Nadege: When Graph Kernels meet Network Anomaly Detection

Hicham Lesfari, Frédéric Giroire

▶ **To cite this version:**

Hicham Lesfari, Frédéric Giroire. Nadege: When Graph Kernels meet Network Anomaly Detection. IEEE International Conference on Computer Communications (INFOCOM), May 2022, London, United Kingdom. hal-03655867

HAL Id: hal-03655867
https://inria.hal.science/hal-03655867

Submitted on 30 Apr 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Nadege: When Graph Kernels meet Network Anomaly Detection

Hicham Lesfari, Frédéric Giroire
Université Côte d'Azur, CNRS, Inria, France

***Abstract*—With the continuous growing level of dynamicity, heterogeneity, and complexity of traffic data, anomaly detection remains one of the most critical tasks to ensure an efficient and flexible management of a network. Recently, driven by their empirical success in many domains, especially bioinformatics and computer vision, graph kernels have attracted increasing attention. Our work aims at investigating their discrimination power for detecting vulnerabilities and distilling traffic in the field of networking.**

In this paper, we propose *Nadege*, a new graph-based learning framework which aims at preventing anomalies from disrupting the network while providing assistance for traffic monitoring. Specifically, we design a graph kernel tailored for network profiling by leveraging propagation schemes which regularly adapt to contextual patterns. Moreover, we provide provably efficient algorithms and consider both offline and online detection policies. Finally, we demonstrate the potential of kernel-based models by conducting extensive experiments on a wide variety of network environments. Under different usage scenarios, *Nadege* significantly outperforms all baseline approaches.

I. Introduction

Learning on graph-structured data has gained a notable momentum with recent advances [49] in Artificial Intelligence (AI). While graphs provide an ubiquitous data structure to represent complex structures and interactions, readily blending them with standard machine learning algorithms brings several challenges [7]. Of particular relevance is the progress in designing flexible and expressive methods [15] for quantifying similarities between graphs.

A graph kernel is a kernel function [37] defined directly on pairs of graphs, and enables to compare between them. It corresponds to an inner product that is equivalent to an embedding of the set of graphs into a possibly high-dimensional Hilbert space [3]. An important benefit of such embedding is to allow efficient machine learning methods to be directly applied on graphs by extending the applicability of the whole arsenal of kernel methods [19] to graphs (e.g., for feature selection, classification, clustering, two-sample tests).

Furthermore, graph kernels offer provable theoretical guarantees and are easy to train thanks to the convexity of their optimization problem [9]. Thereby, they provide a rich connection between graph space and machine learning. However, despite their wide success in other domains, especially in bioinformatics, computer vision and, natural language processing (NLP) [23], they have received little attention in the networking field.

This work has been supported by the French government through the UCA JEDI (ANR-15-IDEX-01) and EUR DS4H (ANR-17-EURE-004) Investments in the Future projects and by Inria associated team EfdyNet.

Among the prominent networking problems, anomaly detection is a crucial task in network security. According to recent studies, cybercrime damage costs are predicted to reach 10.5 trillion dollars annually by 2025 [28]. This alarming situation is further demonstrated by the adoption of recent cybersecurity regulations [44]. However, efficiently identifying the increasingly complex malicious activities from network traffic brings some major challenges.

Most traditional methods consider the statistical variations of traffic volume in order to detect anomalous traffic. Such approach is convenient for high-rate attacks where, for instance, a host is deemed as performing a port scan if a sudden increase in the volume of destination port numbers is noticed. However, low-rate attacks such as link-flooding [27], are not visible in a large network and thus, are very difficult to discover. Moreover, volume-based techniques are inadequate to capture the correlations in communications between hosts and the impact of cross-host anomalous activities since they focus on the statistics of traffic sent by individual hosts.

This paper proposes *Nadege*, a graph-based learning framework which aims at detecting anomalous hosts in networks and classifying traffic. First, we model the communication patterns of each host by a graph structure of multiple network flows that provides a compact representation of its network activity. Then, we develop a new graph kernel to measure the similarities between these activity graphs by leveraging correlations between flows of all the hosts in a network, such that it is possible to detect stealthy anomalies.

While considering the local structural information of each host is important, neglecting the context is insufficient to distinguish whether a network flow is triggered with or without other anomalous flows. Therefore, we increment our graph kernel with the ability to learn representative latent features from activity graphs with the aid of both local and contextual views. The key observation behind *Nadege* is the collective integration of the graph structure and node attributes information in our graph kernel at different network layers. Such integration enables us to characterize each change of anomalies by specific patterns of a particular feature set.
Our contributions can be summarized as follows.
- We advocate a general end-to-end learning framework to detect traffic anomalies and classify network traffic using a graph-based kernel approach.
- We prototype our framework by providing efficient algorithms, with theoretical approximation guarantees, to ensure a fast and accurate feature extraction. To the best of our knowledge, this paper is the first to formulate a graph kernel

crafted from a networking perspective and that is tailored to the analysis of network flows.

- Finally, we perform extensive experiments using real-word traces from different providers (e.g., ISP, University). In our evaluation based on a variety of network environments, both our graph kernel and *Nadege* achieve significant improvements over the popular baselines.

The rest of this paper is organized as follows. In Section II, we review the related works. In Section III, we present details and analysis of each component of our framework. In Section IV, we describe the learning design choices. Then, we evaluate *Nadege* on several traces and discuss the obtained results. Finally, we draw our conclusions in Section V. All proofs can be found in Appendix VI.

II. Related work

Graph kernels. Most of graph kernels have been defined as part of the R-convolutional framework [48] where graphs are decomposed into small substructures, then compared by counting the frequency of their matching substructures such as graphlets, shortest paths, trees, among others. However, it is worth noting that an increase in the size of substructures leads to a decrease in the probability of two graphs sharing common substructures, which results in the so called diagonal dominance issue [23], leading to models prone to overfitting. Our fingerprint kernel differs significantly from the R-convolution based graph kernels. We capture similarity between graphs by an implicit extraction of structural nodes and pairs of nodes, avoiding consequently a direct decomposition into substructures.

Another family of works derives kernels based on spectral, propagation or geometric approaches. For example, optimal assignment kernels [22] represent a departure from the R-convolution framework by computing a matching between substructures of objects such that the overall object pairwise similarity is maximized. Unlike our proposed kernel, they are unfortunately not guaranteed to be positive semi-definite for all choices of base kernels [47]. For a further review of graph kernels, we refer the reader to the surveys [23], [33].

Graph-based anomaly detection. Much interest has been generated for anomaly detection methods using graphs. The first family of techniques explores probabilistic or graph similarity measures [25]. The second family is based on machine learning and comes in supervised or unsupervised branches [21]. Our hybrid framework combines techniques from both families by devising a learning-flavored similarity kernel from a networking perspective. The closest works to ours are [51] and [16].

In [51], the authors combine the standard shortest-path graph kernel [5] with a deep convolutional neural network to analyze network attacks. However, since shortest-paths do not consider neighborhood structures, the graph similarity is only captured at fine granularities [52]. Moreover, the shortest-path kernel requires a quartic time complexity in the size of the graphs and thus, is expensive to compute for very large graphs. In addition, deep neural networks require a large amount of training data and are computationally expensive, especially when considering the entire network flows.

[16] introduces a graph-mining technique to detect network anomalies. In their model, they consider a single host-based reduced graph representation built from self-harvested network flow data. To detect anomalies at the host level, they first count the automorphism orbits of graphlets—small induced subgraphs that describe local topology. Then, a robust algorithm fits the gaussian distribution using the Minimum Covariance Determinant method of [35] on the sequence of graphlets for outlier detection. While the traffic flows collected have similar attributes, our per-host graph model uses a clear-cut representation of all network flow information. We compare the performance of *Nadege* to the ones of existing methods in Section IV.

TABLE I: Glossary of notations.

Symbol	Description
$\mathcal{A}$	Activity graph
S	Source set of size s
T	Destination set of size t
l_i	The i-th layer of $\mathcal{A}$
J	Weighted adjacency matrix
$\widetilde{J}$	Normalized adjacency
D	Weighted degree matrix
P	Transition probability matrix
$\widetilde{\mathcal{L}}$	Normalized Laplacian
$\mathcal{S}$	Set of activity graphs
$\mathcal{M}_{\mathcal{S}}$	Merge graph

III. Framework description

In this section, we define the notation used in the rest of the paper. Then, we introduce each component of *Nadege* (which stands for **N**etwork **A**nomaly **DE**tection with **G**raph k**E**rnels).

A. Preliminaries and Notation

Consider an undirected labeled graph $G = (V, E, l, w)$ where V is a set of graph vertices of size $n = |V|$ and E is a set of graph edges of size $m = |E|$, while $l : V \to \mathbb{N}$ and $w : E \to \mathbb{N}$ are functions that assign labels from a set of positive integers to nodes and edges, respectively. The weighted adjacency matrix J of G is an $n \times n$ symmetric matrix such that $J[i, j] = w_{ij}$. The weighted degree matrix D of G is a diagonal matrix such that $D[i, i] = \sum_{(i,j)\in E} w_{ij}$. The first and last i rows of D are denoted by $D_{:i}$ and $D_{i:}$, respectively. Moreover, we denote a set by $\{a, b, ..., c\}$, an ordered set (tuple) by $(a, b, ..., c)$ and a multiset by $\{\!\{a, b, ..., c\}\!\}$. $\mathcal{N}_v^i$ is the i-th neighborhood of vertex v. Finally, $\odot, \oplus, \wedge, \dot{\vee}$ denote the Hadamard product, the concatenation, logical conjunction and exclusive disjunction operators, respectively.

A walk of arbitrary size k initiated from node v_0 is defined as a sequence of vertices $\{\!\{v_0, \ldots, v_k\}\!\}$ such that $\{v_i, v_{i+1}\} \in E$ for $0 \leq i \leq k-1$. A path is a walk that consists of all distinct vertices and edges. A random walk on G is a discrete-time Markov chain $(X_0, X_1, X_2...)$ endowed with the Markov property [26], which induces the transition matrix $P = JD^{-1}$.

We summarize the notation in Table I.

B. Activity Graphs

Due to privacy, legal, and technological constraints [53], it is important to consider a profiling mechanism that does not expose private or personally identifiable information (PII) and, a fortiori, does not access the packet payload. In addition, computing behavioral statistics over databases of network streams is incomprehensive due to the existence of redundant features and the difficulty in interpretation without further processing. Thus, our aim is to use the flow records gathered by each host in order to build a compact graph structure to track its profile activity, while limiting the redundancy. We model a network flow by a path graph (see Fig. 1(a)) which acts as a summarized indicator.

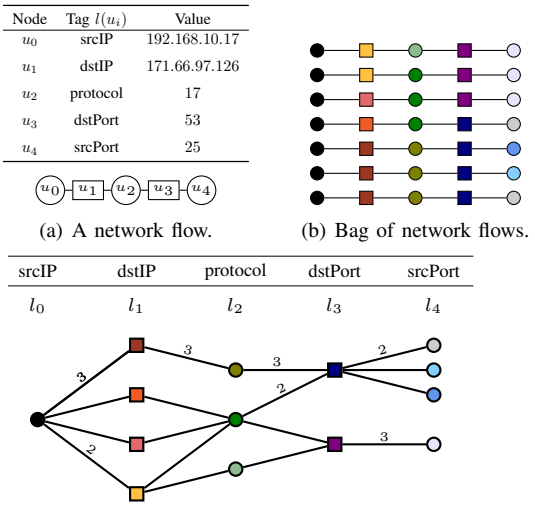
Definition 1. *A bag of network flows associated with a host is a set of flows, i.e., of labelled chains such that each chain $\mathcal{P} = (u_0, ..., u_f)$ is made up of a sequence of identifiers which i-th node is tagged by $l(u_i)$ and, where u_0, u_f correspond to the end-points of the flow.*

Observing all the incoming and outgoing flows of a host during a given time window, gives rise to a set $\mathcal{B}$ of network flows. Clearly, in a bag of flows, the source IP is always unique (as depicted by the black circles in Fig. 1(b)). This collection is then used to build the profile activity of a host according to a two-step procedure: we start by arranging a graph in layers $l_0, ..., l_f$ corresponding to the tagged identifiers in the flows. To make the layers intuitively tractable, we will refer by source (resp. destination) layers those located at an index position which is even (resp. odd). Next, we merge the network flows while counting the frequency of appearance of each edge across the bag $\mathcal{B}$. Thus, each flow creates a path starting from the host IP address on the first layer and traversing entities in each subsequent layer in the graph (see Fig. 1(c)).

Definition 2. *(Activity graph): Let $\mathcal{B}$ be a bag of network flows associated with a host. An activity graph of a host is a weighted connected graph defined by $\mathcal{A} = (\mathcal{B}, E, w)$ where $w : E \to \mathbb{N}$ is the count of flows contributing to each edge.*

Such representation served useful when it comes to identify application patterns [20] in a rather straightforward and intuitive manner, without accessing the packet payload. We further incorporate weights in order to track more latent temporal features, such as the number of packets or bytes for all flows transiting a given path. The main key advantages of handling activity graphs are: (i) they achieve a lossless representation of all flow information while operating in a single structure, (ii) they evolve in a way that reflects evolutionary changes over time and, (iii) they allow intuitive interpretable information through the pairwise interaction between layers. We note that activity graphs can be used in conjunction with several protocols running over TCP/IP in a wide variety of environments such as the MQTT protocol in Internet of Things (IoT), augmenting thereby their diameter while remaining expressive.

We notice that any given activity graph has a bipartite



(a) A network flow. (b) Bag of network flows.

(c) An activity graph reflecting communication patterns.

Fig. 1: Generation instance of an activity graph ($f = 4$).

structure, represented by a partition (S, T) such that $S = \bigcup_{i\equiv 0[2]} l_{i+1}$ collects nodes[1] from source layers, while $T = \bigcup_{i\equiv 1[2]} l_{i+1}$ collects nodes from destination layers. The next proposition shows that the biadjacency matrix, denoted by L_{st}, suffices to uniquely represent an activity graph from its partition (S, T) with source (resp. destination) set S (resp. T) of size denoted by s (resp. t). The proof follows directly from the bipartitness of an activity graph.

Proposition 1. *Let $k \in \mathbb{N}$. The weighted adjacency matrix J for an activity graph and its powers can be expressed as*

$$J = \begin{array}{c} \\ S \\ T \end{array} \begin{array}{c} \begin{array}{cc} S & T \end{array} \\ \begin{bmatrix} \mathbf{0} & \mathbf{L}_{st} \\ \mathbf{L}_{st}^T & \mathbf{0} \end{bmatrix} \end{array}, \quad J^{2k} = \begin{bmatrix} (\mathbf{L}_{st}\mathbf{L}_{st}^T)^k & \mathbf{0} \\ \mathbf{0} & (\mathbf{L}_{st}^T\mathbf{L}_{st})^k \end{bmatrix}$$

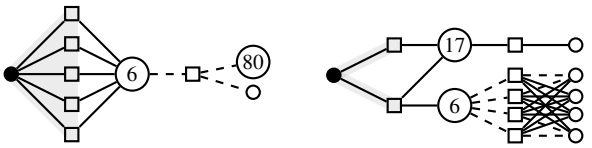
and

$$J^{2k+1} = \begin{bmatrix} \mathbf{0} & (\mathbf{L}_{st}\mathbf{L}_{st}^T)^k\mathbf{L}_{st} \\ (\mathbf{L}_{st}^T\mathbf{L}_{st})^k\mathbf{L}_{st}^T & \mathbf{0} \end{bmatrix}$$

C. Intrinsic Fingerprint

In order to capture the local-global characterization of an activity graph, we need a processing mechanism with the ability to zoom in and out the topological environment of each node. To this end, we capitalize on random-walk based methods for their relationship to multiple graph properties. For instance, [54] derived features based on the isomorphism-invariance property of the return probabilities of random walks. The efficiency of these features lies on their convergence to a certain value (denoted by π) known as the stationary probability in Markov chain theory [26]. However, such a stable distribution does not exist for activity graphs due to their bipartiteness. To circumvent this limitation, we explore the graph using an ergodic random walk variant $\widehat{W}$ where at

[1]The term node indicates the component of an activity graph, while the term host indicates a communicating device.



(a) Benign host. (b) Attack host.

Fig. 2: Structural roles for some intrinsic components.

each time step, with probability $1/2$, the walk remains at its current node, otherwise it jumps to one of its neighbors.

Proposition 2. *Let $\mathcal{A}$ be an activity graph. Let $R_{\mathcal{A}}$ denote the transition probability matrix induced by $\widehat{W}$ on $\mathcal{A}$. Then $R_{\mathcal{A}} = \frac{1}{2}(I_{\mathcal{A}} + J_{\mathcal{A}} D_{\mathcal{A}}^{-1})$.*

Definition 1. *(Intrinsic Fingerprint):*
Let $\tau > 0$. The intrinsic fingerprint of an activity graph $\mathcal{A}$ is the set of vectors

$$i_{\mathcal{A}} = \{i_v, v \in V_{\mathcal{A}}\}, \tag{1}$$

where $i_v = \left[R^1_{\mathcal{A}}[v,v], ..., R^{\tau+1}_{\mathcal{A}}[v,v]\right]^T$ and τ are referred to by the intrinsic component and range, respectively.

The intrinsic fingerprint provides a rich information on the graph structure and can be seen as a subgraph centrality measure. By focusing on closed walks, it captures the interaction of a node with the subgraphs involving it. Moreover, such structural role characterization blends well with the interpretation in terms of the host profile activity. Fig. 2(a) displays an instance of a benign host in our traces corresponding to an HTTP web server. Fig. 2(b) displays an attack where the host is attempting to connect through TCP to potential vulnerable ports at the same destination host.
At the first layer, the intrinsic fingerprint captures the popularity of a host in terms of the number of other hosts it communicates with. At the third level, as illustrated by the dashed lines, the star and cycle subgraphs involving the TCP node reflect functional role patterns where the last layer benefits from the knowledge acquired in the previous layers. In the former subgraph, portrayed by a knot at the destination port layer, the host is providing a service which translates into the use of a single port for the majority of its interactions. In the latter subgraph, closed walks of order four unveil in this case a port scan where the host identifies vulnerabilities at specific destination ports.

A straightforward computation of $i_{\mathcal{A}}$ takes $\mathcal{O}(\tau(s+t)^3)$ by successively computing the powers of R. Since only the diagonals of the transition matrices are required, we use Proposition 3, based on SVD, which leads to Algo. 1 and shows that we can do better. We note that SVD can be approximated in time linear to the number of non-zero entries [13], which makes the practical computation fast due to the sparsity of the biadjacency matrix.

Proposition 3. *Let $\mathcal{A}$ be an activity graph. $i_{\mathcal{A}}$ can be computed in $\mathcal{O}(st\min(s,t)+(\tau+1)(s^2+t^2)+\tau(\tau+1)(s+t))$.*

Clearly, the intrinsic fingerprints are sensitive to the choice of the intrinsic range which intuitively, corresponds to the depth of the graph explorations initiated from the nodes. Therefore, we need a provable guarantee which bridges the exploration intensity with the informativeness of the extracted features. To this end, Theorem 1 tells us how far we need to explore, while ensuring a controlled view on the scope towards the stationary distribution.

Algorithm 1 Intrinsic Fingerprint

Input: An activity graph $\mathcal{A}$, depth parameter δ in (0,1]
Output: $i_{\mathcal{A}}$

```
Compute N = D_{:s}^{-1/2} L_{st} D_{t:}^{-1/2}
N = U Σ V^T                                   ▷ SVD Reduction
Compute τ_δ according to Eq. (2)
parallel for k = 1, ..., τ_δ + 1 do
    Compute i_A^(k) according to Eq. (7)
    parallel for v in V_A do
        Add i_A^(k)[v] to i_v                 ▷ Intrinsic Component
Collect i_A according to Eq. (1)
return i_A
```

Theorem 1. *Let $\mathcal{A}$ be an activity graph and δ in $(0,1]$. If*

$$\tau_\delta \geq \frac{2}{1+\frac{1}{\sqrt{st}}} \log(\frac{n}{\delta}), \tag{2}$$

then $\nu(p_t, \pi) \leq \delta$ where $\nu(p,q) = \|p-q\|_1$ denotes the total variational distance and p_t the probability distribution of the position of the random walk at time t.

When $\delta = \frac{1}{4}$, the depth is equivalent to the mixing time [26].

D. Contextual Fingerprint

Besides identifying the notable structural attributes of activity graphs, it is crucial to consider the context in which they emerge. Different hosts exhibiting different behaviors may induce activity graphs with similar shapes, resulting in limited identification as collective information about the current state of the network is not exploited. In such cases, relying solely on the intrinsic feature is optimistic since two isomorphic graphs have the same intrinsic fingerprints. Thus, analyzing the activity of a host among group of hosts provides more insight about the prospective state of the network.

As an example, let us consider three hosts a, b and c. Fig. 3 presents a *merge* graph constructed by merging their corresponding activity graphs $\mathcal{A}_a$, $\mathcal{A}_b$ and $\mathcal{A}_c$ (depicted by the double circles) into a unified representation. While $\mathcal{A}_b$ and $\mathcal{A}_c$ are isomorphic, they correspond to different malicious hosts, respectively, a UDP-based port scanner and an IP address space scanner. On the other hand, $\mathcal{A}_a$ corresponds to a bot used to perform a Distributed Denial-of-Service (DDoS) attack while acting as a DNS and mail server (indicated by the SMTP source port identifier). Since the layers of activity graphs encode relationships between the hosts at several levels, considering pairs of nodes over a unified graph enables us to take into consideration the temporal nature of host group behaviors and understand complex attacks. For instance, a pair from layers l_0, l_1 enables us to track the host a (as depicted

in red in Fig. 3) across the activity graphs of hosts which fall into the same temporal traffic window. In this scenario, both hosts b, c act as C&C channels for the botmaster a.

We thus devise in this section a *method* (see Algo. 2) *based on a set of merged activity graphs* where pairs of nodes provide latent collective representations according to a *variant of the Weisfeiler-Lehman isomorphism test*. We further equip it with an *efficient approximation calculation technique* (see Algo. 3).

Message passing models are currently the core element of the recent attention in the use of deep learning on graphs. [31], [50] proved somewhat remarkably that graph neural networks (GNNs) [49], which are based on these models, are less expressive than the k-Weisfeiler-Lehman (WL) isomorphism tests hierarchy ($k \in \mathbb{N}^*$) [12]. The popular 1-WL kernel [40] can only support a measure of similarity based on the occurrences of subtree patterns ([39], [2]) over each independent node in the graph, as a byproduct. Meanwhile, going for higher dimensions in the WL hierarchy is computationally expensive as k-WL runs in $\mathcal{O}(n^k)$ where n is the graph size [1]. Thus, we leverage the 2-WL which offers an effective middle ground framework to reason collectively over pairs of nodes. For a review of 2-WL with its different neighborhood rules, we refer the reader to [29], [30].

Our method differs from the standard 2-WL procedure in two important ways. First, 2-WL is executed over each graph g independently, then g is associated with a feature vector $\psi(g)$ where, for each σ in Σ, its coordinate $\psi_\sigma(g)$ counts the frequency of the color σ of the procedure in g. Instead, we operate over a single *merge graph* (see Def. 2 and Fig. 3) to capture the contextual interactions between activity graphs so that each frequency $\psi_\sigma(g)$ depends on all the graphs in $\mathcal{S}$ and not only g. Second, we consider a novel *adaptive neighborhood* policy which adapts the exploration based on a *node centrality* measure that we compute efficiently.

Definition 2. *(Merge graph): Let $\mathcal{S}$ be a family of activity graphs. Let s_u and t_u be two universal nodes acting as source and sink, respectively. The merge graph $\mathcal{M}_\mathcal{S}$ is unweighted undirected, characterized by $V_\mathcal{M} = \bigcup_{g\in\mathcal{S}} V_g \cup \{s_u, t_u\}$ and $E_\mathcal{M} = \bigcup_{g\in\mathcal{S}} E_g \cup \{(s_u, v), (w, t_u), v \in l_0(g), w \in l_4(g)\}$.*

Node centrality. The centrality measure that we consider comes from Lemma 1 which tells us that the eigenpairs, namely the eigenvectors, can be easily retrieved while executing Algo. 1 when computing the intrinsic fingerprints; therefore avoiding a separate centrality computation. Specifically, we associate with each node v its component (denoted by $f(v)$) in the eigenvector corresponding to the largest eigenvalue (which always equals 1) of $\widetilde{J}$. Note that $f(v) > 0$ from the Perron-Frobenius theorem [26]. The use of $f(v)$ as a centrality measure is motivated by the following observation:

$$f(v) = [\widetilde{J}f][v] = \sum_{u\in V_\mathcal{A}} \widetilde{j}_{vu} f(u) = \sum_{u\in\mathcal{N}(v)} \frac{1}{\sqrt{d_v d_u}} j_{vu} f(u). \tag{3}$$

Eq. (3) tells us that assigning $f(v)$ to v is equivalent to implicitly averaging over its neighbors according to their interaction intensity (dictated by the degree function d). Thus, $f(v)$ corresponds to an eigenvector centrality measure that considers that a node v with a neighbor u solely connected to it, provides more information that a same node v for which u is connected to many others at the same time.

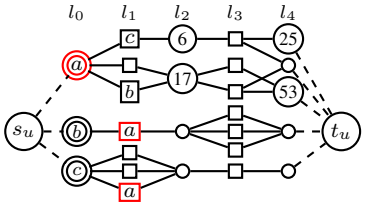


Fig. 3: Instance of a merge graph with hosts a, b and c.

Algorithm 2 Contextual Fingerprint

Input: A set $\mathcal{S}$ of activity graphs, θ an iteration parameter
Output: A set of features $c_\mathcal{S}$

```
Construct the merge graph M_S
Color each pair p = (u, v) of M_S per isomorphism type
parallel for g ∈ S do
    for all r = 1, ..., θ do
        Compute standard 2-WL over M_S using Eq. (4)
        for all σ ∈ Σ_r do
            p^in_σ ← |{(u, v), u ∧ v ∈ V^2_g with color σ}|
            p^out_σ ← |{(u, v), u ∨̇ v ∈ V^2_g with color σ}|
            ψ_{r,σ}(g) ← (p^in, p^out)^T
        ψ_r(g) ← (ψ_{r,σ_1}(g) ... ψ_{r,σ_|Σ_r|}(g))
    c_S[g] ← ⊕_r vec(ψ_r(g))
return c_S
```

Lemma 1. *From the SVD of $N = U\Sigma V^T$, the eigendecomposition of $\widetilde{J}$ can be retrieved by writing*

$$\widetilde{J} = \begin{bmatrix} 0 & N \\ N^T & 0 \end{bmatrix} = B \begin{bmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{bmatrix} B^T \text{where } B = \begin{bmatrix} \widetilde{U} & \widetilde{U} \\ \widetilde{V} & -\widetilde{V} \end{bmatrix}$$

and $\widetilde{U} = \frac{U}{\sqrt{2}}, \widetilde{V} = \frac{V}{\sqrt{2}}$.

Adaptive neighborhood. As seen in Sec. III-B, end-nodes in a pair of the activity graph play different roles depending on the layer they fall into. Thus, instead of treating both end-nodes in the same way, we define a new neighborhood rule where the neighborhood of a pair (u, v) is dynamic and based on the previous node centrality measure:

$$\mathcal{N}_{uv} = \{uw, w \in \mathcal{N}_v^*\} \cup \{wv, w \in \mathcal{N}_v^*\}, \tag{4}$$

where $\widetilde{f}(v) = \frac{f(v)}{\sum_{v\in V_\mathcal{A}} f(v)} \in (0, 1]$ and $\mathcal{N}_v^* = \mathcal{N}_v^1 \cup \mathcal{N}_v^2$ if $\widetilde{f}(v) < 0.5$, otherwise $\mathcal{N}_v^* = \mathcal{N}_v^1$. Intuitively, a low-central node needs to cover a larger portion of its neighborhood compared to a high-central node which is already connected to high-central nodes according to Eq. (3).

Approximation speedup. Since the propagation procedure of 2-WL runs in $\mathcal{O}(n^2)$, to make sure that *Nadege* is able

Algorithm 3 Speedy Contextual Fingerprint

Input: A set $\mathcal{S}$ of activity graphs, θ a refinement parameter, ν confidence probability, ε error tolerance
Output: $\widetilde{c}_{\mathcal{S}}$

Construct the merge graph $\mathcal{M}_{\mathcal{S}}$
Populate $\Gamma^{\mathcal{S}}_{\nu,\varepsilon,\theta}$ according to (9) ▷ Uniform sampling
parallel for $p = (u, v) \in \Gamma^{\mathcal{S}}_{\nu,\varepsilon,\theta}$ **do**
    $\mathcal{H}_p \leftarrow \{g \in \mathcal{S} | u \in V_g \vee v \in V_g\}$ ▷ Involved hosts
    Build $\mathcal{G}_p$ the induced subgraph over $\mathcal{N}(p)$
    Color each node of $\mathcal{G}_p$ per isomorphism type
    for all $r = 1, ..., \theta$ **do**
        Determine $\sigma(p)$ over $\mathcal{G}_p$ ▷ Run diffusion process
        parallel for $g \in \mathcal{H}_p$ **do**
            $i \leftarrow \mathbb{1}_{|\mathcal{H}_p|=1}$
            $\widetilde{\psi}_{r,\sigma}(g)[i] \leftarrow \widetilde{\psi}_{r,\sigma}(g)[i] + \frac{1}{|\Gamma^{\mathcal{S}}_{\nu,\varepsilon,\theta}|}$
$\widetilde{c}_{\mathcal{S}}[g] \leftarrow \oplus_r \mathrm{vec}(\widetilde{\psi}_r(g))$
return $\widetilde{c}_{\mathcal{S}}$

to operate with high velocity data-streams, we propose an algorithm (see Algo. 3) which provably (see Theorem 2) approximates the exact features in Algo. 2.

Theorem 2. *Let $\mathcal{S}$ be a set of activity graphs and $\mathcal{M}$ its associated merge graph. Let us fix r in $\{1, ..., \theta\}$, ε in $(0, 1]$ and $\nu > 0$. Then, there exists $\Gamma^{\mathcal{S}}_{\nu,\varepsilon,\theta}$ in $V^2_{\mathcal{M}}$ such that with probability ν, it holds that*

$$\forall g \in \mathcal{S} : \left\| \psi_r(g) - \widetilde{\psi}_r(g) \right\|_F \leq \varepsilon,$$

where $\|.\|_F$ denotes the Frobenius norm.

E. Fingerprint Graph Kernel

Let $\mathcal{S}$ be a set of activity graphs. Each g in $\mathcal{S}$ is represented by its fingerprint set $r_g = \{(i_v, c_g)\}_{v \in V_g}$ in a real feature space denoted Ω. Let k be a kernel on Ω such that for (g, h) in $\mathcal{S}^2$:

$$k[(i_v, c_g), (i_u, c_h)] = \left(\frac{\|i_v - i_u\|^2}{2\sigma^2} \right) \exp\left(\frac{-\|c_g - c_h\|^2}{2\sigma^2} \right), \tag{5}$$

with bandwidth $\sigma > 0$. Let ϕ and $\mathcal{H}$ be the feature map and the reproducing kernel Hilbert space (RKHS) associated with k; whose existence is derived from Moore–Aronszajn theorem. We view r_g as a sample following a distribution denoted by $\mathcal{D}_g$ that we embed into a mean map $\mu_{\mathcal{D}_g}$ in $\mathcal{H}$ by kernel mean embedding (KME), as it suffers less from the curse of dimensionality compared to density estimation methods [32]:

$$\mu_{\mathcal{D}_g} := \mathbb{E}_{X \sim \mathcal{D}_g}[k(X, .)] = \mathbb{E}_{X \sim \mathcal{D}_g}[\phi(X)] = \int_{\Omega} \phi(x) d\mathcal{D}_g(x).$$

We choose Gaussian kernels in Eq. (5) as they belong to the kernel class of radial basis functions (RBFs) which enjoy nice properties [46]. In particular, their universality property implies that KME retains all information, namely moments, about the distribution $\mathcal{D}_g$. Further details on the properties of the RKHS and different classes of kernel functions can be found in [46], [32].

Since access to the true distribution $\mathcal{D}_g$ is lacking, we use the empirical mean estimator $\widehat{\mu}_{D_g} = \frac{1}{n_g} \sum_{v \in V_g} \phi(i_v, c_g)$, by treating the data as a probability mass distribution associated with r_g, i.e., $\mathcal{D}_g = \frac{1}{n_g} \sum_v \delta_{(r_v, c_g)}$ with δ_x the Dirac measure defined for x in Ω. We note that $\widehat{\mu}_{\mathcal{D}_g}$ is a good proxy for $\mu_{\mathcal{D}_g}$ as it is is unbiased and consistent [43], with convergence rate $\mathcal{O}(\mathcal{R}_{n_g}(\mathcal{H}) + \frac{1}{\sqrt{n_g}})$ where $\mathcal{R}(\mathcal{H})$ denotes the Rademacher complexity of $\mathcal{H}$.
Thereby, we define the *valid* (see Proposition 4) fingerprint graph kernel K (denoted by FGK) as follows

$$K(g, h) = \langle \widetilde{\mu}_{\mathcal{D}_g}, \widetilde{\mu}_{\mathcal{D}_h} \rangle_{\mathcal{H}} = \frac{1}{n_g n_h} \sum_{v,u} k[(i_v, c_g), (i_u, c_h)].$$

Proposition 4. *The fingerprint graph kernel K is symmetric positive definite.*

Computing K takes $\mathcal{O}(|\mathcal{S}|^2 n^2)$ (we assume for simplicity that all graphs have the same size n). Since $\mathcal{S}$ could be potentially large, we use an approximate kernel transformation known as Fast-Food [24]. For each fingerprint x in $\mathbb{R}^{\mathrm{d}}$, the latter provides a randomized explicit feature representation $\widetilde{\phi}(x)$ in $\mathbb{R}^p$ (with $p < d$), running in $\mathcal{O}(d \log p)$ while ensuring that $\langle \widetilde{\phi}(x_i), \widetilde{\phi}(x_j) \rangle$ converges to $\langle \phi(x_i), \phi(x_j) \rangle$ with a rate of $\mathcal{O}(\frac{\log(\frac{2}{\delta})}{\sqrt{d}})$ where δ is a failure probability.

IV. Experimental evaluation

In this section, we provide a broader evaluation assessment of *Nadege* in terms of its detection and runtime performance. We open by describing the evaluation setup, followed by the learning models, and finally, close by presenting our results.

A. Evaluation setup

Datasets. We conducted experiments on six widely-used and recent datasets covering different types of network environments. Detailed descriptions of these traces can be found in [17]. We provide some relevant summary statistics in Table III. All traces labeled the network anomaly hosts, which means we have ground truth of the traffic. Moreover, they vary in volumes and forms of anomalies mixing both high and low-intensity attacks. Thus, overall these traces offer a good diversity for evaluating *Nadege* robustness across different real-word situations.

Baselines. We compared *Nadage* with two categories of state-of-the-art approaches. *Graph kernel-based* methods include the Shortest-Path kernel (SP) [5], the Random-Walk kernel (RW) [48], the Graphlet kernel (GR) [41], and the Weisfeiler-Lehman subtree kernel (WL) [40]. We also picked GraphSAGE (GS) [14], a representative inductive *learning-based* method which fit into the graph neural network framework proposed by [11]. In particular, these methods serve also at assessing the performance of the Fingerprint kernel. The second category comprises *anomaly-based* methods, namely AnoNG (ANG) [51] and Graphprints (GP) [16] (described in Section II).

B. Learning

Metrics. The detection performance of an algorithm, on a particular trace, can be measured in terms of its true

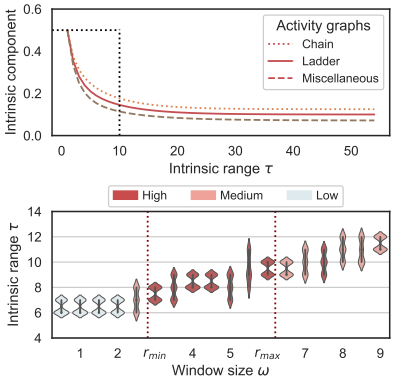


Fig. 4: Parameter sensitivity analysis.

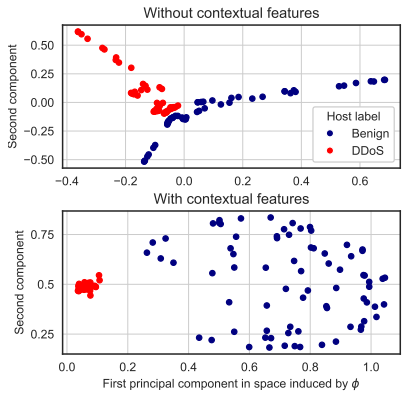


Fig. 5: Kernel-PCA projection of FGK.

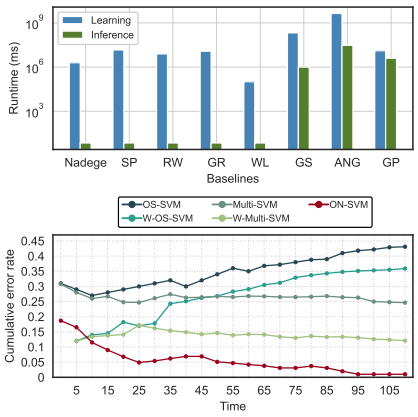


Fig. 6: Runtime and cumulative error rates.

positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). In our evaluations, we consider for each approach three metrics: precision ($\mathcal{P}=\frac{TP}{TP+FP}$), recall ($\mathcal{R}=\frac{TP}{TP+FN}$), and F1-score ($\mathcal{F}=\frac{2\times\mathcal{P}\times\mathcal{R}}{\mathcal{P}+\mathcal{R}}$). In network intrusion detection, it is important to obtain a minimal number of false alarms since it is time consuming and expensive for an analyst to investigate each alarm.

Sliding window principle. We monitor the flow traffic of each trace using overlapping sequential observation windows to mimic a sliding time window model. Each window gives rise to a set $\mathcal{S}$ of labeled (benign or anomalous) activity graphs which are characterized by a similarity matrix $K_{\mathcal{S}}$ resulting from the fingerprint graph kernel. We then feed $K_{\mathcal{S}}$ into a kernelized Support Vector Machine (SVM) learning model, while considering both offline and online execution modes.

Models. As a state-of-the-art kernel method, SVM classifiers [6], [8] find an optimal separating hyperplane specified by a subset of the training data points, known as support vectors (SVs). Their success comes from a statistical learning theory viewpoint where they operate under the principle of risk minimization, providing theoretical guarantees on the generalization error. For further details on kernel methods, we refer the reader to [19]. We consider several learning models which all support class-weighting which is designed to deal with class imbalance:

– *Offline models.* We consider four models: OS-SVM (C_1), Multi-SVM (C_2), W-OS-SVM (C_3), W-Multi-SVM (C_4) where OS stands for One Shot and W for Wide. For OS-SVM, the model is trained only once on the first batch of windows. For Multi-SVM, the model is retrained every day by using, however, only samples from the previous day. W-OS-SVM mimics OS-SVM and W-Multi-SVM mimics Multi-SVM, however, the batch size for training and retraining covers several batches instead of one.

– *Online models.* We consider a popular incremental version of SVM, denoted by ON-SVM (C_5), which processes data on the fly and modifies its hyperplane, if necessary, as new training samples arrive [4]. Moreover, it introduces a SV removal step and supports active selection.

Settings. In all our experiments, we followed common practices of performance evaluation by repeating each 10-fold internal cross validation. Within each fold, we selected the trade-off C of SVM from $\{10^{-3}, ..., 10^{3}\}$ and the depth θ from $\{0, \ldots, 7\}$. We set the parameters δ and ε to 0.1 and 0.05 respectively, ν to 0.95, and σ in the FGK to $1/d_{att}$ with d_{att} the attribute dimension [10]. Instead of using random folds for the validation of the approach, we preserved the temporal order of the network data. Indeed, it is essential for measuring the performance of intrusion detection systems that the training and test data are temporal disjunct and future attacks are not available during learning to avoid experimental bias [34]. Moreover, we made use of the GraKel library [42] for graph kernel implementations and normalized all kernel matrices. We note that GraKel was entirely developed in Python, is compatible with scikit-learn, and exploits the Cython extension to benefit from a fast implementation in C.

C. Results and Discussions

We first discuss the strengths of our fingerprints, then, we evaluate the detection performance of *Nadege*.

Intrinsic fingerprint. We start by a sensitivity analysis over the intrinsic range τ which captures the depth of the closed walks explorations. Fig. 4 shows the maximum range τ_δ across all datasets per window size ω. We note the existence of an optimal region $[r_{min}, r_{max}]$ where the detection for *Nadege* is always high (i.e., F-score > 0.9 on average, across all datasets). When $\omega < r_{min}$, features are under-representative since graphs are mostly chains and very sparse trees and, when $\omega > r_{max}$, the SVM decision problem becomes coarser. As the time step goes to infinity, the intrinsic components will not change much, converging to the stationary probability. Hence, we gain little new information from the intrinsic fingerprint by increasing τ (see Fig. 5).

Contextual fingerprint. In Table. II, we report the average F-score across all datasets along with the node utilization (defined as the *normalized* average value of $\sum_{p\in\Gamma}|\mathcal{N}_p|$ across merge graphs, with Γ the sample size of Algo. 3),

TABLE II: Exploration strategies.

Method	Policy	Score	Utilization
2-WL	Global	0.8941	49.7%
2-FWL	Folklore	0.9157	49.7%
2-LWL	Local	0.8840	23.8%
Nadege	Centrality	0.9716	25.0%

TABLE III: Statistics and evaluation results on 6 traces.

Trace	Approach	Precision	Recall	F1-Score
D-1	***Nadege***	**0.9991**	**0.9987**	**0.9989**
Name: CSE-CIC-IDS-18	*Shortest-Path*	0.6740	0.7290	0.7290
#Total: 16,232,943	*Random-Walk*	0.6706	0.7174	0.6932
Benign: 82%	*Graphlet*	0.9724	0.9423	0.9571
Attacks: 17%	*WL-Subtree*	0.9905	0.9372	0.9631
> Brute force, Dos, DDoS	*GraphSAGE*	0.9962	0.9827	0.9894
PortScan, Web, Infiltration	*AnoNG*	0.9799	0.9960	0.9879
	Graphprints	0.9679	0.9631	0.9679
D-2	***Nadege***	**0.8184**	**0.9842**	**0.8937**
Name: UGR'16	*Shortest-Path*	0.6793	0.6187	0.6476
#Total: 12,517,654	*Random-Walk*	0.5275	0.6392	0.5780
Benign: 64%	*Graphlet*	0.8265	0.7392	0.7804
Attacks: 36%	*WL-Subtree*	0.8739	0.8283	0.8505
> Botnet, Blacklist, Dos,	*GraphSAGE*	0.7776	0.7329	0.7546
Scan, Spam	*AnoNG*	0.8591	0.9021	0.8801
	Graphprints	0.7973	0.8590	0.7973
D-3	***Nadege***	**0.9987**	**0.9985**	**0.9986**
Name: Bot-IoT	*Shortest-Path*	0.8134	0.7593	0.7854
#Total: 3,668,522	*Random-Walk*	0.5168	0.5789	0.5461
Benign: 42%	*Graphlet*	0.6372	0.6719	0.7789
Attacks: 58%	*WL-Subtree*	0.8095	0.8480	0.8283
> Scan, Dos, DDos, Theft	*GraphSAGE*	0.8612	0.7180	0.7831
	AnoNG	0.8612	0.7180	0.9870
	Graphprints	0.6635	0.7370	0.7402
D-4	***Nadege***	**0.9523**	**0.9892**	**0.9704**
Name: N-BaIoT	*Shortest-Path*	0.6537	0.9892	0.7872
#Total: 5,256,390	*Random-Walk*	0.4567	0.8820	0.6018
Benign: 67%	*Graphlet*	0.7137	0.8196	0.7630
Attacks: 33%	*WL-Subtree*	0.8272	0.8741	0.8500
> Botnets (Mirai & Bashlite)	*GraphSAGE*	0.8984	0.8120	0.8530
	AnoNG	0.9736	0.9548	0.9641
	Graphprints	0.6394	0.8548	0.7316
D-5	***Nadege***	**0.9780**	**0.9941**	**0.9860**
Name: Tor-nonTor	*Shortest-Path*	0.7347	0.6492	0.6893
#Total: 67,834	*Random-Walk*	0.7283	0.7239	0.7261
Benign: 78%	*Graphlet*	0.8074	0.7032	0.7517
Anomalies: 22%	*WL-Subtree*	0.8552	0.9290	0.8906
	GraphSAGE	0.9791	0.8410	0.9048
	AnoNG	0.8903	0.9300	0.8903
	Graphprints	0.7526	0.8320	0.7903
D-6	***Nadege***	**0.9757**	**0.9890**	**0.9823**
Name: VPN-nonVPN	*Shortest-Path*	0.5530	0.7930	0.6516
#Total: 125,634	*Random-Walk*	0.7012	0.7153	0.7082
Benign: 87%	*Graphlet*	0.6887	0.6136	0.6490
Anomalies: 13%	*WL-Subtree*	0.8109	0.8693	0.8391
	GraphSAGE	0.9853	0.9382	0.9382
	AnoNG	0.9085	0.8039	0.8530
	Graphprints	0.9332	0.8790	0.9053

which captures the intensity of exploration. We observe that centrality-based explorations offer a nice trade-off between contextual informativeness and the size of the exploration space as they adapt their strategy according to the current topology of the merge graph. In Fig 3, we performed a Kernel PCA [38] projection to the hosts in D-1 based on the FGK. We observe that while the intrinsic features provide an already strong discrimination against most attacks, adding the contextual features improves the classification performance of the FGK in discriminating some collaborative attacks such as Botnets and DDos.

Learning models. In Fig. 6, we report over time, the ratio of misclassifications over the number of classified graphs so far of the SVM models for D-1. C_3 and C_4 surpassing C_1 and C_2 respectively, reveals the necessity of frequently updating models over time to mitigate concept drift. C_3 and C_4 surpassing both C_1 and C_2 indicates the importance of training on more data. We obtained similar error curves on the other datasets where only C_4, C_5 yielded low generalization errors, demonstrating the effectiveness of *Nadege* in both offline and online execution modes.

Detection performance. In Table III, we observe that *Nadege* significantly outperforms all baseline methods in six datasets, while producing the lowest false alarm rates. In addition, *Nadege* yielded the most stable variances (i.e., gaps between maximum and minimum) among all methods. We report the good performance of AnoNG for low frequency attacks at the expense of, however, high memory and time costs (see Fig 6). Even though Graphprints and the Graphlet kernel produce similar features, the former performs better mainly due to the robust MCD [35] removal procedure of outliers. The Fingerprint kernel is adapted for network discrimination against anomalies as demonstrated by its superiority over the kernel-based methods. It is worth noting that WL and GraphSAGE exhibit close average scores as the latter can be seen as a continuous approximation to the WL test [14]. Overall, these observations enable us to advocate *Nadege* as an accurate, fast, and versatile learning framework.

V. Conclusion and Future Work

In this paper, we proposed *Nadege*, a new framework for anomaly detection and traffic classification in networks. In particular, we developed a graph kernel tailored for networking, based on random walks and the Weisfeiler-Lehman hierarchy of isomorphism tests. The advantages of the kernel lie on its effectiveness in capturing rich local and global information on the host profile activity, while incorporating context awareness. The framework was made fast and scalable by using special purpose graphs along with an approximation algorithm for which we derived some theoretical guarantees. In our evaluation using a variety of traces from heterogeneous environments, *Nadege* achieved high F1-scores, significantly outperforming existing state-of-the-art approaches. An interesting future work is assessing the performance of *Nadege* in the context of adversarial attacks and within next generation networks (SDN and NFV).

VI. Appendix

Sketch of Lemma 2 Proof

Proof. The proof follows by noting that $R_{\mathcal{A}} = \frac{1}{2}(I_{\mathcal{A}} + P_{\mathcal{A}})$ where

$$[P_{\mathcal{A}}]_{ij} = \begin{cases} \frac{1}{D[j,j]} & \text{if } \{i,j\} \in E \\ 0 & \text{otherwise} \end{cases}$$

is the probability of going from node j to i. □

Sketch of Proposition 3 Proof

Proof. For clarity of notation, we omit the graph subscript $\mathcal{A}$. Let us fix $k > 0$. We write

$$R^k = \frac{1}{2^k} \sum_{i=0}^{k} \binom{k}{i} (JD^{-1})^i = \frac{1}{2^k} \sum_{i=0}^{k} \binom{k}{i} D^{\frac{1}{2}} \widetilde{J}^i D^{-\frac{1}{2}} \quad (6)$$

where $\widetilde{J} = D^{-\frac{1}{2}} J D^{-\frac{1}{2}}$ is symmetric. We have

$$\widetilde{J} = \begin{bmatrix} 0 & D_s^{-\frac{1}{2}} L_{st} D_t^{-\frac{1}{2}} \\ D_t^{-\frac{1}{2}} L_{st}^T D_s^{-\frac{1}{2}} & 0 \end{bmatrix} \triangleq \begin{bmatrix} 0 & N \\ N^T & 0 \end{bmatrix}$$

Using Proposition 1, we note that

$$\widetilde{J}^{2i} = \begin{bmatrix} U(\Sigma\Sigma^T)^i U^T & 0 \\ 0 & V(\Sigma^T\Sigma)^i V^T \end{bmatrix}$$

and $\widetilde{J}^{2i+1}[v,v] = 0$, where we use a singular value decomposition on $N = U\Sigma V^T$ with U, V real orthogonal

matrices and Σ a diagonal with non-negative real values $\sigma_1 \geq ... \geq \sigma_{f:=\min(s,t)}$. Hence, Eq. (6) yields

$$i^{(k)} \triangleq \left[R^k[v_1,v_1],...,R^k[v_n,v_n]\right]^T = \frac{1}{2^k}\sum_{j\equiv 0[2]}^{k}\binom{k}{j}Y^j \quad (7)$$

where $Y^j = \left((U\odot U)W^{2j}\right)\oplus\left((V\odot V)W^{2j}\right)$ with $W^j = (\sigma_1^j,...,\sigma_f^j)^T$.

Decomposing N via SVD takes $\mathcal{O}(st\min(s,t))$ while computing Y^j for a given j requires $\mathcal{O}(s^2+t^2)$. By performing $\mathcal{O}(\tau(\tau+1)(s+t))$ additions, using Eq. (7), the result follows. □

Sketch of Theorem 1 Proof

Proof. Let p_t denotes the probability distribution of the position of the random walk at time t. First, we express a lower bound on $t(\delta)$ where

$$t(\delta) = \max_{p_0}\arg\min_t\{\nu(p_t,\pi)\leq\delta\}.$$

Observe that $p_t = R^t p_0$ and $\pi(v) = \frac{d(v)}{2|E|}$ where

$$R = \frac{1}{2}(I+JD^{-1}) = I - \frac{1}{2}D^{\frac{1}{2}}\widetilde{\mathcal{L}}D^{-\frac{1}{2}},$$

with $\widetilde{\mathcal{L}} = I - D^{-\frac{1}{2}}JD^{-\frac{1}{2}}$ the normalized Laplacian. Let $\lambda_1 = 0 \leq ... \leq \lambda_n = 2$ denote the spectrum of $\widetilde{\mathcal{L}}$ associated with the eigenvectors $\{r_i\}_{1\leq i\leq n}$. Since

$$\forall i\in\{1,...n\} : RD^{\frac{1}{2}}r_i = \alpha_i D^{\frac{1}{2}}r_i,$$

where $\alpha_i = 1-\frac{\lambda_i}{2}$, then $\{(\alpha_i, D^{\frac{1}{2}}r_i)\}_{1\leq i\leq n}$ are the eigenpairs of R and $\{D^{\frac{1}{2}}r_i\}_i$ forms a basis of $\mathbb{R}^n$ as $D^{\frac{1}{2}}$ has rank n. Hence, for a set of $\{\mu_i \triangleq r_i^T D^{-\frac{1}{2}}p_0\}_{1\leq i\leq n}$ and an arbitrary starting distribution p_0, as $\alpha_1 = 1$ and $\mu_1 D^{\frac{1}{2}}r_1 = \pi$, it holds

$$\nu(p_t,\pi) = \left\|R^t p_0 - \pi\right\|_1 \leq \sqrt{n}\ \left\|\sum_{i=2}^{n}\mu_i\alpha_i^t D^{\frac{1}{2}}r_i\right\|_2,$$

using Cauchy-Schwarz inequality. Also

$$\left\|\sum_{i=2}^{n}\mu_i\alpha_i^t D^{\frac{1}{2}}r_i\right\|_2^2 \leq \alpha_2^{2t}\left\|p_0\right\|_2^2 \leq \alpha_2^{2t}.$$

Consequently, $\nu(p_t,\pi)\leq\delta$ for $t(\delta)\geq\frac{2}{\lambda_2}\log(\frac{n}{\delta})$ using the inequality $(1-x)\leq e^{-x}$ for $x\geq 0$.

The characterization of λ_2 has been studied in [45]. In particular, it was proved that if G is connected bipartite such that $G\neq K_{s,t}$ then $\lambda_2\geq 1+\frac{1}{\sqrt{st}}$. Since $\mathcal{A}$ is a valid candidate, we set $\tau_\delta \triangleq \frac{2}{1+\frac{1}{\sqrt{st}}}\log(\frac{n}{\delta}) \geq t(\delta)$ which only depends on the size of the graph for δ fixed. □

In order to formulate concentration bounds and prove Theorem 2, we first derive an extension of the Hoeffding's inequality [18] to the multivariate case by applying it over several dimensions.

Lemma 2. *(Multivariate Hoeffding's inequality)*
Let $\mathcal{Z}_1,...,\mathcal{Z}_n$ be independent r-dimensional random variables such that $\left\|\mathcal{Z}_i\right\|_2 \leq C$, for each $i=1,...,n$. Set $\mu_{\mathcal{Z}} = \frac{1}{n}\sum_{i=1}^n\mathcal{Z}_i$. Then, for any $\varepsilon>0$, we have

$$P\left(\left\|\mu_{\mathcal{Z}} - \mathbb{E}[\mu_{\mathcal{Z}}]\right\|_2 \geq \varepsilon\right) \leq 2r\exp\left(-\frac{n\varepsilon^2}{2C^2 r}\right).$$

Sketch of Theorem 2 Proof

Proof. Let g in $\mathcal{S}$. First, note that

$$\|\psi_r(g) - \widetilde{\psi}_r(g)\|_F^2 = \sum_{\sigma\in\Sigma_r}\|\psi_{r,\sigma}(g) - \widetilde{\psi}_{r,\sigma}(g)\|_2^2.$$

Using Boole's inequality

$$\begin{aligned}
&P(\bigcup_{g\in\mathcal{S}}\|\psi_r(g)-\widetilde{\psi}_r(g)\|_F \geq \varepsilon)\\
&\leq \sum_{g\in\mathcal{S}}P(\sum_{\sigma\in\Sigma_r}\|\psi_{r,\sigma}(g)-\widetilde{\psi}_{r,\sigma}(g)\|_2^2 \geq \varepsilon^2)\\
&\leq \sum_{g\in\mathcal{S}}\sum_{\sigma\in\Sigma_r}P(\|\psi_{r,\sigma}(g)-\widetilde{\psi}_{r,\sigma}(g)\|_2 \geq \frac{\varepsilon}{\sqrt{|\Sigma_r|}}).
\end{aligned}$$

For clarity of notation, we denote $p_\sigma^{in}, p_\sigma^{out}$ by p_σ^1, p_σ^2 and let P_σ^i be the set of σ-colored pairs of type i with $p_\sigma^i = |P_\sigma^i|$. We set $\gamma_g^i = \sum_\sigma p_\sigma^i$ and $\gamma_{\mathcal{M}} = \binom{V_{\mathcal{M}}}{2}$. Let $\mathcal{Z}_{r,\sigma}(g) = (z_1,z_2)^T$ be a 2-dimensional random variable such that

$$z_i = \alpha_i \mathbb{1}_{P_\sigma^i}, \quad (8)$$

where $\alpha_i = \gamma_g^i/\gamma_{\mathcal{M}}$ and $\mathbb{1}_{\mathcal{E}}$ is 1 if at round r of Algo. 3, we sample p such that p is in $\mathcal{E}$, otherwise 0.

Let $\mu_{\mathcal{Z}_{r,\sigma}}(g) := \frac{1}{|\Gamma|}\sum_{r=1}^{|\Gamma|}\mathcal{Z}_{r,\sigma}(g) = \widetilde{\psi}_{r,\sigma}(g)$. Since for each i, $\mathbb{E}[z_i] = \frac{\alpha_i p_\sigma^i}{\gamma_g^i}$, then $\mathbb{E}[\mu_{\mathcal{Z}_{r,\sigma}}(g)] = \psi_{r,\sigma}(g)$. Note that $\left\|\mathcal{Z}_{r,\sigma}(g)\right\|_2 \leq \sqrt{2}$. By Lemma 2, we have

$$P(\|\psi_{r,\sigma}(g)-\widetilde{\psi}_{r,\sigma}(g)\|_2 \geq \frac{\varepsilon}{\sqrt{|\Sigma_r|}}) \leq 4\exp\left(-\frac{\Gamma\varepsilon^2}{8|\Sigma_r|}\right).$$

Therefore, by setting

$$\Gamma := \Gamma_{\nu,\varepsilon,\theta}^{\mathcal{S}} = \frac{8|\Sigma_\theta|}{\varepsilon^2}\log\left(\frac{4|\mathcal{S}||\Sigma_\theta|}{1-\nu}\right), \quad (9)$$

where Σ_θ is an upper bound on the maximum number of different colors, we obtain

$$P(\bigcup_{g\in\mathcal{S}}\|\psi_r(g)-\widetilde{\psi}_r(g)\|_F \geq \varepsilon) \leq 1-\nu.$$

□

Sketch of Proposition 4 Proof

Proof. Let g and h be two activity graphs. Then

$$K(g,h) = \frac{1}{2n_g n_h\sigma^2}\sum_{v,u} f_1(v,u)f_2(g,h),,$$

where $f_1(v,u) = \|i_v - i_u\|^2$ and $f_2(g,h) = \exp\left(\frac{-\|c_g-c_h\|^2}{2\sigma^2}\right)$. Observe that f_1 is clearly positive definite, while the positive definiteness of f_2 is obtained from Corollary 3 in [36]. Since the sum and multiplication of positive definite kernels are still positive definite, we conclude that K is positive definite. Moreover, it is clearly symmetric. □

REFERENCES

[1] V. Arvind, F. Fuhlbrück, J. Köbler, and O. Verbitsky. On weisfeiler-leman invariance: Subgraph counts and related graph properties. *Journal of Computer and System Sciences*, 113:42–59, 2020.

[2] F. R. Bach. Graph kernels between point clouds. In *ICML 2008*.

[3] A. Berlinet and C. Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

[4] A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6(Sep):1579–1619, 2005.

[5] K. M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pages 8–pp. IEEE, 2005.

[6] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[7] H. Cai, V. W. Zheng, and K. C.-C. Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[8] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[9] S. S. Du, K. Hou, R. R. Salakhutdinov, B. Poczos, R. Wang, and K. Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *NIPS 2019*.

[10] A. Feragen, N. Kasenburg, J. Petersen, M. de Bruijne, and K. M. Borgwardt. Scalable kernels for graphs with continuous attributes. In *NIPS 2013*.

[11] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *ICML 2017*.

[12] M. Grohe. *Descriptive complexity, canonisation, and definable graph structure theory*, volume 47. Cambridge University Press, 2017.

[13] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[14] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *NIPS 2017*.

[15] W. L. Hamilton, R. Ying, and J. Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[16] C. R. Harshaw, R. A. Bridges, M. D. Iannacone, J. W. Reed, and J. R. Goodall. Graphprints: Towards a graph analytic method for network anomaly detection. In *Proceedings of the 11th Annual Cyber and Information Security Research Conference*, pages 1–4, 2016.

[17] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens. A taxonomy of network threats and the effect of current datasets on intrusion detection systems. *IEEE Access*, 2020.

[18] W. Hoeffding. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, pages 409–426. Springer, 1994.

[19] T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[20] T. Karagiannis, K. Papagiannaki, N. Taft, and M. Faloutsos. Profiling the end host. In *International Conference on Passive and Active Network Measurement*, pages 186–196. Springer, 2007.

[21] S. Khatuya, N. Ganguly, J. Basak, M. Bharde, and B. Mitra. Adele: Anomaly detection from event log empiricism. In *IEEE INFOCOM 2018*.

[22] N. M. Kriege, P.-L. Giscard, and R. Wilson. On valid optimal assignment kernels and applications to graph classification. *NIPS 2016*.

[23] N. M. Kriege, F. D. Johansson, and C. Morris. A survey on graph kernels. *Applied Network Science*, 5(1):1–42, 2020.

[24] Q. V. Le, T. Sarlós, and A. J. Smola. Fastfood: Approximate kernel expansions in loglinear time. *arXiv preprint arXiv:1408.3060*, 2014.

[25] B. Le Bars and A. Kalogeratos. A probabilistic framework to node-level anomaly detection in communication networks. In *IEEE INFOCOM 2019*.

[26] D. A. Levin and Y. Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

[27] C. Liaskos, V. Kotronis, and X. Dimitropoulos. A novel framework for modeling and mitigating distributed link flooding attacks. In *IEEE INFOCOM 2016*.

[28] C. magazine. Annual report: Cyberwarfare in the c-suite, 2021.

[29] H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman. Provably powerful graph networks. In *NIPS 2019*.

[30] C. Morris, K. Kersting, and P. Mutzel. Glocalized weisfeiler-lehman graph kernels: Global-local feature maps of graphs. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 327–336. IEEE, 2017.

[31] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019.

[32] K. Muandet, K. Fukumizu, B. Sriperumbudur, and B. Schölkopf. Kernel mean embedding of distributions: A review and beyond. *arXiv preprint arXiv:1605.09522*, 2016.

[33] G. Nikolentzos, G. Siglidis, and M. Vazirgiannis. Graph kernels: A survey. *Journal of Artificial Intelligence Research*, 72:943–1027, 2021.

[34] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro. Tesseract: Eliminating experimental bias in malware classification across space and time. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 729–746, 2019.

[35] P. J. Rousseeuw and K. V. Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999.

[36] I. J. Schoenberg. Metric spaces and positive definite functions. *Transactions of the American Mathematical Society*, 44(3):522–536, 1938.

[37] B. Scholkopf. The kernel trick for distances. In *NIPS 2001*.

[38] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[39] N. Shervashidze and K. M. Borgwardt. Fast subtree kernels on graphs. In *NIPS 2019*.

[40] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

[41] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, pages 488–495, 2009.

[42] G. Siglidis, G. Nikolentzos, S. Limnios, C. Giatsidis, K. Skianis, and M. Vazirgiannis. Grakel: A graph kernel library in python. *Journal of Machine Learning Research*, 21(54):1–5, 2020.

[43] A. Smola, A. Gretton, L. Song, and B. Schölkopf. A hilbert space embedding for distributions. In *International Conference on Algorithmic Learning Theory*, pages 13–31. Springer, 2007.

[44] J. Srinivas, A. K. Das, and N. Kumar. Government regulations in cyber security: Framework, standards and recommendations. *Future Generation Computer Systems*, 92:178–188, 2019.

[45] S. Sun and K. C. Das. On the second largest normalized laplacian eigenvalue of graphs. *Applied Mathematics and Computation*, 348:531–541, 2019.

[46] Z. Szabó and B. K. Sriperumbudur. Characteristic and universal tensor product kernels. *The Journal of Machine Learning Research*, 18(1):8724–8752, 2017.

[47] J.-P. Vert. The optimal assignment kernel is not positive definite. *arXiv preprint arXiv:0801.4061*, 2008.

[48] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. Graph kernels. *The Journal of Machine Learning Research*, 11:1201–1242, 2010.

[49] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[50] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[51] Y. Yao, L. Su, C. Zhang, Z. Lu, and B. Liu. Marrying graph kernel with deep neural network: A case study for network anomaly detection. In *International Conference on Computational Science*, pages 102–115. Springer, 2019.

[52] W. Ye, Z. Wang, R. Redberg, and A. Singh. Tree++: Truncated tree based graph kernels. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[53] S. Yi, Z. Qin, and Q. Li. Security and privacy issues of fog computing: A survey. In *International conference on wireless algorithms, systems, and applications*, pages 685–695. Springer, 2015.

[54] Z. Zhang, M. Wang, Y. Xiang, Y. Huang, and A. Nehorai. Retgk: Graph kernels based on return probabilities of random walks. In *NIPS 2018*.