CHALMERS
UNIVERSITY OF TECHNOLOGY

Trade-Offs in Implementing Unsupervised Anomaly Detection with TAPI-Based Streaming Telemetry

Downloaded from: https://research.chalmers.se, 2024-09-21 06:23 UTC

Citation for the original published paper (version of record):
Lechowicz, P., Natalino Da Silva, C., Karunakaran, V. et al (2024). Trade-Offs in Implementing Unsupervised Anomaly Detection with TAPI-Based Streaming Telemetry. IEEE International Conference on High Performance Switching and Routing, HPSR: 13-18.
http://dx.doi.org/10.1109/HPSR62440.2024.10635922

N.B. When citing this work, cite the original published paper.

© 2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, or reuse of any copyrighted component of this work in other works.

(article starts on next page)

2024 IEEE 25th International Conference on High Performance Switching and Routing (HPSR) | 979-8-3503-6385-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/HPSR62440.2024.10635922

Trade-Offs in Implementing Unsupervised Anomaly Detection with TAPI-Based Streaming Telemetry

Piotr Lechowicz*, Carlos Natalino*, Vignesh Karunakaran†‡,
Achim Autenrieth†, Thomas Bauschert‡, Paolo Monti*
* Dept. of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden
† Adtran Networks SE, Munich, Germany
‡ Chair of Communication Networks, TU Chemnitz, Chemnitz, Germany

E-mail: piotr.lechowicz@chalmers.se

***Abstract*—It is essential to be able to identify hidden anomalies in order to fully automate optical networks. This requires specific features from the application programming interfaces (APIs) used by the control plane and network monitoring solution. One of the solutions, Transport API (TAPI), utilizes advanced techniques in telemetry streaming. The update policy in TAPI enables key performance indicators (KPIs) to be transmitted only when changes are detected. In this paper, we explore how the update policy configuration of TAPI and the use of unsupervised learning (UL) interact in detecting previously unseen anomalies. Results reveal various trade-offs that network operators need to consider, including compute and time overhead, as well as the overall accuracy of UL.**

***Index Terms*—Machine Learning, Optical networks, Unsupervised learning.**

I. Introduction

Optical network automation aims to minimize (or ideally eliminate) human involvement in optical network operations. More recently, with the introduction of disaggregated optical networks, network automation became even more desirable due to the shift in the complexity of controlling and harmonizing an increasing number of devices. In general, network automation is implemented through a loop that periodically, or upon external events, collects the current network status, performs data processing, and takes decisions to be applied in the network.

Network automation requires two critical application programming interfaces (APIs) to interact with devices: *control* and *monitoring*. In terms of *control*, Transport API (TAPI) has been one of the standards to define messages to be exchanged between the network controller and optical devices in order to retrieve the current configuration, or push configuration changes. Several works in the literature already explored the possibilities and benefits of TAPI to control optical devices [1], [2]. However, only recently TAPI established interfaces to target *monitoring*. TAPI streaming telemetry builds on top of gRPC remote procedure call (gRPC) and gRPC network management interface (gNMI), enabling efficient and flexible telemetry configurations [3]. One key feature of TAPI streaming telemetry is the possibility to set the update policy, which once enabled, avoids the transmission of redundant monitoring values, transmitting data only when the monitored value changes. With streaming telemetry, the control plane can subscribe to updates from network devices in a flexible manner. In particular, protocols based on gRPC have shown low latency and low network overhead [4], [5].

Once the data is collected, it needs to be processed by the control plane. Machine learning (ML) is one of the most regarded techniques for the processing of telemetry data for network automation [6]. There are three techniques from ML that can be used for processing of telemetry data. supervised learning (SL) allows for precise classification or regression of values, but require a dataset for training that includes all expected conditions. The requirement of a dataset makes SL not suitable for anomaly detection where, ideally, the algorithm needs to detect never-seen-before anomalies. semi-supervised learning (SSL) is more suitable for anomaly detection, as it separates samples among normal and anomalous. However, it also requires a training dataset containing examples of normal operating conditions, requiring re-training if/when the normal operating conditions change. This is also not ideal for the continuous monitoring of lightpaths, since the introduction or removal of co-propagating lightpaths may drastically change the normal operating conditions of the monitored lightpath. Finally, unsupervised learning (UL) uses a moving window of samples collected from the network to characterize normal operating conditions, and differentiate from anomalies at their start, i.e., when anomalous samples are a minority.

Previous works already studied the use case of telemetry streaming for soft-failure detection [5], [7]. Literature shows that UL are the most suitable ones for anomaly detection in optical networks. However, the area still lacks a comprehensive study of the trade-offs presented by using UL to process streaming telemetry data.

In this work, we study how the specific characteristics of the data collected by TAPI streaming telemetry impact on the application of UL for anomaly detection on established lightpaths. We discuss the characteristics of the data collected,

This work has been funded by the German Federal Ministry of Education and Research and by Sweden's innovation agency VINNOVA within the EUREKA cluster CELTIC-NEXT project AI-NET-PROTECT (#16KIS1279K) and (2020-03506), respectively.

Authorized licensed use limited to: Chalmers University of Technology Sweden. Downloaded on September 13,2024 at 07:48:43 UTC from IEEE Xplore. Restrictions apply.

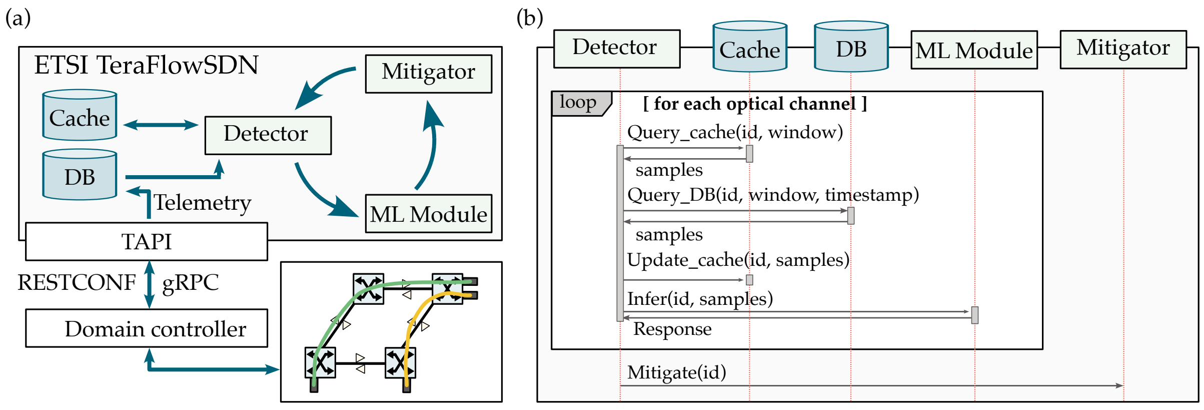

Fig. 1. System architecture; (a) overview; (b) anomaly detection automation loop.

in particular with respect to the update policy. Moreover, we assess how the hyperparameters of a well-known UL algorithm impact the performance of the entire process. The results reveal that the update policy impacts the processing overhead of the data. Moreover, there is a trade-off in the overall anomaly detection accuracy depending on the hyperparameters and how long the network operator is willing to wait to trigger mitigation actions.

II. Monitoring Loop based on TAPI Streaming Telemetry

The TAPI employs RESTCONF for communication with control plane components, utilizing the representational state transfer (REST) protocol with JavaScript object notation (JSON) payloads for various operations. While this is suitable for the exchange of configuration messages, it is not as appropriate for the continuous exchange of telemetry data [4]. TAPI v2.5 [3] addresses this limitation by outlining requirements for efficient telemetry data exchange, focusing on factors such as data volume, streaming techniques, data sources, and related entities. In particular, Section 6.1 of [3] further recommends a solution with the protocol, encoding format, and data structure necessary for effective streaming of performance data.

The recommendations given by TAPI streaming telemetry can be summarized into three main points: *(i)* gNMI/gRPC-based telemetry streaming with protobuf as the encoding format, *(ii)* TLS v1.3 based encryption for data exchange, and *(iii)* a unified data model for subscription requests and responses to ensure consistency across multiple domains. Within the standard, the subscription request includes: *(i)* device target, *(ii)* sample interval, *(iii)* xpath to denote the key performance indicator (KPI) to stream, and *(iv)* update policy (true/false). By enabling the update policy, redundant transmission of performance data is avoided unless there is an actual change, leading to a significant reduction in telemetry traffic. This is because numerous duplicate values are not reported. This is in contrast to REST, where periodic requests and responses are exchanged. Conversely, in gNMI/gRPC streaming, the subscription request is sent only once, while performance data is continuously streamed over the channel reducing the telemetry traffic.

Fig. 1 presents an overview of the system architecture considered in this paper. In Fig. 1(a), the centralized software-defined networking (SDN) controller exchanges information with optical line system (OLS) domain controller(s), using the RESTCONF-based TAPI interface for control messages, and the gRPC/gNMI-based TAPI streaming telemetry for obtaining telemetry data. The OLS domain controller controls and monitors the optical infrastructure, and stream telemetry information to SDN controller with TAPI-compliant gRPC messages. The controller stores received samples in a time-series database (DB). Periodically, the UL anomaly detection automation loop illustrated in Fig. 1(b) is triggered [8]. The detector queries the cache (if available) and the DB for recent samples, according to a predefined observation window. Queried samples are processed and forwarded for the UL anomaly inference in ML module. If an anomaly is detected, ML module raises an alert and notifies the mitigation module for further actions.

In this paper, we investigate how the update policy impacts the implementation of the data preprocessing performed by the detector in Fig. 1(b).In particular, we discuss the trade-offs that arise from this decision, i.e., how this impacts the retrieval of data from the time-series DB located in the data plane, and the processing cost of the loop.

III. Unsupervised Anomaly Detection in Optical Networks

The anomaly detection pipeline considered in this paper is part of the network automation procedures that run in the background for each lightpath. The procedure is presented in Algorithm 1 and takes 3 inputs: the lightpath r for which it should be run, the window w denoting how many historical samples are accounted for detecting anomalies, and list of measured KPIs K for the lightpath r. In general, the algorithm assumes the existence of a cache where recent samples are stored and a DB where all measurements are collected. Each sample in the cache corresponds to the set of measurements

Authorized licensed use limited to: Chalmers University of Technology Sweden. Downloaded on September 13,2024 at 07:48:43 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 Anomaly detection pipeline for single lightpath

```
Require: $r$ - lightpath, $w$ window size, $K$ - list of names of measured KPIs
1: procedure AnomalyDetection($r$, $w$, $K$)
2:     $cs \leftarrow$ queries cache for entries related to lightpath $r$
3:     if $cs$ is empty then
4:         $ts_w \leftarrow$ timestamp for measurements taken $w$-th iterations ago
5:         for $k \in K$ do $ts_k \leftarrow$ the latest timestamp for KPI $k$ stored in the DB such that it is before or equal to $ts_w$
6:         $ts_q \leftarrow$ select the earliest timestamp from $ts_k$ for $k \in K$
7:         $s \leftarrow$ query all measurements in the DB equal or at timestamp $ts_q$ related to lightpath $r$
8:         reshape $s$ to dense format and fill missing KPI values with values from preceding ones
9:         leave $w$ latest samples in $s$
10:        add $s$ to cache
11:    else
12:        $ts_q \leftarrow$ timestamp of the latest sample in the cache
13:        $ns \leftarrow$ query all measurements in the DB after the timestamp $ts_q$ related to lightpath $r$
14:        reshape $ns$ to dense format and fill missing KPI values with values from preceding ones (including values from
           last cached sample)
15:        add $ns$ to cache
16:        remove oldest samples from the cache such that there is at most $w$ samples in it
17:        $s \leftarrow$ latest $w$ samples of $cs \cup ns$
18:    normalize $s$
19:    $anom \leftarrow \mathrm{UL}(s)$
20:    if enough anomalies $anom$ detected, raise alert
```

related to r taken in a single timestamp, i.e., a dense format. Measurements to the DB are added only when a value monitored for r changes, i.e., a sparse format.

In line 2, the algorithm queries the cache for samples related to lightpath r. If cache is empty (lines 3–10), it queries the DB for measurements to fill up the cache. In line 4, timestamp ts_w is retrieved from the DB for measurements that were taken w iterations ago. However, as the measurements in the DB are stored in sparse format, values for some KPIs may be missing at timestamp ts_w. Therefore, in line 5, the algorithm checks for each KPI $k \in K$ what was the latest timestamp ts_k with measurement that was stored before or at the timestamp ts_w. Line 6 selects the earliest timestamp ts_q from all collected timestamps ts_k. Selecting such timestamp assures that it is possible to infer all KPI values at timestamp ts_w. In line 7, the measurements are queried from the DB since timestamp ts_q. Next, they are reshaped to dense format and missing KPI values for each timestamp are inferred from the values for the preceding timestamp (line 8). Next, we extract the latest w samples (line 9) and store them in the cache (line 10). The stored samples s are processed further in the algorithm to detect the anomaly (lines 18–20)

If cache contains samples related to lightpath r, it is possible to use those samples for anomaly detection and reduce the number of samples that need to be queried from the database (lines 11–17). In such a case, in line 12, the algorithm checks what is the latest timestamp ts_q stored in the cache. Line 13 queries the DB for the measurements since timestamp ts_q. In line 14, the retrieved measurement values are reshaped to the dense format, and missing values are restored from the last cached sample (if needed). Next, the algorithm adds the newly acquired samples to the cache (line 15), and the oldest ones are removed (line 16) such that the cache contains at most w samples related to lightpath r. In line 17, the latest w samples are selected from the union of the samples queried from the cache and the ones created from new measurements.

Values in the samples are normalized by, e.g., removing the mean and scaling to the unit variance according to the previously collected values (line 18). In line 19, the UL algorithm is run to assign samples to clusters. Samples that are not belonging to any of the clusters are detected as anomalies. In line 20, if enough number of anomalies has been detected in the window, the algorithm raises an alert.

Enabling the update policy in TAPI allows prevents consecutive duplicate values from being transmitted. Such a behavior allows for storing data telemetry data in the DB in a sparse data format where some values may be missing for some timestamps. However, standard anomaly detection methods assume that all features are present [9]. Therefore, the missing values resulting from sparse telemetry data need to be addressed before applying UL algorithms.

Missing values in ML can be handled by deletion or imputation. The deletion can be performed either list-wise or pair-wise. In list-wise deletion, cases with missing features are removed. In pair-wise deletion, subset of features are considered during analysis, and cases with missing variables in that subset are omitted. In imputation, missing values are replaced by predicted values (e.g., simple imputation with mean/median, regression imputation, hot-deck, expectation maximization, ML-based imputation) [10], [11]. In the considered architecture, restoring exact missing feature values based on the values from preceding timestamps is possible. Such operation is of

Authorized licensed use limited to: Chalmers University of Technology Sweden. Downloaded on September 13,2024 at 07:48:43 UTC from IEEE Xplore. Restrictions apply.

$O(fn)$ complexity, where f is the number of features and n is the number of samples (timestamps). Restoring original values in the given architecture can be motivated two-fold. Firstly, the accuracy of clustering with missing values drops as the number of missing values increases. Missing value mitigation techniques aim to maximize relative loss in performance when compared to a model with full data [9]. Secondly, applying any data imputation or deletion method is of at least $O(fn)$ order, which is also the complexity of restoring the correct values.

IV. Performance Assessment

In this section, we first describe the scenario under which our use case of UL anomaly detection is executed. Then, we present the results from the experiments.

A. Scenario

The data collection for normal operating conditions was performed over a testbed infrastructure, similar to the one reported in [12]. The infrastructure is controlled by a proprietary OLS domain controller, which implements TAPI control and streaming telemetry. The infrastructure is composed of three reconfigurable optical add-drop multiplexers (ROADMs). Each ROADM is connected to two optical terminals (OTs). The OLS domain controller is connected to a version of the ETSI TeraFlowSDN controller (Release 2) which has been enhanced with TAPI streaming telemetry capabilities. Upon receiving new samples, the controller stores them in a time-series database.

We load the system with two lightpaths. Then, we established a third lightpath which is our channel under test (CUT). The CUT is monitored through the subscription-based TAPI streaming telemetry. We monitor 5 signal parameters, namely, signal-to-noise ratio (SNR), chromatic dispersion compensation (CDC), carrier frequency offset, differential group delay (DGD), and Q-factor. We run two sets of experiments, with and without the transmission of repeated values. The anomaly considered is caused by the introduction of an attenuation imposed by a variable optical attenuator (VOA).

We run the UL-based anomaly detection pipeline synchronized with the data collection, i.e., we run the pipeline for every new telemetry collection period. During each execution, the pipeline needs to retrieve the respective data from the database, execute the UL algorithm, and report the result, as described in the previous section. The implementation is based on the density-based spatial clustering of applications with noise (DBSCAN) algorithm [13], which receives as input a set of latest samples for each lightpath, and identifies if an anomaly is present. DBSCAN works under the assumption that, at the beginning of an anomaly, anomalous samples will be present in lower quantity among the samples, and their anomalous properties will make it possible to distinguish normal from anomalous samples. In this work, we assume that 10 anomalous samples are present in the dataset, while the remainder of the samples are composed by samples under normal operating conditions. There are three hyperparameters in DBSCAN. The parameter ϵ defines the distance between two

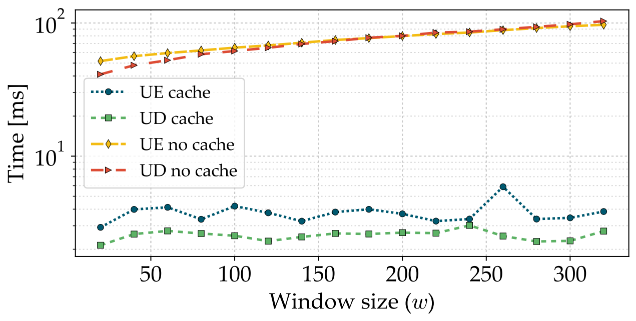

Fig. 2. Data retrieval time for different DBSCAN window sizes (w) with update policy enabled (UE) and disabled (UD).

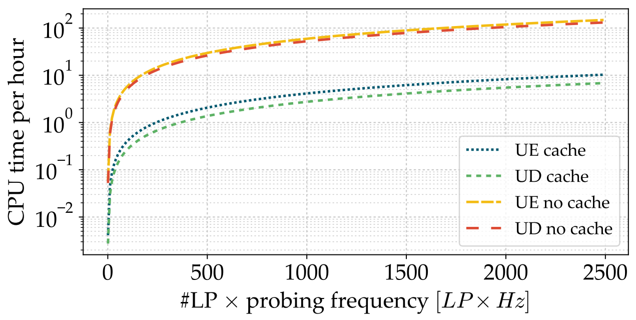

Fig. 3. CPU usage for data retrieval as a function of probing frequency per number of lightpaths (LP) with update policy enabled (UE) and disabled (UD).

samples for them to be considered neighbors. The parameter M defines the minimum number of neighbors that a sample needs to have to be considered a normal sample. Finally, w represents the window size, i.e., the number of samples to be analyzed by DBSCAN.

Two dimensions of the scenario were evaluated. The first dimension analyzes the impact of adopting a cache. In this case, we assess the performance with and without a cache that maintains a sample window. Intuitively, we expect the cache to substantially reduce the time taken to retrieve the samples. The second dimension is related to whether or not repeated values are saved to the database. In this case, the TAPI streaming telemetry agent in TeraFlowSDN may or may not save the sample value, depending on the update policy adopted in the TAPI streaming telemetry configuration.

We assess the performance of the pipeline, and its trade-offs, with the following KPIs:

1) *Data retrieval time* [ms]: the time taken to retrieve the information from the database.
2) *CPU time per hour*: the number of required CPUs per hour to retrieve the information from the database.
3) *True positive rate (TPR)*: the number of correctly classified anomalous samples divided by the total number of anomalous samples.
4) *True negative rate (TNR)*: the number of samples cor-

Authorized licensed use limited to: Chalmers University of Technology Sweden. Downloaded on September 13,2024 at 07:48:43 UTC from IEEE Xplore. Restrictions apply.

rectly flagged as normal divided by the total number of normal samples.

5) *f1-score*: the harmonic mean of the model response to false positives and false negatives.

B. Results

Fig. 2 shows the time taken by the controller to retrieve a given number of samples necessary to execute the UL algorithm. The average time taken to retrieve the data from the database is 75 ms. However, the difference in time from the cases with the update policy enabled and disabled are minor, in the range of 3%. More importantly, the time to retrieve the samples doubles when we move from 50 to 300 samples. Although the time increases relatively slow with respect to the number of samples, i.e., $2\times$ the time vs. $6\times$ the number of samples, this increase is significant once the number of lightpaths to be monitored increases, or when the frequency at which we analyze the lightpaths increases.

Meanwhile, the time taken to retrieve data from the cache is very low, i.e., below 3.7 ms on average. Interestingly, the time does not vary substantially with the number of samples. However, the differences between having update policy enabled and disable are quite relevant. When the update policy is disabled, i.e., it receives every update regardless of the value, the time to retrieve samples decreases by nearly 50% in comparison to the update policy enabled.

Fig. 3 shows CPU time per hour as a function of number of lightpaths and probing frequency. We consider up to 2,500 updates per second, which represents approximately the case where a 50-nodes optical topology has one lightpath per node-pair, receiving updates once per second. In particular, for more loaded system, the number of required CPUs increases from 5 to 10 when cache is considered and the update policy is enabled.

Fig. 4 shows the impact that the window size (w) has on the overall performance of the UL anomaly detection. Fig. 4(a) shows the f1 score, which represents a harmonic mean of the model response to false positives and false negatives. We can see that the higher the M, the higher the f1 score. Moreover, the f1 score saturates with a window size of 60. However, we need to further analyze the interplay between TPR and TNR to understand the impact of deciding M.

Fig. 4(b) shows the TPR, which represents the ability of the UL model to correctly detect positive samples (anomalies). The counterpart to true positives are the false negatives, which are detrimental to the operation of optical networks because they may leave issues unaddressed, potentially leading to the disruption of lightpaths. In this case, the higher the M in DBSCAN, the more neighbors a sample needs to be considered normal. This results in a lower number of false negatives. However, when the window size increases, the higher number of normal samples make it more likely to have anomalous samples categorized as normal, leading to a decrease in the overall metric. Therefore, if false negatives are the most detrimental type of error in the particular optical network setting, a high M combined with a low window size might yield the best trade-off.

Fig. 4(c) shows the TNR, which represents the ability

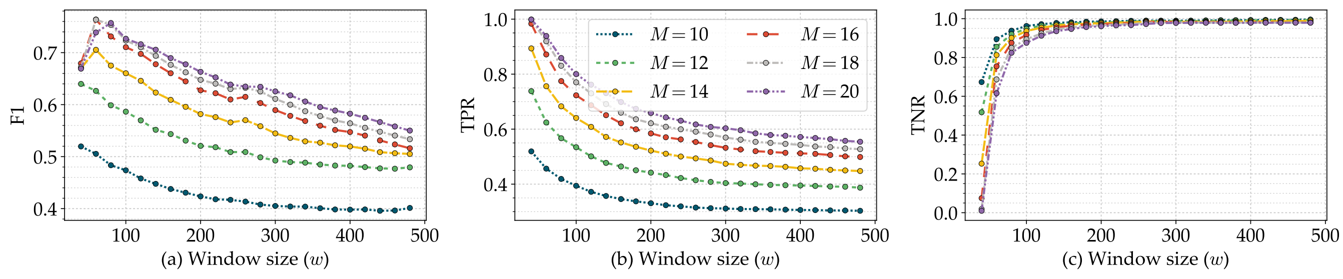

Fig. 4. Performance of DBSCAN with $\epsilon = 1.7$, number of anomalies=20, for different minimum number of neighbours (M) and various window sizes (w) in terms of (a) f1-score, (b) true positive rate (TPR), and (c) true negative rate (TNR).

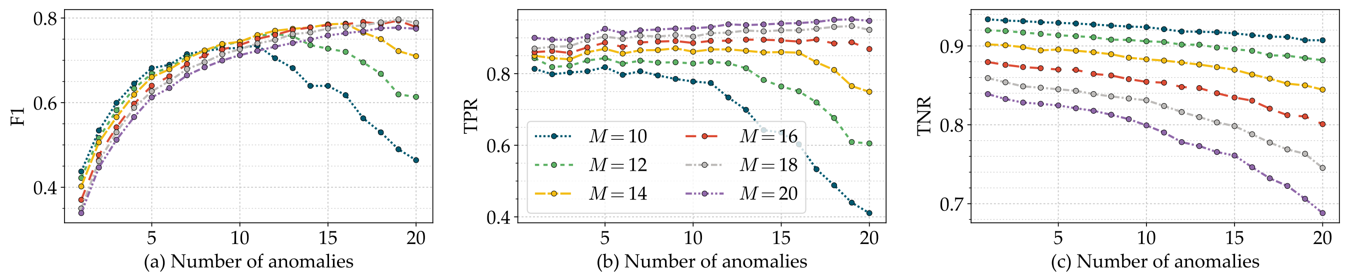

Fig. 5. Performance of DBSCAN with $\epsilon = 1.7$ and $w = 60$ for different minimum number of neighbours (M) as a function of number of anomalies in terms of (a) f1-score, (b) true positive rate (TPR), and (c) true negative rate (TNR).

Authorized licensed use limited to: Chalmers University of Technology Sweden. Downloaded on September 13,2024 at 07:48:43 UTC from IEEE Xplore. Restrictions apply.

of the UL model to correctly detect normal samples (non-anomalies). Their counterpart, false positives, are detrimental to the operation of optical networks because they may incur additional overhead in the form of unnecessary reconfigurations to address incorrectly detected anomalies. In our case, the lower the M, the less neighbors a sample needs to be considered normal. The results show that the best window size is between 100 and 200 samples. Therefore, if false positives are the most detrimental type of error in the optical network setting at hand, a window size in the range of 100-200 is more suitable, combined with a low value of M.

Another important aspect is the time taken to detect an anomaly, which directly impacts the network reaction time.The proposed algorithm alerts the mitigator module when a large enough number of anomalies is detected [14]. Fig. 5 presents the impact that the number of anomalies has on the overall performance of the UL anomaly detection. Fig. 5(a) depicts the f1 score for various settings of M. Intuitively, the performance increases with the number of anomalies, up to the point where it is higher than the value of M. In such a case, there are enough neighbour samples to consider anomalies as normal and categorize them as a normal cluster in UL. Fig. 5(b) shows the model response to detect anomalies, i.e., TPR. Similarly to f1, there is increasing trend up to the point where number of anomalies exceeds the required number of neighbours M to consider samples as normal ones. Fig. 5(c) shows the model response to detect normal samples, i.e., TNR. With the increase of anomalies the performance decreases, which can be justified that as increasing number of anomalies increase the chance that some of them are miss-classified. Therefore, if higher confidence for anomaly detection is required in current optical network settings, setting higher reacting threshold for number of detected anomalies increases f1 score. At the same time, this increases the time that the anomalies affects the system performance.

It is important to mention that the results shown in this paper are representative of the general trends when adopting UL for anomaly detection. However, the specific hyperparameter values and their respective performance will vary depending on the use case. Therefore, a hyperparameter tuning campaign will be needed for each specific network scenario.

V. Conclusions

This paper investigated the trade-offs that network operators need to consider when implementing UL-based anomaly detection over TAPI telemetry streaming. We detail the implementation of an algorithm for anomaly detection, considering the case where a cache exists to store samples of the monitored lightpaths. Intuitively, adopting a cache can substantially decrease the data retrieval time for each monitored lightpath. More importantly, the analysis reveals that there is a beneficial range in the number of samples to be analyzed by the UL. Presenting more samples may have a detrimental impact in terms of false positives and false negatives presented by the UL algorithm. Moreover, despite the trade off between TPR and TNR of UL models, it may be possible to increase overall f1 score and accuracy by using more advanced methods, such as, clustering ensemble.

References

[1] C. Manso, R. Munoz, N. Yoshikane, R. Casellas, R. Vilalta, R. Martinez, T. Tsuritani, and I. Morita, "TAPI-enabled SDN control for partially disaggregated multi-domain (OLS) and multi-layer (WDM over SDM) optical networks," *Journal of Optical Communications and Networking*, vol. 13, no. 1, pp. A21–A33, 2021.

[2] E. Etezadi, C. Natalino, V. Karunakaran, R. Diaz, A. Lindgren, S. Melin, A. Autenrieth, L. Wosinska, P. Monti, and M. Furdek, "Demonstration of DRL-based intelligent spectrum management over a T-API-enabled optical network digital twin," in *European Conference on Optical Communication (ECOC)*, 2023, p. D1.

[3] Open Network Foundation, "TAPI v2.5.0 Reference Implementation Agreement, TR-548, TAPI Streaming, Version 3.1," *Online*, 2024.

[4] F. Paolucci, A. Sgambelluri, F. Cugini, and P. Castoldi, "Network telemetry streaming services in SDN-based disaggregated optical networks," *Journal of Lightwave Technology*, vol. 36, no. 15, pp. 3142–3149, 2018.

[5] R. Vilalta, N. Yoshikane, R. Casellas, R. Martínez, S. Beppu, D. Soma, S. Sumita, T. Tsuritani, I. Morita, and R. Muñoz, "gRPC-based SDN control and telemetry for soft-failure detection of spectral/spacial superchannels," in *45th European Conference on Optical Communication (ECOC)*, 2019, pp. 1–4.

[6] D. Rafique and L. Velasco, "Machine learning for network automation: overview, architecture, and applications [invited tutorial]," *Journal of Optical Communications and Networking*, vol. 10, no. 10, pp. D126–D143, 2018.

[7] F. Paolucci, A. Sgambelluri, M. Dallaglio, F. Cugini, and P. Castoldi, "Demonstration of grpc telemetry for soft failure detection in elastic optical networks," in *European Conference on Optical Communication (ECOC)*, 2017, pp. 1–3.

[8] C. Natalino, L. Gifre, F.-J. Moreno-Muro, S. Gonzalez-Diaz, R. Vilalta, R. Munoz, P. Monti, and M. Furdek, "Flexible and scalable ML-based diagnosis module for optical networks: a security use case," *Journal of Optical Communications and Networking*, vol. 15, no. 8, pp. C155–C165, 2023.

[9] T. G. Dietterich and T. Zemicheal, "Anomaly Detection in the Presence of Missing Values," 2018. [Online]. Available: http://arxiv.org/abs/1809.01605

[10] T. Emmanuel, T. Maupong, D. Mpoeleng, T. Semong, B. Mphago, and O. Tabona, *A survey on missing data in machine learning*. Springer International Publishing, 2021, vol. 8, no. 1. [Online]. Available: https://doi.org/10.1186/s40537-021-00516-9

[11] L. Beretta and A. Santaniello, "Nearest neighbor imputation algorithms: A critical evaluation," *BMC Medical Informatics and Decision Making*, vol. 16, no. Suppl 3, 2016. [Online]. Available: http://dx.doi.org/10.1186/s12911-016-0318-z

[12] V. Karunakaran, C. Natalino, B. Shariati, P. Lechowicz, J. K. Fischer, A. Autenrieth, P. Monti, and T. Bauschert, "TAPI-based telemetry streaming in multi-domain optical transport network," in *Optical Fiber Communications Conference and Exhibition (OFC)*, 2024, p. M3Z.9.

[13] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, p. 226–231.

[14] M. Furdek, C. Natalino, F. Lipp, D. Hock, A. D. Giglio, and M. Schiano, "Machine learning for optical network security monitoring: A practical perspective," *Journal of Lightwave Technology*, vol. 38, no. 11, pp. 2860–2871, 2020.

Authorized licensed use limited to: Chalmers University of Technology Sweden. Downloaded on September 13,2024 at 07:48:43 UTC from IEEE Xplore. Restrictions apply.