Netgen: A Fast and Scalable Tool for the Generation and Labeling of Networking Datasets

Ignacio Martín, José Alberto Hernández, and Óscar González de Dios*
Telematic Engineering Dept. Universidad Carlos III de Madrid, Madrid, Spain
** Telefónica Global CTO, Madrid, Spain*
e-mail: ignmarti@it.uc3m.es

ABSTRACT
This article presents Netgen, an open-source tool for the generation of large networking datasets through the open-source tool Net2Plan. We show Netgen's features and applicability by generating six large labeled datasets for data analysis and Machine Learning. The datasets comprise 100,000 different data points for different traffic matrices and topologies, labeled after solving IP routing, IP routing with link disjoint 1+1 protection and RWA for IP over WDM networks. Both datasets and Netgen's code are publicly available.
Keywords: ML-assisted networking, dataset generation, RWA, path formulations, path protection routing, network planning and optimization.

1. INTRODUCTION

The application of Machine Learning (ML) techniques in the networking field are gaining great relevance lately. Researchers are analyzing the benefits of introducing modern ML techniques to enhance certain network functions, pointing towards a new paradigm, often referred to as Cognitive Network Management [1]. As a result, there is an increasing interest in evaluating the opportunities of ML-powered network applications [2]: Indeed, starting from the traditional Traffic Classification [3], [4] to the more recent proposals for protocol enhancement [5] and paradigm shifts [6] fueled by emerging technologies like Deep Learning. A nice survey of Machine Learning, Artificial Intelligence and their applications to networking problems, showing applications, frameworks, tools and open challenges, can be found on [7].

ML algorithms require large datasets to produce optimal results and avoid overfitting, which appears when there are not enough samples to build accurate models. In general, the applications of ML in the networking field have traditionally suffered from data scarcity [2], since quality datasets are hard to obtain and can be collected some few stakeholders like telcos or vendors. Even though data collection is raising new challenges [8], there is a real need for clean and labelled datasets [9]. Thus, having large realistic datasets in the networking context has become key for the successful implementation of ML-assisted network protocols.

As a result, in this paper, we introduce Netgen, a python-based tool to generate large labeled datasets of networking data. Netgen interfaces the well-known Net2Plan [10], an open-source network planning tool based on Java to design, simulate and solve network optimization problems (routing and wavelength allocation, traffic engineering, etc.). Netgen builds on top of Net2Plan to provide the latter with scalable and efficient ways for large dataset generation. The rest of this work is structured as follows: Section 2 describes Netgen workflow, components and architecture in detail; Section 3 demonstrates the Netgen system capabilities through the generation of various large datasets. Finally, Section 4 concludes and wraps the work.

2. ARCHITECTURE

Net2Plan[1] is a Java-based open-source tool designed for planning, optimization and performance evaluation of communication networks in general, with many popular algorithms already implemented for optical WDM networks. For that, Net2Plan provides a Graphical User Interface (GUI) version that allows the design of abstract representations of networks, including network topologies, traffic matrices, physical layer characteristics, etc. In particular, for a given a topology and traffic matrix, Net2Plan can solve a number of classical networking problems, such as classical IP routing with or without path protection, Routing and Wavelength Allocation (RWA) in IP over optical WDM networks or optimal configuration of physical-layer parameters in wireless networks, either using heuristics or Mixed Integer Linear Programming formulations via optimization toolboxes like CPLEX. In addition to the GUI, Net2plan also provides a Command Line Interface (CLI) to interface its features in a machine-friendly format.

[1] See the Net2Plan website: http://www.net2plan.com, last access Feb 2019

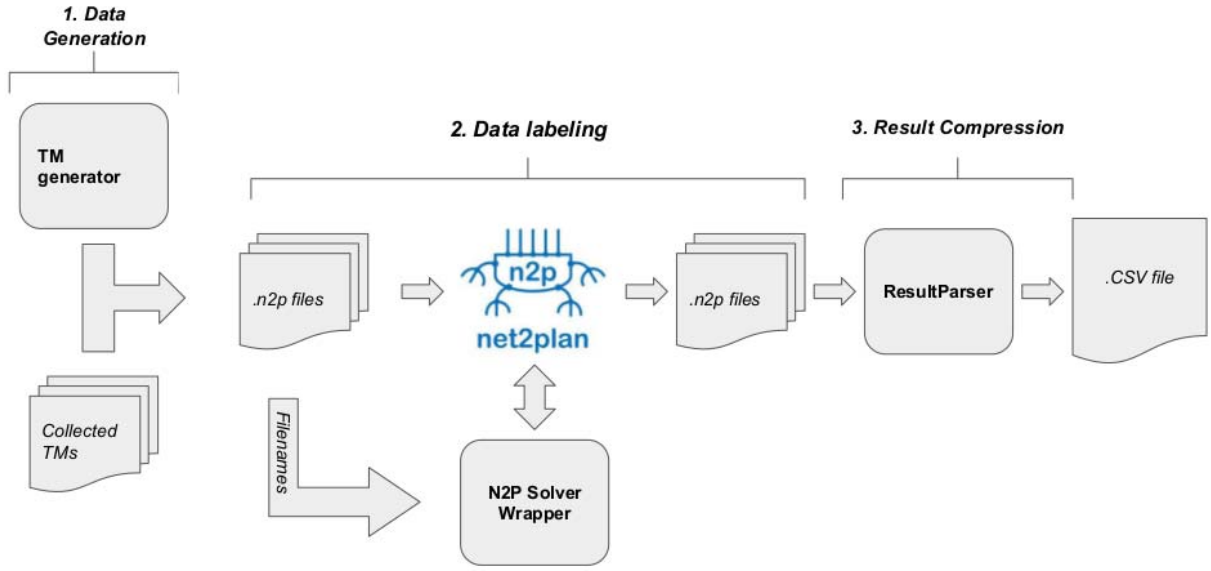

Figure 1. Netgen architecture overview.

Net2Plan can be fed with a network topology and traffic matrix using predefined .n2p files which following conventional XML syntax, executed to solve some algorithm and output its results again in a .n2p file or some other format for post-processing.

In this light, Netgen is conceived as a Net2Plan wrapper system that executes multiple instances of Net2Plan in parallel and collects its output results faster. Figure 1 depicts the Netgen workflow and its main components in a comprehensive way. The source-code of Netgen is publicly available at Github[2].

2.1 Traffic Matrix Generation

The first module in Netgen is the Traffic Matrix generator, which is coded in the python script TMGenerator.py. This script takes as input a ”canonical” net2plan file, containing the network topology and a base traffic matrix (i.e. a population-distance model or a real-network observation) and generates random traffic matrices by introducing noise to the canonical matrix. The module corresponds to phase 1 (Data generation) in the workflow diagram in Fig. 1.

When invoked, the program reads the topology and traffic matrix from the input file, modifies the canonical matrix through introducing random noise as many times as required and stores the resulting matrices along with the topology in a new n2p file that can be interpreted and solved by Net2Plan. Random noise is introduced through statistical distributions centered in the canonical value of the matrix and of configurable size and variability. At present Gaussian and Uniform noise distributions are supported.

In addition, Netgen provides a traffic matrix loading module that takes as input CSV-encoded traffic matrices and the canonical matrix containing the expected network topology (including expected active demands) and returns the n2p files containing each of the former traffic matrices and the later topology. This component is encoded in the TMLoader.py script file.

2.2 Net2Plan Solver Wrapper

The Net2plan Wrapper is a python script that manages the invocation of Net2Plan to solve the desired algorithm for the input traffic matrices in parallel. Several instances of Net2plan are called to speed the resolution of a given network problem (of those supported by Net2Plan), either by means of ILP or heuristic approaches. The python file name is Net2planInvoker.py. In essence, this component is a wrapper that identifies alln2p file locations and invoke different Net2plan instances to solve them, attempting parallel processing whenever possible. The resulting n2p files are stored in another given location. This component is algorithm-independent and thus supports most of the algorithms Net2Plan solves. Algorithms and their location files along with each algorithm specific parameters are introduced via configuration files.

This module is the largest one and corresponds to phase 2 (Data labelling) of the workflow in Fig. 1. Indeed, this module is the core of Netgen, as it contains its most innovative element: The ability to label Traffic matrices with different network algorithm solutions. The reader should note that Netgen acts as scheduler and manager of a pool of Net2Plan instances aiming at better execution performance in the data generation field.

Moreover, this is the key component to obtain the optimal configurations (labels) of real-world data coming from deployed networks. Netgen also supports this process through its TMLoader.py script, that manages the

[2] See Netgen source code at Github: https://github.com/ignmarti/Netgen, last access Feb 2019

mapping of traffic matrices in CSV format into n2p files, that contains network topology information and is the required input format for Net2Plan.

2.3 Net2Plan Results Parser

Previous components are able to randomly generate synthetic traffic matrices and manage Net2Plan execution to compute their optimal solution according to some planning or routing algorithm. In spite of it, Net2Plan results are still in n2p file format, which is not a very useful format for data analysis. Thus, the final component of Netgen is the Result Parser, which reads all n2p files in a folder and compress them into a csv file, where each network setting is a single entry. This component is conceived as a large scale Net2Plan file parser, so any collection of files can be analyzed, as long as they share topology and algorithm resolution.

When given a file location (folder), this program reads all files in such location and parses the relevant information of each supported algorithm into a structured Comma-separated values (CSV) format. This way, any collection of dense XML-based n2p files can be turned into analysis-friendly tabular format. The reader must note that this component is highly impacted by the algorithm to be parsed, as the output format should change depending on the expected configuration. Currently a few algorithms are supported for parsing: *Routing and Wavelength Allocation*, *XdeFormulations* and *1+1PathProtection*. The python file of the file parser is called *ResultParser.py* and is associated to phase 3 (Result compression) in Fig. 1.

3. NETGEN IN ACTION: DATASET GENERATION AND ML TEST

3.1 Dataset generation with Netgen and Net2Plan

We demonstrate Netgen by generating six 100,000-sample datasets after solving three classical networking problems, namely (1) conventional shortest path IP Routing, (2) shortest path IP routing with 1+1 link-disjoint protection, and (3) routing and wavelength allocation in IP over WDM networks algorithms, on two different network topologies: (1) a five-node simple topology and (2) the NSFnet. Table 1 shows the time taken by each component of Netgen in generating 100,000-point datasets for three different algorithms and two network topologies3. The experiments were conducted on an Intel Xeon E5-2630 server with 24 cores and 190 GB of RAM memory.

Table 1. Netgen in Action: completion time for the different tests performed.

Algorithm	IP Routing	1+1 Prot.	RWA	IP Routing	1+1 Prot.	RWA
Topology	NSFNet	NSFNet	NSFNet	5Spain	5Spain	5Spain
Data Generation	454 s	454 s	454 s	180 s	180 s	180 s
Data Labelling	~15 hours	~94 hours	~10 hours	~4 hours	~14 hours	~3 hours
Result Compression	123 s	250 s	178 s	19 s	22 s	39 s
Average Per Sample	0.54 s	3.38 s	0.35 s	0.12 s	0.91 s	0.109 s

These figures show that the Netgen approach is very fast, capable of generating large datasets within some hours in a standard server. The reader should note that the Data generation module is not always needed, as sample traffic matrices may be available, notwithstanding that when generating different datasets, random matrices can be reused for more than one algorithm.

Indeed, average per sample indicates that very small overhead per matrix is included to Net2Plan common processing, as the elapsed time associated to each matrix is very close to the time required by Net2Plan optimization. Actually, the time required by Net2Plan to compute a Formulations' solution in our server is approximately 0.9 s and 2 s in the case of backup computation for the NSFNet case. These times so close to Net2Plan times indicate that Netgen introduces a small overhead in Net2plan.

3.2 ML use case: Logistic regression with Lasso regularization

We tested the Netgen generated dataset (RWA the Spanish 5-node topology) on a simple supervised ML-based use case: the classification of optimal global RWA configuration of the whole network (see [11] for further details). Essentially, starting from 10,000 different traffic matrices and their optimal RWA configurations produced by the ILP solution, we reduced the number of RWA configurations to 69 since, these master RWA configurations are applicable to multiple traffic profiles with slightly added average link load and hop-count. Classification results demonstrated high classification precision and recall values, reaching nearly 85% configuration feasibility in the validation dataset.

4. CONCLUSIONS

In summary, this work has presented a novel approach to generate networking synthetic labeled data: Netgen. The Netgen system is a wrapper tool built onto Net2Plan that coordinates, manages and parallelizes Net2Plan

[3]The resulting datasets are publicly available at: https://osf.io/et2ga/

operations to enable the generation of large Networking datasets related to Net2Plan Network optimization problems. Through the paper, we have summarized the most relevant features of Netgen and displayed its time performance and capabilities. In addition to labeling data, Netgen procures random data generation and data introduction support, as well as result compression into CSV format. All these features make Netgen a very interesting tool for batch labeling and generation of labeled network data, that can be used to assist in the application of Machine Learning to networking problems.

Finally, we demonstrated Netgen workflow by solving three classical networking algorithms (IP Routing, IP Routing with 1+1 protection and Routing and Wavelength Allocation) on two network topologies (NSFNet and Spanish 5 node), observing datapoint generation times between 0.1 to 3.5 seconds per datapoint. Source code is available at Github and the 6 datasets are available at a public repository.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the support of the EU-funded Metrohaul project (grant no. 761727) and the Spanish project TEXEO (TEC2016-80339-R). Ignacio Martín would like to acknowledge the support of his FPU grant (FPU15/03518) that support his work in Universidad Carlos III de Madrid.

REFERENCES

[1] Ayoubi *et al*., "Machine learning for cognitive network management," *IEEE Comm. Mag*. 2018.
[2] Wang *et al*., "Machine learning for networking: Workflow, advances and opportunities," *IEEE Network*, 2018.
[3] P. Wang, S.-C. Lin, and M. Luo, "A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNS," in *Proc. IEEE Int. Conf. on Services Computing (SCC)*, 2016.
[4] Zhang *et al*., "Network traffic classification using correlation information," *IEEE Transactions on Parallel and Distributed Systems*.
[5] F. Tang *et al*., "On removing routing protocol from future wireless networks: A real-time deep learning approach for intelligent traffic control," *IEEE Wireless Communications*, 2018.
[6] Mao *et al*., "Routing or computing? the paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Transactions on Computers*, 2017.
[7] Fadlullah *et al*., "State-of-the-art deep learning: Evolving machine intelligence toward tomorrows intelligent network traffic control systems," *IEEE Communications Surveys and Tutorials*, 2017
[8] Zheng *et al*., "Big data-driven optimization for mobile networks toward 5g," *IEEE Network*, 2016.
[9] Mestres *et al*., "Knowledge-defined networking," in *Proc. ACM SIGCOMM CCR*, 2017.
[10] P. Pavon-Marino and J.-L. Izquierdo-Zaragoza, "Net2plan: An open source network planning tool for bridging the gap between academia and industry," *IEEE Network*, 2015.
[11] N. Martin *et al*., "Is machine learning suitable for solving RWA problems in optical networks?" in *Proc. Eur. Conf. Optical Comm. (ECOC)*, 2018.