entropy

MDPI

Article

(Hyper)graph Kernels over Simplicial Complexes

Alessio Martino * and Antonello Rizzi

Department of Information Engineering, Electronics and Telecommunications, University of Rome "La Sapienza", Via Eudossiana 18, 00184 Rome, Italy; antonello.rizzi@uniroma1.it

* Correspondence: alessio.martino@uniroma1.it; Tel.: +39-06-44585745

Received: 18 September 2020; Accepted: 12 October 2020; Published: 14 October 2020

Abstract: Graph kernels are one of the mainstream approaches when dealing with measuring similarity between graphs, especially for pattern recognition and machine learning tasks. In turn, graphs gained a lot of attention due to their modeling capabilities for several real-world phenomena ranging from bioinformatics to social network analysis. However, the attention has been recently moved towards hypergraphs, generalization of plain graphs where multi-way relations (other than pairwise relations) can be considered. In this paper, four (hyper)graph kernels are proposed and their efficiency and effectiveness are compared in a twofold fashion. First, by inferring the simplicial complexes on the top of underlying graphs and by performing a comparison among 18 benchmark datasets against state-of-the-art approaches; second, by facing a real-world case study (i.e., metabolic pathways classification) where input data are natively represented by hypergraphs. With this work, we aim at fostering the extension of graph kernels towards hypergraphs and, more in general, bridging the gap between structural pattern recognition and the domain of hypergraphs.

Keywords: hypergraphs; graph kernels; kernel methods; support vector machines; simplicial complexes; topological data analysis

1. Introduction

Graphs are powerful data structures able to capture semantic and topological information from data. For this reason, graphs are commonly used to model a plethora of real-world, possibly complex, systems [1]. Notable examples include biological systems and chemistry [2–13], social and collaboration networks [14], computer vision and image processing [15–18], natural language processing [19–22], and energy distribution networks [23].

The price to pay for their modeling capabilities relies on the computational complexity needed in order to compare two graphs [24], as graphs lie in non-metric spaces and thus lack algebraic structures by definition [1]. To this end, several techniques have been proposed in literature in order to quantify (dis)similarities between graphs. Herein, three mainstream approaches are sketched, along with their respective major techniques, referring the interested reader to reviews such as those in [1,25].

Graph matching techniques evaluate the similarity directly in the graphs domain, conversely to the other two following approaches. In this family, it is possible to find

- the isomorphism test (i.e., two graphs of the same size can be considered equal if they are isomorphic) or the subgraph isomorphism test (i.e., if two graphs have different sizes, it is possible to check whether the smaller graph is isomorphic to a "portion" of the bigger graph);

- partial matching, which overcomes the drawback of the two above methods by the evaluation of the maximum (or minimum) common subgraph or the maximal common subgraphs [26–29]. In fact, isomorphism (by definition) does not consider partial similarities: two graphs can be isomorphic, not isomorphic or contained one into the other; and
- inexact graph matching, which does not consider an "exact" match between graphs or their respective subgraphs. Rather, (dis)similarity between graphs is evaluated by means of a properly formulated cost function, the seminal example being the (graph) edit distance [30,31], which evaluates the minimum cost sequence of atomic operations (insertion, deletion and substitution) on nodes and edges in order to transform the two graphs into one another.

Building an *embedding space* consists in moving the pattern recognition problem from the graphs domain towards a possibly metric (Euclidean) space: instead of using the former, one can use the latter which can be equipped with algebraic structures. To this family belongs

- dissimilarity spaces [32], where each pattern is described by the pairwise dissimilarities with respect to all training data (or a properly-chosen subset [3,33,34]), and
- embedding via information granulation, where recurrent and meaningful entities called *information granules* (e.g., network motifs, graphlets, and the like) are extracted by the training data and each pattern is cast into an integer-valued vector (*symbolic histogram*) which counts the number of occurrences of each granule within the pattern itself [35–40].

In *kernel methods*, the feature space is implicitly defined by means of a properly defined positive definite kernel function which projects the (possibly structured) data into a (possibly infinite-dimensional) Hilbert space in which data points are more likely to be linearly separable [41–43], e.g., by a maximal margin classifier such as support vector machines [44].

The aim of this paper is the introduction of four hypergraph kernels defined over simplicial complexes. In fact, while plenty of kernels have been designed for working in the domain of (labeled) graphs, there is still lack of kernel methods for hypergraphs. More in general, this is true for structural pattern recognition techniques, with the work in [36] being the only paper whose aim is to define suitable embedding spaces for hypergraph classification purposes. The strengths of the proposed kernels are as follows: their evaluation is not iterative; if the underlying graph is available, they are parameter-free; and, for some of them, the evaluation can be easily vectorized. A potential drawback is that (in their current implementation, at least) they only work on fully node labeled graphs, where nodes belong to a categorical and finite set (e.g., strings). The proposed kernels are compared to state-of-the-art graph kernels both in terms of efficiency and effectiveness on 18 benchmark datasets for graph classification. Computational results show that they generally outperform state-of-the-art graph kernels in terms of effectiveness and they are robust with respect to the size of the dataset. Furthermore, their parameter-free peculiarity makes the training phase appealing despite some state-of-the-art graph kernels outperform the proposed kernels in terms of evaluation time.

The remainder of the paper is structured as follows. In Section 2, the current literature on graph kernels is reviewed. In Section 3, hypergraphs and simplicial complexes are introduced. In Section 4, the four proposed kernels are described. Section 5 addresses the effectiveness and efficiency of the proposed kernels by comparing against state-of-the-art graph kernels (Section 5.1) and by facing a real-world biological case study (Section 5.2) in which input data are natively described by hypergraphs and current graph kernels cannot be employed. The properties of the proposed kernels are discussed in Section 6 (positive semi-definiteness) and Section 7 (computational complexity). Finally, Section 8 draws some conclusions and future directions.

2. Related Works on Graph Kernels

Graph kernels can be divided into four big families [25]. *Model driven kernels*, for example, exploit generative models or transformative models. An example of generative model driven kernel is the Fisher kernel [45], which exploits hidden Markov models. Conversely, an example of transformative model driven kernel is the diffusion kernel [46], which exploits the heat equation in order to diffuse local information to neighbor nodes.

The largest family is composed by *syntax driven kernels*, which aim at analyzing trees, paths, cycles or subgraphs. Notable examples include random walk kernels [47,48], shortest path kernels [49], graphlet kernels [50], and the Weisfeiler–Lehman kernels [51,52]. The first three, respectively, take into consideration the number of random walks between two graphs via their product graph, the number of common shortest paths, and the number of common k-order graphlets. The fourth one, instead, computes the number of subtrees-walks shared between two graphs using the Weisfeiler–Lehman test of graph isomorphism.

Another state-of-the-art approach is given by *propagation kernels* [53], which are based on the spread of information across several graphs using early-stage distributions from propagation schemes (e.g., random walks) in order to model different types of node and edge labels. Such propagation schemes make propagation kernels suitable for dealing also with unlabeled and partially labeled graphs.

Finally, *deep graph kernels* [54] can be seen as an extension of the aforementioned syntax driven kernels, where a positive semi-definite matrix weights (encodes) the relationships between sub-structures (e.g., graphlets).

For more information about these four families and a more exhaustive list of graph kernels belonging to each family, the interest reader is referred to the very detailed survey [25].

Fewer works have been done by using *hypergraphs*: hypergraphs can be seen as generalization of "plain graphs" where *hyperedges* can connect two or more nodes. In other words, hypergraphs extend the modeling capabilities offered by graphs (Section 1) in cases where multi-way relations are of interest; indeed, graphs take into account only pairwise relations [55–59]. A straightforward example may regard a scientific collaboration network in which n authors co-authored a paper: if one has to model this scenario using a graph, then one might consider nodes as authors which are connected by $n(n-1)/2$ edges. This modeling, however, is ambiguous about whether the n authors co-authored a paper or each pair of authors co-authored a paper. Using hypergraphs, an hyperedge can link the n authors, with no ambiguities. A more biologically-oriented example include protein interaction networks, where nodes correspond to proteins and edges exist whether they interact. Again, this representation does not consider protein complexes [60].

To the best of our knowledge, the only attempt to bridge the gap between hypergraphs and kernel methods is described in [61], where the authors proposed edit distance-based hypergraphlet kernels for node and vertex classification and link prediction, e.g., in protein–protein interaction networks.

Conversely to all previous works, in this paper four hypergraph kernels based on simplicial complexes for classification problems are proposed, therefore we exploit the multi-scale organization of complex networks addressing whether this modeling can also suit the definition of efficient and/or effective kernel functions for hypergraph classification tasks and foster the extension of graph kernels towards hypergraphs.

3. A Primer on Simplicial Complexes

A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is defined by a finite set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Conversely, a hypergraph $\mathcal{HG} = \{\mathcal{V}, \mathcal{H}\}$ is identified by a finite set of vertices $\mathcal{V}$ and a set of hyperedges $\mathcal{H}$, namely, non-empty subsets of $\mathcal{V}$. The striking difference between graphs and hypergraphs lies in their respective sets $\mathcal{E}$ and $\mathcal{H}$: in fact, whereas $\mathcal{E}$ collects edges which encode pairwise relationships between nodes,

$\mathcal{H}$ collects hyperedges, whose cardinality can be greater than two. In order words, $\mathcal{H}$ is able to collect multi-way relationships among elements in $\mathcal{V}$.

Depending on the nature of the input data, building the corresponding hypergraph can be straightforward. Let us consider the example of the co-authorship network from Section 2. Given a paper p, with its respective authors list $a_1, \ldots, a_n$, it is possible to add to the hypergraph $a_1, \ldots, a_n$ as nodes and an hyperedge linking them. Furthermore, if authors $a_1, \ldots, a_n$ co-authored more than one paper together, the hyperedge can encode also this semantic information (i.e., the hyperedge is weighted with the number of papers that authors $a_1, \ldots, a_n$ co-authored together). An additional example (see later Section 5.2) may regard the hypergraph representation of a metabolic network. A metabolic network models the chain of chemical reactions occurring within the cell of a given living organism. It is well known from basic chemistry that chemical reactions can involve more than two molecules, therefore the hypergraph corresponding to a given metabolic network sees the molecules involved in any chemical reaction as vertices, with hyperedges that link together groups of molecules involved in the same chemical reaction.

Several representations of hypergraphs are available in the literature [59,62,63]. In the following, let us focus on one of the most commonly used one: simplicial complexes. Before diving into a formal definition of simplicial complex, it is worth defining the *simplex* of order (dimension) k (also, k-simplex) as the convex hull composed by $(k+1)$ points. For example, points, lines, triangles, and tetrahedrons are 0-dimensional, 1-dimensional, 2-dimensional, and 3-dimensional simplices, being composed by 1, 2, 3, and 4 points (vertices), respectively, but also higher-order analogs exist. Every non-empty subset of the $(k+1)$ vertices of a k-simplex is a *face* of the simplex: a face is itself a simplex. Therefore, a *simplicial complex* $\mathcal{S}$ can be formally defined as a finite collection (set) of simplices having the following two properties,

1. if $\sigma \in \mathcal{S}$, then every face of σ is also in $\mathcal{S}$, and
2. if $\sigma_1, \sigma_2 \in \mathcal{S}$, then $\sigma_1 \cap \sigma_2$ is a face of both σ_1 and σ_2.

A *subcomplex* of $\mathcal{S}$ is a subset of $\mathcal{S}$ which is also a simplicial complex: an important subcomplex is the k-skeleton, namely, a simplicial complex with at most simplices of order k. As the edge itself is a simplex, a graph can also be seen as a 1-skeleton (or 1-dimensional simplicial complex—where the *dimension* of a simplicial complex is defined as the maximum order among its simplices), being composed only by edges (1-dimensional simplices) and, eventually, nodes (0-dimensional simplices).

Several types of simplicial complexes exist and the choice mostly depends on their computational complexity and the cardinality of the starting data [64–66]. Simplicial complexes are widely used in the novel field of Topological Data Analysis in order to analyze a set of data by means of tools derived from topology. Therefore, conversely to the previous two examples (i.e., co-authorship network and metabolic network), in which the relationships between elements were perfectly described (i.e., well-known authors—paper and molecules—reaction information), usually data may come as a *point cloud*. A point cloud is defined as a finite set of points lying in some (possibly metric) space equipped with a notion of similarity. In this scenario, there are no a priori information about the connectivity among data points and several simplicial complexes can be employed in order to infer the simplicial structure and highlight the topological organization of the underlying data points. Notable examples include the Alpha complex and the Vietoris–Rips complex. The former relies on performing the Delaunay triangulation and then building the restricted Voronoi regions by intersecting (for each data point) an α-radius ball centered to it with its Voronoi region. The nerve of the set of all restricted regions is the Alpha complex at scale α. The latter is more graph-oriented and is defined as follows; a set of k points forms a $(k-1)$-dimensional simplex to be included in the Vietoris–Rips complex if all pairwise distances between points are less than or equal to a user-defined threshold ϵ. An efficient two-step procedure in order to compute the Vietoris–Rips complex has been proposed in [67]:

1. define the Vietoris–Rips neighborhood graph $\mathcal{G}_{\text{VR}}$ by scoring edges between any two vertices if their distance is less than or equal to ϵ and then
2. evaluate the flag complex of $\mathcal{G}_{\text{VR}}$.

In turn, the flag complex (also, clique complex) is the simplicial complex whose simplices are the maximal cliques in the underlying graph (1-skeleton). In other words, the clique complex is the topological space in which each k-vertex clique is represented by a $(k-1)$-simplex. Despite the minimalistic definition, it is straightforward to demonstrate that the flag complex is a valid simplicial complex: every maximal clique is a clique that cannot be made any larger and since every subset of a clique is also a clique (just like a face of a simplex is a simplex itself), then the closure under inclusion (i.e., closure under taking subsets) is automatically satisfied.

We stress the flag complex because of the following observation: simplicial complexes like the Vietoris–Rips complex strictly depend on a scale parameter (i.e., ϵ) which somewhat determines the resolution of the simplicial complex (i.e., the resolution at which data have been observed). Usually, by means of techniques such as persistence one finds a suitable value for ϵ [5]. However, if the underlying graph is already available, there is no need to build the neighborhood graph and find a suitable value for ϵ in order to build the Vietoris–Rips complex. Therefore, one can directly use the flag complex as it encodes the same information as the underlying graph, but it additionally completes a topological object with its fullest possible simplicial structure, being it a canonical polyadic extension of existing networks (1-skeletons) [55].

4. Proposed Hypergraph Kernels

From Section 3 it is clear that a simplex can be identified by considering the set of corresponding vertices. For example, when each vertex is just identified by a unique progressive number, it is impossible to match two simplicial complexes (e.g., determine how many/which simplices they have in common) and node labels play an important role: a simplex can be represented by the set of labels belonging to the corresponding vertices. Thanks to the node labels, the matching procedure is straightforward and can be done in an exact manner: two simplices (possibly belonging to different simplicial complexes) are equal if they have the same order and they share the same node labels. This observation is at the basis of all of the four kernels herein described.

4.1. The Histogram Cosine Kernel

The Histogram Cosine Kernel (HCK) is loosely based on the symbolic histogram technique. According to the latter, a given structured pattern (be it a graph, a sequence, and so on) can be explicitly embedded towards a Euclidean space by counting the number of recurrent and/or meaningful substructures (subgraphs, subsequences, and so on) properly extracted from the training data and classification can be performed in the Euclidean space spanned by these histogram vectors (for further details, the interested reader is referred to works such as in [1,35,39,68] and references therein). HCK is also inspired by the Graphlet Sampling kernel [50] and the Vertex Histogram kernel [69]. As the kernel evaluation relies on pairwise matching between patterns, the pivotal substructures to be considered for the symbolic histograms evaluation are the unique simplices between the two simplicial complexes to be matched. Specifically, let $\mathcal{S}_i$ and $\mathcal{S}_j$ be two simplicial complexes and let $\mathcal{A} = \mathcal{S}_i \cup \mathcal{S}_j$ be the set of unique simplices belonging to either $\mathcal{S}_i$ or $\mathcal{S}_j$. Then, a given simplicial complex, say $\mathcal{S}$, can be cast into a vector, say $\mathbf{f} \in \mathbb{N}^{|\mathcal{A}|}$, where

$$\mathbf{f}_k = \text{count}(\mathcal{A}_k, \mathcal{S}), \qquad \forall k = 1, \ldots, |\mathcal{A}| \tag{1}$$

where count(a, b) is a function that counts the number of occurrences of a in b and $\mathcal{A}_k$ denotes the k-th element in $\mathcal{A}$.

Therefore, thanks to Equation (1), $\mathcal{S}_i \to \mathbf{f}^{(i)}$ and $\mathcal{S}_j \to \mathbf{f}^{(j)}$, and HCK has the form

$$K_{\text{HC}}(\mathcal{S}_i, \mathcal{S}_j) = \langle \mathbf{f}^{(i)}, \mathbf{f}^{(j)} \rangle, \tag{2}$$

where $\langle \cdot, \cdot \rangle$ denotes the plain dot product. However, HCK defined as in Equation (2) can be skewed by the different number of simplices within each simplicial complex. To this end, the following normalization is adopted,

$$K_{\text{HC}}(\mathcal{S}_i, \mathcal{S}_j) = \frac{\langle \mathbf{f}^{(i)}, \mathbf{f}^{(j)} \rangle}{\sqrt{\langle \mathbf{f}^{(i)}, \mathbf{f}^{(i)} \rangle \cdot \langle \mathbf{f}^{(j)}, \mathbf{f}^{(j)} \rangle}}. \tag{3}$$

Thanks to the product property of square roots, the latter can be written as

$$K_{\text{HC}}(\mathcal{S}_i, \mathcal{S}_j) = \frac{\langle \mathbf{f}^{(i)}, \mathbf{f}^{(j)} \rangle}{\sqrt{\langle \mathbf{f}^{(i)}, \mathbf{f}^{(i)} \rangle} \cdot \sqrt{\langle \mathbf{f}^{(j)}, \mathbf{f}^{(j)} \rangle}}, \tag{4}$$

therefore collapsing into the cosine similarity between the vector representation (i.e., histogram) of the two hypergraphs, hence the name of the kernel.

4.2. The Weighted Jaccard Kernel

From Section 3 it is clear that simplicial complexes are sets. A straightforward idea is to use a similarity measure between sets as the core of the kernel. The seminal example is the Jaccard similarity, defined as the ratio between the size of the intersection divided by the size of the union between two sets. Since a simplex can be identified by the set of labels associated to each node, different simplices within the same simplicial complex can share the same node labels. This means that simplicial complexes are de facto multisets and the Jaccard similarity as previously defined loses its effectiveness and needs to be extended towards multisets. If $\mathcal{S}_i$ and $\mathcal{S}_j$ are two simplicial complexes to be matched, then their respective multiset representations read as

$$\mathcal{S}_i = \left\{ s_1^{\mu_i(s_1)}, \ldots, s_{n_i}^{\mu_i(s_{n_i})} \right\} \qquad \mathcal{S}_j = \left\{ s_1^{\mu_j(s_1)}, \ldots, s_{n_j}^{\mu_j(s_{n_j})} \right\}$$

where n_i and n_j denote the number of simplices in $\mathcal{S}_i$ and $\mathcal{S}_j$, respectively, and $\mu_i : \mathcal{S}_i \to \mathbb{N}_{\geq 0}$, $\mu_j : \mathcal{S}_j \to \mathbb{N}_{\geq 0}$ are the two multiplicity functions for $\mathcal{S}_i$ and $\mathcal{S}_j$. The latter is a function from the multiset (i.e., simplicial complex) towards the non-negative integers that returns the multiplicity (i.e., the number of occurrences) of an element (i.e., simplex s) within the multiset, with the caveat that if an element does not exist in the multiset, its multiplicity is 0. Let us now define the universe $\mathcal{U}$ from which the support of the two simplicial complexes are drawn

$$\mathcal{U} = \underbrace{\{s \in \mathcal{S}_i | \mu_i(s) > 0\}}_{\text{support of } \mathcal{S}_i} \cup \underbrace{\{s \in \mathcal{S}_j | \mu_j(s) > 0\}}_{\text{support of } \mathcal{S}_j} \tag{5}$$

Alike to the Jaccard similarity, the Weighted Jaccard Kernel is defined as the ratio between intersection and union of the two simplicial complexes that, being multisets, generalizes as

$$K_{\text{WJ}}(\mathcal{S}_i, \mathcal{S}_j) = \frac{\min(\mu_i(s), \mu_j(s))}{\max(\mu_i(s), \mu_j(s))} \quad \forall s \in \mathcal{U}. \tag{6}$$

4.3. The Edit Kernel

The Edit Kernel (EK) aims at measuring the similarity between two simplicial complexes according to the number of simplices to be inserted, removed and/or substituted in order to transform the two simplicial complexes into one another. Let $e(\mathcal{S}_i, \mathcal{S}_j)$ be an edit distance (Levenshtein-like) with unitary weights. The first step is to convert the distance measure into a (possibly normalized) similarity measure. In [70], it has been demonstrated that

$$\bar{e}(\mathcal{S}_i, \mathcal{S}_j) = \frac{2 \cdot e(\mathcal{S}_i, \mathcal{S}_j)}{|\mathcal{S}_i| + |\mathcal{S}_j| + e(\mathcal{S}_i, \mathcal{S}_j)} \tag{7}$$

is a normalized edit distance in range $[0, 1]$ which satisfies the properties of a metric. Furthermore, in [71], it has been shown that if d is a normalized metric distance, then $1 - d$ is a normalized metric similarity. Therefore, EK has the form

$$K_{\mathrm{E}}(\mathcal{S}_i, \mathcal{S}_j) = 1 - \bar{e}(\mathcal{S}_i, \mathcal{S}_j). \tag{8}$$

It is noteworthy that edit distances/similarities are sensitive to the order of the input sequences. In order to ease the matching procedure, within each simplicial complex, simplices are jointly sorted both lexicographically (i.e., according to the node labels) and according to their orders.

4.4. The Stratified Edit Kernel

The Stratified Edit Kernel (SEK) takes into account the following issue with EK: the latter can be skewed if the two simplicial complexes have a high variety of simplices per order. Let $\mathcal{K}$ be the set of different orders amongst simplices in the two simplicial complexes to be matched, then the SEK is defined as

$$K_{\mathrm{SE}}(\mathcal{S}_i, \mathcal{S}_j) = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} 1 - \bar{e}(\mathcal{S}_i^{(k)}, \mathcal{S}_j^{(k)}) \tag{9}$$

where $\mathcal{S}^{(k)}$ denotes the subset of k-simplices in the simplicial complex $\mathcal{S}$. The stratification allows to treat independently subsets of simplices having the same order; nonetheless, simplices are sorted lexicographically within each subset.

5. Tests and Results

5.1. On Benchmark Datasets

In order to investigate both the effectiveness and the efficiency of the proposed kernels, 18 datasets for graph classification have been considered. Details on the considered datasets (e.g., average number of nodes, average number of edges, number of samples, nodes and edges attributes), along with download links, can be found in [72]. No preprocessing has been performed on such data.

The four proposed kernels (HCK, WJK, EK and SEK) have been benchmarked against nine state-of-the-art graph kernels, listed below along with their respective parameters (and admissible values) to be tuned:

- Graphlet Sampling (GS) [50]. Parameters: the dimension of the graphlets $k \in \{3, 4, 5\}$
- Neighborhood Hash (NH) [73]. Parameters: number of hashes $R \in \{2, 3, 4, 5\}$ and the byte size of the hashes 2^b where $b \in \{2, 3, 4\}$. Other settings: simple hash type
- ODD-Sth (OS) [74]. Parameters: maximum single DAG height $h \in \{1, 2, \ldots, 8\}$

- Propagation Kernel (PK) [53]. Parameters: number of iterations $t \in \{1, 2, \ldots, 10\}$ and LSH width $w \in [10^{-8}, 10^{-1}]$. Other settings: total variation as distance metric for LSH, base kernel set as the linear kernel
- Pyramid Match (PM) [75]. Parameters: pyramid histogram level $L \in \{2, 3, 4, 5\}$ and hypercube dimension $d \in \{2, \ldots, 10\}$.
- Random Walk (RW) [76]. Parameters: decay factor $\lambda \in (0, 0.5)$
- SVM Theta (ST) [77]. Parameters: maximum size of the vertex set of sampled subgraphs $M \in \{2, 3, \ldots, 10\}$ (minimum size $m = 2$) and number of samples $s \in \{20, 21, \ldots, 100\}$. Other settings: product metric between theta numbers
- Weisfeiler–Lehman Subtree Kernel (WL) [52]. Parameters: number of iterations $t \in \{1, 2, \ldots, 10\}$.
- Weisfeiler–Lehman Shortest Path Kernel (WLSP) [49,52]. Parameters: number of iterations $t \in \{1, 2, \ldots, 10\}$. Other settings: shortest path matrices evaluated thanks to the Dijkstra algorithm.

As discussed in Section 3, the proposed four kernels are parameter-free (the flag complex can easily be evaluated using the Bron–Kerbosch algorithm [78,79]), whereas the same is not true for the nine competitors. A ν-SVM [80] has been used in order to perform classification. The regularization term $\nu \in (0, 1]$, along with kernel parameters (if any) are learned via 10-fold cross-validation, driven by random-search [81]. For all datasets except FIRSTMM_DB, due to heavy unbalancing between number of splits and labels' distribution, the cross-validation procedure has been stratified according to the ground-truth labels.

The four proposed kernels are implemented in Python, using NumPy [82] and NetworkX [83] as external dependencies, whereas the nine competitors belong to the GraKel library [84]. Finally, the ν-SVM implementation from the Scikit-Learn library [85] has been used for classification. The hardware setup includes a Linux CentOS 7 workstation equipped with two hyperthreaded 14-core Intel® Xeon® Gold 5120 CPU @ 2.20 GHz with 192 GB of RAM. Multithreaded parallelism has been exploited in order to speed up the optimization phase, where each thread processes the dataset by using a different set of parameters.

One final facet to stress before diving into the computational results lies on the graphs vs. hypergraphs duality: indeed, the datasets used for experiments are graph datasets (not hypergraphs), yet the proposed kernels are hypergraph kernels. The possibility to infer the simplicial structure from the underlying graph (e.g., thanks to the flag complex) has a crucial role in this regard, thanks to which one can infer hypergraphs starting from graphs, by adding additional information due to the definition of new possible n-ary relations [55,58]. This makes the four proposed kernels comparable with graph kernels (i.e., they both start from the same graphs). Notwithstanding that, it is safe to say that the four proposed kernels are indeed hypergraph kernels since they work on the top of the simplicial complexes that, in this case, have been obtained from the underlying 1-skeletons.

Figure 1 shows the results in terms of 10-fold cross-validation accuracy, a common practice in related research papers on graph kernels. Due to intrinsic randomness in both cross-validation and random search, each experiment has been repeated five times and the average value is shown. From Figure 1 it is possible to see that the four proposed kernels are competitive against the nine competitors. Furthermore, they are also very appealing for large datasets such as DD as no out-of-memory errors have been triggered and all executions terminated within the 24 h deadline (for details, see caption of Figure 1).

	HCK	WJK	EK	SEK	GS	NH	OS	PK	PM	RW	ST	WL	WL-SP
AIDS	89.5	99.5	99.5	99.6	98.8	99.5	99.6	99.6	99.7	99.4	99.3	99.6	99.6
BZR	81.5	83.5	84.2	84.2	72.1	82.0	83.1	81.8	83.0	80.9	65.1	83.6	83.2
COX2	83.4	80.3	79.5	79.2	59.8	77.2	79.9	77.6	78.5	×	52.6	79.4	75.8
DD	78.1	78.9	79.8	79.8	76.1	74.6	†	77.3	79.5	†	73.9	79.2	†
DHFR	69.1	64.6	66.3	66.3	67.4	67.7	71.6	71.1	68.7	×	61.0	71.7	71.4
ENZYMES	27.3	33.2	30.7	32.0	27.1	30.6	31.1	27.8	31.7	×	19.0	35.0	39.8
FIRSTMM_DB	37.2	22.2	23.5	15.2	20.0	18.7	×	24.5	22.0	×	0.8	27.0	†
MSRC_21	92.6	93.2	90.5	91.9	29.6	92.3	92.2	91.6	92.8	28.0	13.0	91.8	92.3
MSRC_9	89.5	92.0	89.5	82.1	18.3	90.8	†	90.3	91.1	×	6.3	88.6	†
MUTAG	79.6	90.9	86.1	86.4	88.7	85.5	89.0	79.9	88.6	88.0	82.9	87.5	86.2
Mutagenicity	73.3	82.0	75.4	75.3	66.6	84.6	†	80.7	76.3	×	55.9	84.7	†
NCI1	59.7	70.7	63.7	63.7	64.0	79.8	†	77.8	70.6	×	59.1	81.7	†
NCI109	57.9	71.5	65.3	65.7	65.8	78.9	†	75.6	70.0	×	61.0	81.8	†
PROTEINS	73.5	75.1	75.3	75.7	72.2	75.6	74.4	75.0	75.2	×	72.0	75.4	†
PTC_FM	64.2	64.2	63.3	63.3	58.0	61.4	62.5	59.4	63.3	60.1	58.8	65.3	64.5
PTC_FR	71.4	67.8	67.5	67.8	65.5	68.2	69.1	66.8	67.1	65.2	57.4	69.7	68.2
PTC_MM	70.8	66.1	65.2	66.3	61.9	65.6	68.6	63.6	65.7	61.5	54.4	69.0	69.2
PTC_MR	61.5	62.7	61.8	60.0	58.6	64.5	63.0	57.3	61.2	58.8	60.6	63.9	64.0

Figure 1. Average accuracy on the test set. The color scale has been normalized row-wise (i.e., for each dataset) from yellow (lower values) towards red (higher values, preferred). The times sign (×) indicates that the experiment has been aborted after passing a 24-h deadline. The dagger (†) indicates that the experiment went out-of-memory.

Conversely, Figure 2 shows the running times (in seconds) for evaluating the kernel matrix over the entire dataset. Wall-clock times refer to a single-threaded evaluation, no parallelism has been exploited. As the execution time is not deterministic due to spurious processes running in background, running times have been averaged across five evaluations of the kernel matrices. For the nine competitors, the best parameters returned by cross-validation have been used. From Figure 2 it is possible to see that RW is by far the slowest kernels to compute, followed by EK and SEK. The latter two kernels lack any vectorized statements, whereas the same is not true for HCK and WJK whose computational burden is competitive with current approaches. Notwithstanding the running times as such, it is worth stressing that the nine graph kernel competitors shall be evaluated several times in order to find a suitable set of parameters, whereas HCK, WJK, EK, and SEK can be evaluated only once.

	HCK	WJK	EK	SEK	GS	NH	OS	PK	PM	RW	ST	WL	WL-SP
AIDS	0.9	2.0	51.1	76.2	9.8	48.4	5.3	31.1	102.8	877.8	10.3	1.1	45.4
BZR	0.3	0.3	10.2	11.3	6.0	3.1	1.8	2.6	4.3	65.3	1.3	0.2	18.1
COX2	0.4	0.4	17.4	18.8	5.3	3.4	0.8	1.5	5.8	×	1.5	0.2	70.1
DD	14.5	150.8	6300.0	2170.7	3690.4	286.6	†	84.0	185.1	†	73.6	17.5	†
DHFR	0.7	0.7	48.8	52.4	26.3	15.1	23.6	25.4	18.0	×	3.1	0.7	69.7
ENZYMES	0.5	0.6	19.1	11.5	88.4	9.0	5.0	13.3	11.2	×	2.2	0.5	45.0
FIRSTMM_DB	3.2	3.2	388.1	189.7	207.3	2.9	×	1.8	12.9	×	41.7	1.9	†
MSRC_21	0.3	0.4	6.4	5.7	160.1	1.5	2.6	1.4	2.2	17.1	0.9	0.2	17.7
MSRC_9	1.6	2.3	164.1	135.8	670.4	16.4	†	13.5	14.3	×	2.8	1.3	†
MUTAG	0.1	0.1	0.6	0.8	1.0	0.7	0.2	1.0	1.4	15.6	0.5	0.1	3.9
Mutagenicity	3.1	5.7	756.6	876.5	111.2	364.2	†	540.6	648.1	×	31.1	3.7	†
NCI1	2.9	5.5	669.1	780.6	72.2	402.9	†	599.1	608.5	×	29.0	7.1	†
NCI109	2.9	5.6	669.3	779.9	73.0	312.0	†	309.4	655.1	×	29.1	4.8	†
PROTEINS	1.2	1.4	95.1	52.0	18.8	40.3	2.0	37.2	37.4	×	7.7	1.8	†
PTC_FM	0.1	0.1	1.3	2.1	2.4	1.6	2.2	1.8	2.8	57.8	0.9	0.2	4.0
PTC_FR	0.1	0.1	1.4	2.1	1.5	1.6	1.3	1.0	3.5	51.6	1.1	0.1	4.7
PTC_MM	0.1	0.1	1.2	1.9	1.4	1.5	0.6	1.5	4.3	50.6	0.8	0.2	8.8
PTC_MR	0.1	0.1	1.3	2.0	1.4	1.3	1.1	1.3	4.3	34.7	0.8	0.2	3.5

Figure 2. Average running times (in seconds) for evaluating the kernel matrix on the entire dataset. The color scale has been normalized row-wise (i.e., for each dataset) from yellow (lower values, preferred) towards red (higher values). The times sign (×) indicates that the experiment has been aborted after passing a 24-h deadline. The dagger (†) indicates that the experiment went out-of-memory. For the four proposed kernels, running times also include the simplicial complexes evaluation starting from the underlying graphs.

5.2. A Biological Case Study: Analysis of Metabolic Pathways

In Section 5.1, a suite of 18 benchmark datasets for graph classification has been used in order to assess the performances of the proposed four kernels. Tests have been carried out against current approaches by inferring the simplicial complex from the underlying 1-skeletons in order to make them comparable. Here we present a biological case study in which entities are natively represented by hypergraphs and regards the classification of metabolic pathways. Metabolic pathways represent the chain of chemical reactions occurring in a cell and allow a straightforward network formalism [39,86,87]: vertices correspond to metabolites (product/substrate of a chemical reaction) and edges are scored whether there exists a chemical reaction transforming the two nodes into one another. Despite the network formalism has been extensively used in order to analyze metabolic pathways, some information is necessarily lost in the network formalization [88,89]. In fact, metabolic reactions often involve more than two reactants (e.g., $A + B \rightarrow C + D$) and can be more conveniently represented by an hyperedge.

From the KEGG database, the metabolic pathways of 5299 organisms have been collected using its API in KGML format: KGML is the KEGG markup-like format in which each reaction in a given metabolic network is returned as a list of substrates and list of products. Therefore, the KEGG output can be interpreted as an hyperedge list, starting from which the native hypergraph modeling is straightforward.

From the initial set of 5299 organisms, four classification problems are investigated by following the Linnaeus' taxonomy at different scales:

- Problem 1 sees the entire set of 5299 organisms divided in two classes, according to their cellular architecture: Eukaryotes and Prokaryotes
- Problem 2 sees the entire set of 5299 organism divided in six classes, depending on their kingdom: Animals, Archaea, Bacteria, Fungi, Plants, and Protists
- Problem 3 sees a subset of 143 organisms (animals) divided in five classes, depending on their (Linnaeus') class: Birds, Fishes, Insects, Mammals, and Reptiles
- Problem 4 sees a subset of 1456 organisms (bacteria) divided in seventeen classes, depending on their (Linnaeus') class: Bacillus, Bifidobacterium, Burkholderia, Campylobacter, Chlamydia, Clostridium, Corynebacterium, Escherichia, Helicobacter, Lactobacillus, Mycobacterium, Mycoplasma, Pseudomonas, Salmonella, Staphylococcus, Streptococcus, and Streptomyces.

For additional information about the dataset, the interested reader is referred to the work in [39]. For these tests, a more thoughtful modeling has been employed in order also to address the generalization capabilities of the proposed kernels: for each of the four problems, the available patterns have been split in training set (50%), validation set (25%), and test set (25%) in a stratified manner in order to preserve labels' distribution across the three splits. The best parameter $\nu^\star$ is the one that maximizes the accuracy of the ν-SVM on the validation set after being trained on the training set. The best resulting model is finally evaluated on the test set. In order to take into account the intrinsic randomness in data splitting and hyperparameter tuning, for each problem, 10 experiments have been performed and results in terms of accuracy, specificity and sensitivity are shown in Tables 1–3, respectively. Clearly, the four proposed kernels show remarkable performances on the test set, regardless of the "resolution" at which the Linnaean taxonomy is investigated. Further discussion about the biological significance of being able to efficiently discriminate organisms according to their metabolic wiring can be found in [39,87].

Table 1. Accuracy on the test set (average $\pm$ standard deviation) for the four metabolic pathways classification problems.

Kernel	Problem 1	Problem 2	Problem 3	Problem 4
HCK	99.98 ± 0.05	99.95 ± 0.05	98.89 ± 1.84	99.89 ± 0.18
WJK	100.0 ± 0.0	99.91 ± 0.07	98.61 ± 1.86	99.89 ± 0.25
EK	99.93 ± 0.14	99.86 ± 0.09	97.78 ± 1.67	99.73 ± 0.27
SEK	99.98 ± 0.03	99.87 ± 0.13	97.78 ± 2.08	99.51 ± 0.32

Table 2. Specificity on the test set (average $\pm$ standard deviation) for the four metabolic pathways classification problems.

Kernel	Problem 1	Problem 2	Problem 3	Problem 4
HCK	99.99 ± 0.03	99.99 ± 0.01	99.74 ± 0.41	99.99 ± 0.01
WJK	100.0 ± 0.0	99.98 ± 0.01	99.69 ± 0.42	99.99 ± 0.02
EK	99.96 ± 0.07	99.98 ± 0.02	99.49 ± 0.38	99.98 ± 0.02
SEK	99.99 ± 0.02	99.98 ± 0.02	99.52 ± 0.44	99.97 ± 0.02

Table 3. Sensitivity on the test set (average ± standard deviation) for the four metabolic pathways classification problems.

Kernel	Problem 1	Problem 2	Problem 3	Problem 4
HCK	99.99 ± 0.03	99.82 ± 0.28	99.4 ± 1.28	99.91 ± 0.15
WJK	100.0 ± 0.0	99.67 ± 0.51	97.2 ± 3.45	99.89 ± 0.27
EK	99.96 ± 0.07	99.59 ± 1.06	95.72 ± 3.27	99.75 ± 0.24
SEK	99.99 ± 0.02	99.37 ± 0.82	96.36 ± 4.01	99.47 ± 0.34

6. On the Positive Definiteness of the Proposed Kernels

It is well known that a proper kernel function must satisfy the Mercer's condition on positive (semi-)definiteness [90]. Herein, the positive (semi-)definiteness of the four proposed kernels is addressed. To this end, let us anticipate three theorems regarding well-known properties of positive (semi-)definite matrices.

Theorem 1. *Let* $\mathbf{U}$ *and* $\mathbf{V}$ *be two positive semidefinite matrices, then their Hadamard product* $\mathbf{U} \odot \mathbf{V}$ *(also known as element-wise product or point-wise product) is positive semidefinite. Furthermore, if* $\mathbf{U}$ *and* $\mathbf{V}$ *are positive definite, then* $\mathbf{U} \odot \mathbf{V}$ *is positive definite.*

Proof of Theorem 1. Given in [91]. See also in [42,92]. □

Theorem 2. *Let* $\mathbf{U}_1, \ldots, \mathbf{U}_m$ *be m positive semidefinite matrices and let* $a_1, \ldots, a_m$ *be m non-negative scalars. Then, the matrix resulting from their linear combination* $\sum_{i=1}^{k} a_i \mathbf{U}_i$ *is also positive semidefinite. Furthermore, if there exists a* $j \in \{1, \ldots, m\}$ *such that* $a_j > 0$ *and* $\mathbf{U}_j$ *is positive definite, then* $\sum_{i=1}^{k} a_i \mathbf{U}_i$ *is positive definite.*

Proof of Theorem 2. Given in [91]. See also in [42]. □

Theorem 3. *Let* $\mathbf{U}$ *be a symmetric matrix (or Hermitian, in the complex case), then* $\mathbf{U}$ *is positive definite if all of its eigenvalues are positive. Furthermore,* $\mathbf{U}$ *is positive semi-definite if all of its eigenvalues are non-negative.*

Proof of Theorem 3. Given in [91]. □

6.1. The Histogram Cosine Kernel

The core of the HCK is the pairwise evaluation of the cosine similarity between histograms. Therefore, showing that the cosine similarity is positive semi-definite suffices in order to show that HCK is a valid kernel. To this end, we give two equivalent proofs that exploit different properties of positive semi-definite matrices.

The first proof sees the cosine pairwise similarity matrix as written as the Hadamard product between two matrices, say $\mathbf{N}$ and $\mathbf{D}$, respectively, defined as

$$\mathbf{N}_{i,j} = \langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle \tag{10}$$

$$\mathbf{D}_{i,j} = \frac{1}{\sqrt{\langle \mathbf{x}^{(i)}, \mathbf{x}^{(i)} \rangle} \cdot \sqrt{\langle \mathbf{x}^{(j)}, \mathbf{x}^{(j)} \rangle}} \tag{11}$$

The former is the Gram matrix between histograms, therefore it trivially satisfies Mercer's condition [91]. The latter has been shown to be positive semi-definite in [92]. Therefore, thanks to Theorem 1, HCK is a valid kernel.

The second proof involves a matrix, say $\mathbf{X}$, with patterns (i.e., histograms) organized as rows, that is

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} \\ \vdots \\ \mathbf{x}^{(N)} \end{bmatrix} \tag{12}$$

Let us now normalize each row according to the ℓ_2 norm of the row itself, that is

$$\overline{\mathbf{X}} = \begin{bmatrix} \mathbf{x}^{(1)}/\|\mathbf{x}^{(1)}\| \\ \vdots \\ \mathbf{x}^{(N)}/\|\mathbf{x}^{(N)}\| \end{bmatrix} \tag{13}$$

The cosine similarity matrix can dually be defined as $\overline{\mathbf{X}} \cdot \overline{\mathbf{X}}^T$, hence as the Gram matrix of $\overline{\mathbf{X}}$, which trivially satisfies Mercer's condition.

It is worth remarking that the space spanned by the symbolic histograms (cf. Equation (1)) can be equipped with any basic kernel for vector data: the choice of cosine similarity (i.e., normalized dot product) rather than, for example, polynomial or radial basis function stems from the work in [50,69].

6.2. The Weighted Jaccard Kernel

In order to show that WJK is a valid kernel, a vectorial representation of the kernel is analyzed. This also provides an efficient WJK evaluation procedure. It is well known that sets can be represented by binary vectors scoring 1 in position i if the i-th element exists in the set, and 0 otherwise. Similarly, multisets can be represented by integer-valued vectors which see in position i the value of the multiplicity function $\mu(\cdot)$ for the i-th element: in other words, two simplicial complexes (say $\mathcal{S}_i$ and $\mathcal{S}_j$) can be transformed into vectorial collections (say $\mathbf{f}^{(i)}$ and $\mathbf{f}^{(j)}$) by following Equation (1). Under this light, the WJK (cf. Equation (6)) can be rewritten as

$$K_{\mathrm{WJ}}(\mathcal{S}_i, \mathcal{S}_j) = J_W(\mathbf{f}^{(i)}, \mathbf{f}^{(j)}), \tag{14}$$

where

$$J_W(\mathbf{a}, \mathbf{b}) = \frac{\sum_{i=1}^{n} \min(\mathbf{a}_i, \mathbf{b}_i)}{\sum_{i=1}^{n} \max(\mathbf{a}_i, \mathbf{b}_i)}, \quad \text{for } \mathbf{a}, \mathbf{b} \in \mathbb{R}^n \tag{15}$$

is known as weighted Jaccard similarity (or Ružička index). In order to begin with the proof, it is worth mentioning the following theorem.

Theorem 4. *Let $K(a, b)$ be a kernel function satisfying Mercer's condition, then the following kernel is still valid,*

$$\hat{K}(a, b) = \frac{K(a, b)}{K(a, a) + K(b, b) - K(a, b)} \tag{16}$$

Proof of Theorem 4. Given in [93]. □

Equation (15) can be rewritten as follows,

$$J_W(\mathbf{a}, \mathbf{b}) = \frac{\sum_{i=1}^{n} \min(\mathbf{a}_i, \mathbf{b}_i)}{\sum_{i=1}^{n} \mathbf{a}_i + \sum_{i=1}^{n} \mathbf{b}_i - \sum_{i=1}^{n} \min(\mathbf{a}_i, \mathbf{b}_i)}, \tag{17}$$

and, by considering Equations (16) and (17), it can be rewritten as

$$J_W(\mathbf{a},\mathbf{b}) = \frac{\sum_{i=1}^{n}\min(\mathbf{a}_i,\mathbf{b}_i)}{\sum_{i=1}^{n}\min(\mathbf{a}_i,\mathbf{a}_i)+\sum_{i=1}^{n}\min(\mathbf{b}_i,\mathbf{b}_i)-\sum_{i=1}^{n}\min(\mathbf{a}_i,\mathbf{b}_i)}. \tag{18}$$

By comparing Equations (16) and (18) and by Theorem 4, it is clear that WJK is a valid kernel if $K(\mathbf{a},\mathbf{b}) = \sum_{i=1}^{n}\min(\mathbf{a}_i,\mathbf{b}_i)$ is a valid kernel as well. In order to demonstrate the latter, it is worth mentioning the following lemma.

Lemma 1. *Let $\{a_1,\ldots,a_m\}$ be a finite set of real-valued numbers, then the matrix $\mathbf{M} \in \mathbb{R}^{m\times m}$ defined as*

$$\mathbf{M}_{i,j} = \min(a_i,a_j) \tag{19}$$

is a valid kernel matrix.

Proof of Lemma 1. Given in [92]. □

As WJK deals with real-valued vectors in $\mathbb{R}^n$ and not with real-valued scalars as in the previous lemma, we shall extend the lemma to the former case. To this end, it is possible to build a series of n matrices, where each matrix (of the form as in Equation (19)) considers the pairwise minimum along a given dimension, say $i = 1,\ldots,n$, of the considered vectors and then perform the element-wise sum of such matrices:

$$K(\mathbf{a},\mathbf{b}) = \mathbf{M}^{(1)} + \ldots + \mathbf{M}^{(i)} + \ldots + \mathbf{M}^{(n)} \tag{20}$$

Finally, Equation (20) can easily be shown to be a valid kernel matrix thanks to Theorem 2.

6.3. The Edit Kernel and Stratified Edit Kernel

Edit distances (similarities) are well known to lead to possibly indefinite kernels [29]. From Theorem 3, positive semi-definite matrices have non-negative eigenvalues. In order to quantify the goodness of the two proposed edit-based kernels, the negative eigenfraction (NEF) is investigated, defined as the relative mass of the negative eigenvalues [94]:

$$NEF = \frac{\sum_{i:\lambda_i<0}|\lambda_i|}{\sum_i|\lambda_i|}. \tag{21}$$

Clearly, $NEF \in [0,1]$: the closer to 0, the better. Figure 3 shows the NEF of the kernel matrices evaluated over the 18 datasets from Section 5 for EK and SEK. Both the proposed kernels have rather low NEF, the maximum being 0.0631 (dataset AIDS when using EK). Another interesting aspect is that SEK always outperforms EK in terms of NEF for all datasets (i.e., always less than or equal to). As they do not satisfy Mercer's condition, it would be probably more suitable refer to those as *indefinite kernels*, following works such as [29,94,95], namely, "kernels" that despite their not being positive (semi-)definite perform reasonably good as confirmed, in our case, by competitive classification performances (Figure 1) and behavior remarkably close to a positive (semi-)definite kernel (Figure 3).

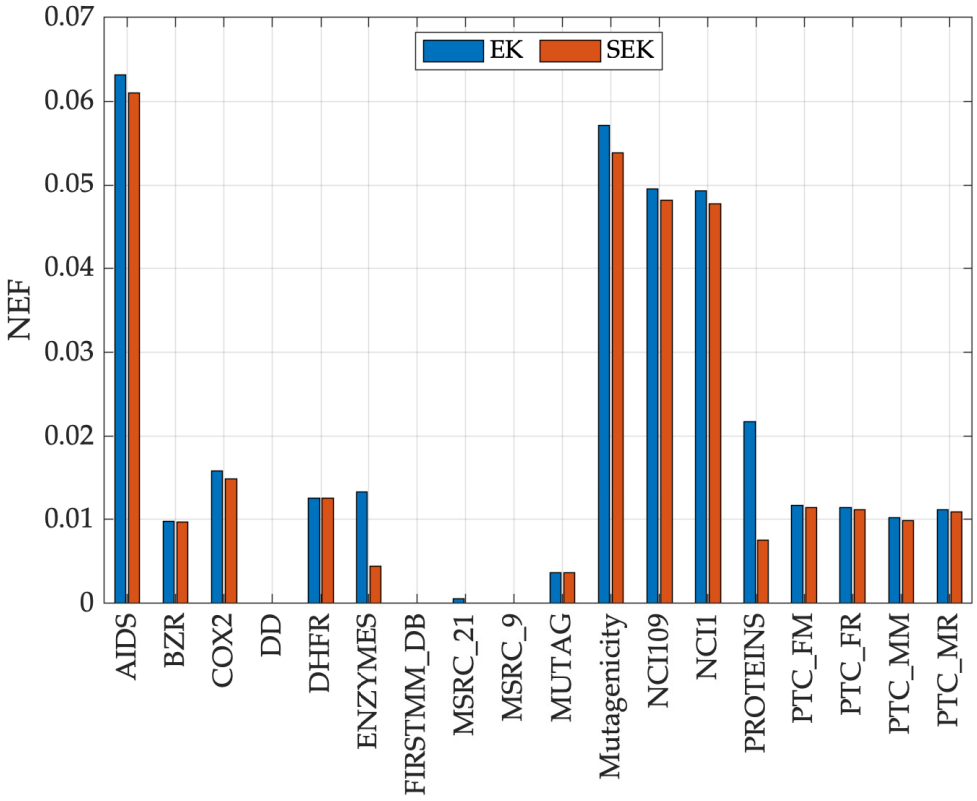

Figure 3. Negative Eigenfraction for the 18 tested datasets.

7. On the Computational Complexity of the Proposed Kernels

Each of the four proposed kernels works on a properly defined simplicial complex. Let us discuss the case of the clique complex used in this work: recall that the clique complex is the topological space in which each k-vertex clique is represented by a $(k-1)$-simplex. For an n-vertex graph, the maximum number of cliques in the worst-case goes like $\mathcal{O}(3^{n/3})$ [96] and the Bron–Kerbosh algorithm matches this bound [97]. Therefore, the worst-case cardinality of the clique complex also follows this bound. For the sake of completeness, the same same holds for the Vietoris–Rips complex [67].

For any two simplicial complexes, say $\mathcal{S}_1$ and $\mathcal{S}_2$ with cardinality $n_1 = |\mathcal{S}_1|$ and $n_2 = |\mathcal{S}_2|$, the evaluation of HCK and WJK is linear with the order of the two simplicial complexes, with a worst-case of $\mathcal{O}(n_1 + n_2)$, that is, when the two simplicial complexes have no simplices in common. For EK and SEK, as a Levenshtein-like edit distance [98,99] has been used, that is, instead of adding/substituting/deleting letters between two words, simplices has been added/substituted/deleted between two simplicial complexes, the computational complexity grows as $\mathcal{O}(n_1 \cdot n_2)$.

8. Conclusions

In this paper, four (hyper)graph kernels have been proposed in order to exploit the native multi-scale organization in complex networks. All of the four kernels rely on simplicial complexes, one of the possible hypergraph representations and, specifically, on the flag complex of the underlying graph. What makes the hypergraph paradigm preferable (for some applications) over the "plain graph" modeling is that the former better captures the information in the data by taking into account multi-way relations, whereas the latter intrinsically takes into account only pairwise relations. The four kernels exploit simplicial complexes under different lights: for HCK, an explicit embedding towards a vector space is performed

before evaluating the linear kernel; WJK relies on the set (or, better, multiset) structure of the simplicial complexes when equipped with semantic information on their respective nodes; EK and SEK measure the similarity in terms of edit operations defined on simplices.

All of the four kernels are easy to compute and rely on exact matching procedures among simplices given the categorical nature of the node labels. This is at the same time a strength and a weakness of the proposed kernels since edge labels and node attributes are not allowed. Future research endeavors can extend the proposed kernels to work with more complex node and/or edge labels.

An interesting aspect of the proposed kernels relies on them being parameter-free: if the underlying graph is available, its simplicial complex can be directly evaluated without any parameters to be tuned. This also holds if data natively comes as an hypergraph. This one-shot evaluation makes the training phase incredibly fast and appealing, as there is no need to evaluate multiple times the kernel matrix with different parameters and the only hyperparameter(s) to be tuned regard the classifier itself. Conversely, if the underlying graph is not available and data (nodes) come as a point cloud, the proposed kernels can still be employed, but they loose their parameter-free peculiarity: indeed, the flag complex cannot be directly used if the 1-skeleton is not natively available and one shall consider different simplicial complexes, such as the Vietoris–Rips complex or the Alpha complex, both of which strongly depend on a scale parameter (ϵ and α, respectively, cf. Section 3) which shall be tuned accordingly.

The four proposed kernels (HCK, WJK, EK, and SEK) have been benchmarked against nine state-of-the-art graph kernels (GS, NH, OS, PK, PM, RW, ST, WL, and WL-SP) on 18 open access datasets for graph classification. From this analysis, the following results emerged; the four kernels have competitive performances against current graph kernels, with HCK being slightly less accurate. Thanks to their parameter-free peculiarity, all of the proposed kernels have very fast training procedures. Furthermore they are also very appealing for large datasets, conversely to kernels such as OS, RW and WL-SP that went over the 24 h deadline or triggered out-of-memory errors. A second analysis regards the classification of metabolic pathways that, due to intrinsic molecular interactions, are conveniently represented by hypergraphs. Four classification problems are considered, following the Linnaeus' taxonomy at different scales: from coarse-grained discrimination (e.g., eukaryotes vs. prokaryotes) to fine-grained discrimination (e.g., among bacteria). Regardless of the resolution at which organisms are classified, all of the four proposed kernels show remarkable performances: a clear sign that the four proposed kernels are able to capture the metabolic diversity across different layers of biological organization allowing, in turn, a consistent classification spanning at different scales.

By jointly considering performances and time required in order to evaluate the kernel matrix, WJK seems the most promising kernel among the proposed ones. HCK is featured by faster evaluation times, but its performances are slightly lower. For EK and SEK, their evaluation time is incredibly high due to lack of vectorization. However, future research avenues can investigate the possibility to use dedicated hardware in order to speed-up the edit distance evaluation between simplicial complexes, see, e.g., in [99]. Finally, WJK has also been demonstrated to be a valid kernel as the Mercer's condition is satisfied.

Author Contributions: Conceptualization, A.M. and A.R.; methodology, A.M.; software, A.M.; validation, A.M.; formal analysis, A.M.; investigation, A.M.; resources, A.R.; data curation, A.M.; writing—original draft preparation, A.M. and A.R.; writing—review and editing, A.M. and A.R.; visualization, A.M.; supervision, A.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript.

EK	Edit Kernel
GS	Graphlet Sampling
HCK	Histogram Cosine Kernel
NEF	Negative EigenFraction
NH	Neighborhood Hash
OS	ODD-Sth
PK	Propagation Kernel
PM	Pyramid Match
RM	Random Walk
SEK	Stratified Edit Kernel
ST	SVM Theta
SVM	Support Vector Machine
WJK	Weighted Jaccard Kernel
WL	Weisfeiler–Lehman (Subtree Kernel)
WLSP	Weisfeiler–Lehman Shortest Path Kernel

References

1. Martino, A.; Giuliani, A.; Rizzi, A. Granular Computing Techniques for Bioinformatics Pattern Recognition Problems in Non-metric Spaces. In *Computational Intelligence for Pattern Recognition*; Pedrycz, W., Chen, S.M., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 53–81. [CrossRef]
2. Bizzarri, M.; Naimark, O.; Nieto-Villar, J.; Fedeli, V.; Giuliani, A. Complexity in Biological Organization: Deconstruction (and Subsequent Restating) of Key Concepts. *Entropy* **2020**, *22*, 885. [CrossRef]
3. Martino, A.; De Santis, E.; Giuliani, A.; Rizzi, A. Modelling and Recognition of Protein Contact Networks by Multiple Kernel Learning and Dissimilarity Representations. *Entropy* **2020**, *22*, 794. [CrossRef]
4. Jeong, H.; Tombor, B.; Albert, R.; Oltvai, Z.N.; Barabási, A.L. The large-scale organization of metabolic networks. *Nature* **2000**, *407*, 651. [CrossRef] [PubMed]
5. Martino, A.; Rizzi, A.; Frattale Mascioli, F.M. Supervised Approaches for Protein Function Prediction by Topological Data Analysis. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8. [CrossRef]
6. Wuchty, S. Scale-Free Behavior in Protein Domain Networks. *Mol. Biol. Evol.* **2001**, *18*, 1694–1702. [CrossRef]
7. Martino, A.; Maiorino, E.; Giuliani, A.; Giampieri, M.; Rizzi, A. Supervised Approaches for Function Prediction of Proteins Contact Networks from Topological Structure Information. In *Image Analysis: 20th Scandinavian Conference, SCIA 2017, Tromsø, Norway, 12–14 June 2017, Proceedings, Part I*; Sharma, P., Bianchi, F.M., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 285–296. [CrossRef]
8. Davidson, E.H.; Rast, J.P.; Oliveri, P.; Ransick, A.; Calestani, C.; Yuh, C.H.; Minokawa, T.; Amore, G.; Hinman, V.; Arenas-Mena, C.; et al. A Genomic Regulatory Network for Development. *Science* **2002**, *295*, 1669–1678. [CrossRef]
9. Gasteiger, J.; Engel, T. *Chemoinformatics: A Textbook*; John Wiley & Sons: Hoboken, NJ, USA, 2006.
10. Krishnan, A.; Zbilut, J.P.; Tomita, M.; Giuliani, A. Proteins as networks: Usefulness of graph theory in protein science. *Curr. Protein Pept. Sci.* **2008**, *9*, 28–38. [CrossRef]
11. Di Paola, L.; De Ruvo, M.; Paci, P.; Santoni, D.; Giuliani, A. Protein contact networks: An emerging paradigm in chemistry. *Chem. Rev.* **2012**, *113*, 1598–1613. [CrossRef]
12. Giuliani, A.; Filippi, S.; Bertolaso, M. Why network approach can promote a new way of thinking in biology. *Front. Genet.* **2014**, *5*, 83. [CrossRef]
13. Di Paola, L.; Giuliani, A. Protein–Protein Interactions: The Structural Foundation of Life Complexity. In *eLS*; John Wiley & Sons Ltd.: Chichester, UK, 2017; pp. 1–12. [CrossRef]

14. Wasserman, S.; Faust, K. *Social Network Analysis: Methods and Applications*; Cambridge University Press: New York, NY, USA, 1994.
15. Aldea, E.; Atif, J.; Bloch, I. Image Classification Using Marginalized Kernels for Graphs. In *Graph-Based Representations in Pattern Recognition*; Escolano, F., Vento, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 103–113. [CrossRef]
16. Harchaoui, Z.; Bach, F. Image Classification with Segmentation Graph Kernels. In Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 7–22 June 2007; pp. 1–8. [CrossRef]
17. Bach, F.R. Graph Kernels Between Point Clouds. In Proceedings of the 25th International Conference on Machine Learning ICML '08, Helsinki, Finland, 5–9 July 2008; ACM: New York, NY, USA, 2008; pp. 25–32. [CrossRef]
18. Rizzi, A.; Del Vescovo, G. Automatic Image Classification by a Granular Computing Approach. In Proceedings of the 2006 16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing, Arlington, VA, USA, 6–8 September 2006; pp. 33–38. [CrossRef]
19. Collins, M.; Duffy, N. Convolution Kernels for Natural Language. In Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic (NIPS'01), Vancouver, BC, Canada, 3–8 December 2001; MIT Press: Cambridge, MA, USA, 2001; pp. 625–632.
20. Das, N.; Ghosh, S.; Gonçalves, T.; Quaresma, P. Comparison of Different Graph Distance Metrics for Semantic Text Based Classification. *Polibits* **2014**, 51–58. [CrossRef]
21. Das, N.; Ghosh, S.; Gonçalves, T.; Quaresma, P. Using Graphs and Semantic Information to Improve Text Classifiers. In *Advances in Natural Language Processing*; Przepiórkowski, A., Ogrodniczuk, M., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 324–336.
22. De Santis, E.; Martino, A.; Rizzi, A. An Infoveillance System for Detecting and Tracking Relevant Topics From Italian Tweets During the COVID-19 Event. *IEEE Access* **2020**, *8*, 132527–132538. [CrossRef]
23. Possemato, F.; Paschero, M.; Livi, L.; Rizzi, A.; Sadeghian, A. On the impact of topological properties of smart grids in power losses optimization problems. *Int. J. Electr. Power Energy Syst.* **2016**, *78*, 755–764. [CrossRef]
24. Bunke, H. Graph-Based Tools for Data Mining and Machine Learning. In *Machine Learning and Data Mining in Pattern Recognition*; Perner, P., Rosenfeld, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 7–19.
25. Ghosh, S.; Das, N.; Gonçalves, T.; Quaresma, P.; Kundu, M. The journey of graph kernels through two decades. *Comput. Sci. Rev.* **2018**, *27*, 88–111. [CrossRef]
26. Bunke, H. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognit. Lett.* **1997**, *18*, 689–694. [CrossRef]
27. Bunke, H.; Shearer, K. A graph distance metric based on the maximal common subgraph. *Pattern Recognit. Lett.* **1998**, *19*, 255–259. [CrossRef]
28. Fernández, M.L.; Valiente, G. A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognit. Lett.* **2001**, *22*, 753–758. [CrossRef]
29. Neuhaus, M.; Bunke, H. Edit distance-based kernel functions for structural pattern classification. *Pattern Recognit.* **2006**, *39*, 1852–1863. [CrossRef]
30. Livi, L.; Rizzi, A. The graph matching problem. *Pattern Anal. Appl.* **2013**, *16*, 253–283. [CrossRef]
31. Livi, L.; Rizzi, A. Graph ambiguity. *Fuzzy Sets Syst.* **2013**, *221*, 24–47. [CrossRef]
32. Duin, R.P.; Pękalska, E. The dissimilarity space: Bridging structural and statistical pattern recognition. *Pattern Recognit. Lett.* **2012**, *33*, 826–832. [CrossRef]
33. Pękalska, E.; Duin, R.P.; Paclík, P. Prototype selection for dissimilarity-based classifiers. *Pattern Recognit.* **2006**, *39*, 189–208. [CrossRef]
34. De Santis, E.; Martino, A.; Rizzi, A.; Frattale Mascioli, F.M. Dissimilarity Space Representations and Automatic Feature Selection for Protein Function Prediction. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8. [CrossRef]
35. Baldini, L.; Martino, A.; Rizzi, A. Stochastic Information Granules Extraction for Graph Embedding and Classification. In Proceedings of the 11th International Joint Conference on Computational Intelligence—Volume 1: NCTA (IJCCI 2019), Dhaka, Bangladesh, 25–26 October 2019; pp. 391–402. [CrossRef]

36. Martino, A.; Giuliani, A.; Rizzi, A. (Hyper)Graph Embedding and Classification via Simplicial Complexes. *Algorithms* **2019**, *12*, 223. [CrossRef]
37. Baldini, L.; Martino, A.; Rizzi, A. Exploiting Cliques for Granular Computing-based Graph Classification. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–9. [CrossRef]
38. Martino, A.; Frattale Mascioli, F.M.; Rizzi, A. On the Optimization of Embedding Spaces via Information Granulation for Pattern Recognition. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8. [CrossRef]
39. Martino, A.; Giuliani, A.; Todde, V.; Bizzarri, M.; Rizzi, A. Metabolic networks classification and knowledge discovery by information granulation. *Comput. Biol. Chem.* **2020**, *84*, 107187. [CrossRef] [PubMed]
40. Martino, A.; De Santis, E.; Rizzi, A. An Ecology-based Index for Text Embedding and Classification. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8. [CrossRef]
41. Cover, T.M. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Trans. Electron. Comput.* **1965**, 326–334. [CrossRef]
42. Schölkopf, B.; Smola, A.J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*; MIT Press: Cambridge, MA, USA, 2002.
43. Shawe-Taylor, J.; Cristianini, N. *Kernel Methods for Pattern Analysis*; Cambridge University Press: Cambridge, UK, 2004.
44. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
45. Jaakkola, T.S.; Haussler, D. Exploiting Generative Models in Discriminative Classifiers. In *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*; MIT Press: Cambridge, MA, USA, 1999; pp. 487–493.
46. Kondor, R.I.; Lafferty, J. Diffusion kernels on graphs and other discrete structures. In Proceedings of the 19th International Conference on Machine Learning, Sydney, Australia, 8–12 July 2002; Volume 2002, pp. 315–322.
47. Vishwanathan, S.V.N.; Schraudolph, N.N.; Kondor, R.; Borgwardt, K.M. Graph kernels. *J. Mach. Learn. Res.* **2010**, *11*, 1201–1242.
48. Gärtner, T.; Flach, P.; Wrobel, S. On Graph Kernels: Hardness Results and Efficient Alternatives. In *Learning Theory and Kernel Machines*; Schölkopf, B., Warmuth, M.K., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 129–143.
49. Borgwardt, K.M.; Kriegel, H.P. Shortest-path kernels on graphs. In Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05), Houston, TX, USA, 27–30 November 2005; p. 8. [CrossRef]
50. Shervashidze, N.; Vishwanathan, S.; Petri, T.; Mehlhorn, K.; Borgwardt, K. Efficient graphlet kernels for large graph comparison. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*; van Dyk, D., Welling, M., Eds.; PMLR: Hilton Clearwater Beach Resort, Clearwater Beach, FL, USA, 2009; Volume 5, pp. 488–495.
51. Shervashidze, N.; Borgwardt, K. Fast subtree kernels on graphs. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2009; pp. 1660–1668.
52. Shervashidze, N.; Schweitzer, P.; van Leeuwen, E.J.; Mehlhorn, K.; Borgwardt, K.M. Weisfeiler-lehman graph kernels. *J. Mach. Learn. Res.* **2011**, *12*, 2539–2561.
53. Neumann, M.; Garnett, R.; Bauckhage, C.; Kersting, K. Propagation kernels: Efficient graph kernels from propagated information. *Mach. Learn.* **2016**, *102*, 209–245. [CrossRef]
54. Yanardag, P.; Vishwanathan, S. Deep Graph Kernels. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15), Sydney, Australia, 10–13 August 2015; ACM: New York, NY, USA, 2015; pp. 1365–1374. [CrossRef]
55. Giusti, C.; Ghrist, R.; Bassett, D.S. Two's company, three (or more) is a simplex. *J. Comput. Neurosci.* **2016**, *41*, 1–14. [CrossRef]
56. Malod-Dognin, N.; Gaudelet, T.; Pržulj, N. Higher-order molecular organization as a source of biological function. *Bioinformatics* **2018**, *34*, i944–i953. [CrossRef]

57. Barbarossa, S.; Sardellitti, S. Topological Signal Processing Over Simplicial Complexes. *IEEE Trans. Signal Process.* **2020**, *68*, 2992–3007. [CrossRef]
58. Barbarossa, S.; Sardellitti, S.; Ceci, E. Learning from Signals Defined over Simplicial Complexes. In Proceedings of the 2018 IEEE Data Science Workshop (DSW), Lausanne, Switzerland, 4–6 June 2018; pp. 51–55. [CrossRef]
59. Barbarossa, S.; Tsitsvero, M. An introduction to hypergraph signal processing. In Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, 20–25 March 2016; pp. 6425–6429. [CrossRef]
60. Ramadan, E.; Tarafdar, A.; Pothen, A. A hypergraph model for the yeast protein complex network. In Proceedings of the 18th International Parallel and Distributed Processing Symposium, 2004, Santa Fe, NM, USA, 26–30 April 2004; p. 189. [CrossRef]
61. Lugo-Martinez, J.; Zeiberg, D.; Gaudelet, T.; Malod-Dognin, N.; Pržulj, N.; Radivojac, P. Classification in biological networks with hypergraphlet kernels. *Bioinformatics* **2020**. [CrossRef] [PubMed]
62. Munkres, J.R. *Elements of Algebraic Topology*; Addison-Wesley: San Francisco, CA, USA, 1984.
63. Grady, L.J.; Polimeni, J.R. *Discrete Calculus: Applied Analysis on Graphs for Computational Science*; Springer Science & Business Media: Cham, Switzerland, 2010.
64. Carlsson, G. Topology and data. *Bull. Am. Math. Soc.* **2009**, *46*, 255–308. [CrossRef]
65. Zomorodian, A. Topological data analysis. *Adv. Appl. Comput. Topol.* **2012**, *70*, 1–39.
66. Wasserman, L. Topological Data Analysis. *Annu. Rev. Stat. Its Appl.* **2018**, *5*, 501–532. [CrossRef]
67. Zomorodian, A. Fast construction of the Vietoris-Rips complex. *Comput. Graph.* **2010**, *34*, 263–271. [CrossRef]
68. Baldini, L.; Martino, A.; Rizzi, A. Towards a Class-Aware Information Granulation for Graph Embedding and Classification. In Proceedings of the Computational Intelligence: 11th International Joint Conference (IJCCI 2019), Vienna, Austria, 17–19 September 2019.
69. Sugiyama, M.; Borgwardt, K. Halting in Random Walk Kernels. In *Advances in Neural Information Processing Systems 28*; Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2015; pp. 1639–1647.
70. Yujian, L.; Bo, L. A Normalized Levenshtein Distance Metric. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 1091–1095. [CrossRef]
71. Chen, S.; Ma, B.; Zhang, K. On the similarity metric and the distance metric. *Theor. Comput. Sci.* **2009**, *410*, 2365–2376. [CrossRef]
72. Kersting, K.; Kriege, N.M.; Morris, C.; Mutzel, P.; Neumann, M. Benchmark Data Sets for Graph Kernels. 2016. Available online: http://graphkernels.cs.tu-dortmund.de (accessed on 14 October 2020)
73. Hido, S.; Kashima, H. A Linear-Time Graph Kernel. In Proceedings of the 2009 Ninth IEEE International Conference on Data Mining, Miami, FL, USA, 6–9 December 2009; pp. 179–188. [CrossRef]
74. Da San Martino, G.; Navarin, N.; Sperduti, A. A tree-based kernel for graphs. In Proceedings of the 2012 SIAM International Conference on Data Mining, SIAM, Anaheim, CA, USA, 26–28 April 2012; pp. 975–986.
75. Nikolentzos, G.; Meladianos, P.; Vazirgiannis, M. Matching Node Embeddings for Graph Similarity. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI'17, San Francisco, CA, USA, 4–9 February 2017; AAAI Press: Palo Alto, CA, USA, 2017; pp. 2429–2435.
76. Vishwanathan, S.V.N.; Borgwardt, K.M.; Schraudolph, N.N. Fast Computation of Graph Kernels. In Proceedings of the 19th International Conference on Neural Information Processing Systems NIPS'06, Vancouver, BC, Canada, 4–7 December 2006; MIT Press: Cambridge, MA, USA, 2006; pp. 1449–1456.
77. Johansson, F.D.; Jethava, V.; Dubhashi, D.; Bhattacharyya, C. Global Graph Kernels Using Geometric Embeddings. In Proceedings of the 31st International Conference on International Conference on Machine Learning ICML'14, Beijing, China, 21–26 June 2014; Volume 32, pp. II-694–II-702.
78. Bron, C.; Kerbosch, J. Algorithm 457: Finding All Cliques of an Undirected Graph. *Commun. ACM* **1973**, *16*, 575–577. [CrossRef]
79. Cazals, F.; Karande, C. A note on the problem of reporting maximal cliques. *Theor. Comput. Sci.* **2008**, *407*, 564–568. [CrossRef]

80. Schölkopf, B.; Smola, A.J.; Williamson, R.C.; Bartlett, P.L. New Support Vector Algorithms. *Neural Comput.* **2000**, *12*, 1207–1245. [CrossRef]
81. Bergstra, J.; Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
82. Oliphant, T.E. Python for Scientific Computing. *Comput. Sci. Eng.* **2007**, *9*, 10–20. [CrossRef]
83. Hagberg, A.A.; Schult, D.A.; Swart, P.J. Exploring Network Structure, Dynamics, and Function using NetworkX. In Proceedings of the 7th Python in Science Conference, Pasadena, CA, USA, 19–24 August 2008; pp. 11–15.
84. Siglidis, G.; Nikolentzos, G.; Limnios, S.; Giatsidis, C.; Skianis, K.; Vazirgiannis, M. GraKeL: A Graph Kernel Library in Python. *J. Mach. Learn. Res.* **2020**, *21*, 1–5.
85. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
86. Tun, K.; Dhar, P.K.; Palumbo, M.C.; Giuliani, A. Metabolic pathways variability and sequence/networks comparisons. *BMC Bioinform.* **2006**, *7*, 24. [CrossRef] [PubMed]
87. Martino, A.; Giuliani, A.; Rizzi, A. The Universal Phenotype. *Org. J. Biol. Sci.* **2019**, *3*, 8–10. [CrossRef]
88. Montañez, R.; Medina, M.A.; Solé, R.V.; Rodríguez-Caso, C. When metabolism meets topology: Reconciling metabolite and reaction networks. *BioEssays* **2010**, *32*, 246–256. [CrossRef]
89. Zhou, W.; Nakhleh, L. Properties of metabolic graphs: Biological organization or representation artifacts? *BMC Bioinform.* **2011**, *12*, 132. [CrossRef]
90. Mercer, J. Functions of positive and negative type, and their connection with the theory of integral equations. *Philos. Trans. R. Soc. Lond. Ser. A Contain. Pap. Math. Phys. Character* **1909**, *209*, 415–446.
91. Horn, R.A.; Johnson, C.R. *Matrix Analysis*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2013.
92. Nader, R.; Bretto, A.; Mourad, B.; Abbas, H. On the positive semi-definite property of similarity matrices. *Theor. Comput. Sci.* **2019**, *755*, 13–28. [CrossRef]
93. Gardner, A.; Duncan, C.A.; Kanno, J.; Selmic, R.R. On the Definiteness of Earth Mover's Distance and Its Relation to Set Intersection. *IEEE Trans. Cybern.* **2018**, *48*, 3184–3196. [CrossRef] [PubMed]
94. Pękalska, E.; Harol, A.; Duin, R.P.W.; Spillmann, B.; Bunke, H. Non-Euclidean or Non-metric Measures Can Be Informative. In *Structural, Syntactic, and Statistical Pattern Recognition*; Yeung, D.Y., Kwok, J.T., Fred, A., Roli, F., de Ridder, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 871–880.
95. Ong, C.S.; Mary, X.; Canu, S.; Smola, A.J. Learning with Non-Positive Kernels. In Proceedings of the ICML 2004, Banff, AB, Canada, 4–8 July 2004; Max-Planck-Gesellschaft; ACM Press: New York, NY, USA, 2004; p. 81.
96. Moon, J.W.; Moser, L. On cliques in graphs. *Isr. J. Math.* **1965**, *3*, 23–28. [CrossRef]
97. Tomita, E.; Tanaka, A.; Takahashi, H. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theor. Comput. Sci.* **2006**, *363*, 28–42. [CrossRef]
98. Levenshtein, V.I. Binary codes capable of correcting deletions, insertions, and reversals. *Sov. Phys. Dokl.* **1966**, *10*, 707–710.
99. Cinti, A.; Bianchi, F.M.; Martino, A.; Rizzi, A. A Novel Algorithm for Online Inexact String Matching and its FPGA Implementation. *Cogn. Comput.* **2020**, *12*, 369–387. [CrossRef]