Faster Inference of Cell Complexes from Flows via Matrix Factorization

1st Til Spreuer
RWTH Aachen University
Aachen, Germany
til.spreuer@rwth-aachen.de

2nd Josef Hoppe
RWTH Aachen University
Aachen, Germany
orcid.org/0000-0003-4383-7049

3rd Michael T. Schaub
RWTH Aachen University
Aachen, Germany
orcid.org/0000-0003-2426-6404

***Abstract*—We consider the following inference problem: Given a set of edge-flow signals observed on a graph, lift the graph to a cell complex, such that the observed edge-flow signals can be represented as a sparse combination of gradient and curl flows on the cell complex. Specifically, we aim to augment the observed graph by a set of 2-cells (polygons encircled by closed, non-intersecting paths), such that the eigenvectors of the Hodge Laplacian of the associated cell complex provide a sparse, interpretable representation of the observed edge flows on the graph. As it has been shown that the general problem is NP-hard in prior work, we here develop a novel matrix-factorization-based heuristic to solve the problem. Using computational experiments, we demonstrate that our new approach is significantly less computationally expensive than prior heuristics, while achieving only marginally worse performance in most settings. In fact, we find that for specifically noisy settings, our new approach outperforms the previous state of the art in both solution quality and computational speed.**

***Index Terms*—Topological signal processing, graph signal processing, cell inference, cell complex, edge flows**

I. Introduction

Graphs have become a prevalent abstraction in data science due to their ability to model many real-life systems [1]. Graph signal processing (GSP) [2], [3] enables signal processing for signals that are defined on such graphs, such as temperatures measured at different locations or neuron activity in different parts of the brain. However, in many applications, the recorded data represents flows, e.g. of people [4], traffic [5], or money [6]. Such flows are often more naturally represented with an (oriented) signal on the edges of a graph. To process such edge signals, recent extensions of GSP towards topological signal processing (TSP) [7]–[12] utilize simplicial complexes or cell complexes, and their associated Hodge Laplacians.

These Hodge Laplacian operators, a generalization of the graph Laplacian to higher dimensions, are a central pillar of TSP, and can be used in lieu of the graph Laplacian as shift operators in signal processing tasks for signals defined on the edges or higher-dimensional cells of a complex. Importantly, the 1-Hodge Laplacian induces a decomposition of the space of edge flows into gradient, curl, and harmonic flows [13], [14]. Similarly to the graph Laplacian in GSP, the eigenvalues and corresponding eigenspaces of this Hodge Laplacian enable the construction of low- and high-pass filters, denoising, signal compression, and other classical signal processing tasks [9].

However, the higher-order cell structure of the complexes used to construct these Hodge-Laplacians is typically not readily available. Specifically, when considering edge-flow data, commonly only the underlying graph structure and the associated edge flows are observable. Therefore, analogously to inferring a graph structure from data on the nodes [3], different methods to infer simplicial complex from edge data on a graph have been proposed [8], [15]. The general problem of inferring a cell complex from such data, i.e., finding an optimal set of 2-cells (polygons) such that the edge-flows can be represented by a sparse set of eigenvectors of the Hodge Laplacian, was introduced in [16]. It was shown that this problem is NP-hard, and thus [16] introduced a heuristic algorithm to solve it.

Contribution. In this paper, we introduce an alternative approach to solving the cell inference problem using matrix decompositions. As we show, our novel approach is significantly faster and incurs only a marginally higher approximation error than the originally proposed heuristic. In our experiments, this trade-off is particularly advantageous on larger networks, when inferring a large absolute number of 2-cells, and in the presence of high noise.

II. Background

This introduction to 2-dimensional cell complexes is inspired from [17].

We consider 2-dimensional complexes constructed by augmenting simple undirected graphs as follows. Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a set of vertices $\mathcal{V} = \{v_1, \dots, v_n\}$ and a set of edges by $\mathcal{E} = \{e_1, \dots, e_m\}$. Each edge e_k consists of a pair of vertices $e = (v_i, v_j)$, with arbitrary but fixed order. We call v_i the *source* and v_j the *target* of e_k. We encode the structure of the graph in an incidence matrix $\mathbf{B}_1 \in \{0, \pm 1\}^{n \times m}$. For every edge e_k, oriented from v_i to v_j, we have $(\mathbf{B}_1)_{i,k} = 1$ and $(\mathbf{B}_1)_{j,k} = -1$, and $(\mathbf{B}_1)_{-,k} = 0$ otherwise.

Definition 1 (Simple cell complexes of dimension 2)**.** An abstract regular cell complex $\mathcal{C}$ of dimension 2 consists of

Funded by the European Union (ERC, HIGH-HOPeS, 101039827). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. Parameter search was performed with computing resources granted by RWTH Aachen University under project *thes1725*.

Evaluation code: https://github.com/til-spreuer/mfci-workflow a graph $\mathcal{G}$, and a non-empty ordered set of 2-cells (polygons) $\mathcal{C}_2$, encoded via a boundary matrix $\mathbf{B}_2 \in \{0, \pm 1\}^{m \times |\mathcal{C}_2|}$. Specifically for every 2-cell $\theta \in \mathcal{C}_2$, we have a set of edges $\{e_1, \ldots, e_m\}$ corresponding to a simple cycle in $\mathcal{G}$ which forms the boundary of θ. Each column of $\mathbf{B}_2$ corresponds to such a cycle and the entries of the associated edges are set to ± 1 such that $\mathbf{B}_1 \mathbf{B}_2 = 0$.

Remark 2 (Orientation). Note that the above construction amounts to equipping each edge and each polygon with a reference orientation. This built-in notion of *orientation* allows for a natural representation of physical data like flows via positive or negative values (with or opposite to the orientation).

Oriented signals on CCs can be represented as *chains*:

Definition 3 (Signal (or chain) space of a cell complex)**.** Given a cell complex $\mathcal{C}$, we denote by $C_k = \mathbb{R}^{|\mathcal{C}_k|}$ the k-th signal space of $\mathcal{C}$ and obtain an associated sequence of signal spaces

$$C_0 \xleftarrow[\mathbf{B}_1]{} C_1 \xleftarrow[\mathbf{B}_2]{} C_2$$

The boundary matrices $\mathbf{B}_k$ are linear maps between the signal spaces C_k. In this paper, we analyze flows from the signal space on edges C_1.

Hodge Laplacian and Hodge decomposition. The Hodge Laplacian $\mathbf{L}_k = \mathbf{B}_k^\top \mathbf{B}_k + \mathbf{B}_{k+1} \mathbf{B}_{k+1}^\top$ is a generalization of the Graph Laplacian [13], [14], [18]. The Hodge decomposition divides the space of edge flows C_1 into three eigenspaces associate to $\mathbf{L}_1$: The *gradient* space $\operatorname{Im} \mathbf{B}_1^\top$, the *curl* space $\operatorname{Im} \mathbf{B}_2$, and the *harmonic* space $\ker \mathbf{B}_1^\top \cap \ker \mathbf{B}_2$. Intuitively, the gradient space consists of flows based on the difference between potentials on the nodes and the curl space consists of flows around the boundaries of 2-cells.

For a flow $\mathbf{f} \in C_1$ on a cell complex $\mathcal{C}$, we define the gradient flow $\operatorname{grad}_{\mathcal{C}}(\mathbf{f}) = \mathbf{B}_1^\top (\mathbf{B}_1^\top)^\dagger \mathbf{f}$, curl flow $\operatorname{curl}_{\mathcal{C}}(\mathbf{f}) = \mathbf{B}_2 (\mathbf{B}_2)^\dagger \mathbf{f}$, and harmonic flow $\operatorname{harm}_{\mathcal{C}}(\mathbf{f}) = (\mathbf{I} - \mathbf{L}_1 (\mathbf{L}_1)^\dagger) \mathbf{f}$. $(\cdot)^\dagger$ denotes the Moore-Penrose pseudoinverse.

III. The Cell Inference Problem

Given a graph and a set of observed flows, the cell inference problem is to find a sparse set of 2-cells that minimize the projection of the flows into the harmonic space when added. We denote the given graph by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and the $s \in \mathbb{N}$ sampled edge flows by $\mathbf{f}_i \in C_1$ for $i \in \{1, \ldots, s\}$. In the following, we use $\mathbf{F}$ to denote the matrix of all flows $\mathbf{F} = [\mathbf{f}_1, \ldots, \mathbf{f}_s]$.

More formally, the problem can be modeled as an optimization problem: find a 2-dimensional cell complex $\mathcal{C}$ which minimizes the harmonic projection of the flows $\mathbf{F}$ onto the harmonic space of $\mathcal{C}$.:

$$\mathcal{L}(\mathcal{C}, \mathbf{F}) = \|\operatorname{harm}_{\mathcal{C}}(\mathbf{F})\|_F = \left(\sum_{i=1}^{s} ||\operatorname{harm}_{\mathcal{C}}(\mathbf{f}_i)||_2^2 \right)^{1/2} \quad (4)$$

As minimizing this loss without constraints on the number of cells would result in a $\mathcal{C}$ with an empty harmonic space, we formulate the minimization problem such that the number of 2-cells is constrained:

$$\min_{\mathcal{C}} \mathcal{L}(\mathcal{C}, \mathbf{F}) \quad \text{s.t.} \quad \mathcal{C} \text{ has } \mathcal{G} \text{ as 1-skeleton and } |\mathcal{C}_2| \leq k \quad (5)$$

Without loss of generality, we will in the following assume that the flows $\mathbf{F}$ are gradient free, as any gradient component will not alter the optimal $\mathcal{C}$ in our optimization problem.

IV. Methods

As proposed in [16], we can proceed in an iterative fashion to solve the cell inference problem. We start with (i) a cell complex $\mathcal{C}^{(0)}$ equivalent to $\mathcal{G}$, (ii) the number of cells to add per iteration l', and (iii) the number l of cell candidates to consider, with $l' \leq l$. In each iteration i, we add l' new 2-cells $\theta_1^{(i)}, ..., \theta_{l'}^{(i)}$ until the desired amount of 2-cells are inferred:

$$\mathcal{C}^{(i)} \leftarrow \mathcal{C}^{(i-1)} \cup \{\theta_j^{(i)} : j \in [1, \ldots, l']\} \quad (6)$$

To operationalize this algorithmic idea, we have to address two questions. First, how do we obtain possible cell candidates $\theta_1^{(i)}, ..., \theta_l^{(i)}$ in each iteration? Second, how do we evaluate the candidates in each iteration and select the best?

In the original work [16], a Spanning Tree Heuristic (SPH) was proposed that constructs candidate 2-cells from one or multiple constructed spanning trees. The candidate cells that (when added to the cell complex) maximally decrease the loss are then selected greedily. This requires computing several harmonic projections of an (augmented) cell complex per iteration, which is the computationally most expensive part of the algorithm (even though efficient algorithms like LSMR [19] can be used to compute the projections).

There is also a more subtle issue associated with the SPH approach, that leads to many iterations required to infer appropriate cells. When using spanning trees to generate candidate cells, each candidate cell contains exactly one edge outside the spanning tree, and at least two edges from the tree. Importantly, candidate cells that significantly lower the loss are likely to share edges in the spanning tree. Adding one of those cells to the complex will account for the flow on all the shared spanning tree edges. The other candidates are thus unlikely to decrease the loss significantly, as they share many edges with the already added cell. Hence, in [16], only *one* cell is added in each iteration.

Matrix-factorization-based approach

In the following we present a novel, modular framework called Matrix-Factorization-Cell-Inference (MFCI) (cf. Figure 1, Algorithm 1) to solve the cell inference problem. For this, we interpret Equation (5) as a problem to find a matrix $\mathbf{B}_2$ that minimizes:

$$\min_{\mathbf{B}_2, \mathbf{C}} \|\mathbf{F} - \mathbf{B}_2 \mathbf{C}\| \quad \text{s.t.} \quad \mathbf{B}_2 \in \mathcal{B}_2^{(k')}, \ \mathbf{C} \in \mathbb{R}^{k' \times s} \quad (7)$$

where $\mathcal{B}_2^{(k')}$ is the set of valid edge-to-cell boundary matrices of cell complexes with $k' \leq k$ 2-cells and a 1-skeleton given by $\mathcal{G}$; $\mathbf{C}$ is a matrix of real coefficients. Given a matrix $\mathbf{B}_2$, an optimal $\mathbf{C}$ can be obtained by solving a least squares problem.

Our key idea now is to first find a low-rank approximation of the flows $\mathbf{F} \approx \mathbf{B} \cdot \mathbf{C}$ using a matrix factorization, without considering the constraints on $\mathbf{B}$; and then discretize the columns of $\mathbf{B}$ to correspond to valid boundary vectors of 2-cells. This

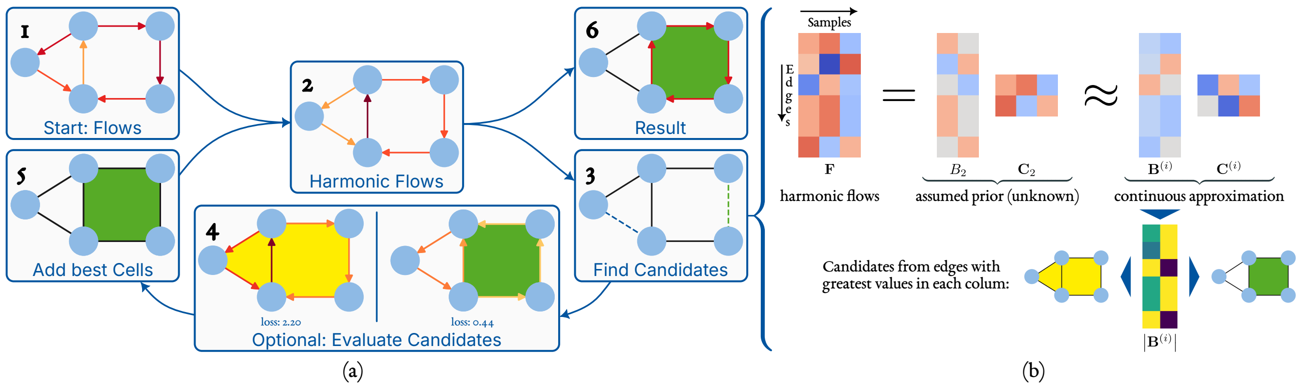

Fig. 1. Overview of the complete inference approach. **(a)** shows an overview of our Cell Inference Algorithm, adapted and modified from [16]. Like the original algorithm, our matrix-based alternative takes a graph with flows on its edges as an input (1) and iteratively adds 2-cells. Both approaches project the flows into the harmonic space (2). Our approach introduces a matrix-factorization-based heuristic for finding candidates (3) that also makes the individual evaluation of the candidates (4) optional in practice. Furthermore, our approach can add multiple cells (instead of one) in each iteration (5). **(b)** shows the concept of the novel candidate heuristic in MFCI. The harmonic flows are decomposed into two matrices, resembling a relaxed version of flows generated by a boundary matrix. Thus, the left matrix can be discretized to one cell candidate per column. Pseudocode of steps 2-5 is given in Algorithm 1

Algorithm 1 One Iteration of Candidate Search for MFCI

```
Input: cell complex CC, the amount of desired 2-cells $l \in \mathbb{N}_+$
Output: a set of $k$ 2-cells
1: Compute the matrix factorization $\mathbf{B} \cdot \mathbf{C} \approx \mathrm{harm}_{CC}(\mathbf{F})$
2: Extract the most promising row vectors $b_1, \ldots, b_l$ from $\mathbf{B}$
3: Discretize $b_1, \ldots, b_l$ to 2-cells $\theta_1, \ldots, \theta_l$
4: Return: $\{\theta_1, \ldots, \theta_l\}$
```

idea can be iteratively applied several times, adding multiple cells in each iteration. To create a practical algorithm from this idea we need to choose a suitable matrix factorization method, a heuristic to find cell candidates from this factorization, and a procedure to evaluate and select the best candidates. As a secondary consideration, we also consider an efficient approximation for calculating the harmonic flows.

Calculating (approximate) harmonic flows. In each step i, SPH calculates the harmonic flows $\mathbf{H}^{(i)} := \mathrm{harm}_{\mathcal{C}^{(i)}}(\mathbf{F})$. To increase speed, MFCI can (optionally) use an approximate update of the harmonic flow instead. Consider that, in sparse configurations, there are few overlaps between 2-cells from different iterations. Thus, the change in harmonic flow depends mostly on the boundaries $\hat{\mathbf{b}}_1^{(i)}, \ldots, \hat{\mathbf{b}}_{l'}^{(i)}$ of the chosen cells. With $\mathbf{H}^{(0)} = \mathbf{F}$, we can approximate the harmonic flow as:

$$\mathbf{H}^{(i)} \leftarrow \mathbf{H}^{(i-1)} - \left[\hat{\mathbf{b}}_1^{(i)} \cdots \hat{\mathbf{b}}_{l'}^{(i)}\right] \cdot \left[\hat{\mathbf{b}}_1^{(i)} \cdots \hat{\mathbf{b}}_{l'}^{(i)}\right]^{\dagger} \cdot \mathbf{B}^{(i)} \mathbf{C}^{(i)} \quad (8)$$

Matrix factorization. To find a suitable matrix factorization $\mathbf{F} \approx \mathbf{B} \cdot \mathbf{C}$ for our application, let us first collect some desirable properties for this factorization. First, the approximation error $\|\mathbf{F} - \mathbf{B} \cdot \mathbf{C}\|$ should be low. To make the discretization of the columns of $\mathbf{B}$ to 2-cells easy, the entries of $\mathbf{B}$ should also almost fulfill the properties of a boundary matrix already: First, the entries of $\mathbf{B}$ should be 0 or ± 1. Second, the boundary of $\mathbf{B}$ should be zero: $\mathbf{B}_1 \mathbf{B} = 0$.

The first possible factorization we consider is a (truncated) Singular Value Decomposition (SVD) of $\mathbf{F}$, which is the optimal low-rank approximation. However, the entries are typically not close to 0 or ± 1. Hence, we also consider Independent Component Analysis (ICA) [20], which separates a matrix into statistically independent components that are not necessarily orthogonal. In practice, this is often closer to the desired decomposition into a boundary matrix $\mathbf{B}_2$ and associated circulations around 2-cells.

Obtaining 2-cell candidates. We apply the chosen matrix factorization strategy iteratively to find 2-cell candidates. In each iteration, we calculate a low-rank matrix approximation $\mathbf{B}^{(i)} \cdot \mathbf{C}^{(i)} \approx \mathbf{H}^{(i-1)}$ (e.g. using the SVD). In each iteration, we first calculate the approximation error of each column of the matrix $\mathbf{B}^{(i)}$ as $\|\mathbf{H}^{(i-1)} - \mathbf{B}_{-,j}^{(i)} \mathbf{C}_{j,-}^{(i)}\|_1$, where $\|A\|_1 = \sum_{i,j} |A_{i,j}|$ denotes the L_1 norm. We keep the l columns $\mathbf{b}_1^{(i)}, \ldots, \mathbf{b}_l^{(i)}$ of $\mathbf{B}^{(i)}$ with the lowest approximation error and discretize them to valid 2-cell candidates using one of two heuristics:

- The *deterministic heuristic* creates a candidate by adding edges in decreasing order of their absolute values in $\mathbf{b}_j^{(i)}$ to an empty graph until a (unique) cycle is formed.
- The *random-walk* based heuristic simulates a random walker, using the corresponding values from chosen vectors of the matrix factorization $\mathbf{b}_j^{(i)}$ as weights for the probability distribution. Once the random walk forms a cycle, this cycle is returned as the candidate. Each cycle identifies a 2-cell $\theta_j'^{(i)}$.

We add the l' best candidates to the cell complex (see Equation (6)); for $l = l'$, we skip the evaluation.

Our approach infers a 2-cell candidate for each column of the approximation. The number of 2-cell additions in each iteration thus depends on the number of chosen columns, and the rank of the matrix approximation. Since the rank of $\mathbf{F}$ is bounded by s, multiple cells can only be added for $s > 1$. As with a larger number of samples s, the low-rank approximation is more likely to eliminate noise in the observed flow data, we expect MFCI to be most useful when many flows are observed.

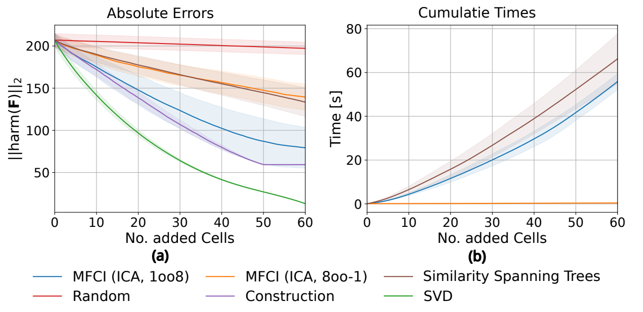

Fig. 2. MFCI with deterministic candidate heuristic and SPH on an Erdős–Rényi with $n = 40$, $p = 0.9$, 50 sampled 2-cells, 64 flows and edge noise with $\sigma = 0.3$. "1oo8" means that the best cell of 8 candidates got added each iteration and "8oo-1" adds all 8 cells that get inferred each iteration. In **(a)** the SVD is the theoretical mathematical optimum. In **(b)** SVD, Construction and Random have no corresponding time, because they are only shown for reference.

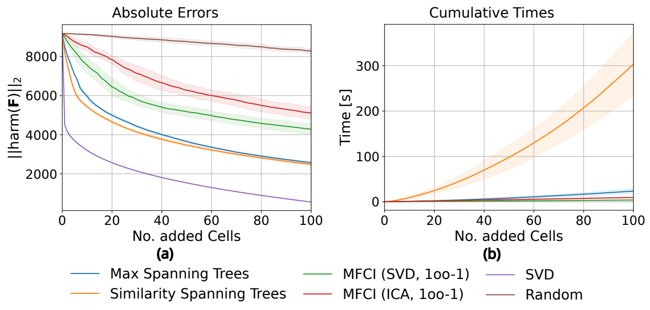

Fig. 3. MFCI (random walk heuristic) vs SPH on Taxi Set with 128 flows. The Max Spanning Trees only evaluates 1 candidate per iteration. The SVD shows the approximation loss of $\mathbf{F}$ using the mathematical optimal approximation without constraints on the matrices as required by problem 7. In subfigure **(a)**, we see the size of the projected harmonic flow depending on the added 2-cells. In subfigure **(b)**, the cumulative times are presented.

V. Experiments

For the empirical evaluation, we compare both the approximation error and the compute time of our new approach to the state-of-the-art method SPH[1] [16] and the non-discretized SVD, which is the optimal continuous solution. We choose the rank of the matrix factorization to be equal to the number of candidates l.

We generate synthetic random CCs using the algorithm from [21]. We sample flow signals $\mathbf{f}_i = \mathbf{B}_2 c_i + \mathbf{f}_i'$, where $c_i \in C_2$ is a signal obtained from an i.i.d. gaussian distribution on all 2-cells and $\mathbf{f}_i' \in C_1$ is i.i.d. gaussian noise on the edges.

We also consider real-world data based on taxi trips in New York City [22], [23]. The transitions between neighborhoods in a driver's trajectory are represented as a directed flow. These flows are then aggregated into s flows via summation.

Synthetic Data. Figure 2 shows the approximation errors and compute times on synthetic data. As the ICA-based factorization approach outperforms the SVD-based approach in both approximation error and compute time, we only show the former. For the most significant speed-up, we skip evaluation of candidates and use the approximate update of the harmonic flow. With this configuration (denoted "8oo-1" in Figure 2), no calls to LSMR are required. Thus, the computation is fast while the approximation error is comparable to SPH. By evaluating candidates and only adding the best in each iteration (like SPH does), MFCI outperforms SPH in terms of approximation error, with a similar (if slightly larger) computational requirement. In fact, the approximation error is close to the ground truth of the sampled cells.

Real-World Data. In Figure 3, we compare MFCI to SPH on the taxi dataset. The maximum spanning tree heuristic is configured to only evaluate one candidate per iteration, which requires only one call to LSMR.

In contrast to the synthetic data, the SPH heuristics are more accurate than MFCI. Furthermore, in MFCI, using the SVD significantly outperforms the ICA, which is just slightly better than the random baseline. However, it is also notable that, after 40 added 2-cells, the difference in error between SPH and MFCI stays approximately constant. The computational times behave similarly to the synthetic data, with MFCI (without candidate evaluation) being significantly faster than SPH.

[1]Unless stated otherwise, we use the *similarity* SPH heuristic with 11 clusters and 11 cell candidates like in the original paper. In our experiments, a larger number did not significantly improve accuracy.

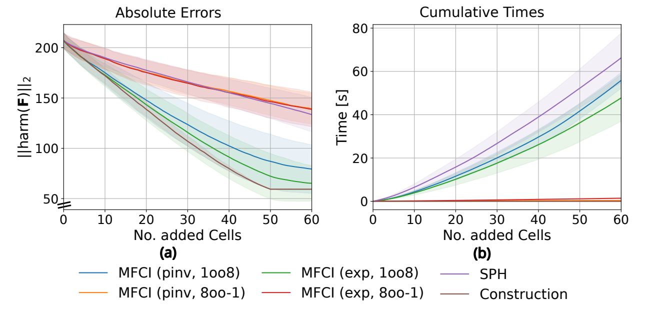

Fig. 4. Comparison of using explicit LSMR projection (exp) and the approximation via pseudoinverse (pinv) to calculate the remaining harmonic flow. Candidates obtained using the deterministic heuristic.

Harmonic Flow Approximation. We evaluate the usefulness of the proposed approximate calculation of the harmonic flow using the pseudoinverse. Figure 4 compares the configurations of MFCI from Figure 2 to the equivalent approximation variants. For MFCI with candidate evaluation, we see a small increase in approximation error with a negligible performance improvement when using the approximation. However, for the faster variant without candidate evaluation, the error does not increase, but the computation time is less than half of the original. Overall, this suggests that the approximation is a good trade-off for the faster variant of MFCI.

Robustness to Noise. Figure 5a shows the relative performance of MFCI compared to SPH for different noise levels. We observe that for low noise levels, SPH is more accurate than MFCI, but the opposite is true for higher noise levels. We hypothesize that the low-rank matrix factorization filters out noise in the data, leading to better candidate cells than in SPH.

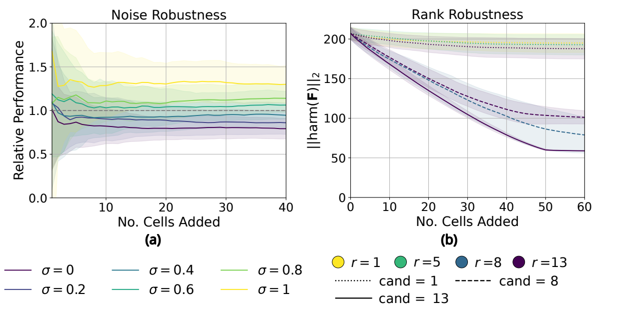

Fig. 5. **(a)** Relative performance of MFCI (SVD, best candidate out of 5, approx. calculation of harmonic flow, deterministic candidate heuristic) compared to SPH on a CC sampled from Erdős–Rényi with $n = 40, p = 0.9$, 80 sampled 2-cells, and 64 flows. The flow and the noise were drawn from normal distributions with mean $\mu = 0$; the former with a standard deviation $\sigma = 1$ and the latter according to the legend. We show the relative performance because the inferred cells and approximation error change with the noise level. The relative performance is calculated as $(r-a)/(r-b)$ where r is the average error of a random algorithm, a is the error of MFCI and b is the error of SPH. As such, a relative performance of 0 is as good as adding random cells; a value of 1 is as good as SPH. A value above 1 indicates that MFCI outperforms SPH. **(b)** Different no. of ranks and candidates for MFCI (ICA, 1oo_); showing negligible impact of rank on accuracy. Experiment settings from Figure 2;

VI. Conclusion

We presented a new framework, MFCI, that approximately solves the general cell inference problem by using matrix factorization. Overall, our experiments show that compared to the previous state-of-the-art, MFCI achieves a significant speed-up at the expense of a small increase in the approximation error in certain settings. Furthermore, in noisy configurations, MFCI achieves a better approximation error.

Importantly, we saw qualitatively different results on synthetic and real-world data. The synthetic data was based on the same assumptions made in the development of MFCI, namely, that there are underlying 2-cells that generate flow *independently* of each other. It is likely that real-world data has strong correlations, leading to a less useful matrix decomposition. This hypothesis is consistent with the observation that ICA, which employs similar inherent assumptions, performs poorly on real-world data. Nevertheless, MFCI with SVD is useful for applications where its significant speed-up (and thus increased scalability) outweighs its relatively small increase in approximation error. The approximation error is based on signal compression and likely translates well to signal processing tasks, but it is unclear how it affects the performance of other downstream tasks built on the inferred cells.

There is a large space of possible configurations for MFCI that exceeds the scope of this paper. First, any component of MFCI could be replaced, mainly the matrix factorization method and the heuristic to obtain candidates. Furthermore, the projection could use a hybrid approach where the approximate method is used, but the projection is done explicitly in some iterations. Second, due to the iterative nature of both MFCI and SPH, it is also possible to combine the two methods. As we have seen, SPH performs particularly well in the first iterations, where it is also comparatively fast. Therefore, it may be advantageous to design a hybrid approach, where the first few iterations are performed with SPH and the remaining iterations with MFCI. Ideally, this could combine the accuracy of SPH in the early iterations with the better computational performance of MFCI in the later, more computationally expensive, iterations.

References

[1] D. Mulder and G. Bianconi, "Network geometry and complexity," *J. of Statistical Physics*, vol. 173, pp. 783–805, 2018.
[2] D. I. Shuman, S. K. Narang *et al.*, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, 2013.
[3] A. Ortega, P. Frossard *et al.*, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, 2018.
[4] S.-W. Lee, J.-S. Park *et al.*, "A study on smart-phone traffic analysis," in *2011 13th Asia-Pacific Network Operations and Management Symp.* IEEE, 2011, pp. 1–7.
[5] Transp. Netw. for Research Core Team, "Transportation Networks for Research," Accessed: 2025-02-28. [Online] https://github.com/bstabler/TransportationNetworks
[6] G. Iori, G. De Masi *et al.*, "A network analysis of the italian overnight money market," *J. of Econ. Dyn. and Control*, vol. 32, no. 1, pp. 259–278, 2008.
[7] S. Barbarossa and S. Sardellitti, "Topological signal processing over simplicial complexes," *IEEE Trans. Signal Process.*, vol. 68, pp. 2992–3007, 2020.
[8] S. Sardellitti, S. Barbarossa, and L. Testa, "Topological signal processing over cell complexes," in *2021 55th Asilomar Conf. on Signals, Systems, and Computers*. IEEE, 2021, pp. 1558–1562.
[9] M. T. Schaub, Y. Zhu *et al.*, "Signal processing on higher-order networks: Livin'on the edge... and beyond," *Signal Processing*, vol. 187, p. 108149, 2021.
[10] M. T. Schaub and S. Segarra, "Flow smoothing and denoising: Graph signal processing in the edge-space," in *2018 IEEE Global Conf. on Signal and Inf. Process. (GlobalSIP)*. IEEE, 2018, pp. 735–739.
[11] M. T. Schaub, J.-B. Seby *et al.*, "Signal processing on simplicial complexes," in *Higher-Order Systems*. Springer, 2022, pp. 301–328.
[12] T. M. Roddenberry, M. T. Schaub, and M. Hajij, "Signal processing on cell complexes," in *2022 IEEE International Conf. on Acoustics, Speech and Signal Process. (ICASSP)*. IEEE, 2022, pp. 8852–8856.
[13] L.-H. Lim, "Hodge laplacians on graphs," *SIAM Review*, vol. 62, no. 3, pp. 685–715, 2020.
[14] M. T. Schaub, A. R. Benson *et al.*, "Random walks on simplicial complexes and the normalized hodge 1-laplacian," *SIAM Review*, vol. 62, no. 2, pp. 353–391, 2020.
[15] C. Battiloro, I. Spinelli *et al.*, "From latent graph to latent topology inference: Differentiable cell complex module," in *The 12th Int. Conf. on Learn. Representations*, 2024.
[16] J. Hoppe and M. T. Schaub, "Representing edge flows on graphs via sparse cell complexes," in *Proc. of the 2nd Learn. on Graphs Conf.*, ser. Proc. of Machine Learn. Research, S. Villar and B. Chamberlain, Eds., vol. 231. PMLR, 27–30 Nov 2023, pp. 1:1–1:22.
[17] J. Hoppe, V. P. Grande, and M. T. Schaub, "Don't be Afraid of Cell Complexes! An Introduction from an Applied Perspective," arXiv Preprints, 2025. arXiv:2506.09726.
[18] L. J. Grady and J. R. Polimeni, *Discrete calculus: Applied analysis on graphs for computational science*. Springer, 2010, vol. 3.
[19] D. C.-L. Fong and M. Saunders, "LSMR: An iterative algorithm for sparse least-squares problems," *SIAM J. on Scientific Comput.*, vol. 33, no. 5, pp. 2950–2971, 2011.
[20] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural Networks*, vol. 13, no. 4, pp. 411–430, 2000.
[21] J. Hoppe and M. T. Schaub, "Random abstract cell complexes," arXiv Preprints, 2024. arXiv:2406.01999.
[22] A. R. Benson, D. F. Gleich, and L.-H. Lim, "The spacey random walk: A stochastic process for higher-order data," *SIAM Review*, vol. 59, no. 2, pp. 321–345, 2017.
[23] C. Whong, "Foiling nyc's taxi trip data," March 2014. [Online] https://chriswhong.com/open-data/foil_nyc_taxi/