Principled Simplicial Neural Networks for Trajectory Prediction

T. Mitchell Roddenberry [*1] **Nicholas Glaze** [*1] **Santiago Segarra** [1]

Abstract

We consider the construction of neural network architectures for data on simplicial complexes. In studying maps on the chain complex of a simplicial complex, we define three desirable properties of a simplicial neural network architecture: namely, permutation equivariance, orientation equivariance, and simplicial awareness. The first two properties respectively account for the fact that the indexing and orientations of simplices in a simplicial complex are arbitrary. The last property requires the output of the neural network to depend on the entire simplicial complex and not on a subset of its dimensions. Based on these properties, we propose a simple convolutional architecture, rooted in tools from algebraic topology, for the problem of trajectory prediction, and show that it obeys all three of these properties when an odd, nonlinear activation function is used. We then demonstrate the effectiveness of this architecture in extrapolating trajectories on synthetic and real datasets, with particular emphasis on the gains in generalizability to unseen trajectories.

1. Introduction

Graph neural networks have shown great promise in combining the representational power of neural networks with the structure imparted by a graph. In essence, graph neural networks compute a sequence of node representations by aggregating information at each node from its neighbors and itself, then applying a nonlinear transformation. Using variations on this approach, many architectures have exhibited state-of-the-art performance in tasks including node classification (Veličković et al., 2018), link prediction (Zhang & Chen, 2018), and graph classification (Hamilton et al., 2017). Indeed, the strength of graph neural networks lies in their ability to incorporate arbitrary pairwise relational structures in their computations.

However, not all data is adequately expressed in terms of pairwise relationships, nor is it strictly supported on the nodes of a graph. Interactions in a social network, for instance, do not solely occur in a pairwise fashion but also among larger groups of people. This warrants a higher-order model in order to represent rich and complex datasets. One such higher-order model is the (abstract) simplicial complex, which describes relational structures that are closed under restriction: if three people are all friends together, then each pair of people in that group are also friends. We can understand data supported on simplicial complexes using tools from algebraic topology (Hatcher, 2002; Carlsson, 2009; Ghrist, 2014); in particular, we can analyze data supported on the edges and higher-order structures using the spectrum of certain linear operators on the simplicial complex. This is analogous to the use of spectral graph theory (Chung, 1997) to understand the smoothness of data supported on the nodes of a graph (Shuman et al., 2013).

1.1. Contribution

We study the extension of graph convolutional network architectures to process data supported on simplicial complexes. After establishing appropriate operators for such data in Section 3, our first contribution is to define in Section 4 a notion of admissibility, in terms of three reasonable properties that we require of neural networks for simplicial complexes. We then focus on the problem of trajectory prediction over a simplicial complex, proposing a simple convolutional neural architecture in Section 5, which we design with admissibility in mind. In particular, we show that the activation functions of the proposed architecture must be odd and nonlinear in order to satisfy our requirements for admissibility. We empirically illustrate how admissibility yields better generalizability to unseen trajectories in Section 6.

2. Related work

2.1. Graph Neural Networks

Graph neural networks extend the success of convolutional neural networks for Euclidean data to the graph domain,

[*]Equal contribution [1]Department of Electrical and Computer Engineering, Rice University, Houston, Texas, USA. Correspondence to: TMR <mitch@rice.edu>, NG <nkg2@rice.edu>, SS <segarra@rice.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

adapting the weight-sharing of convolutional networks in a way that reflects the underlying graph structure. Effectively, graph neural networks interleave nonlinear activation functions with diffusion operators dictated by the graph structure, such as the adjacency or Laplacian matrices. Bruna et al. (2014) developed graph neural networks in the Laplacian spectral domain, which was further simplified by Defferrard et al. (2016), who expressed the diffusion operator at each layer as a low-order Chebyshev polynomial in order to improve scalability. Kipf & Welling (2017) reduced this further, employing a first-order diffusion operator at each layer expressed in terms of the graph Laplacian. We refer the reader to Wu et al. (2020) for a survey of graph neural network architectures and applications.

2.2. Signal Processing on Simplicial Complexes

Aiming to extend the field of signal processing on graphs (Shuman et al., 2013), recent works have used tools from algebraic topology to understand data supported on simplicial complexes. Rooted in discrete calculus on graphs, and in particular combinatorial Hodge theory (Jiang et al., 2011; Lim, 2020), recent works have considered the use of the Hodge Laplacian for the analysis of flows on graphs (Schaub & Segarra, 2018; Barbarossa et al., 2018; Barbarossa & Sardellitti, 2020a;b). This line of research was distinct from existing developments in graph signal processing, in that it handled flows defined with respect to an arbitrary orientation assigned to each edge, much like the analysis of current in an electrical circuit. One application of this perspective was studied by Jia et al. (2019), where the problem of flow interpolation was cast as an optimization problem, minimizing the quadratic form of the Hodge Laplacian subject to observation constraints.

2.3. Simplicial Neural Networks

The first application of discrete Hodge theory to the design of neural networks was proposed by Roddenberry & Segarra (2019). This work analyzed edge-flow data, focusing on the problems of flow interpolation and source localization on simplicial complexes, while drawing attention to the value of permutation and orientation equivariance. Since then, other works (Ebli et al., 2020; Bunch et al., 2020) have also relied on discrete Hodge theory to propose convolutional neural architectures for data supported on simplicial complexes. However, these works do not address important notions of orientation, which we discuss in Section 4. Instead of solely proposing a *specific* neural architecture, our current work develops a principled *framework* to construct generalizable simplicial neural network architectures, proposes a simple architecture following those principles, and illustrates its advantages compared with competing approaches.

3. Background

3.1. Simplicial Complexes

An *abstract simplicial complex* is a set $\mathcal{X}$ of finite subsets of another set $\mathcal{V}$ that is closed under restriction, *i.e.* for all $\sigma \in \mathcal{X}$, if $\sigma' \subseteq \sigma$, we have $\sigma' \in \mathcal{X}$. Each such element of $\mathcal{X}$ is called a *simplex*: in particular, if $|\sigma| = k+1$, we call σ a *k-simplex*. For a k-simplex σ, its *faces* are all of the $(k-1)$-simplices that are also subsets of σ, while its *cofaces* are all $(k+1)$-simplices that have σ as a face.

Grounding these definitions in our intuition for graphs, we refer to the elements of $\mathcal{V}$ as *nodes*, or equivalently the 0-simplices of $\mathcal{X}$. We refer to the 1-simplices of $\mathcal{X}$ as *edges*, and the 2-simplices as *triangles*, corresponding to "filled-in triangles" in a departure from classical graphs. The edges, then, are faces of the triangles, with the nodes being faces of the edges. We refer to higher-order simplices by their order: k-simplices. For convenience, we use the notation $\mathcal{X}_k$ to refer to the collection of k-simplices of $\mathcal{X}$, *e.g.*, $\mathcal{X}_0 = \mathcal{V}$. The *dimension* of a simplicial complex $\mathcal{X}$ is the maximal k such that $\mathcal{X}_k$ is nonempty. The *k-skeleton* of a simplicial complex is the union $\bigcup_{\ell=0}^{k} \mathcal{X}_\ell$. We will later find it convenient to refer to the *neighborhood* of a node i: for a simplicial complex $\mathcal{X}$, denote by $\mathcal{N}(i)$ the set of all $j \in \mathcal{X}_0$ such that $\{i,j\} \in \mathcal{X}_1$, *i.e.*, the set of nodes connected to i by an edge. Or, using our defined terminology for simplicial complexes, the neighborhood of a node i consists of the faces of the cofaces of i, excluding i itself.

We arbitrarily endow each simplex of $\mathcal{X}$ with an *orientation*. An orientation can be thought of as a chosen ordering of the constituent elements of a simplex, modulo even permutations. That is, for a k-simplex $\sigma = \{i_0, i_1 \dots, i_k\} \subseteq \mathcal{V}$, an orientation of σ would be $[i_0, i_1, \dots, i_k]$. Performing an even permutation of this (*e.g.*, $[i_1, i_2, i_0, \dots, i_k]$) leads to an equivalent orientation. For convenience, we label the nodes with the non-negative integers, and let the chosen orientations of all simplices be given by the ordering induced by the node labels.

3.2. Boundary Operators and Hodge Laplacians

Denote by $\mathcal{C}_k$ the vector space with the oriented k-simplices of $\mathcal{X}$ as a canonical orthonormal basis, defined over the field of real numbers. We define notions of matrix multiplication and linear operators, *e.g.*, diagonal matrices and permutation matrices, with respect to this basis. Each element of $\mathcal{C}_k$ is called a *k-chain* and is subject to all the properties of a vector. In particular, multiplying a k-chain by $+1$ is idempotent, while multiplication by -1 reverses its orientation (by reversing the orientations of its basis vectors). For instance, let $[i_0, i_1] \in \mathcal{C}_1$. We then have $[i_0, i_1] = -[i_1, i_0]$. Using this basis of oriented simplices, we endow the vector spaces $\mathcal{C}_k$ with the usual properties of finite-dimensional

real vector spaces such as inner products and linear maps expressed as real matrices.

A particular set of linear maps between chains is the set of *boundary operators*[1] $\{\partial_k\}_{k=1}^K$, where K is the highest order of any simplex in $\mathcal{X}$. For an oriented k-simplex $\sigma = [i_0, i_1, \dots, i_k]$, the boundary operator $\partial_k : \mathcal{C}_k \to \mathcal{C}_{k-1}$ is defined as

$$\partial_k \sigma = \sum_{j=0}^{k} (-1)^j [i_0, \dots, i_{j-1}, i_{j+1}, \dots, i_k]. \quad (1)$$

That is, the boundary operator takes an ordered, alternating sum of the faces that form the boundary of σ. The collection of these vector spaces coupled with the boundary maps forms what is known as a *chain complex*. Indeed, to represent a simplicial complex it is sufficient to use its boundary maps. An important result in algebraic topology relates these boundary operators to each other.

Lemma 1. *The boundary operator squared is null. That is, for all k, $\partial_{k-1} \circ \partial_k = 0$.*

Additionally, the boundary operator induces a *co-boundary operator*, which is the adjoint of ∂, *i.e.* $\partial_k^\top$. Based on the boundary and coboundary operators, we define the k^{th} *Hodge Laplacian* as

$$\Delta_k = \partial_k^\top \partial_k + \partial_{k+1} \partial_{k+1}^\top. \quad (2)$$

Since the linear map ∂_1 can be represented via the signed node-edge incidence matrix, one can check that Δ_0 recovers the graph Laplacian (see supplementary material for details). More generally, $\Delta_k : \mathcal{C}_k \to \mathcal{C}_k$ is a linear operator on the space of k-chains for each k. The *Hodge Decomposition* allows us to view the vector space $\mathcal{C}_k$ in terms of the boundary maps and the Hodge Laplacian.

Theorem 1 (Hodge Decomposition)**.** *For a simplicial complex $\mathcal{X}$ with boundary maps $\partial = \{\partial_k\}_{k=1}^K$ we have that*

$$\mathcal{C}_k = \operatorname{im}(\partial_{k+1}) \oplus \operatorname{im}(\partial_k^\top) \oplus \ker(\Delta_k) \quad (3)$$

for all k, where $\oplus$ represents the (orthogonal) direct sum.

This result is particularly pleasing, as it gives us a convenient representation of a k-chain in terms of the "upper" and "lower" incidence structures of the simplices on which it is supported, as defined by its boundary operators. Moreover, we have that $\dim \ker(\Delta_k) = \beta_k$, where β_k is the k^{th} *Betti number*, which counts the number of "k-dimensional holes" in $\mathcal{X}$ (Carlsson, 2009).

Of particular interest in this work is the Hodge Decomposition of the space $\mathcal{C}_1$, which models "flows" on simplicial complexes, as studied in detail by Schaub et al. (2020). Indeed, 1-chains on a simplicial complex are a natural way to discretize a continuous vector field (Barbarossa & Sardellitti, 2020b), or to model the flow of traffic in a road network (Jia et al., 2019; Roddenberry & Segarra, 2019). That is, we consider the decomposition $\mathcal{C}_1 = \operatorname{im}(\partial_2) \oplus \operatorname{im}(\partial_1^\top) \oplus \ker(\Delta_1)$.

First, we note that $\operatorname{im}(\partial_2)$ corresponds to 1-chains that are *curly* with respect to the triangles in a simplicial complex: that is, such 1-chains consist of flows around the boundary of triangles.

Next, we observe that $\operatorname{im}(\partial_1^\top)$ corresponds to 1-chains induced by node *gradients*. Precisely, 1-chains in the image of $\partial_1^\top$ are determined by a set of scalar values on the nodes, whose local differences dictate the coefficients of the 1-chain, analogously to vector fields in Euclidean space induced by the gradient of a scalar field.

Finally, the subspace of $\mathcal{C}_1$ determined by $\ker(\Delta_1)$ consists of 1-chains that are neither curly nor gradient: we call such chains *harmonic*. Harmonic chains are 1-chains with the property that the sum of the coefficients around any triangle is zero, while the sum of the flow coefficients incident to any node is also zero. This subspace is of particular interest, since it captures a natural notion of a smooth, conservative flow, as leveraged by Ghosh et al. (2018); Jia et al. (2019); Schaub & Segarra (2018); Schaub et al. (2020); Barbarossa et al. (2018); Barbarossa & Sardellitti (2020a). We refer the reader to these works for more in-depth discussion of modeling flows with 1-chains and the relevant applications of discrete Hodge theory, as well as to the supplementary material for a more in-depth discussion of Theorem 1.

4. Admissible Neural Architectures

We define three desirable properties of a graph neural network acting on chains supported by a simplicial complex. These properties will be later leveraged to construct our proposed architecture for the task of trajectory prediction. Throughout, let $\text{SCN}_{\mathbf{W},\partial} : \mathcal{C}_j \to \mathcal{C}_\ell$ denote a neural network architecture acting on input data $\mathbf{c}_j \in \mathcal{C}_j$ and whose output is a chain of possibly different order, where the neural network is parameterized by a collection of weights $\mathbf{W}$ and boundary operators ∂.

4.1. Permutation Equivariance

When representing graphs and related structures with matrices, a key property is *permutation equivariance*. For instance, if we take a graph with adjacency matrix $\mathbf{A}$, then multiplying $\mathbf{A}$ from both sides by a permutation matrix, *i.e.* $\mathbf{PAP}^\top$, corresponds to relabeling the nodes in the original graph. In this way, $\mathbf{A}$ and $\mathbf{PAP}^\top$ are alternative matrix representations of the *same* graph. Therefore, in order to ensure that an operation does not depend on the specific (arbitrary) node labeling, this operation must be impervious to the application of a permutation matrix. More formally, and

[1] We ignore the boundary operator ∂_0, which maps $\mathcal{C}_0 \to \{0\}$.

for simplicial complexes in general, we define permutation equivariance as follows.

Property 1 (Permutation Equivariance)**.** Let $\mathcal{X}$ be a simplicial complex with boundary maps $\partial = \{\partial_k\}_{k=1}^K$. Let $\mathcal{P} = \{\mathbf{P}_k\}_{k=0}^K$ be a collection of permutation matrices matching the dimensions of $\{\mathcal{C}_k\}_{k=0}^K$, *i.e.* $\mathbf{P}_k \in \mathbb{R}^{|\mathcal{X}_k|\times|\mathcal{X}_k|}$, and define $[\mathcal{P}\partial]_k := \mathbf{P}_{k-1}\partial_k\mathbf{P}_k^\top$. We say that SCN satisfies *permutation equivariance* if for any such $\mathcal{P}$, we have that

$$\mathrm{SCN}_{\mathbf{W},\partial}(\mathbf{c}_j) = \mathbf{P}_\ell \mathrm{SCN}_{\mathbf{W},\mathcal{P}\partial}(\mathbf{P}_j\mathbf{c}_j). \tag{4}$$

The above expression guarantees that if we relabel the simplicial complex and apply a neural network, the output is a relabeled version of the output that we would have obtained by applying the neural network prior to relabeling.

4.2. Orientation Equivariance

In defining the boundary operators ∂, we choose an orientation for each simplex in $\mathcal{X}$. The choice of this boundary is arbitrary, and only serves to meaningfully represent boundary operations and useful signals on simplicial complexes. Similar to the arbitrary choice of ordering in the matrix representation motivating permutation equivariance, we also require an architecture to be insensitive to the chosen orientations. Recalling that reversing the orientation of a k-chain is equivalent to multiplying it by -1, we define orientation equivariance as follows.

Property 2 (Orientation Equivariance)**.** Let $\mathcal{X}$ be a simplicial complex with boundary maps $\partial = \{\partial_k\}_{k=1}^K$. Let $\mathcal{D} = \{\mathbf{D}_k\}_{k=0}^K$ be a collection of diagonal matrices with values taking ± 1 with the condition that $\mathbf{D}_0 = \mathbf{I}$, and matching the dimensions of $\{\mathcal{C}_k\}_{k=0}^K$, *i.e.* $\mathbf{D}_k \in \mathbb{R}^{|\mathcal{X}_k|\times|\mathcal{X}_k|}$, and define $[\mathcal{D}\partial]_k := \mathbf{D}_{k-1}\partial_k\mathbf{D}_k$. We say that SCN satisfies *orientation equivariance* if for any given $\mathcal{X}, \mathcal{D}, \mathbf{W}, \mathbf{c}_j$, we have that

$$\mathrm{SCN}_{\mathbf{W},\partial}(\mathbf{c}_j) = \mathbf{D}_\ell \mathrm{SCN}_{\mathbf{W},\mathcal{D}\partial}(\mathbf{D}_j\mathbf{c}_j). \tag{5}$$

The intuition in (5) is analogous to that in (4), but focuses on changes in orientation as opposed to relabeling. Also, notice that orientation is only defined for simplices of order at least 1, *i.e.*, edges and higher. This is due to the simple fact that the nodes of a simplicial complex naturally do not have an orientation: there is only one permutation of a singleton set. For this reason, we require $\mathbf{D}_0 = \mathbf{I}$ in Property 2.

4.3. Simplicial Awareness

The notions of permutation and orientation equivariance are fairly intuitive, corresponding to very common constructs in the analysis of graph-structured data, as well as graph neural networks in particular. However, the higher-order structures in simplicial complexes motivate architectures for data supported on simplices of different order, and regularized by hierarchically organized structure. To this end, we define the notion of *simplicial awareness*, which enforces dependence of an architecture's output on all of the boundary operators.

Property 3 (Simplicial Awareness)**.** Let $\mathrm{SCN}_{\mathbf{W},\partial} : \mathcal{C}_j \to \mathcal{C}_\ell$ and select some integer $k > 0$ such that $k \neq j$ and $k \neq \ell$. Suppose there exists simplicial complexes $\mathcal{X}$ and $\mathcal{X}'$ such that $\mathcal{X}_0 = \mathcal{X}'_0 = \mathcal{V}, \mathcal{X}_j = \mathcal{X}'_j, \mathcal{X}_\ell = \mathcal{X}'_\ell, \mathcal{X}_k \neq \mathcal{X}'_k$. Denote the respective boundary operators of $\mathcal{X}$ and $\mathcal{X}'$ by ∂ and ∂'. If there exists $\mathbf{c}_j \in \mathcal{C}_j$ and weight parameters $\mathbf{W}$ where

$$\mathrm{SCN}_{\mathbf{W},\partial}(\mathbf{c}_j) \neq \mathrm{SCN}_{\mathbf{W},\partial'}(\mathbf{c}_j), \tag{6}$$

we say that SCN satisfies *simplicial awareness of order* k. Moreover, for the set of simplicial complexes of dimension at most K, if the above is satisfied for all $k \leq K$, then we simply say that SCN satisfies *simplicial awareness*.

Put simply, simplicial awareness of order k indicates that an architecture is not independent of the k-simplices in the underlying simplicial complex. For example, consider a simplicial complex $\mathcal{X}$ composed of nodes, edges, and triangles, and $\mathrm{SCN}_{\mathbf{W},\partial} : \mathcal{C}_0 \to \mathcal{C}_0$. One can envision $\mathrm{SCN}_{\mathbf{W},\partial}$ in the form of a standard graph neural network by ignoring the triangles, but this would violate simplicial awareness of order 2.

4.4. Admissibility

We define a notion of *admissibility* that we use to guide our design of neural networks acting on chain complexes.

Definition 1. An architecture is *admissible* if it satisfies permutation equivariance, orientation equivariance, and simplicial awareness.

We define admissibility largely as a suggestion: of course, neural network architectures need not satisfy these three properties. However, much like the permutation equivariance of graph convolutional networks, enforcing the corresponding symmetries in a simplicial neural network ensures that the design is not subject to the user's choice of permutation. Enforcing the property of permutation equivariance enables graph neural networks trained on small graphs to generalize well to larger graphs, since it is not dependent on a set of hand-selected node labels (Hamilton, 2020). Indeed, by enforcing meaningful symmetries under group actions in a domain, neural architectures learn more efficient, generalizable representations (Cohen & Welling, 2016).

For the purposes of a simplicial complex, the same logic applies: the neural architecture itself should reflect the symmetries and invariances of the underlying domain, in order to promote generalization to unseen structures in a way that is not subject to design by the user. The motivation for this in the setting of simplicial complexes is highlighted by the property of orientation equivariance: since the orientation of

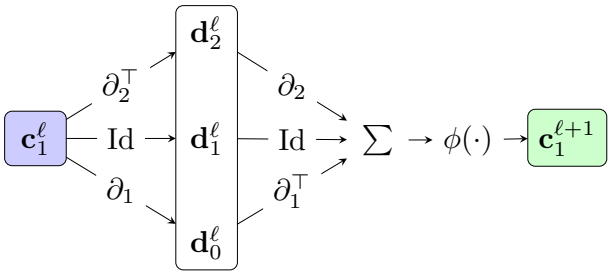


Figure 1. A single layer of SCoNe, with additional intermediate representations $\mathbf{d}_0^\ell \in \mathcal{C}_0, \mathbf{d}_1^\ell \in \mathcal{C}_1, \mathbf{d}_2^\ell \in \mathcal{C}_2$ included, to emphasize the structure of each SCoNe layer as computing over all levels of the simplicial complex.

simplices is arbitrary, we do not want to train an architecture that is dependent on the chosen orientation of the training data, since there is little hope of it working well on unseen structures without careful user-selected orientations.

Finally, simplicial awareness enforces a minimum representational capacity: if a neural architecture is incapable of incorporating information from certain structural features of the simplicial complex, one can construct vastly different datasets with the exact same output, *e.g.*, a simplicial complex that is triangle-dense, compared to its 1-skeleton.

5. Trajectory Prediction with SCoNe

We consider the task of *trajectory prediction* (Benson et al., 2016; Wu et al., 2017; Cordonnier & Loukas, 2019) for agents traveling over a simplicial complex. A *trajectory* over a simplicial complex $\mathcal{X}$ with nodes $\mathcal{V}$ is a sequence $[i_0, i_1, \ldots, i_{m-1}]$ of elements of $\mathcal{V}$, such that i_j is adjacent to i_{j+1} for all $0 \leq j < m-1$. As pointed out by Ghosh et al. (2018) and Schaub et al. (2020), such trajectories are naturally modeled through the lens of the Hodge Laplacian. In particular, a trajectory viewed as an oriented 1-chain (a linear combination of the edges of a simplicial complex) is often harmonic, *i.e.*, conservative and curl-free. This aligns with intuition: a natural walk in space will typically not backtrack on itself nor loop around points locally, and must exit most points it enters.

The trajectory prediction task considers as input a trajectory $[i_0, i_1, \ldots, i_{m-1}]$ and asks what node i_m will be. For simplicity, we do not consider the setting where a trajectory terminates, *i.e.* $i_m = i_{m-1}$.

To this end, we present a neural network architecture called SCoNe (Simplicial Complex Net) for trajectory prediction on simplicial complexes of dimension 2. We specify SCoNe as a map from $\mathcal{C}_1$ to $\mathcal{N}(i_{m-1})$ in Algorithm 1, inspired by the structure of graph convolutional networks (Bruna et al., 2014; Defferrard et al., 2016; Kipf & Welling, 2017).

Algorithm 1 SCoNe for Trajectory Prediction

Input: partial trajectory $[i_0, i_1, \ldots, i_{m-1}]$
Parameters:
boundary operators $\{\partial_k\}_{k=0}^2$
number of layers L
hidden dimensions $\{F_\ell\}_{\ell=0}^{L+1}, F_0 = F_{L+1} = 1$
weight matrices $\{\{\mathbf{W}_k^\ell \in \mathbb{R}^{F_\ell \times F_{\ell+1}}\}_{\ell=0}^L\}_{k=0}^2$
activation function ϕ
Initialize: $\mathbf{c}_1^0 \in \mathcal{C}_1, \mathbf{c}_1^0 = 0$.
for $j = 0$ **to** $m-2$ **do**
  $\mathbf{c}_1^0 \leftarrow \mathbf{c}_1^0 + [i_j, i_{j+1}]$
end for
for $\ell = 0$ **to** $L-1$ **do**
  $\mathbf{c}_1^{\ell+1} \leftarrow \phi(\partial_2 \partial_2^\top \mathbf{c}_1^\ell \mathbf{W}_2^\ell + \mathbf{c}_1^\ell \mathbf{W}_1^\ell + \partial_1^\top \partial_1 \mathbf{c}_1^\ell \mathbf{W}_0^\ell)$
end for
$\mathbf{c}_0^{L+1} \leftarrow \partial_1 \mathbf{c}_1^L \mathbf{W}_0^L$
$\mathbf{z} \leftarrow \mathrm{softmax}(\{[\mathbf{c}_0^{L+1}]_j : j \in \mathcal{N}(i_{m-1})\})$
Return: $\widehat{i}_m \leftarrow \arg\max_j z_j$

5.1. Representation of Trajectories as 1-Chains

In order to leverage the properties of boundary operators and Hodge Laplacians of simplicial complexes, the input to SCoNe needs to be a 1-chain. In particular, we lift the sequence of nodes $[i_0, i_1, \ldots, i_{m-1}]$ to a sequence of oriented edges $[[i_0, i_1], [i_1, i_2], \ldots, [i_{m-2}, i_{m-1}]]$. Then, we "collapse" the sequential structure by summing each edge in the sequence, thus yielding a 1-chain, since each oriented edge is itself a 1-chain in the vector space $\mathcal{C}_1$. Due to trajectories consisting of sequences of *adjacent* nodes, the sequential information is mostly captured by this representation in $\mathcal{C}_1$.

5.2. An Admissible Architecture for 1-Chains

Given the representation of a trajectory as a 1-chain, we now aim to predict the next step in the trajectory prediction task. This consists of a map from $\mathcal{C}_1 \to \mathcal{C}_1$, followed by a mapping to $\mathcal{C}_0$, and then a decision step dependent on the neighborhood of the node i_{m-1}.

We begin by decomposing each layer of SCoNe into two steps. First, we compute $\mathbf{c}_1^{\ell+1}$ from $\mathbf{c}_1^\ell$ as

$$\mathbf{c}_1^{\ell+1} \leftarrow \phi(\partial_2 \partial_2^\top \mathbf{c}_1^\ell \mathbf{W}_2^\ell + \mathbf{c}_1^\ell \mathbf{W}_1^\ell + \partial_1^\top \partial_1 \mathbf{c}_1^\ell \mathbf{W}_0^\ell), \quad (7)$$

where ϕ is an activation function, typically applied "elementwise" in the chosen oriented basis for $\mathcal{C}_1$: we visualize this computation in Fig. 1. We have abused notation here to allow each intermediate representation $\mathbf{c}_1^\ell$ to consist of multiple 1-chains, which are mixed via linear operations from the right via the matrices $\mathbf{W}_k^\ell$. After L such layers, we apply the boundary map ∂_1, yielding a 0-chain $\mathbf{c}_0^{L+1} = \partial_1 \mathbf{c}_1^L \mathbf{W}_0^L$. Then, a distribution over the candidate nodes is computed via the softmax operator ap-

plied to the restriction of $\mathbf{c}_0^{L+1}$ to the nodes in the neighborhood of the terminal node $\mathcal{N}(i_{m-1})$. Using sparse matrix-vector multiplication routines, the ℓ^{th} layer of SCoNe can be evaluated using $\mathcal{O}(|\mathcal{X}_1|F_\ell F_{\ell+1} + |\mathcal{X}_2| \min\{F_\ell, F_{\ell+1}\})$ operations, so that the entire architecture has a runtime of $\mathcal{O}(\sum_{\ell=0}^{L-1}(|\mathcal{X}_1|F_\ell F_{\ell+1} + |\mathcal{X}_2| \min\{F_\ell, F_{\ell+1}\}))$.[2] Moreover, the architecture of SCoNe is localized, in the sense that it computes information based only on an L-hop (simplicial) neighborhood of the terminal node, making this architecture able to work on large simplicial complexes by only operating on a localized region of interest.

Having defined SCoNe, we establish conditions under which the portion of this architecture that maps $\mathcal{C}_1 \to \mathcal{C}_0$ is admissible.[3]

Proposition 1. *Assume that the activation function ϕ is continuous and applied elementwise.* SCoNe *(Algorithm 1) is admissible only if ϕ is an odd, nonlinear function.*

Proof (Sketch). The proof of permutation equivariance for elementwise activation functions is directly analogous to the proof for graph neural networks, so we leave the details to the supplementary materials.

For continuous elementwise activation functions, we now consider the conditions for SCoNe to satisfy orientation equivariance. Since changes in orientation for a basis of $\mathcal{C}_1$ can be expressed as a *sequence* of orientation changes for individual edges, it is sufficient to study such "single-edge" transformations. For some $1 \leq j \leq m$, let $e_j' = -e_j$ be the reversal of the oriented edge e_j. Orientation equivariance for elementwise activation functions can then be written as

$$\phi(\langle \mathbf{c}_1, e_j \rangle) = \phi(-\langle \mathbf{c}_1, e_j' \rangle) = -\phi(\langle \mathbf{c}_1, e_j' \rangle). \tag{8}$$

This condition holds for all inputs if and only if ϕ is an odd function. Under these conditions, SCoNe is the composition of orientation equivariant functions, and is thus orientation equivariant itself.

Finally, we consider simplicial awareness of order 2. Suppose ϕ is an odd, linear function: it is sufficient to assume that ϕ is the identity map. Let a 1-chain $\mathbf{c}_1^0 \in \mathcal{C}_1$ be given arbitrarily. By Theorem 1, there exists $\mathbf{w} \in \mathcal{C}_0, \mathbf{x} \in \ker(\Delta_1) \subseteq \mathcal{C}_1, \mathbf{y} \in \mathcal{C}_2$ such that $\mathbf{c}_1^0 = \partial_1^\top \mathbf{w} + \mathbf{x} + \partial_2 \mathbf{y}$. Some simple algebra, coupled with Lemma 1, shows that when ϕ is the identity map, the 0-chain at the output of SCoNe is given by

$$\mathbf{c}_0^{L+1} = \partial_1 \left(\mathrm{Id} + \partial_1 \partial_1^\top\right)^L \mathbf{w}. \tag{9}$$

That is, the output of SCoNe does not depend on ∂_2, and thus fails to fulfill simplicial awareness of order 2. However, if ϕ is nonlinear, Lemma 1 does not come in to effect, since $\partial_1 \circ \phi \circ \partial_2 \neq 0$, allowing for simplicial awareness. □

A detailed proof can be found in the supplementary materials. Proposition 1 reveals the required properties of ϕ for SCoNe to be admissible. In particular, we propose the use of the hyperbolic tangent activation function $\tanh$, applied to each coefficient (in the standard basis of oriented 1-simplices) of the intermediate chains $\mathbf{c}_1^\ell$; see Fig. 1. The fact that nonlinearities are necessary to incorporate higher-order information is in line with results in Neuhäuser et al. (2020a;b), where it is shown that understanding consensus dynamics on higher-order networks must consider nonlinear behavior, lest the system be equivalently modeled as a linear process on a rescaled pairwise network. Existing works in the development of simplicial neural networks have not discussed the necessity of having odd and nonlinear activation functions in convolutional architectures. In particular, Ebli et al. (2020); Bunch et al. (2020) propose similar convolutional architectures and use ReLU activation functions. We discuss the conditions under which these architectures can be made admissible in the supplementary material.

6. Experiments

6.1. Methods

In evaluating our proposed architecture for trajectory prediction, we consider SCoNe with 3 layers, where each layer has $F_\ell = 16$ hidden features. By default, we use the $\tanh$ activation function, but we also use ReLU and sigmoid activations to compare. In training SCoNe, we minimize the cross-entropy between the softmax output $\mathbf{z}$ and the ground truth final nodes in each batch of training samples.[4]

Seeing how SCoNe consists of a map $\mathcal{C}_1 \to \mathcal{C}_1$ followed by an application of the boundary operator and a softmax function for node selection, we compare SCoNe to methods that employ other natural maps $\mathcal{C}_1 \to \mathcal{C}_1$, followed by the same selection procedure for picking a successor node. In particular, we consider the map that projects the input chain onto the kernel of the Hodge Laplacian $\ker(\Delta_1)$, as well as one that projects the input chain onto the kernel of the boundary map $\ker(\partial_1)$. The first approach reflects the hypothesis that the harmonic subspace $\ker(\Delta_1)$ [*cf.* Theorem 1] is a natural representation for trajectories (Ghosh et al., 2018; Schaub et al., 2020), and the second approach does the same while ignoring the triangular structure of the simplicial complex. We also compare to previously proposed neural net-

[2] Depending on the density of edges and triangles in $\mathcal{X}$, this can be improved in practice. We leave details to the supplementary materials.

[3] Since the final output is not a 0-chain, but an element of $\mathcal{X}_0$, admissibility of this map implies permutation and orientation *invariance* for the entire architecture, rather than equivariance.

[4] Specific hyperparameters and implementation details can be found in the supplementary material. Code available at https://github.com/nglaze00/SCoNe_GCN.

Table 1. Test accuracies for trajectory prediction task. (a) Synthetic dataset with randomly oriented edges, comparing Markov chain, RNN, harmonic projection methods, SCoNe, SCNN (Ebli et al., 2020), and S2CCNN (Bunch et al., 2020). (b) Synthetic dataset with manually oriented edges, compared across different activation functions for SCoNe. (c) Test accuracies for Ocean Drifters and Berlin trajectory datasets.

(a)

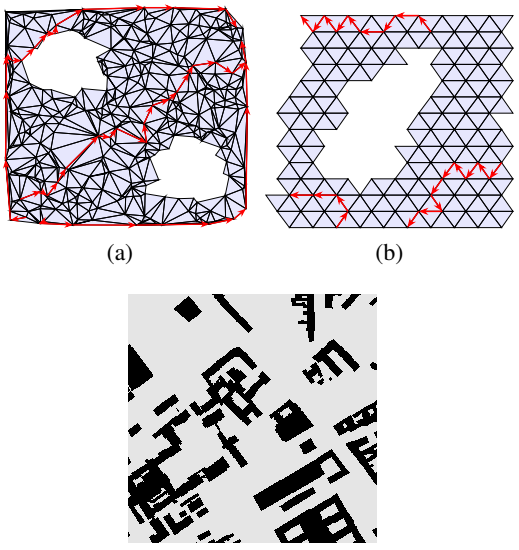
	MARKOV	RNN	$\ker(\Delta_1)$	$\ker(\partial_1)$	SCoNe tanh	SCoNe tanh, NO TRI.	SCoNe ReLU	SCoNe sigm.	SCoNe Id	SCNN	S2CCNN
STD.	0.70	**0.73**	0.55	0.32	0.69	0.55	0.64	0.59	0.31	0.64	0.62
REV.	0.24	0.01	0.58	0.21	**0.59**	0.49	0.57	0.57	0.33	0.48	0.47
TRA.	–	–	0.58	0.40	**0.61**	0.58	0.56	0.53	0.44	0.42	0.57

(b)

	tanh	ReLU	sigm.	Id.
STD.	0.65	0.65	**0.66**	0.27
REV.	**0.63**	0.24	0.10	0.31

(c)

	MARKOV	RNN	$\ker(\Delta_1)$	SCoNe	SCNN	S2CCNN
OCEAN	0.45	0.44	0.45	**0.50**	0.18	0.38
BERLIN	0.76	0.79	0.50	**0.92**	0.85	0.88



(a) (b) (c)

Figure 2. Complexes used for evaluating SCoNe, with sample trajectories from each dataset. (a) Synthetic example, where 400 random points in the unit square have been triangulated, followed by the removal of points from two regions. This yields a simplicial complex where $\dim\ker(\Delta_1) = 2$. (b) Ocean drifters example, where the large hole (Madagascar) yields $\dim\ker(\Delta_1) = 1$. (c) Berlin map example, where each darkly shaded region is an obstacle, yielding a hole in the underlying cubical complex. Figure from (Sturtevant, 2012).

works for simplicial data: SCNN (Ebli et al., 2020), and S2CCNN (Bunch et al., 2020), using leaky ReLU or ReLU nonlinearities as done by the respective authors.

As a baseline, we consider two methods rooted in learning specific sequences, rather than a general rule parameterized by operators underpinning the supporting domain. First, we evaluate a simple Markov chain approach where, at each node, we choose its successor based on the empirically most likely successor in the training set. Second, we apply the RNN model of Wu et al. (2017), with the adjustment of not including geometric coordinates, since we consider trajectories over abstract simplicial complexes, not assuming any geometric structure underlying it. We emphasize that these methods are both incapable of generalizing to unseen structures, since they depend on learning decision rules for each node based on training.

6.2. Synthetic Dataset

Dataset. Following the example of Schaub et al. (2020), we generate a simplicial complex by drawing 400 points uniformly at random in the unit square, and then applying a Delaunay triangulation to obtain a mesh, after which we remove all nodes and edges in two regions, pictured in Fig. 2(a). Then, to generate a set of trajectories, we choose a random starting point in the lower-left corner, connect it via shortest path to a random point in the upper-left, center, or lower-right region, which we then connect via shortest path to a random point in the upper-right corner. Some examples of such trajectories are shown in Fig. 2(a). We generate 1000 such trajectories for our experiment, using 800 of them for training and 200 for testing.

Performance. To start, we evaluate the performance of all methods on the standard train/test split. As shown in Table 1(a), the RNN model performs the best, since the volume of training examples allows for the model to easily learn commonly taken paths, thus leading to good performance on the test set. Compared to this, the kernel projection methods do worse on the test set, but do far better than random guessing. This is due to the natural interpretation of trajectories as being characterized by the harmonic subspace of the

Hodge Laplacian (Ghosh et al., 2018; Schaub et al., 2020). Moreover, projecting onto the kernel of the Hodge Laplacian $\ker(\Delta_1)$ significantly outperforms only projecting onto the kernel of the boundary map $\ker(\partial_1)$. This indicates that incorporating information from $\mathcal{X}_2$ (triangles) is important for this problem, since it allows the harmonic subspace to capture more interesting homological structure.

Finally, we evaluate SCoNe using different nonlinear activation functions and incorporation of simplicial information. When using $\tanh$ activation functions, it is clear that including the triangles in the model improves performance, in line with what we observed for the kernel projection methods. Moreover, we see that using the $\tanh$ activation function over the sigmoid or ReLU activation functions also improves performance, presumably due to the fact that $\tanh$ is odd, as required for admissibility in Proposition 1. Similarly, although the identity (Id.) activation function is odd, it is linear, and thus does not satisfy Proposition 1. This leads to poor performance in all experiments. We also observe that SCoNe tends to outperform the SCNN and S2CCNN models, perhaps due to the extra regularization imposed by admissibility.

Testing Generalization. We demonstrate the generalization properties of SCoNe in two ways, with the common feature of manipulating the training and test sets in order for them to have a mismatch in their characteristics.

First, we evaluate these methods on a "reversed" test set. That is, we keep the training set the same as before, but reverse the direction of the trajectories in the test set. Therefore, a method that is overly dependent on "memorizing" a particular direction for the trajectories will be expected to fare poorly, while methods that leverage more fundamental features relating to the homology of the simplicial complex are expected to perform better. Based on the results in Table 1(a), we see two things: including triangles in the architecture improves performance, and admissible methods outperform inadmissible methods. In particular, SCoNe using $\tanh$ activation and incorporating triangles performs similarly to the projection onto $\ker(\Delta_1)$, and both of these methods outperform competing approaches.

Second, we evaluate how well SCoNe generalizes to trajectories over unseen simplicial structures. To do this, we restrict the training set to trajectories running along the upper-left region of $\mathcal{X}$, and similarly restrict the testing set to trajectories spanning the lower-right region of $\mathcal{X}$. Although in this case SCoNe is still being applied to the same simplicial complex that it was trained on, the locality of the architecture means that the testing set is essentially an unseen structure. We see again in Table 1(a) that SCoNe with $\tanh$ activation outperforms sigmoid and ReLU activations, illustrating the utility of admissibility for designing architectures that generalize well. Note that the Markov chain and the RNN cannot be tested on unseen data.

Finally, to demonstrate the sensitivity of architectures that are not orientation equivariant, we repeat these experiments on the same dataset, except with edge orientations selected carefully in order to reflect the general direction of the training set. That is, using the geometric position of the nodes in Fig. 2(a), we label each node based on the sum of its x and y coordinates, and then orient each edge to be increasing with respect to the ordering of the nodes. This yields a set of oriented edges that "point" from the lower-left of the simplicial complex to the upper-right. Since the training set consists of trajectories that also follow this direction, the coefficients in the representation of each trajectory as a vector in $\mathcal{C}_1$ will be overwhelmingly non-negative.

By manipulating the edge orientation in this way, we have artificially introduced a rule that would work well for the training set, but violates orientation equivariance. Since architectures that violate orientation equivariance are capable of learning such rules, we see in Table 1(b) that the ReLU and sigmoidal architectures perform similarly to the admissible $\tanh$ architecture when the test set matches this rule, with the sigmoidal architecture performing marginally better than the others. However, as soon as the data does not follow this rule, as in the "reversed" case, the non-admissible architectures markedly fail, with the sigmoidal and ReLU activations not satisfying orientation equivariance, and the identity activation not satisfying simplicial awareness.

6.3. Real Data

Datasets. We consider the trajectory prediction problem for the Global Drifter Program dataset, localized around Madagascar.[5] This dataset consists of buoys whose coordinates are logged every 12 hours. We tile the ocean around Madagascar with hexagons and consider the trajectory of buoys based on their presence in hexagonal tiles at each logged moment in time, following the methodology of Schaub et al. (2020). Treating each tile as a node, drawing an edge between adjacent tiles, and filling in all such planar triangles yields a natural simplicial structure, as pictured in Fig. 2(b). Indeed, the homology of the complex shows a large hole, corresponding to the island of Madagascar.

Additionally, we consider a map of a section of Berlin (Sturtevant, 2012), where each point on a grid is designated impassible or passible based on the presence of an obstacle, as pictured in Fig. 2(c). For a set of trajectories, we consider a set of 1000 shortest paths between random pairs of points in the largest connected components, divided into an $80/20$ train/test split. Since the geometry of this map is given as a grid, it is naturally modeled as a *cubical*

[5] Data available from NOAA/AOML at http://www.aoml.noaa.gov/envids/gld/ and as supplementary material.

complex, rather than a simplicial complex. The boundary operators in this setting are quite similar to the simplicial case, highlighting the flexibility of our proposed architecture for more general chain complexes. We discuss the details of this in the supplemental material.

Results. Across the competing methods, SCoNe yields the best trajectory prediction, as shown in Table 1(c). Indeed, by using admissibility as a guiding principle in designing architectures that respect the symmetries of the underlying chain complex, as well as ensuring complete integration of the simplicial structure, we achieve greater prediction accuracy than other methods. Interestingly, we again outperform the method of projecting onto $\ker(\Delta_1)$, which is itself an admissible approach, suggesting that there is utility in the regularization imposed by the successive local aggregations of SCoNe, and in appropriate weighting of the different components of the Hodge Laplacian.

7. Conclusion

A core component of graph neural networks is their equivariance to permutations of the nodes of the graph on which they act. Designing architectures that respect properties such as this enables the design of systems that transfer and scale well. By considering additional symmetries (orientation equivariance) and truly accounting for higher-order structures (simplicial awareness), we construct principled, generalizable neural networks for data supported on simplicial complexes.

Acknowledgements

This work was partially supported by NSF under award CCF-2008555. Research was sponsored by the Army Research Office and was accomplished under Cooperative Agreement Number W911NF-19-2-0269. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. TMR was partially supported by the Ken Kennedy Institute 2020/21 Exxon-Mobil Graduate Fellowship. We would like to thank Michael Schaub for helpfully providing a cleaned dataset for the Ocean Drifters experiment.

References

Barbarossa, S. and Sardellitti, S. Topological signal processing over simplicial complexes. *IEEE Transactions on Signal Processing*, 68:2992–3007, 2020a. doi:10.1109/TSP.2020.2981920.

Barbarossa, S. and Sardellitti, S. Topological signal processing: Making sense of data building on multiway relations. *IEEE Signal Processing Magazine*, 37(6):174–183, 2020b. doi:10.1109/MSP.2020.3014067.

Barbarossa, S., Sardellitti, S., and Ceci, E. Learning from signals defined over simplicial complexes. In *2018 IEEE Data Science Workshop (DSW)*, pp. 51–55. IEEE, 2018. doi:10.1109/DSW.2018.8439885.

Benson, A. R., Kumar, R., and Tomkins, A. Modeling user consumption sequences. In *International Conference on World Wide Web*, pp. 519–529, 2016. doi:10.1145/2872427.2883024.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations*, 2014.

Bunch, E., You, Q., Fung, G., and Singh, V. Simplicial 2-complex convolutional neural networks. In *NeurIPS Workshop on Topological Data Analysis and Beyond*, 2020.

Carlsson, G. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009. doi:10.1090/S0273-0979-09-01249-X.

Chung, F. R. *Spectral graph theory*. Number 92. American Mathematical Society, 1997. doi:10.1090/cbms/092.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *International Conference on Machine Learning*, pp. 2990–2999. PMLR, 2016.

Cordonnier, J.-B. and Loukas, A. Extrapolating paths with graph neural networks. In *International Joint Conference on Artificial Intelligence*, 2019. doi:10.24963/ijcai.2019/303.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems*, 29:3844–3852, 2016.

Ebli, S., Defferrard, M., and Spreemann, G. Simplicial neural networks. In *NeurIPS Workshop on Topological Data Analysis and Beyond*, 2020.

Ghosh, A., Rozemberczki, B., Ramamoorthy, S., and Sarkar, R. Topological signatures for fast mobility analysis. In *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 159–168, 2018. doi:10.1145/3274895.3274952.

Ghrist, R. W. *Elementary applied topology*, volume 1. Createspace Seattle, 2014.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

Hamilton, W. L. *Graph representation learning*, volume 14. Morgan & Claypool Publishers, 2020. doi:10.2200/S01045ED1V01Y202009AIM046.

Hatcher, A. *Algebraic topology*. Cambridge University Press, 2002.

Jia, J., Schaub, M. T., Segarra, S., and Benson, A. R. Graph-based semi-supervised & active learning for edge flows. In *ACM International Conference on Knowledge Discovery and Data Mining*, pp. 761–771, 2019. doi:10.1145/3292500.3330872.

Jiang, X., Lim, L.-H., Yao, Y., and Ye, Y. Statistical ranking and combinatorial Hodge theory. *Mathematical Programming*, 127(1):203–244, 2011. doi:10.1007/s10107-010-0419-x.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Lim, L.-H. Hodge Laplacians on graphs. *SIAM Review*, 62 (3):685–715, 2020. doi:10.1137/18M1223101.

Neuhäuser, L., Mellor, A., and Lambiotte, R. Multibody interactions and nonlinear consensus dynamics on networked systems. *Physical Review E*, 101(3):032310, 2020a. doi:10.1103/PhysRevE.101.032310.

Neuhäuser, L., Schaub, M. T., Mellor, A., and Lambiotte, R. Opinion dynamics with multi-body interactions. *arXiv preprint*, 2020b, arXiv:2004.00901.

Roddenberry, T. M. and Segarra, S. HodgeNet: Graph neural networks for edge data. In *Asilomar Conference on Signals, Systems, and Computers*, pp. 220–224. IEEE, 2019. doi:10.1109/IEEECONF44664.2019.9049000.

Schaub, M. T. and Segarra, S. Flow smoothing and denoising: Graph signal processing in the edge-space. In *IEEE Global Conference on Signal and Information Processing*, pp. 735–739. IEEE, 2018. doi:10.1109/GlobalSIP.2018.8646701.

Schaub, M. T., Benson, A. R., Horn, P., Lippner, G., and Jadbabaie, A. Random walks on simplicial complexes and the normalized Hodge 1-Laplacian. *SIAM Review*, 62(2):353–391, 2020. doi:10.1137/18M1201019.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, Apr. 2013. doi:10.1109/MSP.2012.2235192.

Sturtevant, N. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games*, 4(2):144 – 148, 2012. doi:10.1109/TCIAIG.2012.2197681. URL http://web.cs.du.edu/~sturtevant/papers/benchmarks.pdf.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Wu, H., Chen, Z., Sun, W., Zheng, B., and Wang, W. Modeling trajectories with recurrent neural networks. In *International Joint Conference on Artificial Intelligence*, 2017. doi:10.24963/ijcai.2017/430.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020. doi:10.1109/TNNLS.2020.2978386.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 5165–5175, 2018.