Discovery of Flow Splitting Ratios in ISP Networks with Measurement Noise

Schou, Morten Konggaard; Poese, Ingmar; Srba, Jiri

Published in:
Proceedings - 2023 IEEE 28th Pacific Rim International Symposium on Dependable Computing, PRDC 2023

DOI (link to publication from Publisher):
10.1109/PRDC59308.2023.00017

Publication date:
2023

Document Version
Accepted author manuscript, peer reviewed version

Discovery of Flow Splitting Ratios in ISP Networks with Measurement Noise

Morten Konggaard Schou
Aalborg University, Denmark

Ingmar Poese
BENOCS GmbH, Germany

Jiří Srba
Aalborg University, Denmark

***Abstract*—Network telemetry and analytics is essential for providing highly dependable services in modern computer networks. In particular, network flow analytics for ISP networks allows operators to inspect and reason about traffic patterns in their networks in order to react to anomalies. High performance network analytics systems are designed with scalability in mind, and can consequently only observe partial information about the network traffic. Still, they need to provide a holistic view of the traffic, including the distribution of different traffic flows on each link. It is impractical to monitor such fine-grained telemetry, and in large, heterogeneous networks it is often too complex and error-prone, if not impossible, to access and maintain all technical specifications and router-specific configurations needed to determine e.g. the load balancing weights used when traffic is split onto multiple paths. The ratios by which flows are split on the possible paths must be derived indirectly from the measured flow demands and link utilizations. Motivated by a case study provided by a major European ISP, we suggest an efficient method to estimate the flow splitting ratios. Our approach, based on quadratic linear programming, is scalable and robust to the measurement noise found in a typical network analytics deployment. Finally, we implement an automated tool for estimating the flow splitting ratios and document its applicability on real data from the ISP.**

I. Introduction

Network flow analytics [1, 2] in internet service provider (ISP) networks is often employed by network operators for monitoring the traffic patterns [3]–[5]. This can help to optimize overall network performance and link utilizations. As modern computer networks transfer huge quantities of data, it is impossible to store and analyze every single packet forwarded in the network. By packet sampling (using e.g. NetFlow [6] or IPFIX [7]), a network operator is though capable of estimating, with a relatively high precision, the number of packets transferred by each flow in the network. Similarly, packet counters for each interface can provide reliable information on the current link utilizations. However, answering questions like: "*What traffic caused a spike on this link yesterday?*" requires the analytics to not only show the total traffic dispatched on each link (identifying the spike), but it also needs to break down this traffic into the different flows, in order to determine where the anomaly originates from.

In this paper, we tackle the problem of correlating flows with link traffic in a practical and scalable manner—a problem arising from a case study with a major network analytics company that monitors over 6000 routers across multiple ISPs.

Sampling packet headers on every link in the network can answer such questions; however, it has severe scalability issues. Instead, current high performing network analytics systems sample packet headers at the ingress routers only and combine this with the information from Border Gateway Protocol (BGP) [8] and the interior gateway protocol (IGP) (e.g. IS-IS [9, 10] or OSPF [11]) to determine the links traversed by the packet.

A lookup of the packet's destination in the ingress routing table determines the BGP next-hop, which is the router where the packet will egress the network. The possible paths that the packet can use to go through the network from ingress to egress are obtained from the IGP. All the packets that travel from the same ingress to egress in the network are aggregated into a *flow* that has a certain demand (size in bytes per second) averaged over some time window. Figure 1a shows an example network where links are annotated by the current link utilizations and Figure 1b depicts two flows of demand 12 and 4, respectively. In order to better distribute traffic along the links and thus reduce the maximum link utilization [12], flows can be split along multiple paths as demonstrated in the last column in Figure 1b. The splitting ratios can be uniform among the available paths, or they can depend on the link capacities [13] or custom link weights as shown in Figure 1c.

In practice the flow splitting ratios on the router depend on many technical, vendor-specific implementation details and configurations—some of which may not be accessible. Obtaining and processing this fine-grained information across a large heterogeneous network would require a very complex system. Hence the network analytics company deem it impossible in practice to obtain the flow splitting ratios directly from the router. As it is, moreover, infeasible to sample and categorize the packets traversing each link to the corresponding flows, we need additional information in order to first infer the flow splitting ratios and then estimate how much of each flow contributes to the load on a concrete link in the network.

Fortunately, each router has a byte counter for each interface that measures the total amount of traffic sent out on each link. This information is regularly queried using SNMP [14, 15], and then the link utilization for a given time interval is estimated by linear interpolation between SNMP measurements. In our example from Figure 1a, each link is annotated with the current link utilization. We now want to solve the following *Flow Splitting Ratios* (FSR) problem: given flow demands, their paths and aggregated link utilizations, find the flow splitting ratios such that when we accordingly project flow demands onto the links, the predicted traffic on each link matches (as close as possible) the measured link utilizations.

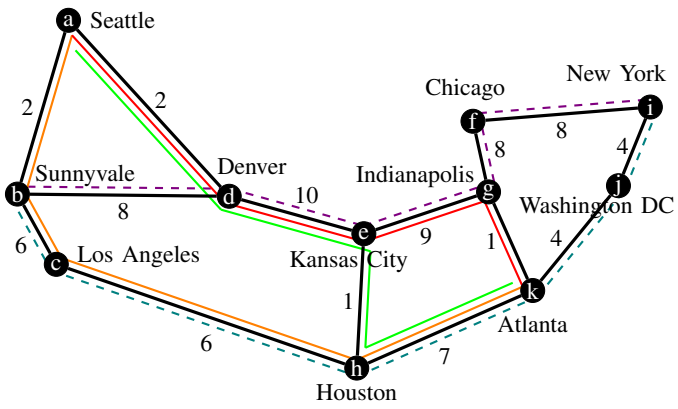
As our main contribution, we provide a practical and efficient solution to the FSR problem, employing quadratic linear programming. As a concrete instance of the FSR problem, consider our running example from Figure 1: given the flow demands, available paths and the link utilizations, our approach automatically predicts the splitting ratios at each node (depicted in the last column of Figure 1c) and hence identifies how much every flow contributes to the total load on each link. Moreover, we suggest a filtering method to compute the splitting ratios even in case of large (but relatively rare) measurement errors that are present in a practical deployment of a real network. The suggested mechanism can efficiently deal with such measurement noise and errors and we demonstrate the robustness of our approach on a large set of simulated networks from the Topology Zoo [16] as well as on real traffic data from a major European ISP (in collaboration with a network analytics company)

We observe that our approach achieves high precision in determining the load balancing weights even in cases where the measured data are imprecise and occasionally significantly deviate from the actual ingress traffic. Based on an extensive statistical evaluation on a benchmark of over 190 real-world ISP topologies, we conclude that our filtering technique helps to improve the precision by an order of magnitude in the best cases and achieves about 66% improvement in the median case. Our approach scales to even large ISP networks with thousands of routers and millions of flows. This allows us to analyze real traffic data from a major European ISP network (which consists of over 3.000 routers and 14.000 links) in a matter of minutes. We automatically identify the load balancing weights in this network (which in this concrete case closely correlates with the capacity-based splitting ratios where the balancing weights are proportional to the link capacities) and put the more precise flow analysis into production (as a part of a network analysis tool developed by the company).

Related Work: Linear programming has been used for network traffic engineering to synthesize optimal splitting ratios [17] even considering the traffic shifts caused by failure recovery [18]–[24]. We, on the other hand, use LP to reverse engineer splitting ratios employed in a real network with the purpose of providing more accurate traffic flow analytics. Contrary to other papers that use linear objective functions, we employ quadratic optimization that is better suited for this application domain. From the network monitoring research, network traffic analysis and visualization tools like NVisionIP [25], Flowyager [26] and VITALflow [27] have been designed for the purpose of network security [25, 28, 29] and management [26, 27]. To the best of our knowledge, none of these tools can reliably project the flow traffic on each link, unless making assumptions on the underlying router configurations, which can be difficult to obtain for larger networks.

II. Problem Formalization

In this section, we shall first define the notion of a network and traffic flows and then formally rephrase the problem of identifying the flow splitting ratios.



(a) Abilene network from the Internet Topology Zoo [16]

Flow	Demand	Paths
Sunnyvale → New York	12	b-d-e-g-f-i b-c-h-k-j-i
Seattle → Atlanta	4	a-d-e-g-k a-d-e-h-k a-b-c-h-k

(b) Two flows in Abilene network and their paths

Flow	Split Node	Split Ratios
Sunnyvale → New York	Sunnyvale	Los Angeles: 1/3 Denver: 2/3
Seattle → Atlanta	Seattle	Denver: 1/2 Sunnyvale: 1/2
	Kansas City	Indianapolis: 1/2 Houston: 1/2

(c) Splitting ratios for the two flows

Fig. 1: Example network topology with link utilizations

A. Network, Paths and Flows

We model a network as a directed simple graph $N = (V, E)$ where V is a finite set of *nodes* (routers) and $E \subseteq V \times V$ is a finite set of *links*. A (simple) *path* in the network is a sequence of distinct nodes $p = v_1 v_2 \dots v_n$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i < n$. The total *link utilization* is a function $U : E \to \mathbb{N}$ that assigns to each link its current load.

A traffic *flow* inside the network is a pair $f = (s, t) \in V \times V$ of ingress and egress router, respectively. The *traffic matrix* of the network is a function $F : V \times V \to \mathbb{N}$ such that $F(f)$ where $f = (s, t)$ indicates the amount of traffic that ingress the network at s and egress at t, averaged over some time interval. The set of *paths* used by a flow (from ingress to egress) is given by the set $P(f) = \{p_1, \dots, p_n\}$ where each path $p \in P$ starts at the ingress node s and ends in the egress node t.

From the set of paths $P(f)$, we can construct a directed subgraph $G(f) \subseteq E$ of the network where there is an edge $(u, v) \in G(f)$ if and only if there is a path $p \in P(f)$ which contains uv as a subpath. When the network computes the set of paths using equal-cost multi-path routing (ECMP) [30], the subgraph for every flow $f = (s, t)$ is acyclic and the set of

possible paths from s to t through $G(f)$ is exactly $P(f)$.

To avoid making assumptions on symmetries of the load balancing weights, we model them independently for each flow f, so that each node v in the flow graph $G(f)$ has a *splitting ratio* $d_v^f : V \to [0;1]$ such that $d_v^f(u)$ denotes for each next-hop u the percentage of traffic of the flow f that splits at the node v and follows the link (v,u). The flow splitting ratios must satisfy $\sum_{u \in V} d_v^f(u) = 1$ and $d_v^f(u) > 0$ only if $(v,u) \in G(f)$.

Now the fraction of traffic from a flow f on a link e can be calculated using the paths and flow splitting ratios as follows:

$$x_e^f \triangleq \sum_{p \in P(f)} \begin{cases} \prod_{i=1}^{j} d_{v_i}^f(v_{i+1}) & \text{if } p = v_1 \dots v_n \text{ and } e = (v_j, v_{j+1}) \\ 0 & \text{otherwise .} \end{cases}$$

The value of x_e^f is the sum the traffic for the flow f over the paths that go through e and for each path we multiply the splitting ratios up until reaching the link e. In the running example in Figure 1, we have for example (ignoring the nodes with no splitting): $x_{hk}^{a \to k} = d_a^{a \to k}(b) + d_a^{a \to k}(d) \cdot d_e^{a \to k}(h) = 1/2 + 1/2 \cdot 1/2 = 3/4$. This means that $3/4$ of the flow size from Seattle to Atlanta passes through the link between Houston and Atlanta.

B. Correlation of Traffic Flow and Link Utilization

By correlating the projection of flow traffic onto the links with the actual link utilization U, we can evaluate various hypotheses about the flow splitting ratios, and in that way improve the accuracy of the forward projection of traffic flow.

In an ideal world, we wish to find flow splitting ratios such that the projected traffic matches the actual link utilization:

$$\forall e \in E. \quad U(e) = \sum_{f \in V \times V} F(f) \cdot x_e^f \qquad (ideal)$$

However, due to the inaccuracies of data introduced by e.g. sampling, timing and delay, misclassification or loss of measurements, we cannot expect the projected flow traffic to exactly match the link utilization. Instead, we define a cost function of how badly the projected traffic on the links differs from the actual link utilization.

$$cost \triangleq \sum_{e \in E} penalty\Big(U(e), \sum_{f \in V \times V} F(f) \cdot x_e^f\Big)$$

where the $penalty$ function describes how undesirable an estimation (est) is given the actual value ($util$), e.g. the absolute error $penalty_{abs}(util, est) \triangleq |util - est|$, or squared relative error $penalty_{rel2}(util, est) \triangleq \left(\frac{util - est}{util}\right)^2$.

Remark 1: In practice, there is a large variety in the size and utilization of links across a network, so $penalty_{abs}$ tends to overfit the large links. Using relative errors alleviates this problem, and squaring the error, like $penalty_{rel2}$ does, penalizes large errors more than several small errors, hence preferring to spread out the inaccuracies over the network.

In the case study, we know that there is some missing data for the traffic matrix F, making it unavoidable that some

Define non-negative variables:
$x_{bc}^{b \to i}, x_{bd}^{b \to i}, x_{ab}^{a \to k}, x_{ad}^{a \to k}, x_{eg}^{a \to k}, x_{eh}^{a \to k}, x_{hk}^{a \to k}$,
$err_{ab}, err_{ad}, err_{bc}, err_{bd}, err_{ch}, err_{de}, err_{eg}$,
$err_{eh}, err_{fi}, err_{gf}, err_{gk}, err_{hk}, err_{ji}, err_{kj}$

Minimize:
$(err_{ab})^2 + (err_{ad})^2 + (err_{bc})^2 + (err_{bd})^2 + (err_{ch})^2 +$
$(err_{de})^2 + (err_{eg})^2 + (err_{eh})^2 + (err_{fi})^2 + (err_{gf})^2 +$
$(err_{gk})^2 + (err_{hk})^2 + (err_{ji})^2 + (err_{kj})^2$

Subject to:
(for the flow Sunnyvale $\to$ New York ($b \to i$))
(1) $x_{bc}^{b \to i} + x_{bd}^{b \to i} = 1$
(for the flow Seattle $\to$ Atlanta ($a \to k$))
(2) $x_{ab}^{a \to k} + x_{ad}^{a \to k} = 1$
(3) $x_{ad}^{a \to k} = x_{eg}^{a \to k} + x_{eh}^{a \to k}$
(4) $x_{ab}^{a \to k} + x_{eh}^{a \to k} = x_{hk}^{a \to k}$
(relative link errors)
(5) $err_{ab} \cdot 2 \geq 4 \cdot x_{ab}^{a \to k} - 2 - c$
(6) $err_{ad} \cdot 2 \geq 4 \cdot x_{ad}^{a \to k} - 2 - c$
(7) $err_{bc} \cdot 6 \geq 12 \cdot x_{bc}^{b \to i} + 4 \cdot x_{ab}^{a \to k} - 6 - c$
(8) $err_{bd} \cdot 8 \geq 12 \cdot x_{bd}^{b \to i} - 8 - c$
(9) $err_{ch} \cdot 6 \geq 12 \cdot x_{bc}^{b \to i} + 4 \cdot x_{ab}^{a \to k} - 6 - c$
(10) $err_{de} \cdot 10 \geq 12 \cdot x_{bd}^{b \to i} + 4 \cdot x_{ad}^{a \to k} - 10 - c$
(11) $err_{eg} \cdot 9 \geq 12 \cdot x_{bd}^{b \to i} + 4 \cdot x_{eg}^{a \to k} - 9 - c$
(12) $err_{eh} \cdot 1 \geq 4 \cdot x_{eh}^{a \to k} - 1 - c$
(13) $err_{fi} \cdot 8 \geq 12 \cdot x_{bd}^{b \to i} - 8 - c$
(14) $err_{gf} \cdot 8 \geq 12 \cdot x_{bd}^{b \to i} - 8 - c$
(15) $err_{gk} \cdot 1 \geq 4 \cdot x_{eg}^{a \to k} - 1 - c$
(16) $err_{hk} \cdot 7 \geq 12 \cdot x_{bc}^{b \to i} + 4 \cdot x_{hk}^{a \to k} - 7 - c$
(17) $err_{ji} \cdot 4 \geq 12 \cdot x_{bc}^{b \to i} - 4 - c$
(18) $err_{kj} \cdot 4 \geq 12 \cdot x_{bc}^{b \to i} - 4 - c$
(19) $c = 0$

Fig. 2: Quadratic programming formulation of the example

estimates become too low, so we decide to only penalize over-estimations. Further, in practice the small flows and links are given less importance, so we want to avoid noise in the data with low magnitude having too big an impact on the relative errors. For this we introduce a constant c that is the acceptable absolute error (e.g. $c = 100$Mbps), and arrive at the following penalty function:

$$penalty(util, est) \triangleq \begin{cases} \left(\frac{est - util - c}{util}\right)^2 & \text{if } est - util > c \\ 0 & \text{otherwise .} \end{cases} \qquad (1)$$

The *flow splitting ratios* (FSR) problem is now to find the splitting ratios d_v^f that minimize the cost function from Equation 1.

III. Solution to Flow Splitting Ratio Synthesis

To solve the FSR problem, we turn to mathematical optimization. In particular, we first encode the FSR problem as the problem of minimizing a linear or quadratic optimization function (depending on the penalty function used) on continuous variables subject to a set of linear constraints. We then study the influence of measurement noise on the precision of splitting ratio estimates.

A. Encoding of FSR to a Linear Program

Linear programming (LP) and quadratic programming are well-studied problems with several industry-standard

	t=1	t=2	t=3
flow size $a \to k$	4.2 / 4.0	8.8 / 8.0	18.0 / 6.0
flow size $b \to i$	11.4 / 12.0	22.8 / 24.0	17.1 / 18.0
ratio $d_a^{a\to k}(b)$	51% / 50%	52% / 50%	66% / 50%
ratio $d_a^{a\to k}(d)$	49% / 50%	48% / 50%	34% / 50%
ratio $d_e^{a\to k}(g)$	50% / 50%	50% / 50%	50% / 50%
ratio $d_e^{a\to k}(h)$	50% / 50%	50% / 50%	50% / 50%
ratio $d_b^{b\to i}(c)$	32% / 33%	31% / 33%	6% / 33%
ratio $d_b^{b\to i}(d)$	68% / 67%	69% / 67%	94% / 67%
mean error	0.83%	1.39%	14.36%
max error	1.30%	2.27%	26.99%
link util. ab	2.1 / 2.0	4.6 / 4.0	11.8 / 3.0
link util. ad	2.1 / 2.0	4.2 / 4.0	6.2 / 3.0
link util. bc	5.8 / 6.0	11.8 / 12.0	12.9 / 9.0
link util. bd	7.7 / 8.0	15.6 / 16.0	16.0 / 12.0
link util. ch	5.8 / 6.0	11.8 / 12.0	12.9 / 9.0
link util. de	9.8 / 10.0	19.8 / 20.0	22.2 / 15.0
link util. eg	8.8 / 9.0	17.7 / 18.0	19.1 / 13.5
link util. eh	1.0 / 1.0	2.1 / 2.0	3.1 / 1.5
link util. fi	7.7 / 8.0	15.6 / 16.0	16.0 / 12.0
link util. gf	7.7 / 8.0	15.6 / 16.0	16.0 / 12.0
link util. gk	1.0 / 1.0	2.1 / 2.0	3.1 / 1.5
link util. hk	6.8 / 7.0	13.9 / 14.0	16.0 / 10.5
link util. ji	3.7 / 4.0	7.2 / 8.0	1.1 / 6.0
link util. kj	3.7 / 4.0	7.2 / 8.0	1.1 / 6.0
penalty value	0.008	0.030	13.403

TABLE I: Result of quadratic program on three separate simulations where cells show estimated vs. real values

solvers [31, 32]. A linear programming problem is to find values for a vector of decision variables x that minimize a given cost function $c^T x$ subject to linear constraints $Ax \geq b$ and $x \geq 0$ for some constant vectors b, c and an integer matrix A. In quadratic programming, the cost function can include products of pairs of decision variables, in general: minimize $c^T x + 1/2 \cdot x^T Q x$ for some symmetric matrix Q. We refer to [33] for further introduction to linear and quadratic programming.

In order to describe the encoding of the FSR problem into LP, we need to introduce some notation. Let $G(f)_v^+ = \{(v,u) \in G(f)\}$ be the outgoing edges from the node v in $G(f)$ and let $G(f)_v^- = \{(u,v) \in G(f)\}$ be the incoming edges to v in $G(f)$. The variables of the optimization problem are x_e^f for every flow f and every link e. The value of the variable x_e^f, $0 \leq x_e^f \leq 1$, expresses the fraction of the traffic of the flow f that is traversing the link e. From the x_e^f variables, we can derive the flow splitting ratios as follows

$$d_v^f(u) = \frac{x_{(v,u)}^f}{\sum_{e \in G(f)_v^+} x_e^f}$$

where $d_v^f(u)$ expresses the flow splitting ratio at node v for the next-hop u. The linearly constrained optimization program is then:

$$\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{e \in E} penalty\big(U(e),\ \sum_{f \in V \times V} F(f) \cdot x_e^f\big) \\
\text{subject to} \quad & \forall f \in V \times V : \quad (\text{let } f = (s,t)) \\
(1) \quad & x_e^f \geq 0 \quad \forall e \in E \\
(2) \quad & \textstyle\sum_{e \in G(f)_s^+} x_e^f = 1 \\
(3) \quad & \textstyle\sum_{e \in G(f)_v^-} x_e^f = \sum_{e \in G(f)_v^+} x_e^f \quad \forall v \in V \setminus \{s,t\}
\end{aligned}$$

Here we minimize the cost function defined in Section II-B and require that (1) the variables a non-negative, (2) the source of the flow initiates all the traffic, and (3) each intermediate router in the flow graph sends out as much traffic as it receives. We do not need a constraint requiring that the target node t receives all the traffic of the flow f, as it is the only sink node in the subgraph $G(f)$, and the constraints (2) and (3) imply that all traffic of f ends in t.

Remark 2: If for some flow f an edge $e = (v,u)$ is the only outgoing edge from v in the subgraph $G(f)$, there is no need to introduce the variable x_e^f as it is redundant.

In order to express the penalty function from the case study (Equation 1), we introduce variables err_e for each link e and rewrite the quadratic program as:

$$\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{e \in E} (err_e)^2 \\
\text{subject to} \quad & (1) \text{ - } (3) \text{ and } \forall e \in E : \\
(4) \quad & err_e \cdot U(e) \geq \textstyle\sum_{f \in V \times V} F(f) \cdot x_e^f - U(e) - c
\end{aligned}$$

where the optimal value of the variable err_e is the positive relative error of the estimation after discounting the acceptable absolute error c. Note that by using an inequality in the constraint (4), we only penalize over-estimation.

Figure 2 shows the quadratic program for our running example. Here, in case of no measurement noise, the optimal zero-cost solution of the linear program gives us exactly the correct splitting ratios from Figure 1c.

B. Measurement Noise

Next, returning to our running example from Figure 1, we synthetically add measurement noise that can vary the size of the measured flow demands. We do this in order to simulate the noise seen in a real network analytics deployment. There is always a small noise variation that reflects the timing and sampling variance, while the large (but less frequent) differences can be caused by late detection of changes in the BGP tables leading to incorrect mapping of ingress-traffic to the flows inside the network.

Table I shows the results of three experiments with increasing levels of measurement noise on the first flow (+5%, +10%, +200%), while the second flow has the same small noise (-5%) for all three simulated time windows (t=1,2,3). The measured value is the left number in the cell, and the actual value the right number. Note that, like in real networks, the amount of traffic changes between the time windows. We use the quadratic programming solver CPLEX 22.1 [31] to solve the programs, and report the computed vs. ideal (real) splitting ratios, as well as the forward projected traffic derived from the computed ratios vs. actual utilization on each link based on the real ratios.

As we can see in Table I, the estimated flow splitting ratios are quite accurate when there is only little noise; however, in the last case (t=3) with a large measurement error for the flow $a \to k$, the estimated ratios are quite far off. This is even the case for the splitting ratios of the other flow $b \to i$.

	combining time series			*filtering*		
	t=1	t=2	t=3	t=1	t=2	t=3
flow size $a \to k$ (measured / real)	4.2 / 4.0	8.8 / 8.0	18.0 / 6.0	4.2 / 4.0	8.8 / 8.0	18.0 / 6.0
flow size $b \to i$ (measured / real)	11.4 / 12.0	22.8 / 24.0	17.1 / 18.0	11.4 / 12.0	22.8 / 24.0	17.1 / 18.0
ratio $d_a^{a\to k}(b)$ (estimated / real)		64% / 50%			53% / 50%	
ratio $d_a^{a\to k}(d)$ (estimated / real)		36% / 50%			47% / 50%	
ratio $d_e^{a\to k}(g)$ (estimated / real)		51% / 50%			49% / 50%	
ratio $d_e^{a\to k}(h)$ (estimated / real)		49% / 50%			51% / 50%	
ratio $d_b^{b\to i}(c)$ (estimated / real)		18% / 33%			33% / 33%	
ratio $d_b^{b\to i}(d)$ (estimated / real)		82% / 67%			67% / 67%	
mean error		10.01%			1.41%	
max error		15.41%			3.16%	
link util. ab (estimated / real)	2.7 / 2.0	5.6 / 4.0	11.5 / 3.0	2.2 / 2.0	4.7 / 4.0	9.6 / 3.0
link util. ad (estimated / real)	1.5 / 2.0	3.2 / 4.0	6.5 / 3.0	2.0 / 2.0	4.1 / 4.0	8.4 / 3.0
link util. bc (estimated / real)	4.7 / 6.0	9.7 / 12.0	14.6 / 9.0	6.0 / 6.0	12.3 / 12.0	15.3 / 9.0
link util. bd (estimated / real)	9.4 / 8.0	18.7 / 16.0	14.0 / 12.0	7.6 / 8.0	15.2 / 16.0	11.4 / 12.0
link util. ch (estimated / real)	4.7 / 6.0	9.7 / 12.0	14.6 / 9.0	6.0 / 6.0	12.3 / 12.0	15.3 / 9.0
link util. de (estimated / real)	10.9 / 10.0	21.9 / 20.0	20.5 / 15.0	9.6 / 10.0	19.3 / 20.0	19.8 / 15.0
link util. eg (estimated / real)	10.1 / 9.0	20.3 / 18.0	17.3 / 13.5	8.5 / 9.0	17.2 / 18.0	15.5 / 13.5
link util. eh (estimated / real)	0.7 / 1.0	1.6 / 2.0	3.2 / 1.5	1.0 / 1.0	2.1 / 2.0	4.3 / 1.5
link util. fi (estimated / real)	9.4 / 8.0	18.7 / 16.0	14.0 / 12.0	7.6 / 8.0	15.2 / 16.0	11.4 / 12.0
link util. gf (estimated / real)	9.4 / 8.0	18.7 / 16.0	14.0 / 12.0	7.6 / 8.0	15.2 / 16.0	11.4 / 12.0
link util. gk (estimated / real)	0.8 / 1.0	1.6 / 2.0	3.3 / 1.5	1.0 / 1.0	2.0 / 2.0	4.1 / 1.5
link util. hk (estimated / real)	5.5 / 7.0	11.3 / 14.0	17.8 / 10.5	7.1 / 7.0	14.4 / 14.0	19.6 / 10.5
link util. ji (estimated / real)	2.0 / 4.0	4.1 / 8.0	3.1 / 6.0	3.8 / 4.0	7.6 / 8.0	5.7 / 6.0
link util. kj (estimated / real)	2.0 / 4.0	4.1 / 8.0	3.1 / 6.0	3.8 / 4.0	7.6 / 8.0	5.7 / 6.0
penalty value		14.162			0.042	

TABLE II: Results of combining the time windows of Table I and filtering out the 20% worst constraints (grey cells).

IV. Dealing with Measurement Noise

To achieve stable and accurate estimations of the flow splitting ratios despite the noise and occasional large errors in the measurement of the size of the flows, we propose two techniques.

First, by *combining time series* of measurements into a single large quadratic program, we exploit that we have data for multiple time intervals (e.g. 24 one-hour measurements of a day) for which the flow splitting ratios are expected to remain (mostly) unchanged.

Second, by *filtering* out the link error constraints with the highest penalty in the optimization function, we can indirectly filter out the flows with large (but rare) measurement errors. The intuition is that when only a few flows have large measurement errors, then only relatively few links will be affected. By not considering the utilization of these links for these specific time periods, we effectively filter out the large flow measurement errors without knowing specifically which flows were measured erroneously.

Table II shows the result of combining the three time windows from Table I of the running example into a single quadratic program. The left side of Table II shows the result without filtering, while the right side shows the results of filtering out the 20% link constraints (highlighted with grey background) that contribute the most to the total penalty value of the LP (in the optimal solution for the unfiltered problem).

We see that combining the three time windows reduces the mean error in estimation of splitting ratios from 14.36% for the worst case (t=3) to 10.01% in the combined problem without filtering. After filtering, the mean error is only 1.41%. This small example shows a strong benefit of combining time series of measurements and filtering out some constraints with high penalty; however, it contains only two flows and only one large measurement error. In order to statistically validate the benefits of our technique, we run an extensive simulation experiment on a large set of network topologies.

A. Simulation Experiments with Synthetic Traffic

We simulate synthetic flow demands and splitting ratios on real world topologies from the Topology Zoo benchmark [16]. We restrict topologies to their largest connected component (disconnected components can be handled independently) and we do not consider topologies with less than eight nodes or where the synthetic traffic encounters no splitting at all. This leaves us with 192 different topologies.

To generate synthetic traffic, we use the gravity model [34] with random node masses and randomly select 25% of all source-destination pairs to have traffic between them—this corresponds to the numbers found in our industrial case study. As an approximate simulation of the variation of traffic during the day, we vary the total traffic in the network over time using a sine wave together with added noise. We generate 24 traffic matrices, corresponding to one for each hour of the day—a similar setup as the data source in the case study. The splitting ratios are generated by assigning random load balancing weights to the links of the graph and then computing ratios based on these link weights. These demands over time and the splitting ratios are the 'ground truth' of the simulation and are used to compute the true total utilization of each link.

To mimic the type of measurement noise found in a real network analytics deployment, we introduce a small random variation of $\pm 1\%$ to the measured traffic of all flows. We also model rare but large measurement errors in flow traffic: with a low probability of 0.5% we vary a flow size by a random factor between 1/10 and 10. From the estimated splitting ratios,

returned by the quadratic programing solver, we compute the error compared to the true splitting ratios, and average each of these errors weighted by the total size of the flow. This avoids small, and hence in practice less important, flows dominating and skewing the results. This weighted average splitting ratio error is then considered as the error of that solution.

For each topology, we create ten different random instances of splitting ratios and traffic demands and average the errors. We then report on the best filtering ratio and the error it achieved, and we compare this to the maximum error of a single time window, and to the error when combining time series without filtering. Table III orders the topologies by the improvement achieved by filtering compared to only combining time series, and shows the results for the top, middle and bottom seven topologies.

It is clear that combining measurements over multiple time windows balances out the measurement noise and reduces the estimation errors compared to the worst single time window. We can further observe that filtering improves the error in the estimation of the real splitting ratios by an order of magnitude in the best cases and in the middle cases it achieves a significant 66% improvement. In the worst seven topologies, our improvement is smaller (but the filtering technique still improves the precision in every single topology). We observe that topologies with smallest improvement have either close to no error in the first place or have a very large diameter of more than 20, where it is likely to create flows that traverse a large number of links in the network. In such cases, we need to filter up to 50% of LP constraints, which is significantly more than what is needed for the other topologies (where 5-10% filtering is sufficient). Hence if a large measurement error is introduced to such a long elephant flow, then there is simply not enough data on the remaining links in the network to accurately approximate the actual splitting ratios.

Topology	Diameter	Max error over all time windows	Error after combining time series	Error after filtering	Filtering percentage	Improvement of filtering
Nextgen	11	40.32%	3.56%	0.20%	5%	94.5%
Bbnplanet	8	56.16%	5.74%	0.43%	5%	92.5%
Psinet	12	50.46%	7.66%	0.64%	5%	91.7%
Goodnet	5	16.36%	2.40%	0.23%	5%	90.5%
Abvt	8	29.62%	4.95%	0.49%	5%	90.1%
Janetlense	5	11.72%	1.69%	0.17%	5%	90.1%
Ibm	7	29.93%	5.46%	0.54%	5%	90.1%
Geant2009	8	22.78%	8.87%	2.87%	10%	67.6%
Geant2001	7	21.48%	10.08%	3.28%	10%	67.5%
Easynet	7	25.87%	4.66%	1.53%	10%	67.1%
Compuserve	5	26.05%	0.82%	0.28%	5%	66.1%
Dfn	7	30.08%	11.41%	3.92%	10%	65.6%
Cesnet201006	7	33.70%	10.65%	3.68%	10%	65.5%
DT	7	19.14%	7.78%	2.71%	10%	65.1%
Ion	26	34.83%	21.13%	15.70%	50%	25.7%
TataNld	29	23.12%	17.05%	13.06%	50%	23.4%
GtsCe	22	28.58%	18.96%	14.59%	50%	23.1%
UsCarrier	36	36.68%	17.00%	13.69%	50%	19.4%
DialtelecomCz	31	34.62%	18.61%	15.17%	45%	18.5%
Gridnet	3	6.16%	0.12%	0.10%	15%	15.7%
Claranet	5	30.28%	0.20%	0.18%	5%	10.4%

TABLE III: Experiments with synthetic traffic data

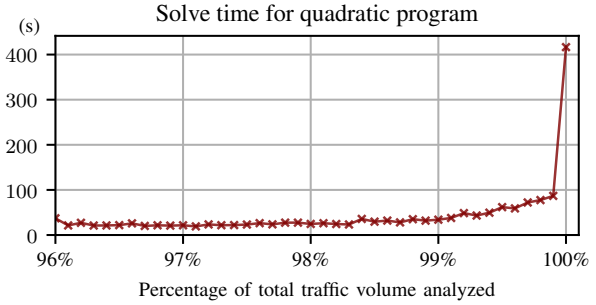


Fig. 3: Scalability of solving FSR on a real, large ISP

V. Scalability Study on Large European ISP

We perform a case study on data from a large European ISP. This network consists of over 3.000 routers and 14.000 links, and the dataset contains hourly traffic matrices, flow paths, and link utilizations for 24 hours of one day. The set of paths used by a flow is in most cases stable in the dataset, but some changes occur during the day. We handle this by assuming that the set of splitting ratios are stable as long as the set of paths is stable, but we introduce new variables for modelling a new set of paths for the flow.

Over the course of one day, more than two and a half million flows have traffic. The quadratic program that analyzes all flows is solved in about seven minutes running on 4 CPU cores at 2.5GHz; however, most of these flows have a very small volume, and in practice the largest flows are the most important. As seen in Figure 3, analyzing the flows that carry 99.9% of the total traffic volume per hour takes only 87 seconds, and analyzing 99% of the traffic volume takes 34 seconds. In conclusion, the method is highly scalable, especially considering the typically uneven distribution of traffic volume in ISP networks.

The diameter of the topology in the case study is 10, so from the experiments on synthetic traffic data presented in Table III we conclude that a 5-10% filtering is expected to provide the largest error reduction. We compute flow splitting ratios using our approach, and observe that they closely match splitting ratios based on link capacities—an insight that is now used in the traffic analytics deployment.

VI. Conclusion

We suggested a method for synthesis of flow splitting ratios from incomplete and noisy network traffic flow measurements. Our methods is based on quadratic linear programming and we documented the accuracy and robustness of our method on an extensive synthetic benchmark. Our method is scalable even to large ISP networks. Based on the analysis by our tool on a case study in collaboration with a network analytics company, flow splitting ratios based on link capacities are now used to improve the accuracy of a real traffic analytics deployment.

Acknowledgement: We thank the anonymous reviewers for their comments, and the DFF project QASNET for supporting this research.

REFERENCES

[1] B. Li, J. Springer, G. Bebis, and M. Hadi Gunes, "A survey of network flow applications," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 567–581, 2013. [Online]. Available: https://doi.org/10.1016/j.jnca.2012.12.020

[2] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014. [Online]. Available: https://doi.org/10.1109/COMST.2014.2321898

[3] J. L. Garcia-Dorado, A. Finamore, M. Mellia, M. Meo, and M. Munafo, "Characterization of ISP traffic: Trends, user habits, and access technology impact," *IEEE Transactions on Network and Service Management*, vol. 9, no. 2, pp. 142–155, 2012. [Online]. Available: https://doi.org/10.1109/TNSM.2012.022412.110184

[4] M. Trevisan, D. Giordano, I. Drago, M. Mellia, and M. Munafo, "Five years at the edge: Watching internet from the ISP network," in *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 1–12. [Online]. Available: https://doi.org/10.1145/3281411.3281433

[5] A. Feldmann, O. Gasser, F. Lichtblau, E. Pujol, I. Poese, C. Dietzel, D. Wagner, M. Wichtlhuber, J. Tapiador, N. Vallina-Rodriguez, O. Hohlfeld, and G. Smaragdakis, "The lockdown effect: Implications of the COVID-19 pandemic on internet traffic," in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–18. [Online]. Available: https://doi.org/10.1145/3419394.3423658

[6] B. Claise, "Cisco systems NetFlow services export version 9," RFC 3954, Oct. 2004. [Online]. Available: https://doi.org/10.17487/RFC3954

[7] P. Aitken, B. Claise, and B. Trammell, "Specification of the IP flow information export (IPFIX) protocol for the exchange of flow information," RFC 7011, Sep. 2013. [Online]. Available: https://doi.org/10.17487/RFC7011

[8] Y. Rekhter, S. Hares, and T. Li, "A border gateway protocol 4 (BGP-4)," RFC 4271, Jan. 2006. [Online]. Available: https://doi.org/10.17487/RFC4271

[9] ISO, "Intermediate system to intermediate system intra-domain routeing exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473)," ISO/IEC 10589:2002, Nov. 2002. [Online]. Available: https://www.iso.org/standard/30932.html

[10] R. Callon, "Use of OSI IS-IS for routing in TCP/IP and dual environments," RFC 1195, Dec. 1990. [Online]. Available: https://doi.org/10.17487/RFC1195

[11] J. Moy, "OSPF version 2," RFC 2328, Apr. 1998. [Online]. Available: https://doi.org/10.17487/RFC2328

[12] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 118–124, 2002. [Online]. Available: https://doi.org/10.1109/MCOM.2002.1039866

[13] J. Networks, "IS-IS user guide: Understanding weighted ECMP traffic distribution on one-hop IS-IS neighbors," jan 2021. [Online]. Available: https://www.juniper.net/documentation/us/en/software/junos/is-is/topics/concept/wecmp-for-one-hop-isis-neighbors-overview.html

[14] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "Simple network management protocol (SNMP)," RFC 1157, May 1990. [Online]. Available: https://doi.org/10.17487/RFC1157

[15] D. Harrington, B. Wijnen, and R. Presuhn, "An architecture for describing simple network management protocol (SNMP) management frameworks," RFC 3411, Dec. 2002. [Online]. Available: https://doi.org/10.17487/RFC3411

[16] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology Zoo," *IEEE Journal on Selected Areas in Comm.*, vol. 29, no. 9, pp. 1765 –1775, 2011.

[17] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven WAN," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 15–26. [Online]. Available: https://doi.org/10.1145/2486001.2486012

[18] Y. Wang, H. Wang, A. Mahimkar, R. Alimi, Y. Zhang, L. Qiu, and Y. R. Yang, "R3: Resilient routing reconfiguration," in *Proceedings of the ACM SIGCOMM 2010 Conference*, ser. SIGCOMM '10. New York, NY, USA: Association for Computing Machinery, 2010, pp. 291–302. [Online]. Available: https://doi.org/10.1145/1851182.1851218

[19] M. Suchara, D. Xu, R. Doverspike, D. Johnson, and J. Rexford, "Network architecture for joint failure recovery and traffic engineering," *SIGMETRICS Perform. Eval. Rev.*, vol. 39, no. 1, pp. 97–108, jun 2011. [Online]. Available: https://doi.org/10.1145/2007116.2007128

[20] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter, "Traffic engineering with forward fault correction," in *Proc. ACM SIGCOMM*, 2014, pp. 527–538.

[21] C. Jiang, S. Rao, and M. Tawarmalani, "PCF: Provably resilient flexible routing," in *Proc. ACM SIGCOMM*, 2020, pp. 139–153.

[22] Y. Chang, C. Jiang, A. Chandra, S. Rao, and M. Tawarmalani, "Lancet: Better network resilience by designing for pruned failure sets," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 3, dec 2019. [Online]. Available: https://doi.org/10.1145/3366697

[23] J. Bogle, N. Bhatia, M. Ghobadi, I. Menache, N. Bjørner, A. Valadarsky, and M. Schapira, "TEAVAR: Striking the right utilization-availability balance in wan traffic engineering," in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 29–43. [Online]. Available: https://doi.org/10.1145/3341302.3342069

[24] C. Jiang, Z. Li, S. Rao, and M. Tawarmalani, "Flexile: Meeting bandwidth objectives almost always," in *Proc. ACM CoNEXT*, 2022, pp. 110–125.

[25] K. Lakkaraju, W. Yurcik, and A. J. Lee, "NVisionIP: Netflow visualizations of system state for security situational awareness," in *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, ser. VizSEC/DMSEC '04. New York, NY, USA: Association for Computing Machinery, 2004, pp. 65–72. [Online]. Available: https://doi.org/10.1145/1029208.1029219

[26] S. J. Saidi, A. Maghsoudlou, D. Foucard, G. Smaragdakis, I. Poese, and A. Feldmann, "Exploring network-wide flow data with Flowyager," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 1988–2006, 2020. [Online]. Available: https://doi.org/10.1109/TNSM.2020.3034278

[27] T. Tremel, J. Kögel, F. Jauernig, S. Meier, D. Thom, F. Becker, C. Müller, and S. Koch, "VITALflow: Visual interactive traffic analysis with netflow," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, 2022, pp. 1–6. [Online]. Available: https://doi.org/10.1109/NOMS54207.2022.9789776

[28] D. Phan, J. Gerth, M. Lee, A. Paepcke, and T. Winograd, *Visual Analysis of Network Flow Data with Timelines and Event Plots*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 85–99. [Online]. Available: https://doi.org/10.1007/978-3-540-78243-8_6

[29] J. R. Goodall and D. R. Tesone, "Visual analytics for network flow analysis," in *2009 Cybersecurity Applications & Technology Conference for Homeland Security*, 2009, pp. 199–204. [Online]. Available: https://doi.org/10.1109/CATCH.2009.47

[30] C. Hopps, "Analysis of an equal-cost multi-path algorithm," RFC 2992, Nov. 2000. [Online]. Available: https://doi.org/10.17487/RFC2992

[31] IBM, "IBM ILOG CPLEX optimization studio 22.1.0." [Online]. Available: https://www.ibm.com/docs/en/icos/22.1.0

[32] Gurobi Optimization, *Gurobi optimizer reference manual - Version 10.0*, 2023. [Online]. Available: https://www.gurobi.com/wp-content/plugins/hd_documentations/documentation/10.0/refman.pdf

[33] R. J. Vanderbei, *Linear programming: Foundations and Extensions*, 5th ed. Springer, 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-39415-8

[34] M. Roughan, "Simplifying the synthesis of internet traffic matrices," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 93–96, oct 2005. [Online]. Available: https://doi.org/10.1145/1096536.1096551