Measurement-Noise Filtering for Automatic Discovery of Flow Splitting Ratios in ISP Networks

MORTEN KONGGAARD SCHOU, Computer Science, Aalborg University, Aalborg, Denmark
INGMAR POESE, BENOCS GmbH, Berlin, Germany
JIRI SRBA, Computer Science, Aalborg University, Aalborg, Denmark

Network telemetry and analytics is essential for providing highly dependable services in modern computer networks. In particular, network flow analytics for internet service provider (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, for example, 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 achieves robustness to the measurement noise found in a typical network analytics deployment by filtering out certain constraints in the linear program. Finally, we implement an automated tool for estimating the flow splitting ratios and document its applicability on real data from the ISP.

CCS Concepts: • **Networks** → **Network measurement**; **Network monitoring**; *Network simulations*; • **Mathematics of computing** → *Linear programming*; *Quadratic programming;*

Additional Key Words and Phrases: Network analytics, network traffic flow, load balancing

ACM Reference Format:
Morten Konggaard Schou, Ingmar Poese, and Jiri Srba. 2024. Measurement-Noise Filtering for Automatic Discovery of Flow Splitting Ratios in ISP Networks. *Form. Asp. Comput.* 36, 4, Article 27 (December 2024), 18 pages. https://doi.org/10.1145/3700602

1 Introduction

Network flow analytics [18, 27] in internet service provider (ISP) networks is often employed by network operators for monitoring the traffic patterns [10, 12, 39]. This can help optimize overall

This work was supported by the DFF project QASNET and by the S4OS Villum Investigator Grant (no. 37819) from Villum Fonden.
Author's Contact Information: Morten Konggaard Schou, Computer Science, Aalborg University, Aalborg, Denmark; e-mail: mksc@cs.aau.dk; Ingmar Poese, BENOCS GmbH, Berlin, Germany; e-mail: ipoese@benocs.com; Jiri Srba, Computer Science, Aalborg University, Aalborg, Denmark; e-mail: srba@cs.aau.dk.

ACM 1433-299X/2024/12-ART27
https://doi.org/10.1145/3700602

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. However, by packet sampling (using, e.g., NetFlow [7] or IPFIX [1]), a network operator is capable of estimating, with 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. Yet, 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, to determine from where the anomaly originates.

In this article, 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 more than 6,000 routers across multiple ISPs.

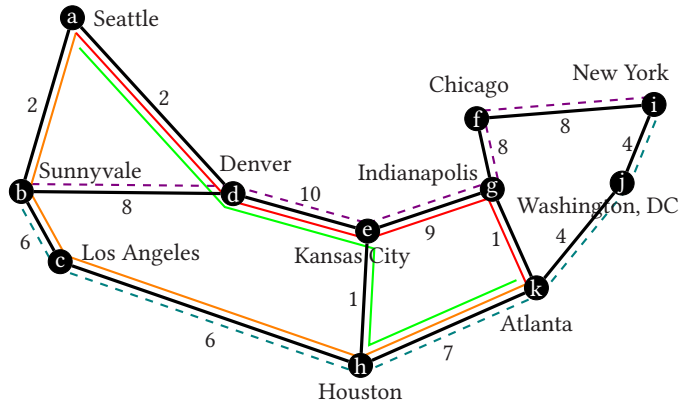
Sampling packet headers on every link in the network can answer such questions; however, it has severe scalability issues. Instead, current high-performing network analytics systems sample packet headers at the ingress routers only and combine this with the information from the border gateway protocol (BGP) [32] and the interior gateway protocol (IGP) (e.g., IS-IS [4, 22] or OSPF [29]) 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 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. Note that this definition of a flow—from the point of view of a single network—differs from the definition based on the source and destination IP addresses, which is typically used in IP networks when discussing packet flows across the internet.

Figure 1(a) shows an example network where links are annotated by the current link utilizations, and Figure 1(b) depicts two flows of demand, 12 and 4, respectively. To better distribute traffic along the links and thus reduce the maximum link utilization [11], flows can be split along multiple paths as demonstrated in the last column in Figure 1(b). The splitting ratios can be uniform among the available paths, or they can depend on the link capacities [30] or custom link weights as shown in Figure 1(c).

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 deems it impossible in practice to obtain the flow splitting ratios directly from the router. As it is, moreover, infeasible to sample the packets traversing each link and categorize them into the corresponding flows, we need additional information 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 [5, 15], then the link utilization for a given time interval is estimated by linear interpolation between SNMP measurements. In our example from Figure 1(a), each link is annotated with the current link utilization. We now want to solve the following Flow Splitting Ratios problem (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.



(a) Abilene network from the Internet Topology Zoo [25]

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 the Abilene network and their paths

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

(c) Splitting ratios for the two flows

Fig. 1. Example network topology with link utilizations

As our main contribution, we provide a practical and efficient solution to the FSR problem, employing quadratic linear programming (LP). 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 1(c)) 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 [25], as well as on real traffic data from a major European ISP (in collaboration with the network analytics company BENOCS [2]).

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 more than 200 real-world ISP topologies, we conclude that our filtering technique helps improve the precision by an order of magnitude in the best cases and achieves about 48% 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 more than 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 BENOCS).

Related Work. LP has been used to solve the classical multi-commodity flow problem [8], and variations of this is used for network traffic engineering to synthesize optimal splitting ratios [19] and balancing throughput and fairness [9]. The work of Hartman et al. [16] takes the LP solution of a flow problem and finds a small set of paths that implements the traffic flow. Further variations of the linear programs consider the traffic shifts caused by failure recovery [3, 6, 23, 24, 28, 37, 41] to synthesize configurations that achieve congestion-free resilience. We, however, 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 works that use linear objective functions, we employ quadratic optimization that allows us to minimize squared relative errors, which is better suited for dealing with the data inaccuracies found in this application domain.

From the network monitoring research, network traffic analysis and visualization tools like NVisionIP [26], Flowyager [34], and VITALflow [38] have been designed for the purpose of network security [13, 26, 31] and management [34, 38]. 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.

This article is an extension of a conference paper [35]. The main new contributions are the extended experiments in Section 4.2, where we investigate how much filtering to apply with our novel method, and the simulation experiments on traffic data from a real ISP network presented in Section 5.1. Moreover, we expand the explanations of the theory.

2 Problem Formalization

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

2.1 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 the ingress and egress routers, 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 $p \in P(f)$ that contains uv as a subpath. When the network computes the set of paths using ECMP (equal-cost multi-path) routing [20], 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.

2.2 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:

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

However, due to the inaccuracies of data introduced, for example, by 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:

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

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

Remark 1. *In practice, there is a large variety in the size and utilization of links across a network, so $\textit{penalty}_{abs}$ tends to overfit the large links. Using relative errors alleviates this problem, and squaring the error, like $\textit{penalty}_{rel2}$ 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 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:

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

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

3 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.

3.1 Encoding of the FSR Problem to a Linear Program

LP and quadratic programming are well-studied problems with several industry-standard solvers [14, 21]. An LP 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 the textbook by Vanderbei [40] for further introduction to linear and quadratic programming.

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^f_e for every flow f and every link e. The value of the variable x^f_e, $0 \leq x^f_e \leq 1$, expresses the fraction of the traffic of the flow f that is traversing the link e. From the x^f_e variables, we can derive the flow splitting ratios as follows:

$$d^f_v(u) = \frac{x^f_{(v,u)}}{\sum_{e \in G(f)^+_v} x^f_e},$$

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

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

Here we minimize the cost function defined in Section 2.2 and require that (1) the variables are non-negative, (2) the source of the flow initiates all of 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 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^f_e, as it is redundant.*

As an example, consider the link (d,e) in Figure 1(a). Here the flow preservation constraints (3) imply that $x^{b\to i}_{(b,d)} = x^{b\to i}_{(d,e)}$ and $x^{a\to k}_{(a,d)} = x^{a\to k}_{(d,e)}$. Instead of explicitly adding these constraints to the LP, we directly use the variables $x^{b\to i}_{(b,d)}$ and $x^{a\to k}_{(a,d)}$ in place of $x^{b\to i}_{(d,e)}$ and $x^{a\to k}_{(d,e)}$, respectively.

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 follows:

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 → New York ($b \to i$))
(1) $x_{bc}^{b\to i} + x_{bd}^{b\to i} = 1$
(for the flow Seattle → 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.

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

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 1(c).

3.2 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 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, whereas 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 1 shows the results of three experiments with increasing levels of measurement noise on the first flow (+5%, +10%, +200%), whereas 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 [21] 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 1, 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 $a \to k$,

Table 1. Result of the quadratic program on three separate simulations where cells show estimated vs. real values

	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

the estimated ratios are quite far off. This is even the case for the splitting ratios of the other flow $b \to i$.

4 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.

Formally, given the set of time windows T, we now, for each time window $t \in T$, have data on the link utilization $U_t : E \to \mathbb{N}$ and traffic matrix $F_t : V \times V \to \mathbb{N}$. The quadratic program introduces variables $err_{t,e}$ for each time window t and link e, and it now contains link constraints for each time window:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{t\in T, e\in E}(err_{t,e})^2 \\ \text{subject to} \quad & \text{(1) - (3) and } \forall t \in T, \forall e \in E : \\ (4) \quad & err_{t,e} \cdot U_t(e) \geq \textstyle\sum_{f\in V\times V} F_t(f) \cdot x_e^f - U_t(e) - c. \end{aligned}$$

Table 2. Results of combining the time windows of Table 1 and filtering out the 20% worst constraints (gray cells)

	combining time series			*filtering*		
	t=1	t=2	t=3	t=1	t=2	t=3
flow size $a \rightarrow k$ (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 $b \rightarrow i$ (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 $d_a^{a\rightarrow k}(b)$ (estimated / real)	64% / 50%			53% / 50%		
ratio $d_a^{a\rightarrow k}(d)$ (estimated / real)	36% / 50%			47% / 50%		
ratio $d_e^{a\rightarrow k}(g)$ (estimated / real)	51% / 50%			49% / 50%		
ratio $d_e^{a\rightarrow k}(h)$ (estimated / real)	49% / 50%			51% / 50%		
ratio $d_b^{b\rightarrow i}(c)$ (estimated / real)	18% / 33%			33% / 33%		
ratio $d_b^{b\rightarrow i}(d)$ (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		

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 2 shows the result of combining the three time windows from Table 1 of the running example into a single quadratic program. The left side of Table 2 shows the result without filtering, whereas the right side shows the results of filtering out the 20% link constraints (highlighted with gray 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. To statistically validate the benefits of our technique, we run an extensive simulation experiment on a large set of network topologies.

4.1 Simulation Experiments with Synthetic Traffic

We simulate synthetic flow demands and splitting ratios on real-world topologies from the Topology Zoo benchmark set [25]. All code and experimental data for these experiments are

made available [36]. We restrict topologies to their largest connected component (disconnected components can be handled independently), and we do not consider topologies with fewer than eight nodes or where the synthetic traffic encounters no splitting at all. This leaves us with 201 different topologies.

To generate synthetic traffic, we use the gravity model [33] 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 presented in Section 5. 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 ±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 programming solver, we compute the error compared to the true splitting ratios and then 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 10 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 3 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 in most cases it 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; in the middle cases, it achieves a significant 48% improvement. In the worst seven topologies, our improvement is smaller, yet still above 20% in all but the worst topology, where the filtering technique nonetheless improves the precision a bit. We observe that most of the topologies with the smallest improvement are comparatively large (number of nodes and edges) and 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 more than 50% of the LP constraints. Hence, if a large measurement error is introduced to such a long elephant flow, then there is less data on the remaining links in the network, which makes it harder to accurately approximate the actual splitting ratios.

4.2 Predicting the Optimal Filtering Ratio

From the initial simulation experiments in the previous section, we observe a large variation in the best filtering ratio for the different network topologies. Table 3 also hints at a tendency that networks with many nodes and edges and a large diameter (the shortest distance between the nodes that are the farthest apart) require more filtering—with the intuition being that in these networks, an erroneous flow measurement tends to impact more links, as the length of the flow's paths are longer.

Table 3. Experiments with synthetic traffic data show the estimation error of applying the solution separately to the data in each time window and compare it to the estimation error when combining data from all time windows without filtering and with the best filtering, respectively. The table shows the top, middle, and bottom seven topologies ordered by the improvement of filtering

Topology	Nodes	Edges	Diameter	Max error over all time windows	Error after combining time series	Error after filtering	Filtering percentage	Improvement of filtering
Fccn	23	25	6	38.77%	4.32%	0.19%	8%	95.5%
KentmanJul2005	16	17	7	51.49%	5.12%	0.25%	7%	95.0%
Eunetworks	14	16	7	44.67%	7.63%	0.46%	9%	93.9%
Nsfcnet	9	10	5	40.27%	2.74%	0.25%	6%	90.8%
TLex	12	13	4	41.44%	5.00%	0.48%	6%	90.4%
Nsfnet	13	15	6	38.06%	5.77%	0.58%	10%	90.0%
York	23	24	11	67.26%	7.31%	0.76%	18%	89.6%
Missouri	67	83	15	36.14%	26.47%	13.73%	79%	48.1%
Janetbackbone	29	45	6	26.30%	18.40%	9.55%	16%	48.1%
Uninett2011	69	96	10	24.73%	29.60%	15.37%	64%	48.1%
Columbus	70	85	19	36.54%	30.68%	15.96%	83%	48.0%
Cesnet200304	29	33	7	24.26%	21.66%	11.32%	87%	47.8%
KentmanAug2005	27	29	9	38.76%	20.58%	10.76%	16%	47.7%
Garr201004	54	68	8	22.46%	28.41%	14.86%	38%	47.7%
VtlWavenet2008	88	92	32	42.58%	34.75%	25.58%	91%	26.4%
Renater2008	33	43	8	34.38%	21.95%	16.53%	15%	24.7%
Colt	153	177	21	34.68%	16.45%	12.41%	53%	24.6%
Cogentco	197	243	29	25.40%	19.47%	14.78%	62%	24.1%
Ion	125	146	26	35.77%	20.35%	15.77%	66%	22.5%
TataNld	145	186	29	23.74%	15.80%	12.47%	66%	21.1%
DialtelecomCz	138	151	31	33.79%	17.40%	16.68%	71%	4.1%

In this section, we will conduct a closer study of the connection between parameters of the network, such as its size and diameter, and the filtering ratio that gives the best estimation of splitting ratios. To also investigate the impact of the amount of noise in the measurements, we run experiments where the parameters for simulating noise and errors in the measurements are varied by scaling the parameters used in the previous experiments up (respectively, down) by a factor of 2.

These experiments give for each network topology and noise level the estimation error for each filtering ratio from 0% to 99%. Combining this with parameters of the network topology, particularly the number of nodes and edges as well as the diameter of the network, we build a decision tree to predict the best filtering ratio given the noise level and network topology parameters. We reserve (the alphabetically first) 1/5 of the network topologies as a test set and use the rest of the networks to train a decision tree using the tool RapidMiner [17] with maximal depth 5 and minimal gain 0.01 using the least square criterion. The decision tree is depicted in Figure 3. We start at the root of the tree, and by querying the network characteristics (number of nodes and edges and network diameter) as well as the expected noise level, we arrive at a predicted filtering ratio that we recommend to use for a given network instance.

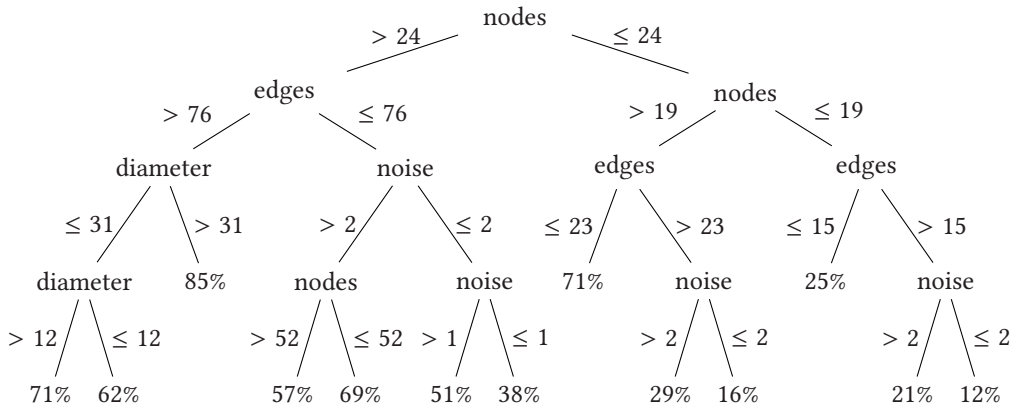


Fig. 3. Decision tree for predicting filtering ratios based on number of nodes, edges, diameter and noise level, where low noise = 1, medium = 2, and high = 3.

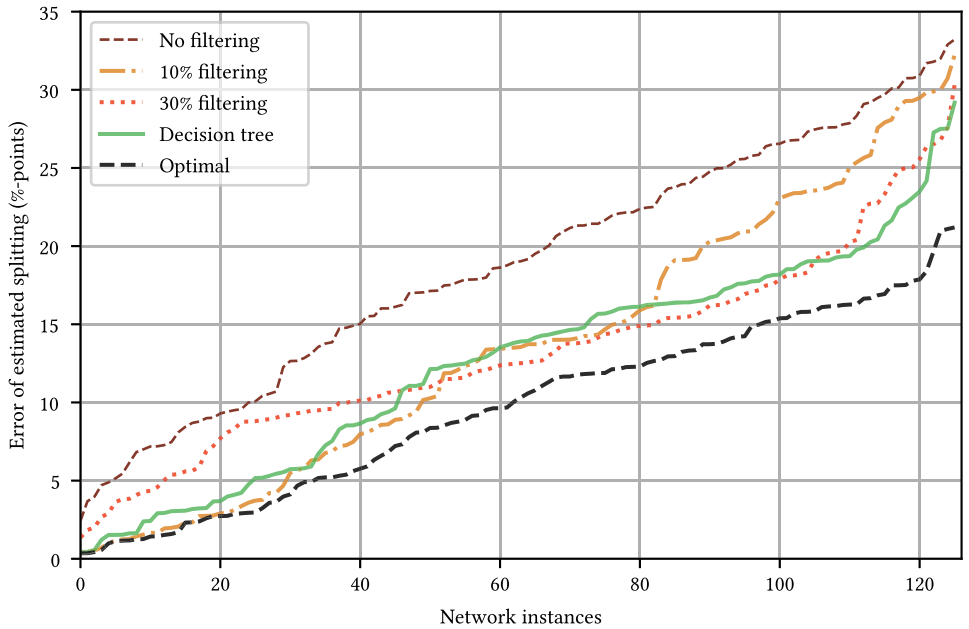


Fig. 4. Error in estimating the flow splitting ratios for various networks on the x-axis sorted by increasing value on the y-axis individually for each approach. The plot compares the filtering ratio from the decision tree with constant ratios as well as the optimal filtering for each instance—found by brute force.

In our experimental evaluation, we use the decision tree to compute suggested filtering ratios for networks in the test set, and compare the estimation error of the suggestion to that of the optimal filtering ratio. We also compare to the estimation error with no filtering and with fixed filtering ratios of 10% and 30% in Figure 4. The x-axis of each line in the figure is sorted by increasing estimation error on the y-axis. We first note that filtering significantly lowers the estimation error compared to not filtering the constraints of the linear program. In particular, 10% filtering works well for the networks with low estimation error on the left side of the plot, whereas 30% filtering

performs better for networks with higher estimation errors. The filtering percentage suggested by the decision tree matches the performance of 10% filtering for the networks with low estimation error on the left side of the plot, and the decision tree matches the performance of 30% filtering for the right side of the plot. Hence, the decision tree manages to find good filtering ratios based on the network parameters, which is preferable in case there is no further information about the noise in the network. If it is possible to estimate the noise in the traffic data by other means (e.g., by increasing the monitoring and analysis at the cost of scalability), this could be used to determine a good filtering ratio instead of using the decision tree.

To get a more in-depth view of the performance of various filtering ratios and the suggestion from the decision tree, we look at eight network topologies in the test benchmark that have the largest absolute improvement of the optimal filtering—that is, the networks with the largest potential for improvement, if a good filtering ratio is chosen. Figure 5 shows, for the three different levels of noise, the estimation error of each filtering ratio from 0% to 99%, and it marks the optimal filtering ratio as well as the ratio suggested by the decision tree in Figure 3.

We notice that in the cases where there is a small interval of good filtering ratios—notably Airtel and Belnet2004—the decision tree manages to suggest a filtering ratio close to the optimal. For other networks, where there is a larger range of good filtering ratios, the decision tree suggests filtering ratios that result in estimation errors close to the optimal.

The Biznet network is an exception, where the suggestion from the decision tree is less good. Looking closer at this network topology, we find that it has a long and narrow structure, which means that many flows will traverse a large proportion of the links in the network. So in cases where the measurement errors happen at these long flows, the filtering ratio needs to be higher than in the cases where the measurement errors happen for flows traversing fewer links. Similarly, the variation in estimation error across the 10 experiments is also higher for Biznet. This variation, caused by the random measurement noise, means that it is inherently hard to predict a good filtering ratio for this network based on the parameters used for building the decision tree.

5 Case Study on a Large European ISP

We perform a case study on data from a large European ISP. This network consists of more than 3,000 routers and 14,000 links, and the dataset contains hourly traffic matrices, flow paths, and link utilizations for 24 hours of 1 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 modeling a new set of paths for the flow.

Over the course of 1 day, more than 2.5 million flows have traffic. The quadratic program that analyzes all flows is solved in about 7 minutes running on four CPU cores at 2.5 GHz; however, most of these flows have a very small volume, and in practice, the largest flows are the most important. As seen in Figure 6, 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.

5.1 Simulation Experiment on Real Traffic Data

To evaluate the accuracy of our estimation of flow splitting ratios on the real ISP network, we perform simulation experiments where the measured traffic flow demands in the dataset are used as the true values, and we simulate measurement noise on top of this and randomly choose the (unknown) splitting ratios—same as for the experiments in Section 4.1. For scalability reasons, we restrict the dataset to the large flows that in total constitute 99% of the total traffic volume.

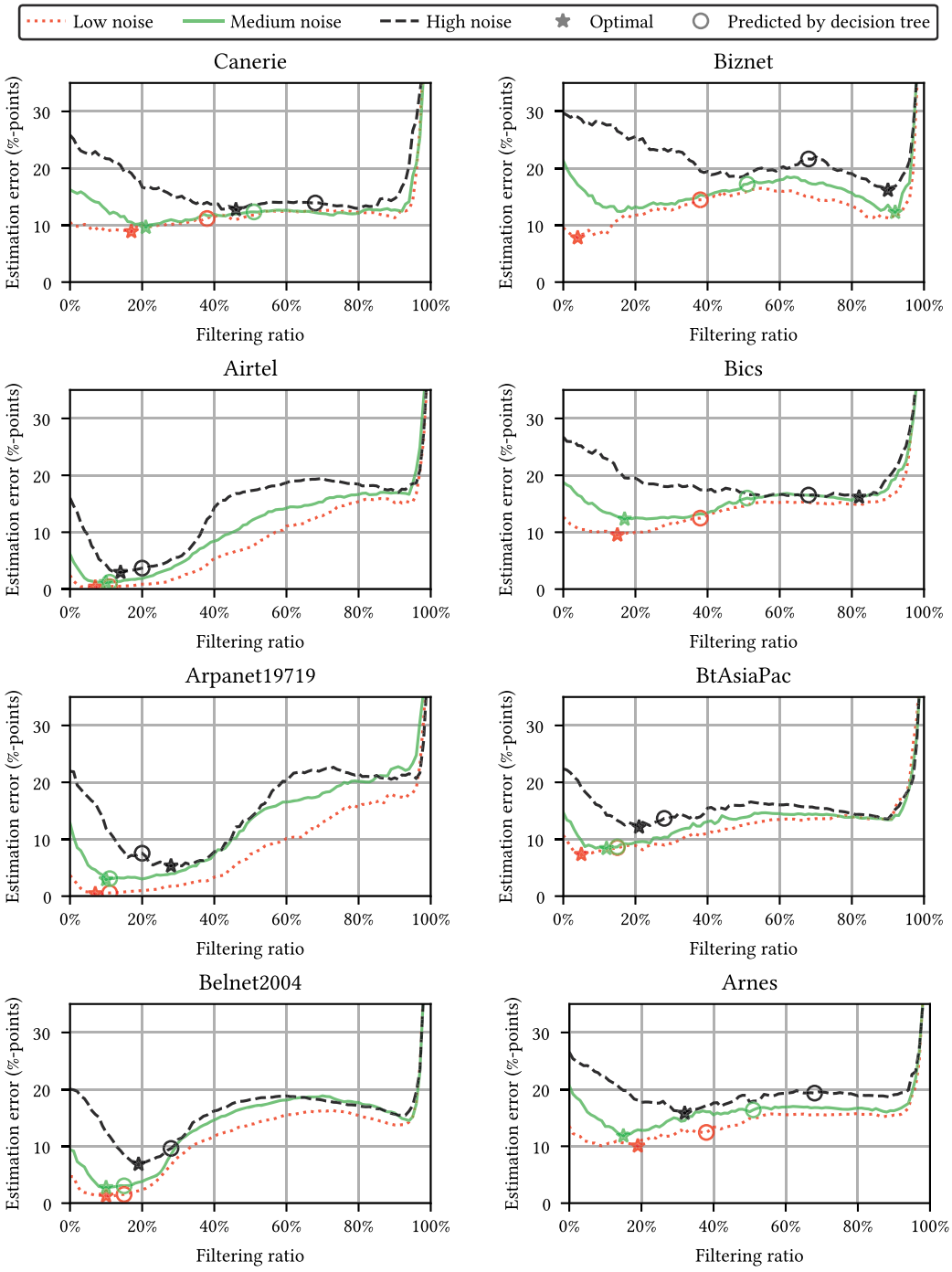


Fig. 5. Estimation error for different filtering ratios on eight networks where filtering makes the largest improvement. The plots show three different levels of simulated noise. The circle marks the suggested filtering ratio from the decision tree, whereas the star marks the optimal filtering ratio for that network and noise level.

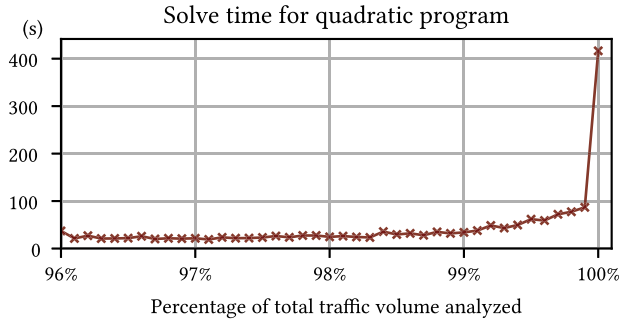


Fig. 6. Scalability of solving the FSR problem on a real, large ISP.

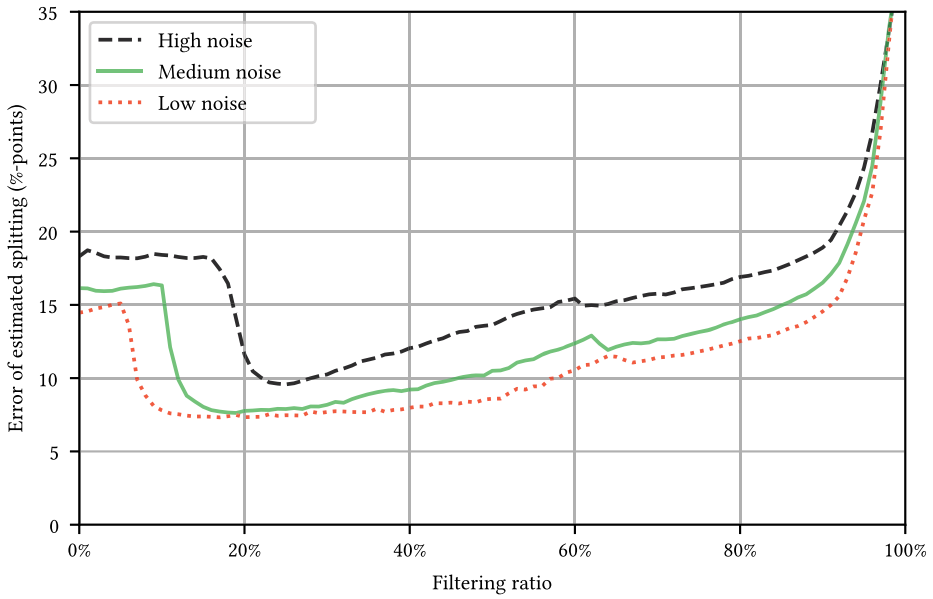


Fig. 7. Error in estimating the flow splitting ratios using different filtering ratios for the traffic data from a large ISP network on which three different levels of measurement noise are simulated.

We run these experiments with three different levels of noise and evaluate the estimated flow splitting ratios when using filtering ratios from 0% to 99%—taking the average over 10 runs. The results, presented in Figure 7, show that when there is more noise in the measurements, more of the constraints in the quadratic program need to be filtered out, and that a filtering ratio of around 25% gives good results in all three cases with an average estimation error of less than 10%-points.

The diameter of the topology in the case study is 10, so the suggestion from the decision tree in Figure 3 is to use a filtering ratio of 62%. The reason this suggestion is further away from the optimal one is that the decision tree was trained on much smaller network topologies (fewer than 200 routers), where the larger of these topologies typically have diameters greater than 10, so the networks in the training set are too different from the our real ISP network to give transferable suggestions for the filtering ratio. Nevertheless, the predicted filtering ratio still makes significant improvements compared to the situation with no filtering, and the error in estimation is at most 5%-points worse than the optimal filtering ratio. To increase the precision of filtering predictions, we

can alternatively learn a decision tree for the specific topology with varying traffic characteristics (e.g., based on the historical data) and using statistics about the causes of measurement errors, gathered by network monitoring, to estimate the noise level.

In our concrete case study, we computed the estimation of the splitting ratios over different time intervals and observed that they better match splitting ratios based on link capacities rather than uniform splitting ratios. To quantify this conclusion, we run an experiment where we use 25% optimal filtering of link constraints and use the value of 10 units of traffic for the acceptable absolute error c in the penalty function in Equation (1) where the units are scaled so that the largest link has a capacity of 1 million. As the LP returns separate splitting ratios for each flow, however, in the case study we assume that routers do not split with different ratios for each flow, and we instead compute splitting ratios for each set of outgoing links as the weighted average of the ratios for the flows that split on those links. We compare these splitting ratios to the ones produced by capacity-based and uniform splitting. Since the penalty function is quadratic, we take the square root of the cost function defined in Section 2.2. This gives an error value of 10 for the splitting ratios based on the LP, 14 for the capacity-based splitting ratios, and 26 for the uniform splitting ratios. We can so conclude that capacity-based splitting ratios match the data in the case study almost as accurately as the flow-independent splitting ratios derived from the optimal solution to the LP, and significantly better than the uniform splitting ratios. This insight is now used in the traffic analytics deployment implemented by BENOCS GmbH.

6 Conclusion

We suggested a method for synthesis of flow splitting ratios from incomplete and noisy network traffic flow measurements. Our method is based on quadratic LP, and it handles the measurement noise and errors experienced in a real traffic analytics deployment by filtering out constraints with high penalty. We studied the amount of filtering to apply based on network topology characteristics and expected noise level.

We documented the accuracy and robustness of our method on an extensive synthetic benchmark as well as simulations based on real traffic data, and we showed that our filtering technique increases the accuracy of the synthesized flow splitting ratios. 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.

A limitation of our approach is that the more noise there is in the traffic data, the more filtering is needed in the linear program. This can lead to information loss and increase the imprecision of our estimates. To address this issue and further improve the accuracy of network traffic analytics based on our method, we suggest to develop techniques that will allow us to estimate the amount of noise in the network traffic data. Another approach is to extend the monitoring system to improve the reliability of packet classification during the monitoring, hence reducing the amount of measurement errors. This can, however, become expensive and resource demanding.

References

[1] Paul Aitken, Benoît Claise, and Brian Trammell. 2013. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information.* RFC 7011. RFC Editor. https://doi.org/10.17487/RFC7011

[2] BENOCS. n.d. BENOCS Home Page. Retrieved October 15, 2024 from https://www.benocs.com/

[3] Jeremy Bogle, Nikhil Bhatia, Manya Ghobadi, Ishai Menache, Nikolaj Bjørner, Asaf Valadarsky, and Michael Schapira. 2019. TEAVAR: Striking the right utilization-availability balance in WAN traffic engineering. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM '19)*. ACM, 29–43. https://doi.org/10.1145/3341302.3342069

[4] R. Callon. 1990. *Use of OSI IS-IS for Routing in TCP/IP and Dual Environments.* RFC 1195. RFC Editor. https://doi.org/10.17487/RFC1195

[5] Jeffrey Case, Mark Fedor, Martin Schoffstall, and James Davin. 1990. *Simple Network Management Protocol (SNMP).* RFC 1157. RFC Editor. https://doi.org/10.17487/RFC1157

[6] Yiyang Chang, Chuan Jiang, Ashish Chandra, Sanjay Rao, and Mohit Tawarmalani. 2019. Lancet: Better network resilience by designing for pruned failure sets. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3, 3 (2019), Article 49, 26 pages. https://doi.org/10.1145/3366697

[7] Benoît Claise. 2004. *Cisco Systems NetFlow Services Export Version 9.* RFC 3954. RFC Editor. https://doi.org/10.17487/RFC3954

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2022. *Introduction to Algorithms* (4th ed.). MIT Press.

[9] Emilie Danna, Subhasree Mandal, and Arjun Singh. 2012. A practical algorithm for balancing the max-min fairness and throughput objectives in traffic engineering. In *Proceedings of IEEE INFOCOM.* 846–854. https://doi.org/10.1109/INFCOM.2012.6195833

[10] Anja Feldmann, Oliver Gasser, Franziska Lichtblau, Enric Pujol, Ingmar Poese, Christoph Dietzel, Daniel Wagner, Matthias Wichtlhuber, Juan Tapiador, Narseo Vallina-Rodriguez, Oliver Hohlfeld, and Georgios Smaragdakis. 2020. The lockdown effect: Implications of the COVID-19 pandemic on internet traffic. In *Proceedings of the ACM Internet Measurement Conference (IMC '20).* ACM, 1–18. https://doi.org/10.1145/3419394.3423658

[11] B. Fortz, J. Rexford, and M. Thorup. 2002. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine* 40, 10 (2002), 118–124. https://doi.org/10.1109/MCOM.2002.1039866

[12] J. L. Garcia-Dorado, A. Finamore, M. Mellia, M. Meo, and M. Munafo. 2012. Characterization of ISP traffic: Trends, user habits, and access technology impact. *IEEE Transactions on Network and Service Management* 9, 2 (2012), 142–155. https://doi.org/10.1109/TNSM.2012.022412.110184

[13] John R. Goodall and Daniel R. Tesone. 2009. Visual analytics for network flow analysis. In *Proceedings of the 2009 Cybersecurity Applications and Technology Conference for Homeland Security.* 199–204. https://doi.org/10.1109/CATCH.2009.47

[14] Gurobi Optimization. 2023. *Gurobi Optimizer Reference Manual—Version 10.0.* Gurobi Optimization. https://www.gurobi.com/wp-content/plugins/hd_documentations/documentation/10.0/refman.pdf

[15] David Harrington, Bert Wijnen, and Randy Presuhn. 2002. *An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks.* RFC 3411. RFC Editor. https://doi.org/10.17487/RFC3411

[16] Tzvika Hartman, Avinatan Hassidim, Haim Kaplan, Danny Raz, and Michal Segalov. 2012. How to split a flow? In *Proceedings of IEEE INFOCOM.* 828–836. https://doi.org/10.1109/INFCOM.2012.6195830

[17] Markus Hofmann and Ralf Klinkenberg (Eds.). 2013. *RapidMiner: Data Mining Use Cases and Business Analytics Applications.* CRC Press.

[18] Rick Hofstede, Pavel Čeleda, Brian Trammell, Idilio Drago, Ramin Sadre, Anna Sperotto, and Aiko Pras. 2014. Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX. *IEEE Communications Surveys & Tutorials* 16, 4 (2014), 2037–2064. https://doi.org/10.1109/COMST.2014.2321898

[19] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. 2013. Achieving high utilization with software-driven WAN. In *Proceedings of the 2013 ACM SIGCOMM Conference (SIGCOMM '13).* ACM, 15–26. https://doi.org/10.1145/2486001.2486012

[20] Christian Hopps. 2000. *Analysis of an Equal-Cost Multi-Path Algorithm.* RFC 2992. RFC Editor. https://doi.org/10.17487/RFC2992

[21] IBM. n.d. IBM ILOG CPLEX Optimization Studio 22.1.0. Retrieved October 15, 2024 from https://www.ibm.com/docs/en/icos/22.1.0

[22] ISO. 2002. *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. ISO. https://www.iso.org/standard/30932.html

[23] Chuan Jiang, Zixuan Li, Sanjay Rao, and Mohit Tawarmalani. 2022. Flexile: Meeting bandwidth objectives almost always. In *Proceedings of the 18th International Conference on Emerging Network EXperiments and Technologies (CoNEXT '22).* 110–125. https://doi.org/10.1145/3555050.3569119

[24] Chuan Jiang, Sanjay Rao, and Mohit Tawarmalani. 2020. PCF: Provably resilient flexible routing. In *Proceedings of the 2020 ACM SIGCOMM Conference (SIGCOMM '20).* 139–153. https://doi.org/10.1145/3387514.3405858

[25] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. 2011. The Internet Topology Zoo. *IEEE Journal on Selected Areas in Communications* 29, 9 (2011), 1765–1775.

[26] Kiran Lakkaraju, William Yurcik, and Adam J. Lee. 2004. 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 (VizSEC/DMSEC '04).* ACM, 65–72. https://doi.org/10.1145/1029208.1029219

[27] Bingdong Li, Jeff Springer, George Bebis, and Mehmet Hadi Gunes. 2013. A survey of network flow applications. *Journal of Network and Computer Applications* 36, 2 (2013), 567–581. https://doi.org/10.1016/j.jnca.2012.12.020

[28] Hongqiang Harry Liu, Srikanth Kandula, Ratul Mahajan, Ming Zhang, and David Gelernter. 2014. Traffic engineering with forward fault correction. In *Proceedings of the 2014 ACM SIGCOMM Conference (SIGCOMM '14)*. 527–538. https://doi.org/10.1145/2619239.2626314

[29] John Moy. 1998. *OSPF Version 2*. RFC 2328. RFC Editor. https://doi.org/10.17487/RFC2328

[30] Juniper Networks. 2021. *IS-IS User Guide: Understanding Weighted ECMP Traffic Distribution on One-Hop IS-IS Neighbors*. Juniper Networks. https://www.juniper.net/documentation/us/en/software/junos/is-is/topics/concept/wecmp-for-one-hop-isis-neighbors-overview.html

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

[32] Yakov Rekhter, Susan Hares, and Tony Li. 2006. *A Border Gateway Protocol 4 (BGP-4)*. RFC 4271. RFC Editor. https://doi.org/10.17487/RFC4271

[33] Matthew Roughan. 2005. Simplifying the synthesis of internet traffic matrices. *ACM SIGCOMM Computer Communication Review* 35, 5 (Oct. 2005), 93–96. https://doi.org/10.1145/1096536.1096551

[34] Said Jawad Saidi, Aniss Maghsoudlou, Damien Foucard, Georgios Smaragdakis, Ingmar Poese, and Anja Feldmann. 2020. Exploring network-wide flow data with Flowyager. *IEEE Transactions on Network and Service Management* 17, 4 (2020), 1988–2006. https://doi.org/10.1109/TNSM.2020.3034278

[35] Morten Konggaard Schou, Ingmar Poese, and Jiří Srba. 2023. Discovery of flow splitting ratios in ISP networks with measurement noise. In *Proceedings of the 28th Pacific Rim International Symposium on Dependable Computing (PRDC '23)*. IEEE, 64–70. https://doi.org/10.1109/PRDC59308.2023.00017

[36] Morten Konggaard Schou, Ingmar Poese, and Jiří Srba. 2024. *Artifact for "Measurement-Noise Filtering for Automatic Discovery of Flow Splitting Ratios in ISP Networks."* https://doi.org/10.5281/zenodo.13326011

[37] Martin Suchara, Dahai Xu, Robert Doverspike, David Johnson, and Jennifer Rexford. 2011. Network architecture for joint failure recovery and traffic engineering. *ACM SIGMETRICS Performance Evaluation Review* 39, 1 (June 2011), 97–108. https://doi.org/10.1145/2007116.2007128

[38] Tina Tremel, Jochen Kögel, Florian Jauernig, Sebastian Meier, Dennis Thom, Franziska Becker, Christoph Müller, and Steffen Koch. 2022. VITALflow: Visual interactive traffic analysis with NetFlow. In *Proceedings of the 2022 IEEE/IFIP Network Operations and Management Symposium (NOMS '22)*. 1–6. https://doi.org/10.1109/NOMS54207.2022.9789776

[39] Martino Trevisan, Danilo Giordano, Idilio Drago, Marco Mellia, and Maurizio Munafo. 2018. Five years at the edge: Watching internet from the ISP network. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '18)*. ACM, 1–12. https://doi.org/10.1145/3281411.3281433

[40] Robert J. Vanderbei. 2020. *Linear Programming: Foundations and Extensions* (5th ed.). Springer. https://doi.org/10.1007/978-3-030-39415-8

[41] Ye Wang, Hao Wang, Ajay Mahimkar, Richard Alimi, Yin Zhang, Lili Qiu, and Yang Richard Yang. 2010. R3: Resilient routing reconfiguration. In *Proceedings of the 2010 ACM SIGCOMM Conference (SIGCOMM '10)*. ACM, 291–302. https://doi.org/10.1145/1851182.1851218

Received 3 May 2024; revised 15 August 2024; accepted 24 September 2024