Probabilistic Analysis of Network Availability

Yunmo Zhang*, Hong Xu†, Chun Jason Xue*, Tei-Wei Kuo‡
*Department of Computer Science, City University of Hong Kong
†Department of Computer Science and Engineering, Chinese University of Hong Kong
‡Department of Computer Science and Information Engineering, National Taiwan University

***Abstract*—In recent years, quantitative verification concerning network availability has received increasing attention due to its practical importance in network management. Existing work extends upon qualitative verification and tries to answer whether a network would have any link overload under failures and dynamic traffic. The simple yes-or-no question falls short of accurately characterizing the robustness of a network under uncertainties, which the operators are in dire need of. Thus, we argue that it is necessary to design a probabilistic framework to analyze network availability comprehensively. We propose Pita, a novel network analysis framework that outputs the overall probability of the network being unavailable under a range of failure scenarios and traffic demands. We formalize the problem and show that it is #P-hard which does not admit deterministic approximation solutions. We further develop an improved randomized approximation that exploits the structural property of our problem to reduce the computational cost of the so-called boundary oracle procedure, a key bottleneck of the approximation, without any accuracy loss. Evaluation with real topologies shows that Pita provides up to 2.25x speedup over state-of-the-art solutions, and can be effectively used in many network management tasks such as identifying high-risk failure scenarios, and aiding robust traffic engineering design.**

I. Introduction

Internet-scale services ranging from search, advertisement, and social networks require stringent performance and availability guarantees from the underlying networks. Yet managing a network at scale is inherently difficult, especially under uncertainties such as failures and dynamic traffic. In this regard, various verification tools have been developed to verify that certain important properties can be guaranteed in the network. Most of them focus on verifying *qualitative* properties that are path-related [4], [15], [25], [32], e.g., reachability [32], path isolation (two paths do not share any links) [25], waypointing (traffic traverses a chain of services) [4], etc.

Very recently, *quantitative* verification has started to gain attention due to their imminent practical values. This line of work concentrates on the issue of *network availability* under failures where traffic is re-routed to other links, and with dynamic traffic. For example, given the range of traffic fluctuation, QARC [42] tells the operators if the network is guaranteed congestion-free or overload-free which reflects network availability. Using the same problem setup, Chang et al. [7] calculate the maximum link utilization to reflect how severe the overload is if any.

We observe that simple yes or no answers in these work are useful but still far from accurately and comprehensively characterizing network availability under dynamics, which the operators are in dire need of. Take Fig. 1 as an example. When

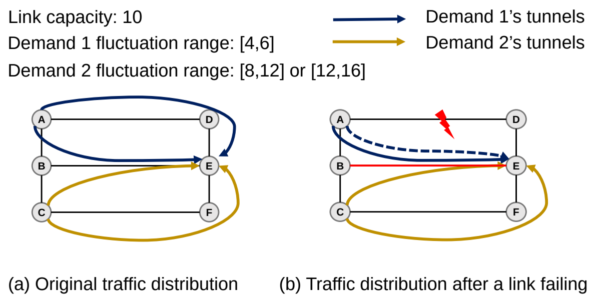
Link capacity: 10
Demand 1 fluctuation range: [4,6]
Demand 2 fluctuation range: [8,12] or [12,16]
Demand 1's tunnels
Demand 2's tunnels
A
B
C
D
E
F
(a) Original traffic distribution
(b) Traffic distribution after a link failing

Fig. 1: An example of profiling the traffic load under a failure and dynamic traffic. When AD fails, BE could be overloaded under both fluctuation ranges of demand 2 but with different probabilities (0.5 and 1, respectively).

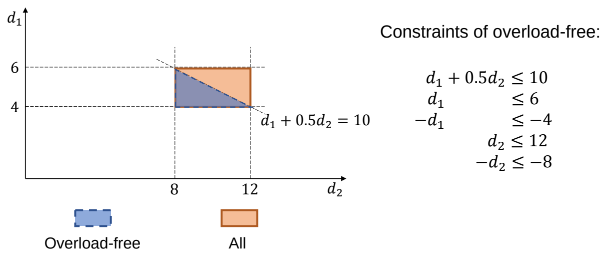

Fig. 2: If demand 2's range is $[8, 12]$, the probability of BE being overload-free after a failure is 0.5, which is obtained by computing the volume of a convex polygon (2-dimensional polytope).

link AD fails, demand 1's traffic on tunnel A-D-E is re-routed to the remaining tunnel A-B-E. Assuming the tunnel splitting weights of the two demands are both 1:1, demand 2's traffic on link BE varies between 4 to 6, and demand 1's traffic is also 4 to 6. In this case, existing solutions determine that the network is not overload-free since it is possible for the overall traffic to exceed BE's capacity. This analysis is not accurate since it is also entirely possible (with probability 0.5 as shown in Fig. 2) for BE to be not overloaded. If demand 2's range is changed to $[12, 16]$ (demand 2's traffic on link BE is 6 to 8), BE is still not overload-free in existing solutions, but in fact it is overloaded in all cases with probability one, and BE availability vanishes to zero in this scenario.

Thus, we argue that verification ought to give quantitative results to accurately outline network availability, that is, to answer the question "how likely is the network going to be overload-free", in addition to the question of whether the network can be guaranteed overload-free. This can be useful for many network management tasks, such as network design and planning, traffic engineering, failover mechanism design, etc. A probabilistic framework naturally fits here to capture the dynamic traffic demands, and their interactions due to failures and re-routing.

It is quite challenging to probabilistically characterize network availability. Consider a much simplified problem of computing the probability of a single link being overload-free given the specific failures, re-routing scheme, and demand ranges. As each demand is continuous, this is equivalent to computing a Lebesgue integration of a measurable set, or geometrically, computing the volume of a polytope defined by the ranges of re-routed demands on this link, and the constraint that total traffic does not exceed link capacity. As the number of demands (i.e. dimensions of the polytope) is large, this problem can be shown to be #P-hard which does not admit deterministic approximation solutions (see §III for details). The probability of the network being overload-free is clearly even more difficult, because links are not independent of each other for the overload event due to routing.

As a result, we resort to randomized approximation to obtain the network overload-free probability under a given failure scenario. Specifically, we resort to Multiphase Markov Chain Monte Carlo (MCMC), a common framework for approximating the volume of high-dimensional convex bodies [10], [11], [24], [31]. On a high level, it works by defining a sequence of convex bodies such that multiplying the ratios of the volume of two consecutive bodies yields the volume of our target body, which effectively captures all the cases of the network being overload-free after failure. Each volume ratio is then estimated by MCMC using a random walk algorithm to sample points.

The challenge here is that this approach has prohibitively high computational cost even with state-of-the-art solutions [31], especially considering that a WAN has many demands in general. The main bottleneck is the so-called boundary oracle procedure which checks if a point generated by random walk is inside the convex body or not. Thus we design an optimization to substantially reduce the complexity of boundary oracle that does not need any assumptions on network topology, traffic engineering, or failures. Our key insight is that the polytope that geometrically represents our problem constraints, although irregular such that we have to resort to a general Multiphase MCMC, has many special hyperplanes that provide the opportunity to optimize the general algorithm. Specifically, the hyperplanes representing the demand ranges are parallel to the possible random walk directions (axes), which implies they can be safely bypassed in the boundary oracle checking. We further show that a substantial number of constraints or hyperplanes satisfy this property, making it possible to reduce the overall cost of this procedure.

Based on the improved boundary oracle, we propose a new probabilistic analysis framework which we call Pita. We perform realistic evaluations of Pita using real WAN topologies, traffic engineering schemes, and failover mechanism. The results show that Pita is up to 2.25x faster than existing solutions, and the speedup is more salient as the network scales up. We further demonstrate that with Pita, operators can quantitatively identify the high-risk failure scenarios, and can aid the TE selection in terms of the resulting network availability.

We make the following contributions.

- We identify probabilistic verification of a traffic-related property, namely availability, as a new and important area of verification research with imminent practical value.
- We propose a new probabilistic analysis framework, Pita, as the first step towards quantitative verification of network availability. We systematically attack the problem, by first showing its theoretical #P-hardness, then adopting the classical Multiphase MCMC for randomized approximation. We further design an improved boundary oracle procedure to reduce the computational cost.
- We present realistic evaluations of Pita to show its effectiveness and potential applications in network management.

II. Related Work and Motivation

In this section, we first present related work on traffic management and show that though taken seriously in the literature, availability is still impossible to guarantee in practice for many reasons. This motivates us to seek to more precisely characterize availability through network analysis. We then introduce a simple taxonomy of existing verification work and position our probabilistic analysis of availability in contrast to them. Finally, we present a few usecases to demonstrate how operators may find our tool useful for network management.

A. *Traffic Management Cannot Guarantee Availability*

Before failures. WANs apply centralized traffic engineering (TE) to split the aggregated traffic of a pair of source and destination across multiple pre-established tunnels [20], [28], [43]. However, failures in both control and data plane are common in production WANs [17], [35], and can degrade network availability with link overload. Thus, a line of work represented by FFC [28] proposes to proactively handle failures in the TE formulation. The basic idea is to reserve some bandwidth headroom at each link so that even with failures, overload and congestion would be minimized when traffic still follows the original TE plan. This trades efficiency for availability as part of the expensive WAN bandwidth has to be left idle most of the time [44], and to better guarantee availability more bandwidth needs to be set aside which is undesirable [20], [23], [38].

After failures. In addition to TE, failover mechanisms are employed to rapidly redirect traffic away from the failed links. There are two common approaches. One is local rescaling with ECMP [22], the default load balancing scheme in data plane. Upon detecting a failure, ingress switches remove the tunnels using the failed links and re-distribute the affected traffic among the remaining tunnels (of the same source-destination pair) according to the original splitting ratios. This restores connectivity swiftly, but links may be overloaded since the original TE plan is not optimized for the new topology.

The other mechanism is to use pre-computed backup tunnels once failures occur [21], [44]. Backup tunnels are computed in addition to the regular TE to ensure victim traffic can be maximally re-distributed without causing congestion to the remaining tunnels. They are installed in the switch flow tables in advance and activated upon detecting a corresponding failure. Compared to local rescaling, backup tunnels recover traffic more effectively. However, the fact that only a small number of forwarding rules are supported in commodity switches constrains the deployment of this approach [44].

To quickly recap, TE solutions improve network availability but cannot guarantee it due to the exponentially many failure cases to consider and the extremely high resource requirements (bandwidth, flow table entries) to bear. In this sense, it becomes imperative to instead quantitatively characterize availability in face of failures and traffic dynamics for operators to make their tradeoffs, for which formal analysis is the primary means.

B. Existing Verification is Insufficient

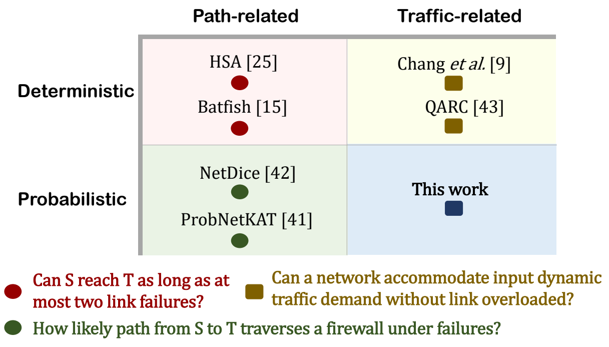
Path-related
Traffic-related
Deterministic
Probabilistic
HSA [25]
Batfish [15]
Chang et al. [9]
QARC [43]
NetDice [42]
ProbNetKAT [41]
This work
Can S reach T as long as at most two link failures?
Can a network accommodate input dynamic traffic demand without link overloaded?
How likely path from S to T traverses a firewall under failures?

Fig. 3: Pita compared to existing work on verification. We also show the verification questions each work aims to answer.

A rich literature exists on network verification. We categorize the most related work to ours on two dimensions, the target properties, and the analysis approach as depicted in Fig. 3. Verification has been focusing on *path-related* properties such as reachability and waypointing that depend on topological information and switch configurations [4], [14]–[16], [25], [32]. Recent work such as QARC [42] and Chang et al. [7] extends the scope and explores target properties that are also *traffic-related*, for example network availability in terms of link load conditions. Inherently, availability by definition concerns *both* connectivity and bandwidth, i.e. link load [19], [21], which implies that traffic dynamics has to be taken into consideration in addition to path-related properties. These two lines of work share the same conventional approach which is *deterministic* verification. That is, one studies the worst-case scenario under different failure cases and verifies if the target property can be guaranteed or not.

As argued in §I, deterministic verification is insufficient for operators to comprehensively and accurately understand the impact of network uncertainties and the exact robustness level of their network. Thus, some recent work such as NetDice [41] and ProbNetKAT [40] has adopted a *probabilistic* approach to verification: they compute the probability that the target property is going to hold in all possible cases. Due to the complexity of probabilistic verification, they constrain themselves to simple path-related properties again for which probability may be obtained by enumeration. Moreover, the stochastic branch of network calculus [8] models statistical multiplexing and scheduling to obtain performance bounds on quantitative properties such as delay and congestion. Yet, its fine-grained packet-level modeling inherently hinders it from being applied to model traffic management strategies and to reason about availability.

We explore the unchartered territory of probabilistic verification of traffic-related properties, i.e. network availability, in this work. The essential question we want to answer is, what is the probability that the network is still available without any link overload under failures and dynamic demands? The interaction across the varying network demands and its impact on link load makes the probabilistic analysis much more challenging as introduced in §I and will be formally shown in §III. Before that, we present a few usecases of probabilistic analysis of network availability to motivate its practical relevance and importance.

C. Usecases

The following tasks in network management can directly benefit from probabilistic analysis of availability.

Network design and planning. The network needs to be constantly upgraded in response to the rapid growth of demands [18]. Operators need to determine where to add links and how much capacity to provision, which are very difficult to change afterwards. Thus, proactive probabilistic analysis can provide quantitative measures to evaluate candidate network designs under uncertainties and make better decisions.

Tunnel selection. Tunnel selection is essential in TE. A good choice of tunnels improves the performance and robustness of the network. Currently TE tunnels are selected heuristically based on path distance and/or cost. Given a probabilistic analysis of the network availability using different tunnels, operators can now assess and thereby select them more strategically. It also applies to backup tunnel selection for failover.

Service differentiation. A quantitative analysis of network availability further allows operators to orchestrate different applications running over their WANs to make their own tradeoff decisions according to their needs, thereby enabling service differentiation.

III. Problem Definition and Challenges

We now formulate our problem and highlight its technical challenges.

A. Model

Let (V, E) be the network topology, where the nodes V represent routers, and edges E represent directed links. Each link e has a capacity C_e. D is a set of independent demands, where each demand d has (aggregated) traffic volume ν_d. Similar to previous work [7], [42], we model ν_d as a continuous random variable uniformly distributed in its range $[L_d, U_d]$, where L_d is a lower bound and U_d is an upper bound known a priori. A TE plan specifies the traffic splitting weights W_d^t across the tunnels T_d for each demand d such that $\sum_{t \in T_d} W_d^t = 1$.

B. General Problem Formulation

We now formally define the overload-free property and the network analysis problem with regards to this property in our probabilistic framework.

Definition 1. ***Overload-free property*** ϕ_f. *Given a failure scenario f, where f is a set of failed links $f \subseteq E$, let $G^f = \{g_e^f | e \in E\}$ be a set of functions where each function g_e^f takes as input a vector of demands $\nu = [\nu_1, \nu_2, \dots, \nu_n]$, where $L_d \leq \nu_d \leq U_d, \forall d \in \{1, ..., n\}$, and outputs the resulting traffic load on link e.*

The overload-free property is defined as:

$$\phi_f = \bigwedge_{e \in E \setminus f} (g_e^f(\nu) \leq C_e), \tag{1}$$

Clearly, a network is overload-free if and only if all the links (other than the failed ones) are not overloaded. We assume that failed links lose their capacity completely and are removed from the topology. The key here is the functions G^f, which encode the traffic management scheme that determines the total link load (in the steady state). We refer to them as traffic management (TM) functions hereafter. Before we present the exact forms of TM function, let us first define the overload-free probability of the network for a given failure scenario f which does not depend on the details of $g_e^f(\cdot)$.

Definition 2. ***Overload-free probability*** $Pr(\phi_f)$. *Given ϕ_f in Eq. (1), the overload-free probability is:*

$$Pr(\phi_f) = \frac{\mu(R_f)}{\prod_d (U_d - L_d)}, \tag{2}$$

where R_f is the set of demand values for which the overload-free property ϕ_f under f holds, that is,

$$\{\nu \mid \phi_f \bigwedge_d \nu_d \leq U_d \bigwedge_d -\nu_d \leq -L_d\}, \tag{3}$$

and $\mu(R_f)$ is the Lebesgue measure of R_f.

R_f can be regarded geometrically as a high-dimensional convex body enclosed by hyperplanes defined by inequalities (3), and $\mu(R_f)$ is the volume of this convex body. Therefore, $Pr(\phi_f)$ is the ratio of two volumes, namely the volume of R_f and the volume of a hyperrectangle defined by the ranges of demands.

Our analysis problem can then be formulated.

Definition 3. ***Network availability analysis****. Given a set of failure scenarios F, our analysis procedure computes the probability of the network being available, i.e. overload-free:*

$$Pr(\phi) = \sum_{f \in F} Pr(\phi|f)Pr(f) = \sum_{f \in F} Pr(\phi_f)Pr(f). \tag{4}$$

We expect the failure set F and failure probabilities $Pr(f)$ to be given by operators. The main challenge is then to figure out the (failure-specific) overload-free probability $Pr(\phi_f)$. For tractability, we consider at most k link failures as much prior work has similarly done [28]. To differentiate from the (failure-specific) overload-free probability $Pr(\phi_f)$, we refer to $Pr(\phi)$ as the *network availability* hereafter.

Now let us present the concrete form of the TM function $g_e^f(\cdot)$. A TM function has two parts. First, it captures the traffic on link e following the TE plan defined by tunnel splitting weights. In addition, it models the failover mechanism which re-routes the victim traffic to e in response to failures. We use an indicator variable H_e^t to describe if tunnel t traverses through link e or not. Then the TM function can be written as:

$$g_e^f(\nu) = \sum_{d \in D, t \in T_d} \nu_d H_e^t W_d^t (1 - S_f^t) + Re(f, e), \tag{5}$$
$$\text{where } S_f^t = \mathbb{1}_{(\sum_{e' \in f} H_{e'}^t) > 0}.$$

Here S_f^t is an indicator that equals 1 when tunnel t is affected by the failure scenario and 0 otherwise, and $Re(f, e)$ is the function that returns the re-distributed traffic load caused by failure scenario f on link e. We present $Re(f, e)$ that models the two failover mechanisms we introduced in §II.

- **Local rescaling.** If tunnel $t \in T_d$ is affected by f (i.e. $S_f^t = 1$), demand d's traffic on it is re-routed to the remaining tunnels in T_d. The amount of traffic to be re-routed is thus $\sum_{t \in T_d} \nu_d W_d^t S_f^t$. Since rescaling uses the original splitting weights, traffic re-routed to a remaining tunnel t' has a relative weight of $\frac{W_d^{t'}}{\sum_{t \in T_d} (1 - S_f^t) W_d^t}$ among all remaining tunnels. Thus, the re-distributed traffic on link e due to f is:

$$Re(f, e) = \sum_{d \in D} \Bigg(\frac{\sum_{t' \in T_d} (1 - S_f^{t'}) H_e^{t'} W_d^{t'}}{\sum_{t \in T_d} (1 - S_f^t) W_d^t} \times \sum_{t \in T_d} \nu_d W_d^t S_f^t \Bigg). \tag{6}$$

- **Backup tunnels.** For a failure scenario f where backup tunnels are pre-computed, traffic on an affected tunnel t is re-directed to its backup tunnel $b_{t,f}$. Notice here we assume at most one backup tunnel is used for each original tunnel due to the switch's limited rule space [44]. So we have

$$Re(f, e) = \sum_{d \in D} \sum_{t \in T_d} H_e^{b_{t,f}} S_f^t W_d^t \nu_d. \tag{7}$$

C. Challenges of Computing Overload-Free Probability

We discuss the challenges of computing $Pr(\phi_f)$ in this section, by first establishing the hardness of the problem, followed by presenting the limitations of prior work and our key idea.

In fact the overload-free probability $Pr(\phi_f)$ is intractable to compute as a counting problem. We prove that probabilistic analysis of ϕ_f (a #SMT problem) is #P-hard [3], as stated below.

Theorem 4. *The overload-free probability $Pr(\phi_f)$ is #P-hard to compute.*

Proof. We prove it by reducing the #P-hard problem of volume computation of polytopes [11] to $Pr(\phi_f)$. The overload-free convex body R_f defined in Eq.(3) can be written as $\{\nu | A\nu \leq b\}$, $A \in \mathbb{R}^{m \times n}$, where m is the number of inequalities in R_f

and n is the dimension of the demand vector ν. This is because ϕ_f consists of TM functions defined by Eq. (5)–(7) which are clearly affine. Thus, R_f is a n-dimensional polytope defined by m hyperplanes, and according to Eq. (2), the problem of polytope volume computation can be reduced to our problem of calculating $Pr(\phi_f)$ and vice versa. $\square$

#P-hard problems are more difficult than NP-hard, because they consist of counting problems associated with the decision problems in NP [3]. In other words, determining whether there is an assignment satisfying SMT constraints is NP-hard, while counting "how many" assignments satisfy SMT constraints is #P-hard. Further, due to the continuous and non-enumerable nature of dynamic demands [7], our problem is a continuous model counting on SMT constraints. Therefore we note that probabilistic verification methods based on model checking [27] are not applicable here.

Specifically, notice that n here is the number of dynamic demands in the network which is at most $|V|^2$, and the number of hyperplanes m is $O(|E|+2n)$. Thus the polytope R_f related to the overload-free probability $Pr(\phi_f)$ is extremely complex with high dimensions and with number of hyperplanes increasing in n. It has been shown that when the number of dimensions n is larger than 15, the polytope's volume cannot be exactly computed [6]. Further, no deterministic approximation scheme is known for finding the volume in the general case [31]. So we turn to randomized approximation that approximates the high-dimensional volume within arbitrarily small error ϵ with high probability in polynomial time.

The common approach to randomized polytope volume approximation is Multiphase Markov Chain Monte Carlo. Readers are referred to the survey [37] or a series of work [10], [11], [24], [31] for more details. Roughly speaking, it works by strategically generating a sequence of convex bodies, estimating the ratio of the volumes of two consecutive bodies by sampling many points using Markov Chain random walk, and computing the polytope volume as the product of all such volume ratios. A series of theoretical work from [10] to [31] has improved the efficiency of randomized approximation of polytope volume from requiring $O(n^{23})$ to $O(n^4)$ *boundary oracle* calls. The boundary oracle is a procedure necessary to ensure the sample points are inside the polytope. Yet this is still inefficient for practical use. The single-process implementation of the state-of-the-art $O(n^4)$ algorithm [31] takes more than two hours for n up to 9 [30], and a GPU implementation only scales to $n = 20$ [33]. This does not work for a reasonable-sized WAN that can easily have more than 100 demands.

D. Our Key Idea

We notice that the overall computational cost of polytope volume approximation has two parts: (i) the number of boundary oracle calls which most theoretic work focuses on, and (ii) the number of operations required per oracle call, i.e., the *oracle complexity*. Since the former is difficult to reduce, we concentrates on mitigating the oracle complexity in this work.

Recall that the polytope boundaries are defined by m hyperplanes. Our key insight is that we do not need to compute the position relationship of a point with all the hyperplanes. We find that a large number of these hyperplanes are actually parallel to the random walk direction, in which case the sample point could not step outside the boundary defined by this hyperplane. Thus we can safely bypass checking these hyperplanes and save the associated computation cost. We will show that this can reduce the oracle complexity of our random walk algorithm from $O(m) = O(|E| + 2n)$ to $O(|E|)$ in our problem.

IV. Our Solution

In this section, we present the complete design of Pita, a probabilistic analysis tool on network availability. We first present the overall framework based on Multiphase Markov Chain Monte Carlo (Multiphase MCMC) (§IV-A). Next we introduce our optimization to reduce the computational cost of the random walk oracle (§IV-B). Finally, we present a technique to rapidly compute the overload-free probability in special failure scenarios (§IV-C).

A. Overview

Algorithm 1 shows the overall working of Pita. Pita takes as input network topology, TE plan, failover mechanism, demand ranges, failure scenarios and probability model, and approximates the overload-free probability $Pr(\phi_f)$. First we check if f belongs to a special case where the overload-free probability degenerates to an SMT problem which can be readily handled (§IV-C). In the general case when it is not an SMT problem, $Pr(\phi_f)$ is approximated following the common Multiphase MCMC framework with our revised boundary oracle (§IV-B). Finally, Pita produces the overall availability $Pr(\phi)$.

Algorithm 1 Pita: Computing availability $Pr(\phi)$

```
Input: The network topology $(V, E)$, tunnels $T_d$ assigned for each
    $d \in \{1, ..., n\}$ with weight $W_d^t$ for each $t$, dynamic demands
    $\{\nu_d\}$ with $[L_d, U_d]$ for each $d$, failures scenarios $F$, failure model
    $\{Pr(f)\}$ for $f \in F$, and failover mechanisms.
Output: Approximation of availability $Pr(\phi)$.
 1: $Pr'(\phi) \leftarrow 0$;
 2: for $f \in F$ do
 3:     Calculate $\phi_f$;                                  ▷ §III-C
 4:     $Pr(\phi_f) \leftarrow$ SMT_Check($\phi_f$);         ▷ Algorithm 3
 5:     if $Pr(\phi_f)$ is unknown then
 6:         Approximate $Pr'(\phi_f)$;   ▷ Algorithm 4 in Appendix §A
 7:     end if
 8:     $Pr'(\phi) \leftarrow Pr'(\phi) + Pr'(\phi_f)Pr(f)$;
 9: end for
10: return $Pr'(\phi)$;
```

We briefly introduce the Multiphase MCMC framework here. Algorithm 4 in Appendix §A has the complete details. We construct a sequence of convex bodies $K_\alpha \subseteq K_{\alpha+1} \subseteq ... \subseteq K_{\beta-1} \subseteq K_\beta = R_f$, where the first body K_α has known volume and the last body K_β is our target polytope R_f. Each convex body is the intersection of R_f and a n-ball B_i. The smallest ball corresponding to K_α is the Chebyshev ball of R_f, i.e. the largest ball totally enclosed by R_f, and the largest ball corresponding to K_β is a ball (almost) enclosing R_f (Line 4 in Algorithm 4). Then the polytope volume $\mu(R_f)$ is given by

Algorithm 2 RandomWalk(p, K, w)

```
Input: A point $p$ in the convex body $K$, the walk length $w$.
Output: A new point $p' \in K$.
1: for $i = 1$ to $w$ do
2:     Pick a line $l$ with a random uniformly distributed coordinate direction through $p$;
3:     Compute chord $ch = l \cap K$;                    ▷ §IV-B
4:     Pick a random point $p'$ uniformly distributed on $ch$;
5: end for
6: return $p'$;
```

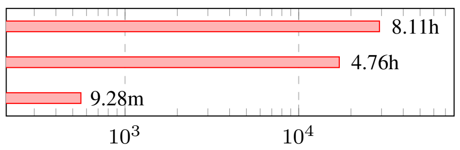

Fig. 4: Running time (seconds) of the Multiphase MCMC for $Pr(\phi_f)$ (Algorithm 4 in Appendix §A) with different random walk algorithms for a network with $n = 100$.

the product of volume ratios between two consecutive convex bodies. To compute the ratio, say $vol(K_i)/vol(K_{i-1})$, we generate many uniformly distributed points in K_i by Markov Chain random walk, and count the fraction of them falling in K_{i-1}.

Random Walk. Before we optimize the oracle complexity involved in each step of the random walk, we need to first determine which random walk algorithm should be used for our problem. We focus on hit-and-run random walk which yields the fastest algorithms today [5], [12], [29], [39]. There are two variants: Random Direction Hit-and-Run (RDHR) [5] and Coordinate Direction Hit-and-Run (CDHR) [39]. Given the current point p, both RDHR and CDHR determine the next point $p' \in K$ by picking a random line l through p, and moving p to a random point p' uniformly distributed on the chord $K \cap l$. The difference is that RDHR picks l with a direction uniformly distributed in the space, while CDHR picks the directions uniformly from all the axes (i.e. the line is parallel to one axis).

To choose between RDHR and CDHR, we note that RDHR's oracle complexity is $O(mn)$ operations per oracle, while CDHR's is $O(m)$ [2]. To verify the impact of oracle complexity on the actual running time of our solution, we also conduct an empirical experiment with different random walk algorithms. We set the walk length according to its empirical bound suggested in [12], [13] and use the same error parameter ϵ. As shown in Fig. 4, CDHR is 31× faster than RDHR and 54× faster than ball walks in analyzing a network with 100 demands. Thus we choose CDHR as the random walk algorithm in our framework.

B. An Optimization for CDHR: OptHR

We now present our optimization to CDHR called OptHR, which reduces its oracle complexity from $O(m) = O(|E|+2n)$ to $O(|E|)$ without any accuracy loss in our problem.

We find that the main computational overhead in oracle is computing the intersection points of line l with R_f, i.e., m hyperplanes in $K = R_f \cap B$, since the intersection of l with

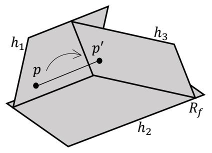

Fig. 5: A point could not walk outside the boundary defined by hyperplanes parallel to the direction of a step.

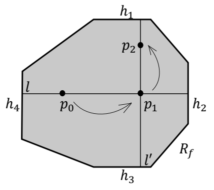

Fig. 6: Updating a point in OptHR at the first and second steps. The shown polytope is of $n = 2$, $m = 8$.

a n-ball B is straightforward. This is necessary to check if the sample point is within R_f or not. Our insight is that we do not need to traverse every hyperplane as in the original CDHR. As shown in Fig. 5, if the walk direction of p is parallel to hyperplane h_1 and h_2, it is clearly impossible for p' to step outside the boundaries defined by h_1 and h_2, and we can safely bypass this hyperplane in the computation without any issue.

More formally, recall that $R_f = \{D \in \mathbb{R}^n | AD \le b\}$, where $A \in \mathbb{R}^{m \times n}$. Let p_0 be the current (initial) point and $l = \{y \in \mathbb{R}^n | y = p_0 + \lambda v, \lambda \in \mathbb{R}\}$ be the randomly selected coordinate line through p_0, where $v \in \mathbb{R}^n$ is the direction of l. Fig. 6 shows an example. Then, we use $p^{\pm} = p_0 + \lambda^{\pm} v$ to represent the intersection points between l and R_f, where

$$\lambda^{\pm} = \max\{\pm\lambda | A(p_0 \pm \lambda v) \le b\}, \tag{8}$$

namely $\lambda^{\pm}$ have the largest absolute value making points $p_0 \pm \lambda v$ on l within the area $AD \le b$. To obtain the intersection points $p^{\pm}$, we only need to compute $\lambda^{\pm}$. Since a point walks in the coordinate direction, v is a coordinate vector. Let j be the walk coordinate of current step in CDHR, i.e., $v = e_j$, where e_j is the j-th unit coordinate vector. Then Eq. (8) becomes $\pm\lambda A v = \pm\lambda A_{*j} \le -Ap_0 + b$, where A_{*j} is the j-th column of A. Let $z = -Ap_0 + b \in \mathbb{R}^m$, then in the original CDHR,

$$\begin{aligned} \lambda^+ &= \min_{1 \le i \le m} \max_{\lambda} \{\lambda > 0 | \lambda a_{ij} \le z_i\} \\ \lambda^- &= \max_{1 \le i \le m} \min_{\lambda} \{\lambda < 0 | \lambda a_{ij} \ge -z_i\}, \end{aligned} \tag{9}$$

where a_{ij} is the i-th element of A_{*j}, and z_i is the i-th element of z. Thus, getting $\lambda^{\pm}$ in CDHR requires $O(m)$ (arithmetic and comparison) operations. In OptHR, assume there are s hyperplane parallel to e_j and without loss of generality, they are the 1-th to s-th hyperplanes, we obtain $\hat{\lambda}^{\pm}$ by

$$\begin{aligned} \hat{\lambda}^+ &= \min_{s+1 \le i \le m} \max_{\lambda} \{\lambda > 0 | \lambda a_{ij} \le z_i\} \\ \hat{\lambda}^- &= \max_{s+1 \le i \le m} \min_{\lambda} \{\lambda < 0 | \lambda a_{ij} \ge -z_i\}. \end{aligned} \tag{10}$$

Lemma 5. $\lambda^+ = \hat{\lambda}^+, \lambda^- = \hat{\lambda}^-$.

Proof. We prove the case for λ^+ and $\hat{\lambda}^+$ for brevity. Since e_j is parallel to s hyperplanes, $A_{q*} \cdot e_j = a_{qj} = 0$ for $q = 1, ..., s$ where A_{q*} is the q-th row of A. Notice p_0 is within the boundary, i.e., $Ap_0 \le b$, so $z_i \ge 0$ for $\forall i$. Thus $\hat{\lambda}^+$ in Eq. (10) also satisfy $\lambda a_{qj} \le z_i$ for $i = 1, ..., s$. □

Now if the number of such hyperplanes that can be skipped s is substantial, this can save a lot of computational cost. Fortunately, we can show that this is the case in our problem.

Lemma 6. *For any R_f defined by Eq. (3), there are $2n-2$ hyperplanes parallel to the random walk direction of CDHR at each step.*

Proof. We prove it by constructing a mapping between a unit coordinate vector and a set of hyperplanes. For a unit coordinate vector e_j, let $\tau : e_j \to \Psi_j$, where $\Psi_j = \{\nu_d \le U_d, -\nu_d \le -L_d\}_{d=1,\dots,j-1,j+1,\dots,n}$. Assume a hyperplane in Ψ_j is the q-th hyperplane in R_f. We have $a_{qi} = \{1, -1\}$ for $i = d$ and $a_{qi} = 0$ for other i. Obviously, $j \neq d$. Thus $a_{qj} = 0$. Then this hyperplane is parallel to e_j because $A_{q*} \cdot e_j = 0$. Notice that $|\Psi_j| = 2n-2$. □

There are some other details to conclude that the oracle complexity of OptHR is $O(|E|)$. Notice that at the first step of random walk, aside from $\lambda^{\pm}$ in $O(|E|)$ operations, obtaining $z = Ap_0 + b$ consumes $O(mn)$ operations. Thus, updating p_0 requires $O(mn)$ operations in total. Yet this only holds for the first step. Let $p_1 = p_0 + ce_j$ be the second point walked from p_0, where $c > 0$ is the randomly chosen step size at the first step. Let p_1 walks in $v' = e_k$ at the second step. The new $\lambda^{\pm}$ is obtained by maximizing $\pm\lambda A_{*k} \le -Ap_1 + b = z'$. This time, the computation of z', which is $z - cA_{*j}$, takes $O(|E|)$ operations (we omit this proof as it is very similar to that of Lemma 5).

Furthermore, as can be seen in Algorithm 2, we obtain the N sample points for the volume ratios as follows: starting from an interior point, when we perform w random walk steps, we take the last point as a sample point. So the expensive boundary oracle call with $O(mn)$ complexity is only done once; after that all the Nw boundary calls are in $O(|E|)$ each. As the total walk steps Nw is far larger than mn, the overall (amortized) cost of each oracle is $O(|E|)$.

Theorem 7. *The oracle complexity of OptHR is $O(|E|)$.*

We can further show the overall complexity of obtaining $Pr(\phi_f)$ with OptHR. Specifically,

Proposition 8. *Computing $Pr(\phi_f)$ (Algorithm 4) with OptHR consumes $O(|E|n^3 \log n \log(\rho_{max}/\rho_{min}))$ operations.*

Proof. According to Algorithm 4 in Appendix §A, we need a total of $n \log \rho_{max}/\rho_{min}$ convex bodies. For each of them, we need to use $N = O(n \log n)$ sample points for the volume ratios. Each sample point is generated after $w = O(n)$ random walk steps with each step consuming $O(|E|)$ operations by OptHR. Putting everything together, the algorithm needs $O(|E|n^3 \log n \log(\rho_{max}/\rho_{min}))$ operations. □

In contrast, $O((|E|n^3 + n^4) \log n \log(\rho_{max}/\rho_{min}))$ operations are needed for CDHR to compute $Pr(\phi_f)$ [2]. Here ρ_{max}/ρ_{min} is fixed given R_f as defined in Eq. (3), which reflects the problem structure related to the network topology, capacity of links and TE. Therefore, the efficiencies of both CDHR and OptHR are dominated by terms associated with the number of dynamic demands n. From the above analysis, we can conclude that the performance of OptHR necessarily exceeds CDHR. Recall that n is at most $|V|^2$. Thus, n grows quadratically with the network scale increasing, which makes OptHR overshadow CDHR even further.

C. Special Cases When $Pr(\phi_f)$ Can Be Determined

Another factor of the complexity of availability analysis $Pr(\phi)$ is the combinatorial nature of the failure scenarios. Consider a WAN with 20 links (40 edges). If operators wish to consider up to 2 link failures, the number of failure scenarios is $20 + \binom{20}{2} = 210$, which implies a naive solution needs to call the highly complex Multiphase MCMC based procedure 210 times to obtain the overload-free probability in each scenario.

We notice that in our problem there exists some special cases in which the overload-free probability is either 0 or 1 and does not require running Multiphase MCMC. Consider the running example in Fig. 1, and suppose demand 2's range is $[9, 12]$. If the splitting weights of tunnels C-B-E and C-F-E is 1:2, obviously the network never would be overloaded when link AD fails and local rescaling is used. On the other hand if the weights are 2:1, the network is definitely overloaded. These cases do not require any randomized approximation to obtain an intuitive answer to the overload-free probability.

To detect these special cases, we can express the corresponding SMT constraints as follows. For the case when the network is never overloaded, the constraint can be expressed as:

$$\phi_f^+ := \big(\bigwedge_d \nu_d = U_d\big) \ \wedge \ \phi_f, \tag{11}$$

that is, the network is overload-free under f even when all demands are at their maximum values. Similarly, the SMT constraint for the case $Pr(\phi_f)$ is 0 is

$$\phi_f^- := \big(\bigwedge_d \nu_d = L_d\big) \ \wedge \ \neg\phi_f. \tag{12}$$

When there is at least one overloaded link when all the demands are at their minimum, $Pr(\phi_f) = 0$.

SMT problems are typically NP-hard [9] and solving them is intrinsically more efficient than that of #SMT problems. Thus we leverage modern SMT solvers to directly verify if the constraints ϕ_f^+ and ϕ_f^- hold, which is summarized in Algorithm 3. This is used as the first step in the overall solution Algorithm 1 before running the expensive Multiphase MCMC to further reduce the running time.

Algorithm 3 SMT_Check(ϕ_f)

```
1: if $\phi_f^+$ is SAT then
2:     return 1;
3: else if $\phi_f^-$ is SAT then
4:     return 0;
5: else
6:     return $unknown$;
7: end if
```

We comment that techniques from prior work [41] that prune the failure cases cannot be effectively applied to our problem. This is because the nature of traffic-related properties is different from path-related properties. Traffic loads in the network are highly correlated: The failure of one link could affect the

Network	Nodes	Edges	Demands
GridNet	9	40	(max.) 81
Abilene	12	30	(max.) 144
B4	12	38	(max.) 144
ANS	18	50	(max.) 324

TABLE I: Network topologies used in evaluation.

demands on paths that do not involve this link since it may cause traffic from victim demands to be redistributed.

V. Evaluation

In this section, we evaluate the effectiveness of our solution Pita by answering the following questions:

- How quickly can Pita analyze overload-free probability? How does our optimization speed up the analysis process? (§V-A)
- What factors influence the performance of Pita? (§V-A)
- How Pita can be used to (i) identify high-risk failure scenarios, (ii) select appropriate TE schemes? (§V-B)
- How available are real WANs across uncertain network conditions? (§V-B)

We implement Pita in Python and C++ with Z3 [34] as the SMT solver and the open-source implementation of Emiris and Fisikopoulos [12] as the basis for our OptHR-based Multiphase MCMC. Our experiments run on a machine with 64GB RAM and ten CPU cores at 3.7GHz. To evaluate our work on different network scales, we use Google's B4 [23] and three real topologies from Topology Zoo [1]: GridNet, Abilene, and ANS, as shown in Table I. The fourth column is the number of demands used in the evaluation unless indicated otherwise. We use the gravity model [36] to generate traffic matrices for TE such that the resulting link utilization without failures is between 0.6 and 0.8. We evaluate three TE schemes: k-shortest paths (KSP) where a demand's traffic is equally split across the tunnels, maximum flow (MaxFlow), and maximum flow with minimum latency (MinLatency) [23]. These traffic volumes are taken as the lower bounds of the demands, and we set the upper bounds to be 105% of the lower bounds. To characterize the network availability in the most general form, we set four networks to have all their possible demands dynamic. Thus the maximum dimension of polytope n is 324 in this evaluation.

We consider failures with at most 2 links. Moreover, we adopt the failure model that each link in the network fails independently at the probability of 0.001 based on the empirical study [17]. Local rescaling is the default failover mechanism. We set $\epsilon = 1$ as default in all random walk algorithms. All data points report the average of five runs.

A. Pita's Running Time

We now evaluate our solution's running time for analyzing the failure-specific overload-free probability and the overall availability. Recall that as shown in Lemma 5, our solution with OptHR produces the same result as the original CDHR.

Analyzing overload-free probability. We record the time taken to compute $Pr(\phi_f)$ for each failure scenario and report

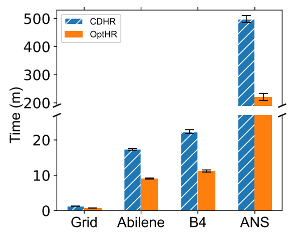

Fig. 7: OptHR is faster than CDHR in analyzing overload-free probabilities across all networks and their failure scenarios.

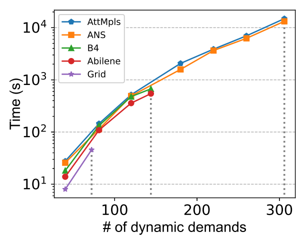

Fig. 8: Time (seconds) respect to the number of dynamic demands in networks. Dashed lines represents the max. demand number.

the average over all failure scenarios and TE schemes of the same network, including probabilities of three TE schemes under 1-link and 2-link failures, as they share the same computation complexity.

Fig. 7 shows the results. We make three observations. *First*, OptHR significantly outperforms CDHR across all topologies upon all failure scenarios, especially with a large number of demands. OptHR improves analysis time by 1.71x, 1.92x, 1.99x, and 2.25x on average, respectively, for the four networks. *Second*, the analysis time generally increases with the number of demands. *Third*, the analysis time is also affected by the network scale, though to a lesser extent: Abilene and B4 have the same demand numbers, while B4 with more edges takes a bit more time.

We further illustrate the last two observations in Fig. 8 which depicts the running time of analyzing a network with different numbers of dynamic demands. This time, we set partial demands as dynamic and keep other demands constants at their lower bounds. We set the number of dynamic demands varying from 40 to the maximum number of demands a network could support (in the 4-th column of Table I). From Fig. 8, the analysis time of the same network grows rapidly when the number of dynamic demands increases. This is as expected since the complexity of our algorithm has the highest power in n (Proposition 8).

Besides, we could find from Fig. 8 that under the same number of dynamic demands, networks with a larger scale have a longer running time, but the increase is limited compared to that brought by more numbers of dynamic demands. To make this observation more convincing, we supplement a larger network AttMpls with 25 nodes and 112 edges and make it under the same set of numbers of dynamic demand as ANS. Notice the blue line and orange line in Fig 8. Even if AttMpls's scale is nearly twice as bigger as ANS's, the running time of a network is 10x when the number of the dynamic demand increases from 100 to 200, while far less than 2x when the network scale becomes twice.

In summary, our answer to **the first question** is that **our domain-specific optimization OptHR reduces the running time sequentially**, with more speedup at larger networks. Pita could analyze failure-specific overload-free probabilities in tens of seconds, minutes, and an hour under forty dynamic demands,

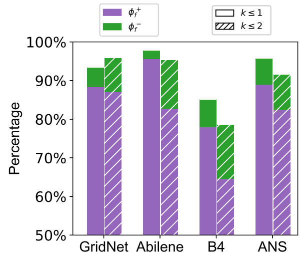

Fig. 9: Fraction of scenarios figured out by SMT_Check in average when analyzing the availability of at most 1-link and 2-link failures.

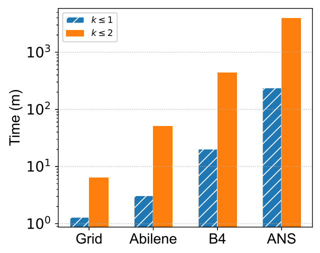

Fig. 10: Average time (minutes) spent on analyzing the overall availability of four real networks under at most 1-link and 2-link failures.

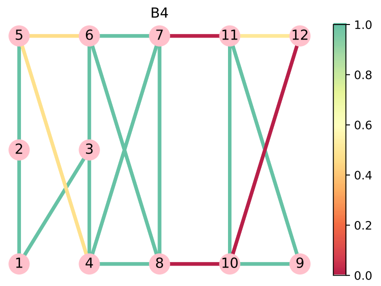

Fig. 11: Overload-free probabilities under the failure of each single link in B4.

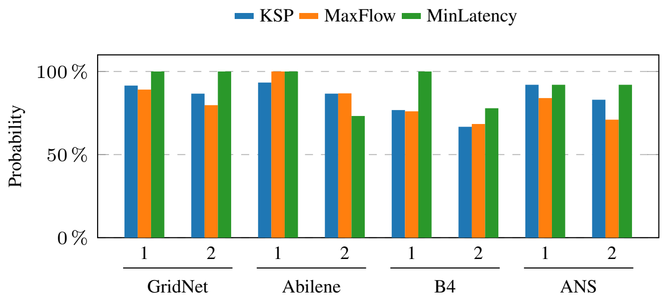

Fig. 12: Overload-free probabilities of networks under 1-link and 2-link failures.

one hundred demands, and two hundred dynamic demands, respectively.

Analyzing overall availability. We now present the time of Pita to analyze overall availability under at most 1-link failure and 2-link failure settings separately. Again, we report the average over all TE schemes.

We first illustrate the fraction of scenarios whose overload-free probabilities are identified directly by SMT_Check in Fig. 9. In our setting, each network has some failure scenarios of either 0% overload-free (ϕ_f^-) or 100% overload-free (ϕ_f^+). These fractions are specific in our setting. They are determined by the traffic situation in the network.

We report the time required for analyzing an overall availability in Fig. 10. Intuitively, networks with less scenarios figured out by SMT_Check take more time on analyzing the availability. B4 requires more than 20 minutes and 7 hours for the validation of at most 1-link and 2-link failures availability, respectively. While the corresponding time for Abilene is less than 10 minutes and 1 hour, albeit Abilene of same dynamic demand numbers as B4.

In general, time on analyzing an availability grows sequentially with k in "at most k-link failures" specified by operators. For ANS, 4 hours and 65 hours are needed for the validation of an availability under at most 1-link and 2-link failures. But we can see from TABLE II that the largest difference of validated availability between two failure spaces is less than 0.01%. So operators can select the target failure spaces based on the required service level and their time budget.

In summary, our answer to **the second question** is that **the running time of Pita is decided by both the efficiency of analyzing a failure-specific overload-free probability** and the fraction of failure scenarios that could not be handled by SMT_Check. The former is mainly determined by the number of dynamic demands in the network. The latter is decided by the traffic situations in the network after failures.

B. Analyzing Real Networks

This part presents the analysis results produced by Pita on four real networks with all their demands being dynamic. To highlight how Pita functions and interesting results found, we first analyze the availability under k-link failures, namely the failure space incorporates all scenarios with k links failing. Given the above failure model, all scenarios of the same k share the same failure probability. We report our results of $k = 1, 2$ in Fig. 12.

Pita discovers that networks undergo varying degrees of risk under uncertain network conditions and convey two main observations. First, the probability of overload increases when more links fail: the best overload-free probability of networks ranges from 92% to 100% under 1-link failure and 77% to 100% under 2-link failures. Second, the choice of TE scheme affects the result to some extent. Interestingly, the best choice is not fixed for different networks in our settings: for GridNet, B4, and ANS, MinLatency is the best, while for Abilene, MaxFlow is the best. So the result of Pita could be another metric for operators to select the appropriate TE scheme. Note that our metric is different from measuring an "availability" given the trace of traffic, which is used in some TE work to evaluate their scheme [26]. We model the dynamic conditions directly and in a more general manner: it does not adhere to any concrete traffic matrix. Thus Pita is more flexible and usable for operators to *proactively* profile their network availability.

Besides, we present the overload-free probability of B4 under single link failure in Fig. 11. The failure of links 8-10 or 10-12 definitely causes network overload. Link 7-11 failure is also dangerous, although it is not 100% leading to network overload. Failures of another three links are not relatively imperative: the overload-free probability exceeds 50% under link 11-12 failure and is no more than 50% under link 5-6 or 5-4 failure. Probabilistic analysis achieves more accurate availability profiling than deterministic verification like QARC [42], which simply and roughly tells operators that all six links mentioned above

Network	KSP		MaxFlow		MinLatency	
	$k \leq 1$	$k \leq 2$	$k \leq 1$	$k \leq 2$	$k \leq 1$	$k \leq 2$
GridNet	99.833%	99.830%	99.786%	99.782%‘	100%	100%
Abilene	99.900%	99.900%	100%	99.999%	100%	99.997%
B4	99.567%	99.561%	99.552%	99.548%	100%	99.996%
ANS	99.805%	99.800%	99.609%	99.601%	99.805%	99.802%

TABLE II: Network availabilities under various TE schemes.

could lead to overload.

In summary, our answer to **the third question** is that **Pita could help operators identify the severity of various failure scenarios** and select an appropriate TE scheme by figuring out the overload-free probabilities given uncertainties concerned.

Based on the above failure-specific overload-free probability results, we now compute the network availability. Recall that we define availability as the overload-free probability under "at most" link failures. This time we adopt the same failure model but normalize the failure probability of a scenario upon all scenarios in F including no failure one such that $\sum_{f \in F} Pr(f) = 1$. For example, if operators consider at most 2-link failures, for a failure scenario f of $k = 1$ happening at Abilene with 15 links (30 directed edges in our network model), $Pr(f)$ is $0.001 \times 0.999^{14} / (0.999^{15} + \binom{15}{1} \times 0.001 \times 0.999^{14} + \binom{15}{2} \times 0.001^2 \times 0.999^{13})$. We report the availability result in TABLE II.

Applications in different service classes require various availability SLOs. For example, there are four service classes in B4 requiring SLO 99%, 99.9%, 99.95%, 99.99%, from the first service class to the fourth service class [21]. So our results are recorded to the third digit after the decimal point to cover these four class requirements. Intuitively, considering more failure scenarios makes analysis characterize availability more precisely. But we observe that considering at most 2-link failures is enough in most cases, which only changes the third digit after the decimal point compared to scenarios under at most 1-link failures.

Again, networks with various TE scheme selections have different availability results. TE scheme in our setting could even affect the service class provided in some networks. For Abilene, the availability under KSP reaches the second service class (99.9%), while under MinLatency reaches the fourth service class (99.99%). Note that availability is determined by network topology, traffic allocations, and network condition variations comprehensively. The above results are produced under our evaluation settings and might differ for other topology and traffic patterns.

In summary, our answer to **the fourth question** is that **in our settings, Pita finds that at the best, three networks could provide the fourth service class** and one network provides the first service class.

VI. Limitations and Future Work

We now discuss limitations of our work and possible future directions.

Independent demands. Our framework can be adapted to demands with dependencies. When the correlation of demands is affine, e.g., complementary or linear growth, the constraints in our problem could be simplified since some demand variables could be represented by their dependent variables. We leave the modeling and analysis of dependent demands to future work.

Uniform demands. Our framework focuses on demands with uniform distribution in their range. To extend to the general case with non-uniform demand distributions, the computation of the problems in these scenarios requires different sampling algorithms corresponding to the new distribution, instead of uniform sampling in the polytope as used in the current approach. We plan to extend Pita to consider non-uniform demands in the future.

VII. Conclusion

We presented Pita, a novel probabilistic analysis tool for network availability. First, we have presented a quantitative analysis framework that operators can use to characterize their network availability under continuous and non-enumerable uncertainties. Second, we have developed a practical algorithm to obtain the overload-free probability without any assumption on network topologies, failures, or traffic variations. Third, we have demonstrated the promise of our framework on real networks and its effectiveness in helping operators to understand network availability comprehensively in a proactive manner. Our results encourage us to extend our work on richer demand patterns, understand more complex network performance under failures and study other challenging tasks in network management in the future.

Acknowledgements This work is supported in part by funding from the Research Grants Council of Hong Kong (11209520).

Appendix A
Multiphase MCMC

Algorithm 4 Multiphase MCMC for $Pr(\phi_f)$

```
Input: Overload-free property $\phi_f$, error parameter $\epsilon$.
Output: The approximation to the overload-free probability $Pr(\phi_f)$.
 1: Calculate $R_f$;                                        ▷ Eq. (3)
 2: $N \leftarrow 400\epsilon^{-2} n \log n$;               ▷ # of points per ball
 3: Compute the Chebyshev ball $B(c, \rho_{min})$;
 4: Compute the largest ball $B(c, \rho_{max})$, where $\rho_{max}$ is the largest
    distance between $c$ and $N$ points generated additionally;
 5: $\alpha \leftarrow \lfloor n \log \rho_{min} \rfloor$; $\beta \leftarrow \lceil n \log \rho_{max} \rceil$;   ▷ sequence range
 6: generate a random point $p$ in $K_\alpha = R_f \cap B(c, \rho_{min})$;
 7: $\mu'(R_f) \leftarrow vol(K_\alpha) = 4\pi\rho_{min}^3/3$;   ▷ initialization
 8: $w \leftarrow \lfloor 10 + n/10 \rfloor$;               ▷ walk length
 9: for $i = \alpha + 1$ to $\beta$ do
10:     $K_i \leftarrow R_f \cap B(c, 2^{i/n})$;
11:     $count \leftarrow 0$;
12:     for $j = 1$ to $N$ do
13:         $p \leftarrow$ RandomWalk$(p, K_i, w)$;         ▷ Algorithm 2
14:         if $p \in B(c, 2^{(i-1)/n})$ then
15:             $count \leftarrow count + 1$;
16:         end if
17:     end for
18:     $\mu'(R_f) \leftarrow \mu'(R_f) \cdot (N/count)$;
19: end for
20: return $\mu'(R_f) / \prod_d (U_d - L_d)$;
```

References

[1] "Topology zoo." http://www.topology-zoo.org/.

[2] H. C. Andersen and P. Diaconis, "Hit and Run as a Unifying Device," *Journal de la société française de statistique*, vol. 148, no. 4, pp. 5–28, January 2007.

[3] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[4] R. Beckett, A. Gupta, R. Mahajan, and D. Walker, "A General Approach to Network Configuration Verification," in *Proc. ACM SIGCOMM*, 2017.

[5] H. Berbee, C. Boender, A. Rinnooy Ran, C. Scheffer, R. L. Smith, and J. Telgen, "Hit-And-Run Algorithms for the Identification of Nonredundant Linear Inequalities," *Mathematical Programming*, vol. 37, no. 2, pp. 184–207, June 1987.

[6] B. Büeler, A. Enge, and K. Fukuda, "Exact Volume Computation for Polytopes: A Practical Study," in *Polytopes — combinatorics and computation*, 2000, pp. 131–154.

[7] Y. Chang, S. Rao, and M. Tawarmalani, "Robust Validation of Network Designs under Uncertain Demands and Failures," in *Proc. USENIX NSDI*, 2017.

[8] R. L. Cruz, "A Calculus for Network Delay," *IEEE Trans. Information Theory*, vol. 37, no. 1, pp. 114–131, 1991.

[9] L. De Moura and N. Bjørner, "Satisfiability Modulo Theories: Introduction and Applications," *Communications of the ACM*, vol. 54, no. 9, pp. 69–77, September 2011.

[10] M. Dyer, A. Frieze, and R. Kannan, "A Random Polynomial-Time Algorithm for Approximating the Volume of Convex Bodies," *Journal of the ACM*, vol. 38, no. 1, pp. 1–17, January 1991.

[11] M. E. Dyer and A. M. Frieze, "On the Complexity of Computing the Volume of a Polyhedron," *SIAM Journal on Computing*, vol. 17, no. 5, pp. 967–974, October 1988.

[12] I. Z. Emiris and V. Fisikopoulos, "Efficient Random-Walk Methods for Approximating Polytope Volume," in *Proc. ACM symposium on Computational geometry*, 2014.

[13] ——, "Practical Polytope volume Approximation," *ACM Trans. on Mathematical Software*, vol. 44, no. 4, pp. 1–21, December 2018.

[14] S. K. Fayaz, T. Sharma, A. Fogel, R. Mahajan, T. Millstein, V. Sekar, and G. Varghese, "Efficient Network Reachability Analysis Using a Succinct Control Plane Representation," in *Proc. USENIX OSDI*, 2016.

[15] A. Fogel, S. Fung, L. Pedrosa, M. Walraed-Sullivan, R. Govindan, R. Mahajan, and T. Millstein, "A General Approach to Network Configuration Analysis," in *Proc. USENIX NSDI*, 2015.

[16] A. Gember-Jacobson, R. Viswanathan, A. Akella, and R. Mahajan, "Fast Control Plane Analysis Using an Abstract Representation," in *Proc. ACM SIGCOMM*, 2016.

[17] P. Gill, N. Jain, and N. Nagappan, "Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications," in *Proc. ACM SIGCOMM*, 2011.

[18] R. Govindan, I. Minei, M. Kallahalla, B. Koley, and A. Vahdat, "Evolve or Die: High-Availability Design Principles Drawn from Googles Network Infrastructure," in *Proc. ACM SIGCOMM*, 2016.

[19] T. Hauer, P. Hoffmann, J. Lunney, D. Ardelean, and A. Diwan, "Meaningful Availability," in *Proc. USENIX NSDI*, 2020.

[20] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *Proc. ACM SIGCOMM*, 2013.

[21] C.-Y. Hong, S. Mandal, M. Al-Fares, M. Zhu, R. Alimi, C. Bhagat, S. Jain, J. Kaimal, S. Liang, K. Mendelev *et al.*, "B4 and After: Managing Hierarchy, Partitioning, and Asymmetry for Availability and Scale in Google's Software-Defined WAN," in *Proc. ACM SIGCOMM*, 2018.

[22] C. Hopps *et al.*, "Analysis of an equal-cost multi-path algorithm," RFC 2992, Internet Engineering Task Force, Tech. Rep., 2000.

[23] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a Globally-Deployed Software Defined WAN," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, October 2013.

[24] R. Kannan, L. Lovász, and M. Simonovits, "Random Walks and an $O^*(n^5)$ Volume Algorithm for Convex Bodies," *Random Structures & Algorithms*, vol. 11, no. 1, pp. 1–50, December 1997.

[25] P. Kazemian, G. Varghese, and N. McKeown, "Header Space Analysis: Static Checking for Networks," in *Proc. USENIX NSDI*, 2012.

[26] P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, P. Lapukhov, C. L. Lim, and R. Soulé, "Semi-Oblivious Traffic Engineering: The Road Not Taken," in *Proc. USENIX NSDI*, 2018.

[27] M. Kwiatkowska, G. Norman, and D. Parker, "Prism 4.0: Verification of Probabilistic Real-Time Systems," in *International Conference on computer Aided Verification*, 2011.

[28] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter, "Traffic Engineering with Forward Fault Correction," in *Proc. ACM SIGCOMM*, 2014.

[29] L. Lovász, "Hit-And-Run Mixes Fast," *Mathematical programming*, vol. 86, no. 3, pp. 443–461, December 1999.

[30] L. Lovász and I. Deák, "Computational Results of an $O(n^4)$ Volume Algorithm," *European journal of operational research*, vol. 216, no. 1, pp. 152–161, January 2012.

[31] L. Lovász and S. Vempala, "Simulated Annealing in Convex Bodies and an $O(n^4)$ Volume Algorithm," *Journal of Computer and System Sciences*, vol. 72, no. 2, pp. 392–417, March 2006.

[32] H. Mai, A. Khurshid, R. Agarwal, M. Caesar, P. B. Godfrey, and S. T. King, "Debugging the Data Plane with Anteater," in *Proc. ACM SIGCOMM*, 2011.

[33] L. Mohacsi and I. Deák, "A Parallel Implementation of an $O^*(n^4)$ Volume Algorithm," *Central European Journal of Operations Research*, vol. 23, pp. 925–952, December 2015.

[34] L. d. Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2008.

[35] R. Potharaju and N. Jain, "When the Network Crumbles: An Empirical Study of Cloud Network Failures and Their Impact on Services," in *Proc. ACM SOCC*, 2013.

[36] M. Roughan, "Simplifying the Synthesis of Internet Traffic Matrices," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 93–96, October 2005.

[37] M. Simonovits, "How to compute the volume in high dimension?" *Mathematical programming*, vol. 97, no. 1, pp. 337–374, July 2003.

[38] R. Singh, S. Agarwal, M. Calder, and P. Bahl, "Cost-Effective Cloud Edge Traffic Engineering with Cascara," in *Proc. USENIX NSDI*, 2021.

[39] R. L. Smith, "Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions," *Operations Research*, vol. 32, no. 6, pp. 1296–1308, 1984.

[40] S. Smolka, P. Kumar, N. Foster, D. Kozen, and A. Silva, "Cantor Meets Scott: Semantic Foundations for Probabilistic Networks," in *Proc. ACM POPL*, 2017.

[41] S. Steffen, T. Gehr, P. Tsankov, L. Vanbever, and M. Vechev, "Probabilistic Verification of Network Configurations," in *Proc. ACM SIGCOMM*, 2020.

[42] K. Subramanian, A. Abhashkumar, L. D'Antoni, and A. Akella, "Detecting Network Load Violations for Distributed Control Planes," in *Proc. ACM PLDI*, 2020.

[43] A. Viswanathan, E. C. Rosen, and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, Jan. 2001. [Online]. Available: https://www.rfc-editor.org/info/rfc3031

[44] J. Zheng, H. Xu, X. Zhu, G. Chen, and Y. Geng, "We've Got You Covered: Failure Recovery with Backup Tunnels in Traffic Engineering," in *Proc. IEEE ICNP*, 2016.