Constrained maximum flow in stochastic networks

Kuipers, FA; Yang, S; Trajanovski, S; Orda, A

DOI
10.1109/ICNP.2014.63

Publication date
2014

Document Version
Accepted author manuscript

Published in
22nd IEEE International Conference on Network Protocols

Constrained Maximum Flow in Stochastic Networks

Fernando A. Kuipers, Song Yang, Stojan Trajanovski
Delft University of Technology
Delft, The Netherlands
{F.A.Kuipers, S.Yang, S.Trajanovski}@tudelft.nl

Ariel Orda
Technion
Haifa, Israel
ariel@ee.technion.ac.il

***Abstract*—Solving network flow problems is a fundamental component of traffic engineering and many communications applications, such as content delivery or multi-processor scheduling. While a rich body of work has addressed network flow problems in "deterministic networks," finding flows in "stochastic networks," where performance metrics like bandwidth and delay are uncertain and solely known by a probability distribution based on historical data, has received less attention. The work on stochastic networks has predominantly been directed to developing single-path routing algorithms, instead of addressing multi-path routing or flow problems.**

In this paper, we study constrained maximum flow problems in stochastic networks, where the delay and bandwidth of links are assumed to follow a log-concave probability distribution, which is the case for many distributions that could represent bandwidth and delay. We formulate the maximum-flow problem in such stochastic networks as a convex optimization problem, with a polynomial (in the input) number of variables. When an additional delay constraint is imposed, we show that the problem becomes NP-hard and we propose an approximation algorithm based on convex optimization. Furthermore, we develop a fast heuristic algorithm that, with a tuning parameter, is able to balance accuracy and speed. In a simulation-based evaluation of our algorithms in terms of success ratio, flow values, and running time, our heuristic is shown to give good results in a short running time.

***Index Terms*—Maximum flow, Stochastic networks, QoS, Convex optimization.**

I. Introduction

The ability to solve network flow problems is crucial to the successful operation of many different kinds of applications and networks, e.g. transportation networks, energy networks and communication networks. The precise problem could differ per application domain and for instance range from unconstrained to constrained flow or from single commodity to multi-commodity flow. Particularly within the field of communication networks, much research, already since 1955, has been directed to developing network flow algorithms (an excellent discourse of the subject is presented by Ahuja *et al.* [1]). These studies typically consider deterministic networks, in which the link weights (such as bandwidth and delay) are fixed. However, in many real-life networks, the delay or the available bandwidth of a link usually varies and is uncertain. For example, due to the size and complexity of data communication networks, it is difficult and expensive to obtain an accurate view on the states of the links. Another example is that the delay and available bandwidth are affected by diurnal patterns, interference in wireless networks, or by failure and maintenance events. In this paper, we consider so-called stochastic networks, where the link capacities and delays are characterized by a stochastic/probabilistic model (introduced in Section II) and therein study the maximum-flow problem.

In deterministic networks, the maximum-flow problem asks to send as much flow (information or goods) from a source to a destination, without exceeding the capacity of any of the used links. Solving maximum-flow problems is for instance important to avoid congestion and improve network utilization in computer networks or data centers, or to improve fault tolerance. Fortunately, the maximum-flow problem in deterministic networks is solvable in polynomial time [2]. The delay-constrained maximum-flow problem in deterministic networks [3] is to find a set of paths, each path obeying a given delay constraint, over which as much flow as possible is to be transported. This problem for instance appears in (real-time) video delivery over bandwidth-limited networks.

TABLE I: Hardness of computing network flows.

max-flow	deterministic	stochastic
without delay constraint	P [1], [2]	P*
with delay constraint	NP-hard [3], [4]	NP-hard*

* This paper, for the considered stochastic model.

We will consider these two problems in stochastic networks. Our main contributions are as follows (also see Table I):

- We prove that the maximum-flow problem in stochastic networks with log-concave probability distributions and some integrality assumptions is polynomially solvable.
- We prove that the delay-constrained maximum-flow problem in stochastic networks is NP-hard and propose an approximation algorithm and a faster heuristic.
- We evaluate the algorithms in terms of acceptance ratio, flow values, and running time.

The remainder of this paper is organized as follows: Section II introduces the considered stochastic link model and presents the unconstrained stochastic maximum-flow problem. We formulate this problem as a convex optimization problem with a polynomial (in the input) number of variables and discuss its computational complexity. Section III defines the stochastic maximum-flow problem with a delay constraint imposed on the used paths. We prove that this delay-constrained problem is NP-hard. Sections IV and V provide an approximation algorithm and a heuristic algorithm, respectively, for the

NP-hard stochastic delay-constrained maximum-flow problem. We evaluate the approximation and heuristic algorithms in Section VI. An overview of related work is presented in Section VII and we conclude in Section VIII.

II. Stochastic Maximum Flow

In this section, we introduce our stochastic network model, define the maximum-flow problem in stochastic networks, present a corresponding convex optimization formulation and discuss the computational complexity of the problem.

A. Stochastic Link Model

For completeness, we first present the standard definitions [5] of a convex set and a convex function. We define $\mathcal{R}^n$ to be the set of n-dimensional real vectors.

Definition 1: Convex Set. A subset C of $\mathcal{R}^n$ is called convex if $\alpha x + (1-\alpha)y \in C$, $\forall x, y \in C, \forall \alpha \in [0,1]$.

Definition 2: Convex Function. Let C be a convex subset of $\mathcal{R}^n$. We say that a function f: $C \rightarrow R$ is convex if $f(\alpha x + (1-\alpha)) \leq \alpha f(x) + (1-\alpha) f(y)$, $\forall x, y \in C, \forall \alpha \in [0,1]$.

A function f is said to be concave if and only if $-f$ is convex. If the second derivative of f is always nonnegative, then f is convex. Linear functions can be regarded to be convex or concave.

Many common (complementary) cumulative distribution functions $(C)CDF$[1] are log-concave (e.g., see [6] or [7] for more information). In convex analysis, a nonnegative function f: $\mathcal{R}^n \rightarrow \mathcal{R}$ is logarithmically concave or log-concave if its domain is a convex set, and if it satisfies the following inequality:

$$f(\theta x + (1-\theta)y) \geq f(x)^{\theta} f(y)^{1-\theta} \tag{2.1}$$

for all $x, y \in$ dom f and $0 < \theta < 1$. If f is strictly positive, then Eq. (2.1) can be rewritten as:

$$\log f(\theta x + (1-\theta)y) \geq \theta \log f(x) + (1-\theta) \log f(y) \tag{2.2}$$

Since bandwidth and delay values, and consequently their distributions, cannot be negative, several representative nonnegative log-concave distributions are the exponential, uniform, and chi-square distributions, although our results apply to all nonnegative distributions with log-concave (convex) CDF and $CCDF$.

B. Problem Definition

We consider a directed network $G(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ represents a set of N nodes and $\mathcal{L}$ denotes a set of L links. For each link $l \in \mathcal{L}$ the allocated bandwidth has a known cumulative distribution function $c_l(b_l)$, which gives the probability of being able to allocate no more than b_l units of bandwidth. Moreover, if the possible bandwidth allocated on each link l ranges from 0 to $b_l^{\max}$ ($0 < b_l^{\max}$), then the probability of allocating a bandwidth out of this range is 0, i.e. $c_l(b_l^{\max}) = 1$.

Definition 3: The Maximum Flow in Stochastic Networks (MFSN) problem is to push as much flow F as possible from a source node s to a terminal node t as long as the probability of actually realizing that flow is no **less** than a user-defined value P_B.

The MFSN problem is considered in the setting where the maximum flow F is computed upfront, i.e., before the realization of the random variables $c_l(b_l)$, $l \in \mathcal{L}$; then, the computed links/paths are used to (try to) send the flow F. The random variables are assumed to be independent. Although, the utilized bandwidth of the links used is correlated if only one flow is sent through a network, a network is typically used to transport many flows between various source-destination pairs. In such a setting the correlation, if any, is much less, and our independence assumption justified.

In our scenario, the total flow will not be realized if one link, out of the set of links that are used to transport the flow, is not able to deliver its portion of the total flow. Hence, in order to realize a maximum flow with probability at least P_B, the product of all the individual link probabilities needs to be bounded by P_B.

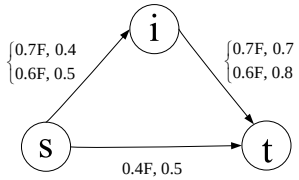


Fig. 1: An example to illustrate that all the link probabilities need to be included.

For instance, in Fig. 1, let us consider a network of 3 nodes, s, i, t, directed from s to t. The maximum flow F that obeys $P_B = 0.2$ pushes $0.4F$ over link (s,t) with probability 0.5 and $0.6F$ over path $s-i-t$ with probability 0.4. If the direct link (s,t) does not deliver its portion of the flow, then it might still happen that the path $s-i-t$ can deliver $0.7F$. However, the probability of this occurring is smaller than 0.4, otherwise $1.1F$ could have been transported with $P_B = 0.2$. In fact, since $-\log(CCDF)$ is convex, a decrease of ΔF flow on one link being compensated by an increase in ΔF flow on another link leads to a smaller total probability. The same holds if either (s,i) or (i,t) fails to deliver its portion of the flow, which indicates that the links can be considered independently.

As a result, the traditional min-cut max-flow theorem [8] does not apply anymore in stochastic networks, since all link probabilities (instead of only those of the minimum cut) play a role in realizing the maximum flow. Instead, the minimum cut in a stochastic network gives an upper bound on the amount of flow that can be pushed from source to destination with probability no less than P_B.

Fortunately, the topological independence of the MFSN problem greatly simplifies solving it. We will show that

[1] A $CCDF$ is defined as $\overline{F}(x) = \Pr[X > x]$, where the right-hand side represents the probability that the random variable X takes on a value greater than x. In the literature [6], it is sometimes referred to as *reliability function*.

the MFSN problem can be formulated as a simple convex optimization problem. The latter is a problem in which the objective function is either a maximization of a concave function or a minimization of a convex function, and its constraints are all convex. Convex optimization problems can usually be solved quickly and accurately with convex optimization solvers. The convex optimization formulation of the MFSN problem is as follows:

Objective:

$$\max \ F \tag{2.3}$$

Constraints:

$$\sum_{v:(u,v)\in\mathcal{L}} f(u,v) - \sum_{v:(v,u)\in\mathcal{L}} f(v,u) = \begin{cases} F & u = s \\ 0 & \forall u \in \mathcal{N}\backslash\{s,t\} \\ -F & u = t \end{cases} \tag{2.4}$$

$$-\sum_{(u,v)\in\mathcal{L}} \log\left[CCDF_{(u,v)}(f(u,v))\right] \leq -\log(P_B) \tag{2.5}$$

The variable $f(u,v)$ denotes the flow through link (u,v) and F is the amount of flow from s to t. The objective can be regarded as a maximization of a concave function. Constraints (2.4) are flow conservation constraints and they are linear. $CCDF_{(u,v)}(f(u,v))$ denotes the probability of allocating at least $f(u,v)$ bandwidth on link (u,v), i.e., $1-c_{(u,v)}(f(u,v))$. Constraint (2.5) is equivalent to

$$\prod_{(u,v)\in\mathcal{L}} CCDF_{(u,v)}(f(u,v)) \geq P_B$$

which reflects the condition to have a flow F with probability at least P_B. As explained, the product of the $CCDF$s does not depend on the network topology. The network topology is included in the flow conservation constraints (2.4). The link capacity constraints are implicitly defined by the $CCDF$.

Given that the $CCDF$ is log-concave, we obtain that constraint (2.5) is convex, therefore the MFSN problem is a convex optimization problem.

In Appendix A, we also present a convex optimization formulation for the min-cut in stochastic networks problem.

C. Computational Complexity

The convex optimization formulation of the MFSN problem has $O(L)$ variables and constraints. Convex optimization problems can be solved to ε-optimality in polynomial time if certain computability conditions (as stipulated in [9]) are satisfied. We will provide an alternative (simpler) proof of polynomial solvability of the MFSN problem. We will prove that when a piecewise linear convex function[2] is used, or, alternatively, when it is a convex function where the feasible solution needs to be integral (both models are introduced and justified in [1, Ch. 14]), the MFSN problem can be solved in polynomial time. In [1, Ch. 14] a (weakly) polynomial-time algorithm is provided that solves the following problem:

Objective:

$$\min \sum_{(u,v)\in\mathcal{L}} C_{uv}(f(u,v)) \tag{2.6}$$

Constraints:

$$\sum_{v:(u,v)\in\mathcal{L}} f(u,v) - \sum_{v:(v,u)\in\mathcal{L}} f(v,u) = \begin{cases} F & u = s \\ 0 & \forall u \in \mathcal{N}\backslash\{s,t\} \\ -F & u = t \end{cases} \tag{2.7}$$

$$0 \leq f(u,v) \leq b(u,v) \text{ for all } (u,v) \in \mathcal{L} \tag{2.8}$$

$$f(u,v) \text{ is integer for all } (u,v) \in \mathcal{L} \tag{2.9}$$

where $C_{uv}(f(u,v))$ is a convex cost function of the flow $f(u,v)$ on link (u,v) and $b(u,v)$ is the capacity of link (u,v).

As a starting point for the amount of flow to be sent from source s to destination t, let us take the minimum cut of the network when considering the maximum possible bandwidth values $b_l^{\max}$ of the links. The link costs are set similarly as before to $C_{uv}(f(u,v)) = -\sum_{(u,v)\in\mathcal{L}} \log\left[CCDF_{(u,v)}(f(u,v))\right]$. Via a binary search on the flow value F to be transported from source to destination, one can search for the largest flow F whose cost is still within the constraint $-\log(P_B)$, which demonstrates that the MSFN problem is solvable in polynomial time.

III. Stochastic Delay-Constrained Flow

In this section, we add a delay constraint to the MFSN problem. Each link $l \in \mathcal{L}$ now has two known CDFs, namely, (1) $c_l(b_l)$, the probability of being able to allocate no more than b_l units of bandwidth and (2) $p_l(d_l)$, the probability of transporting data with no more than d_l units of delay. We assume that the bandwidth allocated on each link l ranges from 0 to $b_l^{\max}$, and the delay on each link l ranges from $d_l^{\min} > 0$ to $d_l^{\max}$. For ease of notation we will sometimes write c_l and p_l to denote $c_l(b_l)$ and $p_l(d_l)$.

Definition 4: The Maximum Delay-Constrained Flow (MDCF) problem in stochastic networks is to find a set of paths Ψ from s to t such that:

1) The flow F transported by all paths in Ψ is maximum under the condition that the probability to realize that flow F is **no less** than P_B.
2) Each path in Ψ should have a delay no more than D and the probability of transporting it within **at most** D time is **no less** than P_D.

Contrary to the MFSN problem, the MDCF problem is NP-hard (as will be proved in Section III-A).

[2] A continuous convex function can, with arbitrary precision ε, be approximated by a piecewise linear convex function.

A. Computational Complexity

To prove that the MDCF problem is NP-hard, we first introduce the NP-hard partition problem [10], defined as follows: Given a set A that consists of elements $a_1, a_2, ..., a_{2n}$, where the size $S(a_i)$ of a_i is a nonnegative number and $n \geq 1$. The problem is to find a subset $A' \subseteq A$ such that A' contains exactly one element of $\{a_{2i-1}, a_{2i}\}$ for every $1 \leq i \leq n$ and $\sum_{a \in A'} S(a) = \sum_{a \in A \setminus A'} S(a)$.

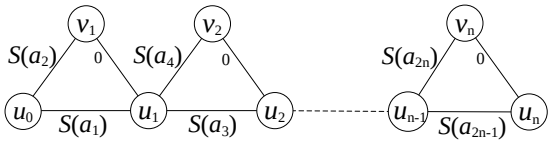


Fig. 2: A reduction to the partition problem.

Theorem 5: The MDCF problem is NP-hard.

Proof: We construct a graph $\mathcal{G}(\mathcal{N}, \mathcal{L})$ (as exemplified in Fig. 2), where the node set $\mathcal{N} = \{u_0, v_1, u_1, ..., u_{n-1}, v_n, u_n\}$. The links $(u_i, u_{i+1}), (u_i, v_{i+1})$, for all $i = 0, 1, ..., n-1$, have a Gamma distributed delay in the range 0 to $S(a_{2i+1})$, and from 0 to $S(a_{2i+2})$, respectively, and the links (v_i, u_{i+1}), for all $i = 0, 1, ..., n-1$, have zero delay (with probability 1). The allocated link bandwidth is Gamma distributed and ranges from 0 to 1. The maximum possible (unconstrained) flow is 2. Suppose we want to find, with probability no less than 0, a flow from s to t of at least 2 and for which each used path has a delay no more than $\frac{\sum_{a \in A} S(a)}{2}$ with probability of 1. Hence, we need to find two link-disjoint paths, such that the delay of each path is no greater $\frac{\sum_{a \in A} S(a)}{2}$. Each path either passes through (a) (u_i, u_{i+1}) or (b) (u_i, v_i) and (v_i, u_{i+1}). If both paths have a delay no more than $\frac{\sum_{a \in A} S(a)}{2}$ each, then they have to be exactly $\frac{\sum_{a \in A} S(a)}{2}$, which means that solving the MDCF problem would provide a solution to the NP-hard partition problem (the links in one path correspond to the elements in one set). ∎

IV. Approximation Algorithm

We propose, in this section, an approximation algorithm for the MDCF problem. The algorithm consists of three parts:

1) we round and scale the link weights (in Sections IV-A and IV-B) to have a polynomial number of variables;
2) we create a transformation graph (in Sections IV-C and IV-D) that captures the delay constraint and probability values as nodes in the graph;
3) we present a convex optimization formulation (in Section IV-E) that aims to solve the maximum-flow problem in this transformation graph, such that the requested bandwidth and delay probabilities are met.

The resulting flow in the transformation graph can directly be mapped to a flow in the original graph.

We first explain the procedure of rounding and scaling to obtain new delay and probability values. The rounding and scaling depends on two input parameters, namely ε, which quantifies the desired accuracy in the delay probability constraint P_D, and η, which quantifies the desired accuracy in the delay constraint D. The new link probabilities will be defined as

$$w_l = \left\lfloor -\log_{\left(1+\frac{\varepsilon}{N}\right)}(p_l) \right\rfloor$$

and the corresponding new requested delay probability as

$$W = \left\lceil -\log_{1+\frac{\varepsilon}{N}} P_D \right\rceil.$$

The new link delay values will be defined as

$$\delta_l = \left\lfloor \frac{d_l N}{D\eta} \right\rfloor$$

and the new requested delay constraint as

$$\Delta = \left\lceil \frac{N}{\eta} \right\rceil.$$

In Sections IV-A and IV-B, we will prove that solving MDCF based on such rounded and scaled values results in returned paths ψ for which it holds that $\sum_{l \in \psi} d_l \leq (1+\eta) D$ with probability at least $\frac{P_D}{(1+\varepsilon)}$. A solution satisfying these relaxed constraints is called an (ε, η)-solution. If only one parameter of accuracy is required, one could simply set $\eta = \varepsilon$.

A. Rounding the Requested Delay Probability

For the rounding of the requested delay probability, we make use of a technique presented in [11]. For each probability p_l, there exists an integer w_l, such that:

$$\left(1+\frac{\varepsilon}{N}\right)^{-(w_l+1)} < p_l \leq \left(1+\frac{\varepsilon}{N}\right)^{-w_l}$$

which implies $w_l \leq -\log_{\left(1+\frac{\varepsilon}{N}\right)}(p_l) < w_l + 1$, or equivalently $w_l = \left\lfloor -\log_{\left(1+\frac{\varepsilon}{N}\right)}(p_l) \right\rfloor$. Instead of considering p_l, we will use w_l. The same is done for P_D, but rounded slightly differently

$$\left(1+\frac{\varepsilon}{N}\right)^{-W} \leq P_D < \left(1+\frac{\varepsilon}{N}\right)^{-(W-1)} \tag{4.1}$$

which gives $W = \left\lceil -\log_{\left(1+\frac{\varepsilon}{N}\right)}(P_D) \right\rceil$. Therefore, if a path ψ satisfies the probability constraint $\prod_{l \in \psi} p_l \geq P_D$, we have

$$\prod_{l \in \psi} \left(1+\frac{\varepsilon}{N}\right)^{-w_l} \geq \prod_{l \in \psi} p_l \geq P_D \geq \left(1+\frac{\varepsilon}{N}\right)^{-W}.$$

Hence, we obtain $\sum_{l \in \psi} w_l \leq W$.

For the total probability θ, we will show that $\theta = \prod_{l \in \psi} \left(1+\frac{\varepsilon}{N}\right)^{-w_l} \geq \frac{P_D}{(1+\varepsilon)}$. Given that the maximum hopcount in the network is $H_{\max} \leq N-1$. Then:

$$\begin{aligned} \theta &\geq \prod_{l \in \psi} \left(1+\frac{\varepsilon}{N}\right)^{-(w_l+1)} \\ &= \left(1+\frac{\varepsilon}{N}\right)^{-\sum_{l \in \psi} w_l} \left(1+\frac{\varepsilon}{N}\right)^{\sum_{l \in \psi} 1} \\ &\geq \left(1+\frac{\varepsilon}{N}\right)^{-(W+H_{\max})} \end{aligned} \tag{4.2}$$

Using (4.1) into (4.2), we arrive at

$$\begin{aligned}\theta &\geq \left(1+\frac{\varepsilon}{N}\right)^{-W}\left(1+\frac{\varepsilon}{N}\right)^{-H_{\max}} \\ &= \left(1+\frac{\varepsilon}{N}\right)^{-(W-1)}\left(1+\frac{\varepsilon}{N}\right)^{-H_{\max}-1} \\ &\geq P_D\left(1+\frac{\varepsilon}{N}\right)^{-N} = \frac{P_D}{\left(1+\frac{\varepsilon}{N}\right)^{N}} \\ &\approx \frac{P_D}{1+\frac{\varepsilon}{N}N} \\ &= \frac{P_D}{1+\varepsilon}\end{aligned}$$

B. Rounding and Scaling the Delay Constraint

The delay values d_l will be scaled and rounded to $\delta_l = \left\lfloor \frac{d_l N}{D\eta} \right\rfloor$ and the delay constraint is set to $\Delta = \left\lceil \frac{N}{\eta} \right\rceil$. If a path ψ satisfies the delay constraint $\sum_{l\in\psi} d_l \leq D$, then $\sum_{l\in\psi} \delta_l \leq \sum_{l\in\psi} \frac{d_l N}{D\eta} \leq \frac{N}{\eta} \leq \Delta$. If we find a path ψ for which $\sum_{l\in\psi} \delta_l \leq \Delta$, then also $\sum_{l\in\psi}(\frac{d_l N}{D\eta} - 1) \leq \frac{N}{\eta} + 1$. Considering there are at most $N-1$ links in a path, we obtain $\sum_{l\in\psi} \frac{d_l N}{D\eta} \leq \frac{N}{\eta} + N$. It follows that $\sum_{l\in\psi} d_l \leq (1+\eta)\, D$.

After rounding and scaling, we proceed to build a *transformation graph* that has at most $(\Delta+1)(W+1)N$ nodes. In order to construct the transformation graph, we first create an *auxiliary graph* that, by its construction, ensures that every feasible path from s to t will have a delay no bigger than Δ, but it does not involve the requested probability P_D or W. The transformation graph builds on the auxiliary graph by including the probability P_D in W.

C. Auxiliary graph

The *auxiliary graph* $\mathcal{G}^A(\mathcal{N}^A, \mathcal{L}^A)$ is constructed as follows:

1) For each node $u \in \mathcal{N}$, $\mathcal{N}^A$ contains $(\Delta+1)$ nodes $u_0, u_1, ..., u_\Delta$.
2) For each link $(u,v) \in \mathcal{L}$, $\mathcal{L}^A$ has at most $\Delta + (\Delta - 1) + \cdots + 1 = \frac{\Delta^2+\Delta}{2}$ links in the form of (u_i, v_{i+j}), where $i = \delta_l^{\min}, ..., \min\{\Delta, \delta_l^{\max}\} - 1$ and $j = i+1, ..., \min\{\Delta, \delta_l^{\max}\}$
3) There are also Δ links in the form of (t_i, t_{i+1}) where $0 \leq i \leq \Delta - 1$. These links can allocate any bandwidth and zero delay with probability 1. Alternatively, the nodes t_i could be considered as one node t.

Fig. 4 gives the auxiliary graph of the graph in Fig. 3. The source and the destination are s_0 and t_Δ, respectively. In this example, a $\Delta = 3$ constrained flow is requested from s to t in the original graph.

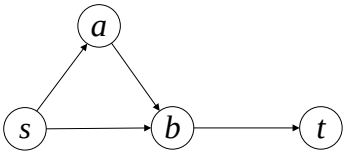


Fig. 3: Original network.

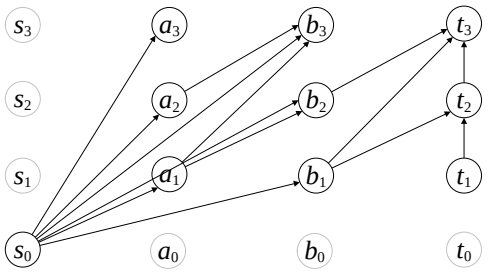


Fig. 4: Auxiliary graph for the network in Fig. 3.

D. Transformation Graph

We proceed to extend the auxiliary graph $\mathcal{G}^A$ to reach our transformation graph $\mathcal{G}^T(\mathcal{N}^T, \mathcal{L}^T)$.

1) For each node $u_i \in \mathcal{N}^A$, where $i = 0, 1, ..., \Delta$, $\mathcal{N}^T$ contains $(W+1)$ nodes $u_{i0}, u_{i1}, ..., u_{iW}$.
2) For each link $(u_i, v_j) \in \mathcal{L}^A$, $\mathcal{L}^T$ has at most $W - w(u_i, v_j) + 1$ links $(u_{ir}, v_{j(r+w(u_i,v_j))})$, where $r = 0, 1, ..., W - w(u_i, v_j)$.
3) There are W links $(t_{ir}, t_{i(r+1)})$, for $i = 0, 1, ..., \Delta$ and $r = 0, ..., W-1$. These links can allocate any bandwidth and zero delay with probability 1.

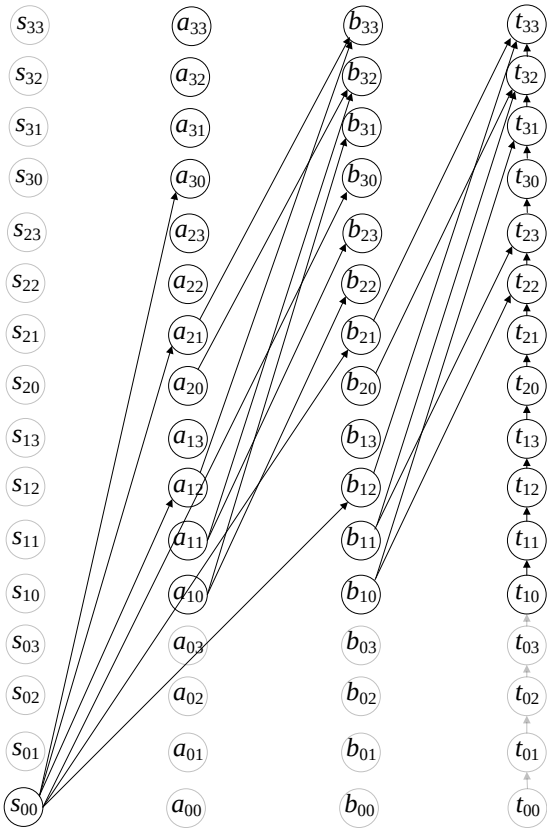


Fig. 5: Transformation graph for the network in Fig. 3.

The source and destination nodes in $\mathcal{G}^T$ are s_{00} and $t_{\Delta W}$, respectively. Redundant nodes could be pruned for faster solvability by, for instance, redirecting the links and computing

a shortest paths tree rooted at $t_{\Delta W}$: nodes not in that tree can be pruned and if s_{00} is not in the tree, a solution does not exist. In Section VI, we demonstrate that significant time savings can be reached by pruning redundant nodes.

The delay of each feasible path is no more than Δ (similarly as in the auxiliary graph). Hence, if an index $i = \Delta$ is achieved for a node different from t, then the maximum delay is achieved and the path is considered to be unfeasible. Similarly, the sum of the link weights w_l cannot exceed W.

Let us continue the example based on Fig. 3. Assume all links have a $-\log$(probability) of 0 (i.e., probability 1) for having delay within 3 time units, 0.001 for delay within 2 time units and 0.002 for delay within 1 time unit. The smaller $-\log$(probability), the higher the probability. The requested $-\log$(probability) is 0.003. Assume further, for simplicity, that these values are rounded and scaled with an ε in such a way that it is equivalent to multiplying by 1000. Under these assumptions, Fig. 5 presents the transformation graph, where node u_{ij} corresponds to $u \in \mathcal{N}$. Node u_{ij} has two numbers i and j. The first number i is at most Δ and indicates the (rounded and scaled) delay from the source node (which should not exceed Δ), and the second number j, with value not exceeding W, represents the (rounded and scaled) log-probability at which delay i can be realized.

The final part is a convex optimization formulation that aims to solve a maximum-flow problem in this transformation graph. The resulting flow in the transformation graph can directly be mapped to a flow in the original graph that obeys the requested bandwidth and delay probabilities with (ε, η)-accuracy, as will be shown in the following section.

E. Convex Optimization Formulation

We first introduce some notation.

$G(\mathcal{N}, \mathcal{L})$: original graph.

$G^A(\mathcal{N}^A, \mathcal{L}^A)$: auxiliary graph of $G(\mathcal{N}, \mathcal{L})$ with rounded and scaled delay values.

$G^T(\mathcal{N}^T, \mathcal{L}^T)$: transformation graph of $G(\mathcal{N}, \mathcal{L})$ with rounded and scaled delay and probability values.

$f^T(u, v)$: the total flow along link (u, v) in the transformation graph $\mathcal{G}^T(\mathcal{N}^T, \mathcal{L}^T)$.

$f^A(u, v)$: the total flow along link (u, v) in the auxiliary graph $\mathcal{G}^A(\mathcal{N}^A, \mathcal{L}^A)$.

$f(u, v)$: the total flow along link (u, v) in the original graph $\mathcal{G}(\mathcal{N}, \mathcal{L})$.

Objective:

$$\max \ F \tag{4.3}$$

Constraints:

$$\sum_{(u,v)\in\mathcal{L}^T} f(u,v) - \sum_{(v,u)\in\mathcal{L}^T} f(v,u) = \begin{cases} F & u = s_{00} \\ -F & u = t_{\Delta W} \\ 0 & else \end{cases} \tag{4.4}$$

$$f^A(u,v) = \sum_{m=0}^{i\le W-1} \sum_{n=i+1}^{j\le W} f^{\Delta}(u_{im}, v_{jn}) \tag{4.5}$$
$$\forall (u,v) \in \mathcal{L}^T, (u_i, v_j) \in \mathcal{L}^A$$

$$f(u,v) = \sum_{i=0}^{i\le \Delta-1} \sum_{j=i+1}^{j\le \Delta} f^A(u_i, v_j) \tag{4.6}$$
$$\forall (u,v) \in \mathcal{L}, (u_i, v_j) \in \mathcal{L}^A$$

$$-\sum_{(u,v)\in\mathcal{L}} \log(CCDF_{(u,v)}(f(u,v))) \le -\log(P_B) \tag{4.7}$$

$$0 \le f(u,v) \le b_{(u,v)}^{\max} \ \forall (u,v) \in \mathcal{L} \tag{4.8}$$

Constraint (4.4) represents the nodal flow conservation constraint in the transformation graph $\mathcal{G}^T$. Constraint (4.5) calculates the total flow on each link $(u_i, v_j) \in \mathcal{L}^A$, and Constraint (4.6) calculates the total flow on each link $(u, v) \in \mathcal{L}^T$. Constraint (4.7) ensures that the probability of allocating bandwidth in the original graph is not smaller than requested. With Constraint (4.8) we obey the link capacity.

To find the set of paths Ψ, requested in the MDCF problem, that obey the delay constraint, we deploy a conventional flow decomposition algorithm (e.g., see [1, pp. 79-83]).

We conclude by demonstrating that a maximum-flow in the transformation graph translates to an (ε, η)-approximate solution to the MDCF problem in the original graph. In Sections IV-A and IV-B, we have proved that rounding and scaling leads to an (ε, η)-solution, hence it remains to demonstrate that the translation to (and from) a maximum-flow problem in the transformation graph is equivalent to solving the rounded and scaled MDCF problem directly on the original graph.

The rounding and scaling step has discretized the convex functions and all possible values for the delay and delay probability are expressed as nodes. For instance, node u in the original graph has $(\Delta + 1)(W + 1)$ corresponding nodes u_{ij} in the transformation graph, where $i = 0, ..., \Delta$ reflects the (rounded and scaled) delay from the source node to u, and $j = 0, ..., W$, represents the (rounded and scaled) log-probability at which delay i can be realized. Consequently, all possible rounded and scaled delays and log-probabilities of the original graph are represented via links in the transformation graph, provided they do not exceed Δ and W. If a feasible path does not exist in the transformation graph, i.e. a path can only achieve one of the "upper nodes" without outgoing links, this means either a delay of Δ and/or a value of W is already spent. Since paths from s to t in the transformation graph obey Δ and W, and therefore safe-guard the (ε, η)-solution, it remains to show that a maximum flow from s to t in the transformation graph corresponds to an (ε, η)-solution for the MDCF problem in the original network.

Our convex optimization explicitly takes the link capacity constraints of the original network into account (in Constraint

(4.8)). Since nodes are linked in the transformation graph if and only if their corresponding nodes are linked in the original graph and the link capacities are not exceeded, a maximum flow in the transformation graph corresponds to an (ε, η)-solution to the MDCF problem in the original network. The nodes in the transformation graph are directly mapped to nodes in the original graph, which means that the paths in the transformation graph are also directly mapped to the original graph.

F. Computational Complexity

According to [5], the computation time for solving a convex optimization problem is (roughly) proportional to $\max\{n^3, n^2m, \phi\}$, where n reflects the number of variables, m the number of constraints, and ϕ is the cost of evaluating the first and second derivatives of the convex functions. In our convex optimization formulation there are $O(\Delta^2 WL)$ variables and constraints, which for the approximation algorithm corresponds to $O\left(\frac{N^2}{\eta^2}\left\lceil -\log_{1+\frac{\varepsilon}{N}} P_D \right\rceil L\right)$, i.e., a polynomial expression of the input size and the accuracy parameters (ε, η).

V. Heuristic Algorithm

Although the approximation algorithm has a polynomial number of variables, its running time for large networks may be too high. Therefore, we also propose a faster heuristic algorithm, called Multi-Constrained Maximum Flow (MCMF), for the MDCF problem. The idea of MCMF is that it discretizes the stochastic network and then iteratively (similar to an augmenting flow approach) runs a heuristic multi-constrained routing algorithm. We have chosen a heuristic, since multi-constrained path selection is an NP-hard problem [12].

The network discretization is as follows: k samples are taken (uniformly, at random, or non-uniformly, such as more samples from the low-bandwidth or high-delay regimes) from each link probability distribution, where k is determined by the user. For instance, for the delay we would end up, per link, with k (delay, -log(probability)) pairs, and similarly for the bandwidth. With these $2k$ pairs per link, we transform all links in the network as illustrated in Fig. 6. We make the following observations indicating that it is wise to keep k small:

1) Increasing k does not always lead to better results (as explained in Appendix B).
2) Including $2k$ pairs per link, as illustrated in Fig. 6, increases the search space of possible paths between two nodes exponentially.

Subsequently, we run TAMCRA [13], a heuristic multi-constrained path selection algorithm. The delay constraint is D, the delay probability constraint is $-log(P_D)$, and the bandwidth probability constraint is $-log(P_B)$. There is no constraint on the bandwidth. Instead, we modify the length function of TAMCRA slightly, by choosing to minimize $\sum_{l\in\psi} \frac{1}{b_l}$, which is a function that strives to increase bandwidth while minimizing the hopcount. TAMCRA never returns paths that exceed one of the constraints, but it is heuristic in the sense that it may fail to find a feasible path. TAMCRA is tunable in how many paths it stores per node, a parameter that we set equal to $q = ck$, a constant c times the number of samples k.

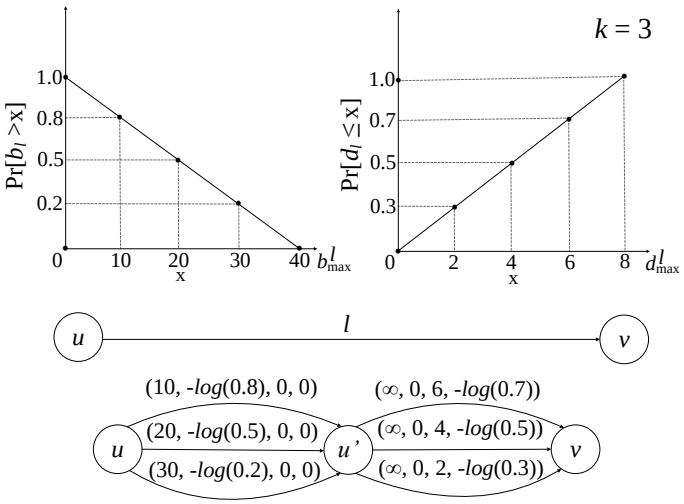


Fig. 6: Link transformation for the heuristic algorithm.

In the discretized network, we will iteratively run TAMCRA, prune its selected path, and adjust the remaining network. Since the delay probabilities and constraint are per path, they need not be adjusted. The bandwidth values do however. We decrease the bandwidth values of the traversed links by the bandwidth of the path. We do that for all k parallel links and prune the links with zero or negative values. To prevent the -log(probability) to be counted twice, we reduce the max k parallel links by the probability of the link that was used. An example is given in Fig. 7 for $k = 1$. In step 1, the network

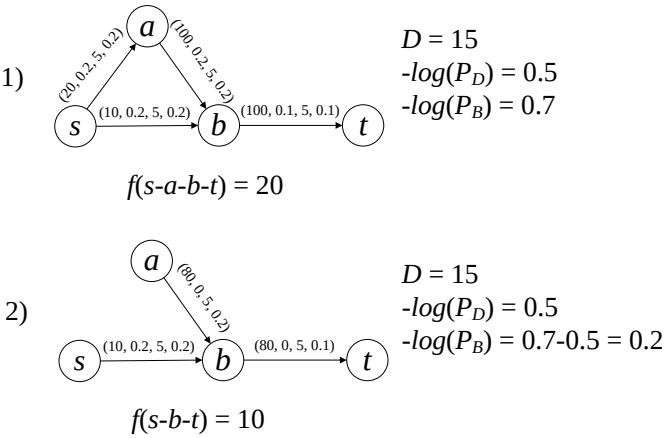


Fig. 7: An example of the heuristic link changes after an iteration for $k = 1$.

of Fig. 3 is discretized. To not clutter the figure, we have chosen to merge the two 4-tuples into one 4-tuple (we have used fictitious values) and to omit the extra node per link as illustrated in Fig. 6. TAMCRA returns path $s - a - b - t$ with bandwidth 20 and which satisfies the constraints. In step 2, the bandwidth along the links of this path is reduced by

20 and the corresponding probabilities are also adjusted. In the new graph, TAMCRA is executed again and returns path $s-b-t$ with bandwidth 10. In a third iteration, s would get disconnected and hence no more paths exist. The resulting flow therefore is 20 units of bandwidth via $s-a-b-t$ and 10 units of bandwidth via $s-b-t$.

For four parameters (delay and bandwidth probabilities and constraints), TAMCRA has a complexity of $O(qN\log(qN)+q^2kL)$. Since TAMCRA is iterated at most kL times, the total complexity becomes $O(qkNL\log(qN)+q^2k^2L^2)$.

VI. Simulation-Based Evaluation

A. Simulation Setup

We conduct simulations on two networks: USANet, displayed in Fig. 8, which is a realistic carrier backbone network consisting of 24 nodes and 43 links, and GÉANT, shown in Fig. 9, which is a pan-European communications infrastructure consisting of 40 nodes and 63 links.

We present simulation results for three representative log-concave distributions, namely the exponential, uniform, and chi-square distributions. In the exponential distribution $1-e^{-\lambda x}$, we choose $\lambda \in [0.001, 0.01]$ for the bandwidth distributions of different links and $\lambda \in [0.5, 1.5]$ for the delay distributions of different links. In the uniform distribution $\frac{x-\alpha}{\beta-\alpha}$, we choose $\alpha = 0$ for both bandwidth and delay, and $\beta \in [12, 20]$ for bandwidth and $\beta \in [4, 8]$ for delay. In the chi-square distribution $\Gamma(\frac{\kappa}{2}, \frac{x}{2})$, where $\Gamma()$ denotes the regularized gamma function, we choose $\kappa = 1$ for bandwidth and $\kappa \in [4, 9]$ for delay.

We generate 100 different requests, whose source and destination nodes are randomly selected. If constraints are set too tight, no solutions will exist, and if they are set too loose, the problem turns into the simpler MFSN problem. Our aim is to set constraints from loose (exponential) to medium (uniform) to tight (chi-square). We have set, for both topologies, the constraints for the exponential distribution to $P_B = 0.5$, $D = 12$, and $P_D = 0.5$, for the uniform distribution to $P_B = 0.1$, $D = 12$, $P_D = 0.5$, and for the chi-square distribution to $P_B = 0.01$, $D = 12$, $P_D = 0.5$. In all cases $d_l^{min} = 0$, $d_l^{\max} \in [4, 8]$, $b_l^{\max} \in [12, 20]$. The simulation is run on a desktop PC with 3.00 GHz CPU and 4 GB memory. We use CVX in Matlab, a package for specifying and solving convex optimization problems [14].

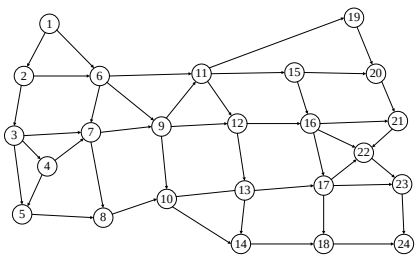


Fig. 8: USA carrier backbone network.

We compare the approximation algorithm with $\varepsilon = \eta = 0.1$, and our heuristic algorithm MCMF with $k = 1, 4, 8, 16$ uniformly chosen samples per link. As remarked before, better (than uniform) sampling methods may exist for a specific distribution. For each k in MCMF, the maximum number of stored paths q for each node is set to $q = k$ when compared to the approximation algorithm. Later for a fixed $k = 8$, we will vary q in order to further evaluate the heuristic algorithm. The heuristic algorithm guarantees to only return results that obey (P_B, D, and P_D), while the heuristic does not guarantee to always find a result when one exists. The approximation algorithm also obeys P_B and guarantees to return a result when a feasible one exists, but may violate D and P_D by a factor of $(1+\eta)$ and $(1+\varepsilon)$, respectively. To have a fair comparison, we (1) run the approximation algorithm with the constraints D and P_D, which leads to an upper bound, and (2) we use constraints $D' = \frac{D}{1+\eta}$ and $P_D' = (1+\varepsilon)P_D$, in which case the returned paths are guaranteed to obey D and P_D, but due to the stricter constraints, a solution may not be found, i.e. we have a lower bound.

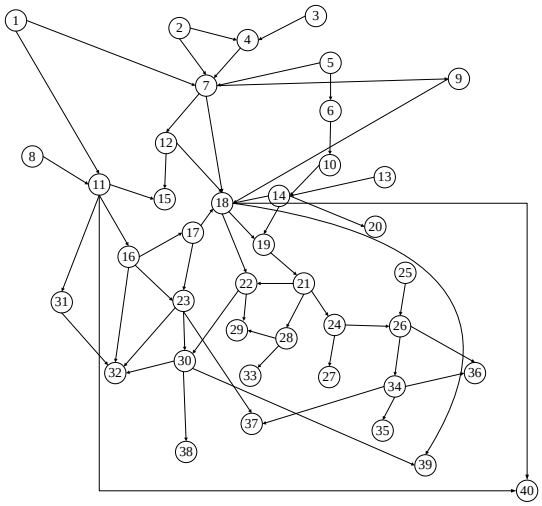
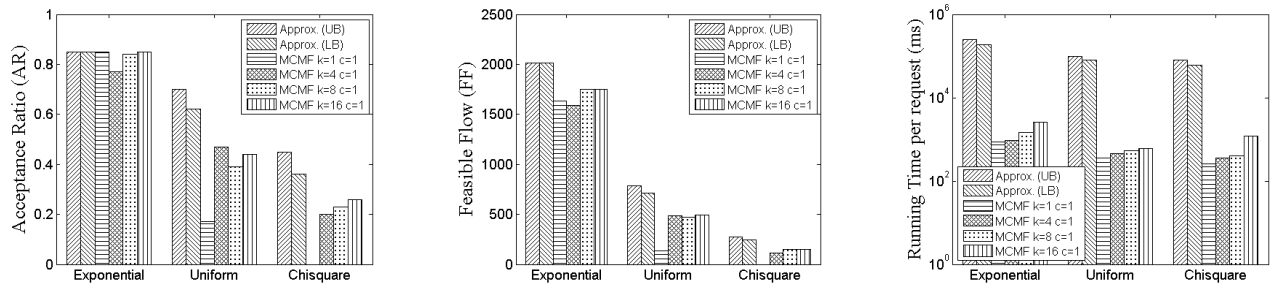


Fig. 9: GÉANT pan-European research network.

B. Simulation Results

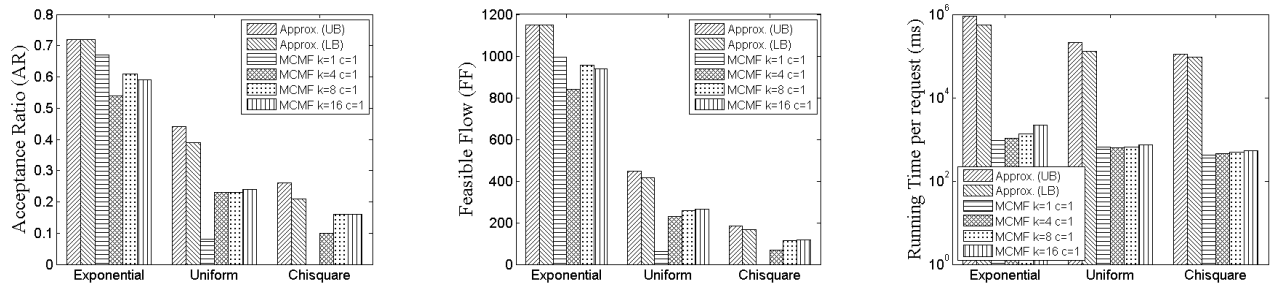
Figures 10 and 11 present our simulation results in terms of Acceptance Ratio (AR), Feasible Flow (FF) and Running Time.

Figures 10(a) and 11(a) depict the Acceptance Ratio, which is defined as (the number of times the algorithm returned a feasible result, i.e. obeying the three constraints) divided by (the total number of requests). As expected, the algorithms have the highest AR for the exponential distribution and the lowest AR for the chi-square distribution (where $k = 1$ did not return a feasible flow). Moreover, since we have chosen a small error parameter ($\varepsilon = \eta = 0.1$) for the approximation algorithm, its upper and lower bound values are quite close, which suggests that it has near-optimal performance. For MCMF, when k increases, its achieved AR approaches that of the (close-to-optimal) approximation, but much faster. However, this trend is not always increasing with k, as explained in Appendix B.



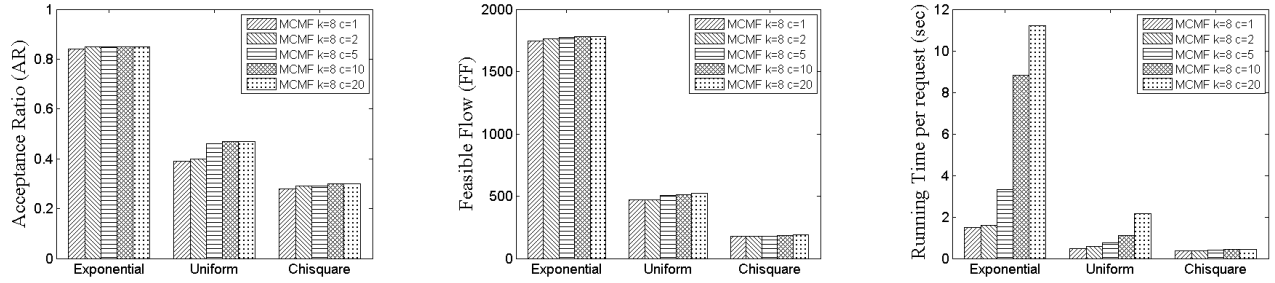
(a) Acceptance Ratio (AR) (b) Feasible Flow (FF) (c) Running Time

Fig. 10: (a) Acceptance Ratio (AR) (b) Feasible Flow (FF) (c) Running Time for 100 requests on USANet for $q = k$.



(a) Acceptance Ratio (AR) (b) Feasible Flow (FF) (c) Running Time

Fig. 11: (a) Acceptance Ratio (AR) (b) Feasible Flow (FF) (c) Running Time for 100 requests on GÉANT for $q = k$.



(a) Acceptance Ratio (AR) (b) Feasible Flow (FF) (c) Running Time

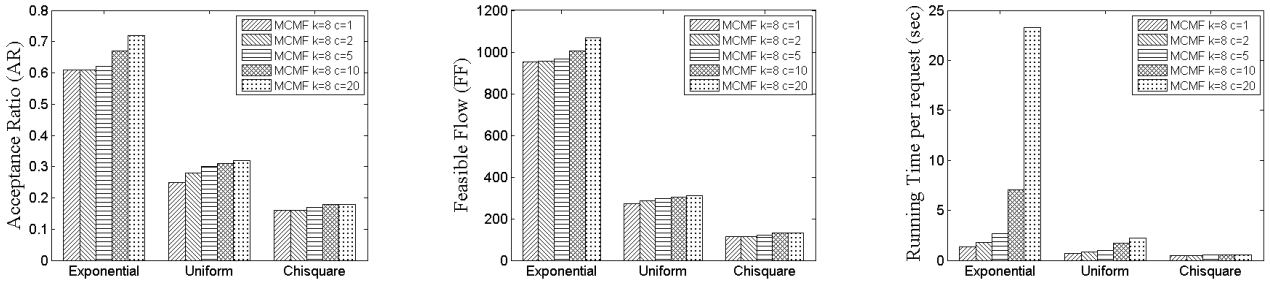
Fig. 12: Simulation results over 100 requests on USANet for $k = 8$ and $q = c \cdot k$, where $c = 1, 2, 5, 10, 20$: (a) Acceptance Ratio (AR) (b) Feasible Flow (FF) (c) Running Time.

Figures 10(b) and 11(b) plot the Feasible Flow (FF), which is defined as the total amount of flow returned by the algorithm that obeys the constraints. We observe that the lower bound of the approximation algorithm has achieved the same value as the upper bound of the approximation algorithm for the exponential distribution. This is due to the loose delay constraint. For the heuristic algorithm, again the flow values increase with k (most of the time) and approach the FF values of the approximation algorithm.

The running times of the algorithms are shown (on a log-scale) in Figures 10(c) and 11(c). The running time of the heuristic is much smaller than the approximation algorithm and therefore is the preferred choice for when flows need to be computed fast.

We notice in the above figures that when q is equal to k, the performance (either AR or FF) of the heuristic algorithm does not always increase with k. One reason is explained in Appendix B and another reason is the exponentially increasing search space. The latter could be solved by increasing the maximum number of stored paths q in MCMF. We therefore fix $k = 8$ in $q = ck$, and vary $c = 1, 2, 5, 10, 20$.

Figures 12 and 13 show the AR, FF and Running Time for



(a) Acceptance Ratio (AR) (b) Feasible Flow (FF) (c) Running Time

Fig. 13: Simulation results over 100 requests on GÉANT for $k = 8$ and $q = c \cdot k$, where $c = 1, 2, 5, 10, 20$: (a) Acceptance Ratio (AR) (b) Feasible Flow (FF) (c) Running Time.

MCMF. We can see from these figures that when q increases, the performance values indeed increase.

Finally, in Tables II and III, we study the effect of pruning the redundant nodes from the transformation graph. Since the approximation algorithm without pruning nodes takes too long for $\varepsilon = \eta = 0.1$, we have set it to $\varepsilon = \eta = 0.5$ instead, while we use $\varepsilon = \eta = 0.1$ for the case with pruning.

TABLE II: Running time per request for USANet (sec).

	Exponential	Uniform	Chi-square
Approx. (UB) (With Pruning)	252.3	96.4	79.1
Approx. (LB) (With Pruning)	183.6	79.6	61.3
Approx. (UB) (No Pruning)	350.2	805.6	842.7
Approx. (LB) (No Pruning)	206.3	450.9	470.8

TABLE III: Running time per request for GÉANT (sec).

	Exponential	Uniform	Chi-square
Approx. (UB) (With Pruning)	921.7	211.1	112.5
Approx. (LB) (With Pruning)	566.8	128.5	93.8
Approx. (UB) (No Pruning)	1386.6	1271.4	1003.6
Approx. (LB) (No Pruning)	621.7	818.8	592.9

As can be seen, our approximation algorithm with pruning and $\varepsilon = \eta = 0.1$ outperforms the approximation algorithm with $\varepsilon = \eta = 0.5$ but without pruning both in terms of accuracy and speed.

VII. Related Work

A small overview of techniques for dealing with stochastic networks and traffic uncertainty in general is provided in [15].

A. Flows

Although there is work on stochastic flow problems, e.g. see the work of Tahmasbi *et al.* [16] and the references therein, the stochastic model there differs from that in our paper and is concerned with a reliability perspective, where the expected maximum flow is computed under a random link-failure model. Contrary to our model, these problems have an implicit topological dependence, hence are NP-hard to solve.

Also the class of fuzzy flow problems, e.g. see the work of Diamond [17], deals with the maximum-flow problem under uncertainty. There, the uncertainty in capacity is represented by a fuzzy number.

Another class of (slightly) related network flow problems are dynamic (discrete or continuous) network flow problems, in which the capacities of the links vary over time (e.g., see a survey by Kotnyek [18]). Orda and Rom [19] extended dynamic network flow problems by also including time-dependent link delays $d_{(u,v)}(x)$, i.e. a flow leaving node u on link (u, v) at time x will arrive at node v at time $x+d_{(u,v)}(x)$. In dynamic flow problems, the time-dependent bandwidth and delay functions should be precisely known, which is difficult to realize in practice. Instead, probability distribution functions, as used in our paper, can be easily based on historical data and take uncertainty into account.

Sarangan *et al.* [20] studied how to estimate the maximum flow in a domain, where the domains are regarded as stochastic networks, and devised a capacity-aware inter-domain routing algorithm. However, their work does not take the requested probability (P_B) into account.

B. Path Selection

The stochastic models more closely related to our work have focused on finding a single path. Korkmaz and Krunz [21] considered the case where link delays are represented by non-negative Gaussian random variables. In that case, given a delay constraint D, the probability $\pi_D(\psi)$ that the delay of a path is no larger than D is

$$\pi_D(\psi) \approx \Phi\left(\frac{D - \mu(\psi)}{\sigma(\psi)}\right),$$

where $\Phi(x)$ is the Probability Density Function (PDF) of a Gaussian distribution, and $\mu(\psi)$ and $\sigma(\psi)$ denote the mean and standard deviation of the delay in path ψ. Since $\Phi(x)$ is an increasing function, to maximize $\pi_D(\psi)$ is to maximize $\frac{D-\mu(\psi)}{\sigma(\psi)}$. Korkmaz and Krunz referred to the problem of finding a path for which $\frac{D-\mu(\psi)}{\sigma(\psi)}$ is maximum as the Most Probable Delay Constrained Path (MPDCP) problem, and proposed a heuristic algorithm to solve it. Xiao *et al.* [22]

subsequently proved that the MPDCP problem is NP-hard and developed a Fully Polynomial Time Approximation Scheme (FPTAS) for when there exists a path whose mean delay is no more than D and an approximation scheme for when no such path exists.

Lorenz and Orda [23] considered the more general case where each link (i, j) has a function $\pi_{ij}(D_{ij})$ that represents the probability that link (i, j) introduces a delay of no more than D_{ij} time units. This so-called Delay-Based Routing (DBR) problem is to find a path that has the biggest probability of not exceeding D. Lorenz and Orda proved that the DBR problem is NP-hard, and by decomposing the end-to-end delay constraint D into local delay constraints, managed to develop an FPTAS.

C. Bandwidth and Delay Constraints

Banner and Orda [11] addressed the Restricted Multipath (RMP) problem, which is to find a set of paths Ψ that minimize network congestion, while each path $\psi \in \Psi$ should have a length no more than a given value. Misra *et al.* [3] studied the Multipath routing with Bandwidth and Delay constraints problem (MPBD), where the objective is to find a set of paths Ψ such that the total bandwidth is no less than a given value and the delay of each path $\psi \in \Psi$ should be minimized. Based on the MPBD problem, Zhang *et al.* [4] studied the Reliable Adaptive Multipath Provisioning (RAMP) problem, which refers to finding a set of paths Ψ such that the total bandwidth is no less than a given value, the delay of each path $\psi \in \Psi$ should be within some range and the bandwidth of each path $\psi \in \Psi$ should not exceed a given value. All these three problems are studied in deterministic networks and are proved to be weakly NP-hard. Accordingly, pseudo-polynomial time algorithms and polynomial approximation algorithms have been devised.

VIII. Conclusion

In this paper, we have studied the maximum-flow problem without and with delay constraints in stochastic networks. Under a general log-concave probability distribution model to represent bandwidth and delay and some mild assumptions, we have shown that the maximum-flow problem in stochastic networks is polynomially solvable and presented a convex optimization formulation. When a delay constraint is imposed on each path, the problem becomes NP-hard. To solve it, we have proposed a convex optimization formulation for an approximation algorithm and developed a faster tunable heuristic algorithm. We have performed simulations that show that our heuristic is fast and tunable to return close-to-optimal results.

Acknowledgement

This research has been partly supported by the EU FP7 Network of Excellence in Internet Science EINS (project no. 288021).

References

[1] R. Ahuja, T. Magnanti, and J. Orlin, *Network flows: theory, algorithms, and applications*. Prentice Hall, 1993.

[2] A. V. Goldberg and R. E. Tarjan, "Efficient maximum flow algorithms," *Communications of the ACM*, vol. 57, no. 8, pp. 82–89, 2014.

[3] S. Misra, G. Xue, and D. Yang, "Polynomial time approximations for multi-path routing with bandwidth and delay constraints," in *INFOCOM*. IEEE, 2009, pp. 558–566.

[4] W. Zhang, J. Tang, C. Wang, and S. De Soysa, "Reliable adaptive multipath provisioning with bandwidth and differential delay constraints," in *INFOCOM*. IEEE, 2010, pp. 1–9.

[5] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[6] M. Bagnoli and T. Bergstrom, "Log-concave probability and its applications," *Economic Theory*, vol. 26, no. 2, pp. 445–469, 08 2005.

[7] G. R. Mohtashami Borzadaran and H. A. Mohtashami Borzadaran, "log-concavity property for some well-known distributions," *Surveys in Mathematics and its Applications*, vol. 6, pp. 203–219, December 2011.

[8] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, pp. 399–404.

[9] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM Studies in Applied and Numerical Mathematics, 1994.

[10] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[11] R. Banner and A. Orda, "Multipath routing algorithms for congestion minimization," *IEEE/ACM Trans. on Netw.*, vol. 15, no. 2, pp. 413–424, 2007.

[12] F. A. Kuipers and P. Van Mieghem, "Conditions that impact the complexity of qos routing," *IEEE/ACM Transactions on Networking*, vol. 13, no. 4, pp. 717–730, 2005.

[13] P. Van Mieghem and F. Kuipers, "Concepts of exact quality of service algorithms," *IEEE/ACM Trans. on Netw.*, vol. 12, no. 5, pp. 851 – 864, 2004.

[14] CVX Research, Inc., "CVX: Matlab software for disciplined convex programming, version 2.0."

[15] S. Yang and F. A. Kuipers, "Traffic uncertainty models in network planning," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 172 – 177, 2014.

[16] R. Tahmasbi, E. Nasrabadi, and S. M. Hashemi, "The value of information in stochastic maximum flow problems," *Computers & Operations Research*, vol. 40, no. 7, pp. 1744 – 1751, 2013.

[17] P. Diamond, "A fuzzy max-flow min-cut theorem," *Fuzzy Sets and Systems*, vol. 119, pp. 139 – 148, 2001.

[18] B. Kotnyek, "An annotated overview of dynamic network flows," INRIA, Tech. Rep. RR-4936, Sep. 2003. [Online]. Available: http://hal.inria.fr/inria-00071643

[19] A. Orda and R. Rom, "On continuous network flows," *Operations Research Letters*, vol. 17, no. 1, pp. 27 – 36, 1995.

[20] V. Sarangan, D. Ghosh, and R. Acharya, "State aggregation using network flows for stochastic networks," in *GLOBECOM. IEEE*, vol. 3, 2002, pp. 2430–2434.

[21] T. Korkmaz and M. Krunz, "Bandwidth-delay constrained path selection under inaccurate state information," *IEEE/ACM Trans. on Netw.*, vol. 11, no. 3, pp. 384–398, 2003.

[22] Y. Xiao, K. Thulasiraman, X. Fang, D. Yang, and G. Xue, "Computing a most probable delay constrained path: NP-hardness and approximation schemes," *IEEE Trans. on Computers*, vol. 61, no. 5, pp. 738–744, 2012.

[23] D. Lorenz and A. Orda, "QoS routing in networks with uncertain parameters," *IEEE/ACM Trans. on Netw.*, vol. 6, no. 6, pp. 768–778, 1998.

[24] A. Tamir, "Polynomial formulations of min-cut problems," *Manuscript, Department of Statistic and Operations Research, Tel Aviv University, Israel*, 1994.

Appendix A
Min-cut in Stochastic Networks

Definition 6: The Min-Cut in Stochastic Networks (MCSN) problem is to find a cut C which partitions G into two disjoint subsets X ($X \in \mathcal{N}$) and $\mathcal{N} - X$ such that the source

s and the terminal t are in different subsets and the sum of allocated bandwidth for the links belonging to the cut should be as small as possible as long as the probability of realizing that bandwidth is no less than P_C.

According to the definition, we should use a CDF (instead of a $CCDF$ in the MFSN problem) to represent its realizing probability:

$$\prod_{(u,v)\in L: u\in X v\in \mathcal{N}-X} c_{(u,v)}\left(f(u,v)\right) \geq P_C \tag{1.1}$$

where $f(u,v)$ denotes the allocated bandwidth for link (u,v).

According to [24], the min-cut problem in a deterministic network can be solved by the following Linear Programming (LP) formulation:

Objective:

$$\min \sum_{(u,v)\in\mathcal{L}} b^{\max}_{(u,v)} \cdot y_{u,v} \tag{1.2}$$

Constraints:

$$y_{s,t} \geq 1 \tag{1.3}$$

$$y_{u,v} + y_{v,w} \geq y_{u,w},\ \forall u,v,w \in \mathcal{N} : u \neq v \neq w \tag{1.4}$$

$$y_{u,v} \geq 0,\ \forall u,v \in \mathcal{N} : u \neq v \tag{1.5}$$

where $b^{\max}_{(u,v)}$ stands for the capacity of link (u,v) in the deterministic network and $y_{u,v}$ is an indicator denoting whether (u,v) belongs to the cut. Similarly, the MCSN problem can be solved by the following convex optimization formulation:

Objective:

$$\min \sum_{(u,v)\in\mathcal{L}} f(u,v) \cdot y_{u,v} \tag{1.6}$$

Constraints:

$$-\sum_{(u,v)\in\mathcal{L}} \log\left[CDF_{(u,v)}(f(u,v))\right] \leq -\log(P_C) \tag{1.7}$$

$$0 \leq f(u,v) \leq b^{\max}_{(u,v)}\ \ \forall (u,v) \in \mathcal{L} \tag{1.8}$$

$$y_{s,t} \geq 1 \tag{1.9}$$

$$y_{u,v} + y_{v,w} \geq y_{u,w},\ \forall u,v,w \in \mathcal{N} : u \neq v \neq w \tag{1.10}$$

$$y_{u,v} \geq 0,\ \forall u,v \in \mathcal{N} : u \neq v \tag{1.11}$$

where $f(u,v)$ indicates the flow through link (u,v). According to [7], if the density function of a distribution is log-concave, then its CDF and $CCDF$ are also log-concave. Since we consider a log-concave CDF and $CCDF$ distribution for the allocated bandwidth in this paper, constraints (1.7)-(1.11) are convex. In particular, Eq. (1.7) ensures that the probability of realizing the sum of allocated bandwidth of the min-cut is no less than P_C. Although Eq. (1.7) takes the sum of the $-\log$ over all the links in the network, if link (u,v) does not belong to the cut in the optimal solution, the convex optimization formulation will "force" it to achieve its maximum value with probability equal to $CDF(b^{\max}_{(u,v)}) = 1$. In this sense, Eq. (1.7) only calculates the bandwidth allocating probability of the links belonging to the cut.

It remains to show that Eq. (1.6) is convex. In general, the product of two convex functions is not always convex, however, according to [5, pp. 119], one special case is: "If functions f and g are convex, both nondecreasing (or nonincreasing), and positive (nonnegative) functions on an interval, then $f \cdot g$ is convex." Therefore, for each $(u,v) \in \mathcal{L}$, $f(u,v) \cdot y_{u,v}$ is convex.

Appendix B
More samples may reduce performance

Although, typically, increasing the number of samples k leads to better performance, we will illustrate that in some cases it may reduce the performance.

In the uniform distribution, the Complementary Cumulative Density Function ($CCDF$) of a link l with maximum bandwidth value b_l^{max} can be expressed as:

$$CCDF = \frac{b_l^{max} - b_l}{b_l^{max}} \tag{2.1}$$

where b represents the allocated bandwidth.

In Fig. 14, there are three nodes and two links with $b_l^{max} = 17$. Our aim is to find a maximum flow from node 1 to node 3 with requested probability $P_B = 0.1$. We compare $k = 2$ with $k = 4$.

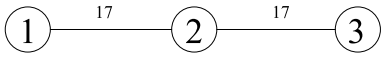


Fig. 14: An example graph to illustrate that a bigger k does not necessarily lead to more flow.

When $k = 4$, the bandwidth of the links is sampled into 3.4, 6.8, 10.2 and 13.6. Since there is only one path (1-2-3) from node 1 to node 3, the algorithm will choose this path. The maximum bandwidth 13.6 of this path cannot be chosen since the total probability of allocating this value for these two links according to Eq. (2.1) is equal to $\frac{17-13.6}{17} \times \frac{17-13.6}{17} = 0.04$, which is less than $P_B = 0.1$. Instead, the algorithm will select 10.2 with probability equal to $\frac{17-10.2}{17} \times \frac{17-10.2}{17} = 0.16$.

When $k = 2$, the bandwidth of the links is sampled into 5.6 and 11.3. Again, since there is only one path (1-2-3) from node 1 to node 3, the algorithm will choose this path. If each link allocates 11.3 bandwidth, then the total probability is equal to $\frac{17-11.3}{17} \times \frac{17-11.3}{17} = 0.112$, which is greater than $P_B = 0.1$, and greater than with $k = 4$.