to 2007, when it was retired and upgraded into the current "Internet2 Network". Taiwan Advance Research and Edu-

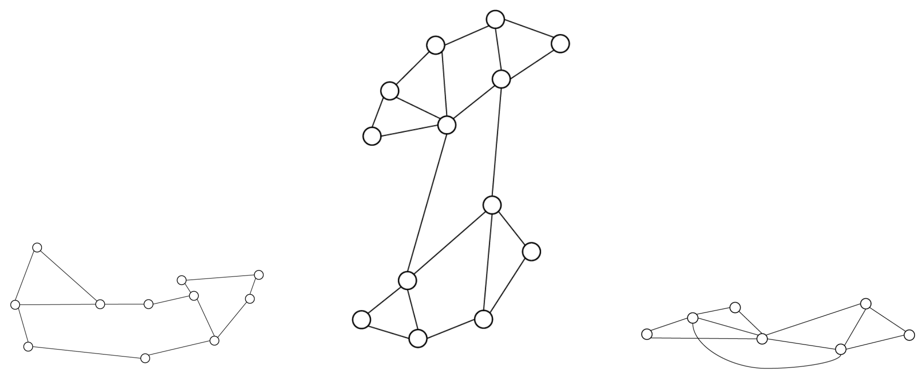

(a) Abilene Network (b) TWAREN (c) CANARIE

Figure 2: NREN Networks

cation Network (TWAREN) [3] is the current Taiwanese NREN. Canada's Advanced Research and Innovation Network (CANARIE) [2] is the current Canadian NREN.

Abilene Network contains 11 nodes and 14 arcs; TWAREN contains 13 nodes and 20 arcs; CANARIE contains 7 nodes and 11 arcs. The networks are illustrated in Figure 2.

For each network, we take the hourly traffic demands for a one-week period as input and solve mathematical program (8) for each. For Abilene, traffic demands between all pairs of nodes are collected, so we directly use them. For TWAREN and CANARIE, we estimate the traffic demands using link utilization. The α in the solutions are then used in mathematical program (3) to determine the objective value. We then use mathematical program (5) to check whether the obtained paths induced by α are realizable. The objective values over one week of instances are compared with those using the industry standard routing policies and are illustrated by the dashed and solid lines in Figure 4. It is of no surprise that a policy determined for each instance using mathematical program (8) performs better than the industry policy, as mathematical program (8) considers the demand for that instance when setting the routing policy. We look at setting one policy for all demand instances in the next subsection.

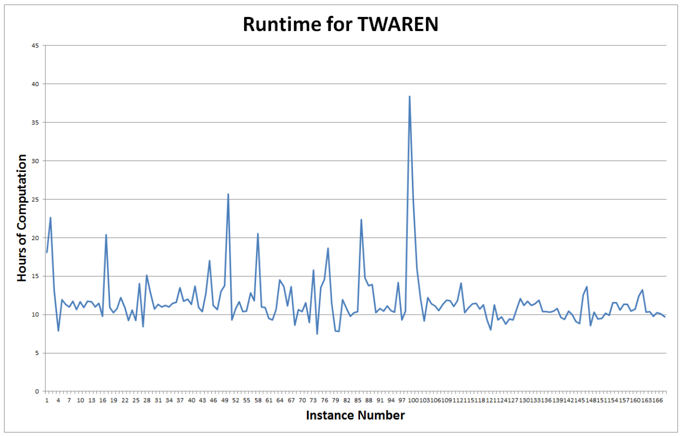
5.1.2 Robust Model Numerical Results

Figure 4 shows the comparison of three sets of routing policies: the industry benchmark, the policy found by mathematical program (8) and the policy found by the robust mathematical program model (9). As expected, the robust model performs slightly worse than the optimal policy found in mathematical program (8), however, it still outperforms the benchmark in most cases. Note that the robust policy is determined using data from a week other than the week on which it is tested that occurred prior to the tested week and the robust instances were determined by averaging three hour time intervals into a single instance, leading to eight demand instances for the robust problem.

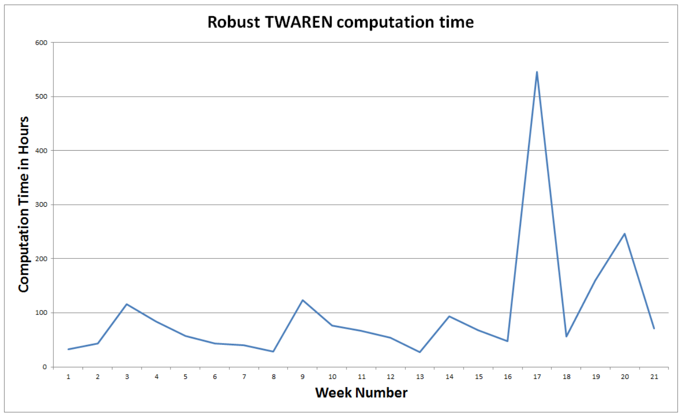
Before we continue we would like to discuss the computational time of the problems we consider. For example, the largest network we consider is the TWAREN network, and for a week to compute the optimal OSPF routing policies that account for RED may take up to 38 hours, with an average of about 12 hours and a standard deviation of about 3.5 hours; see Figure 3(a). As expected the computational time for the robust instance increases: it may take up to 540 hours to complete, though the average computation time is 100 hours with a minimum of 27 hours (Figure 3(b)). The key takeaway from our computational time discussion is: even though we improved the computational time of our method, using improved models, even the new method does not scale well for larger networks. We were not able to solve larger problem instances, for example, networks of 20 or more nodes were not solved after a week or more of computation. Finding computationally more attractive models and methods is something of interest especially if larger networks are considered. One issue to keep in mind is that larger networks may not require all-pairs communication which may make the problem easier, something we do not consider in our research.

5.1.3 Analysis

For all three networks, the traffic demands over time all share a diurnal pattern as well as the weekday-weekend pattern, as seen in Figure 4. Specifically, the demands are higher in the daytime versus in the night time, and higher during weekdays than weekends. For Abilene, the highest total demand in



(a) TWAREN computation time



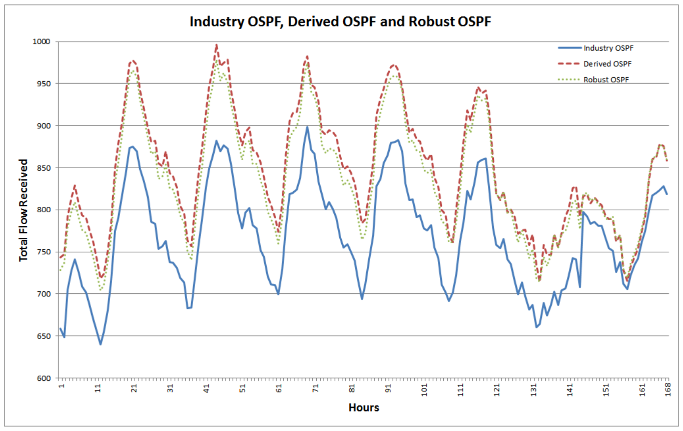
(b) TWAREN robust instance computation time

Figure 3: Computation time for TWAREN network

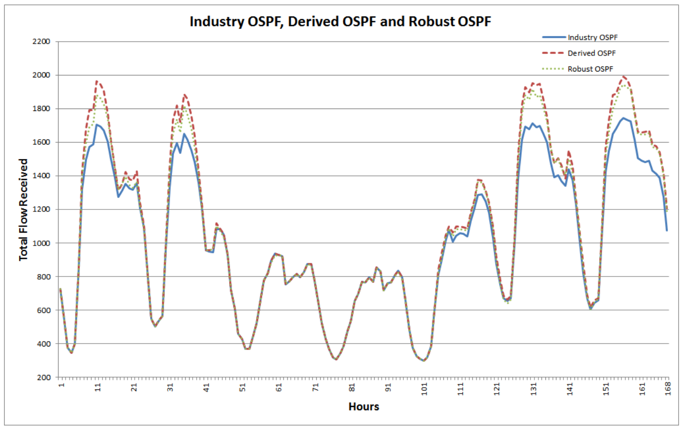
a week is two times the lowest total demand, while for TWAREN and CANARIE, the highest is about four times the lowest. Therefore, for Abilene, the total flow received is relatively stable (Figure 4(a)) but TWAREN and CANARIE show a much greater variation in total flow received over a week; see Figures 4(b) and 4(c). The demand variation is significantly exhibited in the comparison between the Industry policy and our policy; suppose the demand drops below the minimum threshold, β in equation (2), then under RED, every node will retain all the flow that is sent over, and therefore, the total flow received would be the same for any policy. In addition, we note that the robust policy does not perform that much worse relative to the best found policy for each instance, something we explore further in the next set of plots. As shown in Figure 5, for TWAREN and CANARIE, the percentage improvement of the total flow received from the industry policy compared to our derived robust policy shows a similar diurnal pattern. For Abilene, the percentage increase is around 10%. Overall, these results show that our method outperforms the industry benchmark, for OSPF routing problem under RED.

5.1.4 Realizability

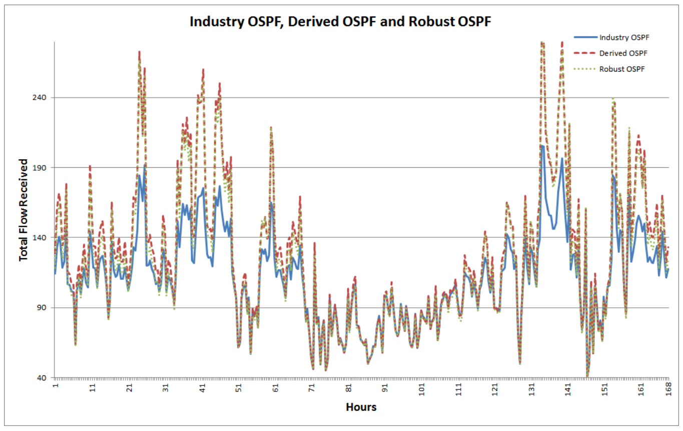
So far, we have obtained a feasible policy for each demand instance of Abilene, TWAREN and CANARIE. In order to make sure that these policies gain better performance without losing the OSPF compatibility, we need to solve model mathematical program (5) to make sure we can find the corresponding OSPF arc weights for each policy. Recall that in mathematical program (8), we introduce a tree constraint, a necessary condition of OSPF configurability. Our results show that every single one of the policies, regardless of whether it is a regular policy or robust policy, is OSPF-configurable, i.e., we obtain feasible arc metrics for each policy. This means that not only are we generating routing policies that deliver more traffic, but these routing policies may be implemented in practice using OSPF.



(a) Abilene

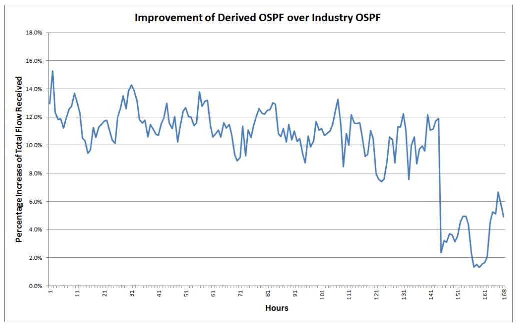


(b) TWAREN

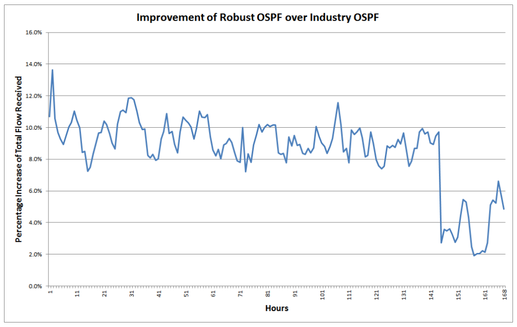


(c) CANARIE

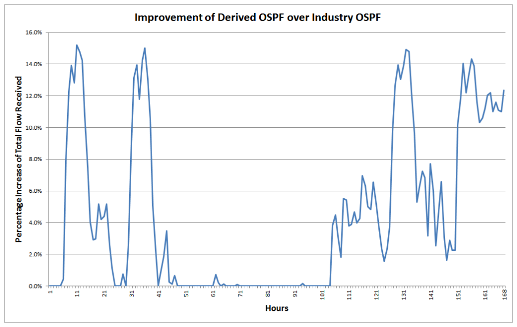
Figure 4: Industry OSPF, Derived OSPF and Robust OSPF



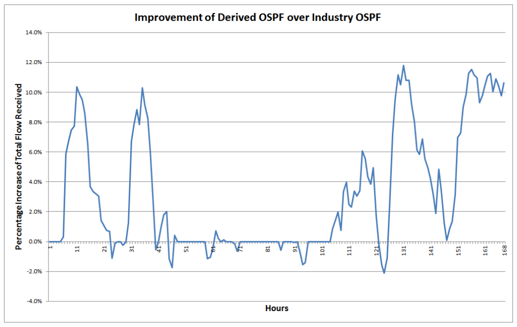
(a) % Improvement derived vs industry Abilene



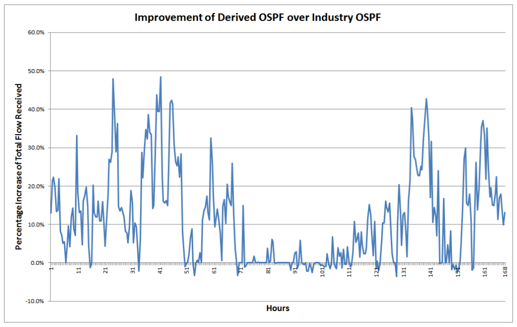
(b) % Improvement of robust vs industry Abilene



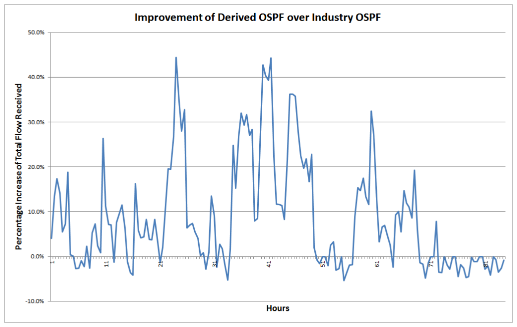
(c) % Improvement derived vs industry TWAREN



(d) % Improvement of robust vs industry TWAREN



(e) % Improvement derived vs industry CANARIE



(f) % Improvement of robust vs industry CANARIE

Figure 5: Percent improvement of derived and robust vs industry for NREN networks

6 Conclusion

In this paper, we presented a mathematical model for single source single path end-to-end routing problem that accounts for active congestion control, namely Random Early Detection (RED). We showed that the resulting problem not only is non-convex, but also NP-Hard. Using an off the shelf solver, we showed that routing policies that perform better than those currently used in three real-world backbone networks may be found when considering the robust instance of our model. Further, we showed that when determining if a routing policy is OSPF realizable in the all-pairs setting, arc weights may be found such that solving the all-pairs shortest path problem generates the same routing policy, is also NP-Hard. However, all routing policies generated in our computational experiments are OSPF realizable.

In the future, we would like to consider more than just OSPF by considering all shortest path protocols in general, by removing the single path between origin and destination requirement. In addition, in the presented work we assumed that the RED parameters were exogenously given and fixed, so one natural extension is to endogenize the value of the RED parameters. Finally, it is surprising that all of the over one thousand real-world instances we considered (we only displayed a subset of the ones we considered) had routing policies that were realizable. We would like to understand if our instances have special structure making all instances realizable.

7 Acknowledgments

This work was partially funded by the Natural Sciences and Engineering Research Council of Canada (NSERC). The Abilene Network traffic demands used in our work were collected by Prof. Yin Zhang at University of Texas, Austin. The traffic data for CANARIE was provided by Thomas Lam and Jun Jian at CANARIE. We would like to thank them for providing all the real-world information for completing our work.

References

[1] Bonmin home page: Basic open-source nonlinear mixed integer programming, http://www.coin-or.org/Bonmin/.

[2] CANARIE network operation center, http://www.canarie.ca/en/network/overview/.

[3] TWAREN network operation center, http://noc.twaren.net/noc_eng/index.php/.

[4] Configuring OSPF, http://www.cisco.com/c/en/us/td/docs/security/asa/asa82/configuration/guide/config/route_ospf.pdf.

[5] Cisco visual networking index: Global mobile data traffic forecast update 2009-2014, White Paper.

[6] Cisco visual networking index: Global mobile data traffic forecast update, 2013–2018, White Paper.

[7] A. Altın, B. Fortz, M. Thorup, and H. Ümit, Intra-domain traffic engineering with shortest path routing protocols, 4OR 7 (2009), 301–335.

[8] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, Overview and principles of internet traffic engineering, RFC 3272 (Informational) Updated by RFC 5462.

[9] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, Requirements for traffic engineering over MPLS, RFC 2702 (Informational).

[10] W. Ben-Ameur, Multi-hour design of survivable classical IP networks, Int J Commun Syst 15 (2002), 553–572.

[11] W. Ben-Ameur and E. Gourdin, Internet routing and related topology issues, SIAM J Discr Math 17 (2003), 18–49.

[12] A. Bley, Inapproximability results for the inverse shortest paths problem with integer lengths and unique shortest paths, Networks 50 (2007), 29–36.

[13] A. Bley, Routing and capacity optimization for IP networks, PhD thesis, TU Berlin, 2007.

[14] A. Bley, B. Fortz, E. Gourdin, K. Holmberg, O. Klopfenstein, M. Pióro, A. Tomaszewski, and H. Ümit, "Optimization of OSPF routing in IP networks," Graphs and algorithms in communication networks, A. Koster and X. Muñoz (Editors), Springer Berlin Heidelberg, 2010, pp. 199–240.

[15] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, Recommendations on queue management and congestion avoidance in the internet, RFC 2309 (Informational) Updated by RFC 7141.

[16] P. Broström and K. Holmberg, Determining the non-existence of compatible OSPF weights, NordicMPS 2004, no. 14 Linkoping Electron Conference Proc, 2004, pp. 7–21.

[17] P. Broström and K. Holmberg, On the extremal structure of an related cone, Vietnam J Math 35 (2007), 507–522.

[18] P. Broström and K. Holmberg, Valid cycles: A source of infeasibility in open shortest path first routing, Networks 52 (2008), 206–215.

[19] D. Burton and P.L. Toint, On an instance of the inverse shortest paths problem, Math Program 53 (1992), 45–61.

[20] M. Call and K. Holmberg, Complexity of inverse shortest path routing, 5th Int Conference, INOC 2011, 2011, pp. 339–353.

[21] J. Czyzyk, M.P. Mesnier, and J.J. Moré, The NEOS server, IEEE J Comput Sci Eng 5 (1998), 68–75.

[22] S. Dimitrov, M. Epelman, and D. Sharma, New models of network routing under active congestion control, Technical report, University of Michigan, Industrial and Operations Engineering Department, Ann Arbor, Michigan, September 2009.

[23] E.D. Dolan, The NEOS server 4.0 administrative guide, Technical memorandum ANL/MCS-TM-250, Mathematics and Computer Science Division, Argonne National Laboratory, 2001.

[24] S. Even, A. Itai, and A. Shamir, On the complexity of time table and multi-commodity flow problems, Proc 16th Ann Symp Foundations Comput Sci, IEEE Computer Society 1975, pp. 184–193.

[25] A. Faragó, Á. Szentesi, and B. Szviatovszki, Inverse optimization in high-speed networks, Discr Appl Math 129 (2003), 83–98.

[26] S. Floyd, Congestion control principles, RFC 2914 (Best Current Practice) Updated by RFC 7141.

[27] S. Floyd and V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Trans Networking 1 (1993), 397–413.

[28] B. Fortz and M. Thorup, Internet traffic engineering by optimizing OSPF weights, INFOCOM 2000. Nineteenth Ann Joint Conference IEEE Comput Commun Societies. Proceedings., Vol. 2, IEEE 2000, pp. 519–528.

[29] B. Fortz and M. Thorup, Optimizing OSPF/IS-IS weights in a changing world, IEEE J Selected Areas Commun 20 (2002), 756–767.

[30] M.R. Garey and D.S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, W. H. Freeman & Co., New York, NY, USA, 1979.

[31] T.F. Gonzalez, Clustering to minimize the maximum intercluster distance, Theor Comput Sci 38 (1985), 293–306.

[32] W. Gropp and J.J. Moré, "Optimization environments and the NEOS server," Approximation theory and optimization, M.D. Buhman and A. Iserles (Editors), Cambridge University Press, 1997, pp. 167 – 182.

[33] R. Izmailov, B. Sengupta, and A. Iwata, Administrative weight allocation for PNNI routing algorithms, 2001 IEEE Workshop High Performance Switching Routing (IEEE Cat. No.01TH8552), IEEE, 2001, pp. 347–352.

[34] J. Liu, An optimization problem of internet routing, Master's Thesis, University of Waterloo, 2014.

[35] J. Moy, OSPF version 2, RFC 2328 (INTERNET STANDARD) Updated by RFCs 5709, 6549, 6845, 6860.

[36] J. Nagle, Congestion control in IP/TCP internetworks, SIGCOMM Comput. Commun. Rev. 14 (1984), 11–17.