Adversarial Network Algorithm Benchmarking

Sebastian Lettner
TU München
sebastian.lettner@tum.de

Andreas Blenk
TU München
andreas.blenk@tum.de

ABSTRACT

Most research papers should have one thing in common: a clear and expressive evaluation of proposed solutions to problems. However, evaluating solutions is interestingly a challenging task: when using human-constructed examples or real-world data, it is difficult to assess to which degree the data represents the input spectrum also of future demands. Moreover, evaluations which fail to show generalization might hide algorithm weak-spots, which could eventually lead to reliability and security issues later on. To solve this problem we propose Toxin, a framework for automated, data-driven benchmarking of, e.g., network algorithms. In a first proof-of-concept implementation, we use Toxin to generate challenging traffic datasets for a data center networking use case.

CCS CONCEPTS

• **Networks** → *Traffic engineering algorithms*; *Network simulations*; *Network performance analysis*; • **Computing methodologies** → *Machine learning*; Artificial intelligence.

KEYWORDS

adversarial traffic generation, artificial intelligence, data center

ACM Reference Format:
Sebastian Lettner and Andreas Blenk. 2019. Adversarial Network Algorithm Benchmarking. In *The 15th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '19 Companion), December 9–12, 2019, Orlando, FL, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3360468.3366779

1 INTRODUCTION

Traditional benchmarks of network algorithms are either performed on randomly created traffic data [3, 15, 16] or based on real traffics traces [6, 11]. But either way of data acquisition implies several problems. Traffic traces are not always publicly available due to privacy or security concerns and randomly generated traffic could over-simplify reality. This might not only leave presented results questionable, it could even hinder the reproducibility of published results, which are important for future improvements and comparisons of new solutions. Yet another major problem is that algorithms might be tweaked towards the evaluation data. The test set might fail to expose performance issues which only occur for samples not included in the evaluation data.

CoNEXT '19 Companion, December 9–12, 2019, Orlando, FL, USA

ACM ISBN 978-1-4503-7006-6/19/12.
https://doi.org/10.1145/3360468.3366779

Unfortunately, obtaining challenging input data is a problem of its own. Even human experts are often not able to construct inputs exposing these weaknesses [5, 7, 12], not at least because of the high effort it takes. As a consequence, evaluations might sometimes be biased and actually fail to show generalization. This is, however, problematic since overlooked performance issues can have negative implications not only on the reliability but also on the security of the system [9, 14] because it could open the door for exploitation.

To address this problem we propose Toxin, an automated, data-driven benchmarking framework for data center network algorithms. We demonstrate that creating challenging evaluation data sets is a suitable task for machine learning and artificial intelligence. Those machine generated data-sets consist, e.g., of traffic matrices (demands), which are trained to maximize certain network metrics, e.g. the Flow Completion Time (FCT) in data centers. Using an automated, data-aware and unified way of benchmarking (i.e., attacking) algorithms, evaluation becomes more representative and even reproducibility might be simplified.

Previous work on algorithm complexity attacks has already shown methods for generating challenging, often called adversary, algorithms inputs [8, 10, 13, 17, 18]. With the help of these inputs the authors were able to improve algorithm performance and close security holes. As we see, while the idea is not new in general, to the best of our knowledge it has not yet been applied to networking use cases such as data center traffic scheduling.

In a first proof-of-concept, Toxin uses a *Genetic Algorithm* to schedule flows in a way that maximizes FCT in a data center environment. First simulation results show that Toxin can generate traffic with an increase of 40 % in FCT compared to a random search.

2 TOXIN: ADVERSARIAL EVALUATION

Data Center Traffic Scheduling. A fat-tree serves as the network topology [2]. Traffic consists of multiple flows which arrive over time. Each flow has a defined source and destination host, as well as a volume and an arrival time. For scheduling flows, we use a simple approach first: If multiple flows request the same resource, we schedule them based on their volume (large flows are preferred) and once the flow is in the network we route it from host to host based on shortest paths.

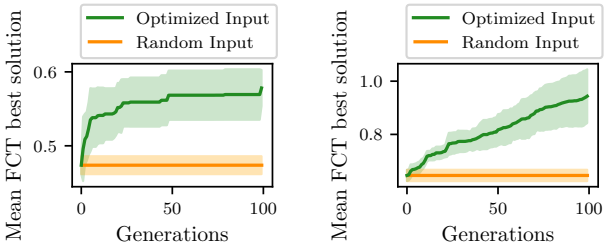
Problem Formulation. More formally, current traffic is defined by a list of N flows $F_N := \{f_0, f_1, ..., f_N\}$ ordered by their arrival time. Each flow f_i has four attributes: source, destination, volume and arrival time. The goal is to generate a set of flows that maximizes a measure of how harmful the set is to the network performance (called $Q(x)$), in our case mean FCT over all flows. The FCT of a flow f_i is defined by the duration between the time when the flow is requested and when it is completely transmitted, consequently

$$\underset{F_N}{\operatorname{argmax}}\, Q(x) \equiv \underset{F_N}{\operatorname{argmax}}\, \frac{1}{N+1} \sum_{i=0}^{N} FCT(f_i) \tag{1}$$

The generated flow sequence needs to fulfill certain constraints in order to be a meaningful example of network traffic. First, source and destination of each flow should be valid identifiers within the network. Second, the arrival time and the flow volume should be bounded such that it is not allowed to generate unusual high volumes or schedule all flows at the exact same time or path. To incorporate these constraints, we narrow the problem down to an ordering problem. The goal is to schedule the volumes $V_N = \{v_0, v_1, ..., v_N\}$ of a flow sequence F_N in a way that maximizes $Q(x)$, while keeping the other attributes fixed. This formulation has two advantages. First, source, destination and arrival time only need proper initial values. Second, because only the order of volumes changes instead of the volumes themselves, the total amount of traffic in a sequence will remain the same during optimization.

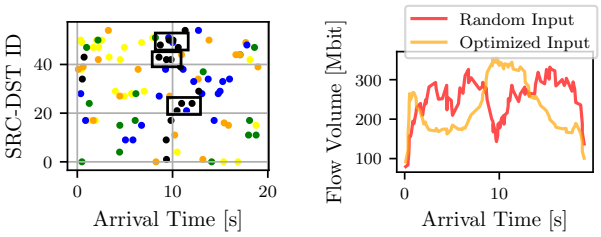
Genetic Algorithm. At the heart of `Toxin` lies artificial intelligence and machine learning. It is the task of an appropriate algorithm to create a challenging input for evaluating algorithms. In a first proof-of-concept, we deploy a Genetic Algorithm (GA). GAs belong to the family of evolutionary algorithms [1, 4]. During operation, they maintain a pool of sample solutions, called *population*, which is iteratively improved by applying a set of operators to select and combine population members with high fitness values. `Toxin` uses the mean FCT as the fitness indicator. The GA only requires to evaluate the fitness function in order to maximize it.

First, we sample N initial flows using appropriate distributions. Let V_N be the list of volumes from the initial sequence of flows. To form an initial population, we generate sample solutions by randomly permuting the order of V_N. During optimization, the GA is going to modify the ordering of volumes by applying *partially-mapped-crossover*[1]. To evaluate the fitness of a newly generated population member we replace the volumes in the initial flow sequence F_N with the modified volume list from the new population member and derive its mean FCT by simulation.



(a) N=10. **(b) N=100.**

Figure 1: Development of the mean FCT over the generations. Confidence is calculated over 16 seeds with $\alpha = 0.99\%$.



(a) Spatial Behavior. **(b) Temporal Behavior.**

Figure 2: Comparison between spatial and temporal behavior. Fig. 2a shows the spatial distribution of flow volumes over time. The y-axis gives the source to destination id. The connections are structured such that similar connections, e.g., one end-point is shared, are closer to each other. Colors mark the volume of flows (ascending: yellow, green, orange, blue, black). Fig. 2b visualizes the amount of volume over the simulation time, averaged over 8 seeds.

3 MAXIMIZING FLOW-COMPLETION TIME

Experiment Description. We use an event-based, flow-level network simulator to evaluate the FCT of inputs F_N. All experiments use a $k = 4$ fat-tree with 500 *Mbps* links. The initial population member (i.e., flows) of the GA use a uniform distribution to sample the source and destination from the hosts. The arrival times are drawn from a Poisson process with a mean of 0.7 seconds. The volumes are sampled uniformly between 1 and 500 *Mbps*.

Observations. `Toxin` is evaluated for $N = 10$, as an example of very few flows, and $N = 100$ as a more complex ordering task. Fig. 1 shows the development of the best fitness values of traffic samples (FCT) over the course of generations. It shows that the GA is able to increase the mean FCT by 23 % for $N = 10$ and 38 % for $N = 100$. `Toxin` cannot only create more challenging input sets for a small number of flows, but also for a larger one efficiently.

Fig. 2a visualizes the spatial distribution of flow volumes for $N = 100$. The colored dots represent flows arriving over time. The y-axis shows the host-to-host connection the flows were assigned to. The black rectangles emphasize a pattern, where large flows with small inter-arrival times are assigned to the same or close connections. This leads to a high probability of large flows sharing a link, thereby exhausting the links capacity. `Toxin` finds such challenging patterns in an automated manner. Fig. 2b visualizes the total volume of all flows over time. In contrast to the random input the adversarial input shows a clear pattern: `Toxin` schedules large flows together, illustrated by the peak at 10 seconds. Similar, the smaller flows are present in two groups, one in front of the peak and one behind. This behavior is indeed reasonable: concentrating large flows increases the probability that a link already has a high utilization when other flows arrive, consequently increasing FCT.

4 CONCLUSION AND FUTURE WORK

`Toxin` has shown its capability to generate challenging traffic loads for data center networks. Currently, `Toxin` only controls the order of volumes in a flow sequence. Giving it control over more parameters might lead to even more challenging traffic patterns.

ACKNOWLEDGMENT

This work is part of a project that has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation program (grant agreement No 647158 - FlexNets).

REFERENCES

[1] Zakir H Ahmed. 2010. Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator. *International Journal of Biometrics & Bioinformatics (IJBB)* 3, 6 (2010), 96.

[2] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review*, Vol. 38. ACM, 63–74.

[3] Li Chen, Justinas Lingys, Kai Chen, and Feng Liu. 2018. Auto: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, 191–205.

[4] Lawrence Davis. 1991. Handbook of genetic algorithms. (1991).

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[6] Brandon Heller, Srinivasan Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown. 2010. Elastictree: Saving energy in data center networks.. In *Nsdi*, Vol. 10. 249–264.

[7] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284* (2017).

[8] Caroline Lemieux, Rohan Padhye, Koushik Sen, and Dawn Song. 2018. Perffuzz: Automatically generating pathological inputs. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*. ACM, 254–265.

[9] M. Douglas McIlroy. 1999. A killer adversary for quicksort. *Software: Practice and Experience* 29, 4 (1999), 341–344.

[10] Wei Meng, Chenxiong Qian, Shuang Hao, Kevin Borgolte, Giovanni Vigna, Christopher Kruegel, and Wenke Lee. 2018. Rampart: Protecting Web Applications from CPU-Exhaustion Denial-of-Service Attacks. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Baltimore, MD, 393–410. https://www.usenix.org/conference/usenixsecurity18/presentation/meng

[11] Xiaoqiao Meng, Vasileios Pappas, and Li Zhang. 2010. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *2010 Proceedings IEEE INFOCOM*. IEEE, 1–9.

[12] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. 2017. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 27–38.

[13] Theofilos Petsios, Jason Zhao, Angelos D Keromytis, and Suman Jana. 2017. Slowfuzz: Automated domain-independent detection of algorithmic complexity vulnerabilities. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2155–2168.

[14] Emil Sit and Robert Morris. 2002. Security considerations for peer-to-peer distributed hash tables. In *International Workshop on Peer-to-Peer Systems*. Springer, 261–269.

[15] Asaf Valadarsky, Michael Schapira, Dafna Shahaf, and Aviv Tamar. 2017. Learning to route. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*. ACM, 185–191.

[16] Mowei Wang, Yong Cui, Shihan Xiao, Xin Wang, Dan Yang, Kai Chen, and Jun Zhu. 2018. Neural network meets DCN: Traffic-driven topology adaptation with deep learning. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 2, 2 (2018), 26.

[17] Jiayi Wei, Jia Chen, Yu Feng, Kostas Ferles, and Isil Dillig. 2018. Singularity: Pattern fuzzing for worst case complexity. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 213–223.

[18] Johannes Zerwas, Patrick Kalmbach, Laurenz Henkel, Gábor Rétvári, Wolfgang Kellerer, Andreas Blenk, and Stefan Schmid. 2019. NetBOA: Self-Driving Network Benchmarking. In *Proceedings of the 2019 Workshop on Network Meets AI & ML (NetAI'19)*. ACM, New York, NY, USA, 8–14.