Continuous Product Graph Neural Networks (CITRUS) A Four-Page Report with Key Equations and Experiments

Benaziza Chems Eddine [1] **Braham Mohamed Amir** [1]

Abstract

In this short summary of the Continuous Product Graph Neural Networks (CITRUS) framework, we present an intuitive overview of its core motivation for learning from multidomain data on multiple graphs. We outline the key mathematical derivations, highlight the model's stability and over-smoothing properties, and summarize the main experimental findings. Additionally, we compare CITRUS to a closely related discrete approach (GTCNN) and discuss how CITRUS generalizes beyond polynomial filtering to a continuous, PDE-based paradigm. We also faced some issues in our experiment code and made this pull request to improve the open-source repository: `github.com/ArefEinizade2/CITRUS/pull/1`.

1. Introduction and Motivation

Many real-world applications generate data that reside on multiple interacting graphs, which we refer to as *factor graphs*. Examples include traffic data that can be modeled on a spatial road network and a temporal graph of successive time steps, as well as multi-dimensional signals (e.g., images or videos) that can be represented by the product of multiple graphs. The primary challenge in learning from multidomain graph data is developing efficient frameworks that can effectively capture *joint* interactions between graphs.

Previous work in this area has proposed discrete graph filtering operations in product graphs (PGs). However, these methods inherit well-known issues from traditional Graph Neural Networks (GNNs), including:

- **Over-smoothing** and **over-squashing**, particularly in discrete polynomial-based GNNs that require careful selection of filter orders.
- **Computationally costly** or **ad hoc** hyperparameter tuning when dealing with multiple domains, typically involving sequential processing of different modalities (e.g., space, time, frequency).

These limitations restrict the graph's receptive field and hinder the modeling of long-range interactions, ultimately constraining the performance of GNNs on multidomain graph data. To overcome these challenges, **CITRUS** (*Continuous Product Graph Neural Networks*), makes a novel framework that addresses these issues by modeling *tensorial data* on product graphs *continuously*, using partial differential equations (PDEs). Specifically, CITRUS employs two key strategies: i) continuous graph filtering, which allows for more effective capture of complex interactions, and ii) a general framework that can accommodate any number of factor graphs, making it a versatile solution for a wide range of applications. By leveraging these innovations, CITRUS introduces a closed-form exponential filter that is lightweight, stable, and can be extended to an arbitrary number of domains, thereby unlocking the full potential of graph-based models for real-world applications.

2. Key Equations and Theoretical Underpinnings

2.1. Tensorial PDE on Graphs

Let there be P factor graphs, each with an undirected Laplacian $L_p \in \mathbb{R}^{N_p \times N_p}$. Denote our multidomain signal as $\widetilde{U}_t \in \mathbb{R}^{N_1 \times \cdots \times N_P}$, a P-way tensor evolving over continuous time t. CITRUS proposes the **tensorial PDE on graphs (TPDEG)**:

$$\frac{\partial \widetilde{U}_t}{\partial t} \;=\; -\sum_{p=1}^{P} \widetilde{U}_t \;\times_p\; L_p, \tag{1}$$

where $\times_p$ denotes a mode-p product *(i.e., multiplication on the p-th dimension)*.

Closed-Form Solution. If the initial tensor at $t = 0$ is $\widetilde{U}_0$, the PDE (1) admits

$$\widetilde{U}_t \;=\; \widetilde{U}_0 \;\times_1\; e^{-t\,L_1} \;\times_2\; e^{-t\,L_2} \;\cdots\; \times_P\; e^{-t\,L_P}. \tag{2}$$

Hence, each factor Laplacian contributes an exponential graph filter. CITRUS generalizes from two-factor (spatiotemporal) to multi-domain PDEs.

2.2. CITRUS Layer Formulation

A CITRUS network consists of L layers; each layer ℓ applies a learnable *continuous* filter:

$$U_{\ell+1} = \sigma\Big(U_\ell \times_1 e^{-t_\ell^{(1)} L_1} \times_2 \cdots \times_P e^{-t_\ell^{(P)} L_P} \times_{P+1} W_\ell^\top\Big), \tag{3}$$

where:

- $U_\ell \in \mathbb{R}^{N_1 \times \cdots \times N_P \times F_\ell}$ is the input to layer ℓ.
- $t_\ell^{(p)}$ is a *learnable* "time" parameter controlling diffusion on factor graph p.
- W_ℓ is a trainable linear weight for channel mixing, and $\sigma(\cdot)$ is a nonlinearity (e.g. ReLU).

Proposition 3.3. (From the paper) The core function of CITRUS in (5) can be rewritten as:

$$\big[f\left(U_l\right)_{(P+1)}\big]^\top = e^{-t_l L_\odot} \big[U_l\big]_{(P+1)}^\top W_l, \tag{4}$$

where $L_\odot := \bigoplus_{p=1}^P L_p$ is the Laplacian of the Cartesian product graph.

Proposition 3.3 is the main building block for implementing CITRUS. More precisely, we use the spectral decompositions of the factor graphs $\{L_p = V_p \Lambda_p V_p^\top\}_{p=1}^P$ and product graph $\{L_\odot = V_\odot \Lambda_\odot V_\odot^\top\}$, where $V_\odot := \bigotimes_{p=1}^P V_p$ and $\Lambda_\odot := \bigoplus_{p=1}^P \Lambda_p$. Let $K_p \le N_p$ be the number of selected eigenvalue-eigenvector pairs of the p-th factor Laplacian. When $K_p = N_p$, it can be shown that we can rewrite (6) as follows:

$$\big[f\left(U_l\right)_{(P+1)}\big]^\top = V_\odot^{(K_p)} \Big(\tilde{\lambda} \odot \big(V_\odot^{(K_p)\top} \big[U_l\big]_{(P+1)}\big)\Big) W_l, \tag{5}$$

with

$$\tilde{\lambda} := \bigoplus_{p=1}^P e^{-t_l \lambda_p^{(K_p)}}, \tag{6}$$

where $\lambda_p^{(K_p)} \in \mathbb{R}^{K_p \times 1}$ and $V_p^{(K_p)} \in \mathbb{R}^{N_p \times K_p}$ are the first $K_p \le N_p$ selected eigenvalues and eigenvectors of L_p based on largest eigenvalue magnitudes, respectively, and $\odot$ is the pointwise multiplication operation. Finally, we can define the output of the l-th layer of CITRUS as

$$U_{l+1(P+1)} = \sigma\big(f\left(U_l\right)_{(P+1)}\big),$$

where $\sigma(\cdot)$ is some proper activation function.

2.3. Stability and Over-Smoothing

Stability. Suppose each factor adjacency A_p is perturbed by a small noise matrix E_p. Then the overall Cartesian product adjacency A_{prod} is perturbed by $\sum_p E_p$ in a product sense, leading to a *bounded additive effect* on $\widetilde{U}_t$. Formally, CITRUS's solution error grows linearly with factor perturbations, yielding

$$\|\phi(u,t) - \phi_{\text{noisy}}(u,t)\| \;\approx\; \sum_{p=1}^P \|E_p\|.$$

Over-Smoothing. In deep GNNs, node embeddings often collapse to a constant when many layers are stacked. To analyze this in *CITRUS*, let

$$U \in \mathbb{R}^{N_1 \times \cdots \times N_P \times F}$$

be a multidomain feature tensor, and denote by $\widehat{L}^p$ the normalized Laplacian of the p-th factor graph. The *tensorial Dirichlet energy* is:

$$E(U) := \frac{1}{P} \sum_{f=1}^F \sum_{p=1}^P \operatorname{tr}\Big(U_{f(p)}^\top \widehat{L}^p U_{f(p)}\Big),$$

where $U_{f(p)}$ is the mode-p unfolding of U along the feature index f. Large diffusion times $t_\ell(p)$ on each domain can drive $E(U)$ to zero, yielding an over-smoothed (constant) embedding.

When $\ell \to \infty$, $E(X_\ell)$ exponentially converges to 0 if

$$\ln(s) \;-\; \frac{2\,\hat{\lambda}}{P} \;<\; 0.$$

Interpretation. Theorem 3.10 shows that the factor graph with the smallest non-zero eigenvalue (spectral gap)—multiplied by its receptive field—dominates the overall over-smoothing behavior. The smaller the spectral gap, the lower the probability that the factor graph is connected.

3. Experiments

We briefly highlight key experimental insights (full details in (1)).

Datasets. *MetrLA* and *PemsBay*, standard road sensor benchmarks. Each node is a sensor, connected in a spatial graph. The time steps form a second factor graph. We predict future traffic speeds given the past few minutes.

Setup. We compare CITRUS with various baselines:

- *GRUGCN*, a TTS pipeline using GRU (temporal) then GNN (spatial).
- *GTCNN*, a discrete product-graph (2 graphs only : temporal and spatial) approach (polynomial filters).
- *Others* (e.g. Graph Wavenet, STGCN).

Illustrative Results (MetrLA, Horizon=12). Below is a snippet from the longer-horizon (60-min ahead) prediction. Smaller MAE is better.

Method	MAE	Reference
GRUGCN	3.62	(1)
GTCNN	3.55	(2)
CITRUS	**3.44**	(1)

Table 1. Excerpt of results on MetrLA (H=12).

CITRUS outperforms or matches baselines, particularly for longer horizons, supporting its continuous multi-domain design.

Below is a comparison between our reproduced results and those from the paper for H=3 :

Table 2. Paper Results

Dataset	MAE	MAPE	RMSE
MetrLA	2.70	6.74%	5.14
PemsBay	1.21	2.51%	2.61

Table 3. Our Results

Dataset	MAE	MAPE	RMSE
MetrLA	2.70	7.07%	5.10
PemsBay	1.11	2.22%	2.31

Table 4. Comparison of our reproduced results (left) with the original paper results (right).

In *Molene* (32 stations, hourly data) and *NOAA* (109 stations, thousands of hours), CITRUS similarly showed robust gains over discrete product-graph baselines, with the domain-specific diffusion times controlling smoothing across space and time.

3.1. Ablation

The ablation study in the paper looks into the parts of CITRUS that are in charge of learning the spatiotemporal relationships in the data.The authors compare two popular architectures: TTS and STT. In the TTS setup, they first run the data through an RNN (specifically, a GRU) and then pass the output to a GNN, while in the STT approach the order is reversed. They also introduce continuous versions—CTTS and CSTT—where the regular GNN is swapped out for a continuous GNN (CGNN).

The experiments on the MetrLA dataset show that CITRUS performs the best. This might be because it learns the spatial and temporal dependencies together rather than one after the other. Additionally, the continuous models (CTTS and CSTT) outperform the regular ones (TTS and STT),

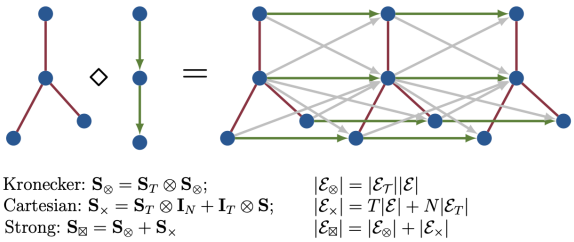

Figure 1. Product graphs

which suggests that using adaptive graph neighborhoods is more effective than relying on simple 1-hop connections in standard GNNs.

4. Relation to GTCNN

4.1. Choice of GTCNN

We selected GTCNN as our second reference because it was consistently cited in the CITRUS paper. GTCNN provides a foundational discrete approach to product-graph processing—specifically for the case of two factor graphs (space and time)—and in so doing, establishes benchmarks and methodologies upon which CITRUS builds. Revisiting GTCNN's core ideas illuminates how CITRUS extends that discrete paradigm to a continuous, PDE-driven framework and clarifies the reasoning behind many of CITRUS's design choices.

4.2. Summary of GTCNN

GTCNN proposes a **discrete polynomial filtering** paradigm on a product graph that fuses space and time. Specifically, it constructs a parametric product-graph shift operator

$$S_\otimes = \sum_{i=0}^{1}\sum_{j=0}^{1} s_{ij}\left(S_T^i \otimes S^j\right),$$

where $S \in \mathbb{R}^{N\times N}$ and $S_T \in \mathbb{R}^{T\times T}$ are shift operators for the spatial and temporal graphs, respectively, and $\{s_{ij}\}$ are learnable scalars capturing spatiotemporal coupling strength. Once $S_\otimes$ is set, GTCNN learns a *polynomial* filter of order K:

$$u_\otimes = \sum_{k=0}^{K} h_k \left(S_\otimes\right)^k x_\otimes = \sum_{k=0}^{K} h_k \left(\sum_{i=0}^{1}\sum_{j=0}^{1} s_{ij}\left(S_T^i \otimes S^j\right)\right)^k x_\otimes, \tag{7}$$

where $x_\otimes \in \mathbb{R}^{NT}$ is the vectorized spatiotemporal signal, h_k are trainable coefficients, and K is the polynomial order controlling the range of spatiotemporal interactions. This operation (often called "shift-and-sum") accumulates higher-order neighborhoods in both space *and* time, pro-

viding a powerful yet discrete means to capture rich spatiotemporal dependencies.

Beyond filtering, GTCNN also incorporates zero-pad pooling layers and nonlinearity, producing a *compositional* architecture. It has shown strong empirical performance on tasks like traffic forecasting and environmental data modeling, serving as a widely cited reference for product graph-based deep learning.

4.3. Relation to CITRUS

While GTCNN relies on finite-order polynomial filtering (Equation 7), **CITRUS** introduces a *continuous* exponential filter $e^{-tL_{\text{prod}}}$ derived from a partial differential equation on product graphs. This leads to several distinctions:

- **No polynomial order hyperparameter.** GTCNN must specify an integer K for $(S_{\otimes})^k$. By contrast, CITRUS's diffusion times $t_\ell^{(p)}$ in each domain are *continuous* parameters, potentially simplifying model tuning and reducing the risk of over-smoothing for large K.
- **Straightforward extension to multiple domains.** GTCNN focuses primarily on a two-domain scenario (space $\times$ time). CITRUS's PDE-based derivation naturally generalizes to P factor graphs (e.g. space, time, and additional axes like frequency), each with its own Laplacian exponent.
- **Continuous stability.** The PDE viewpoint in CITRUS facilitates theoretical analysis of stability and diffusion, whereas GTCNN's discrete polynomial expansion can be less transparent when K grows.

In short, CITRUS can be viewed as a *continuous* generalization of GTCNN: it shares the same *product-graph* foundation but replaces discrete polynomials with a PDE-driven exponential filter. This yields more flexible smoothing control, broader domain scalability, and a compact alternative to high-order polynomial expansions.

5. Conclusion

CITRUS leverages *continuous* product-graph filtering to handle multidomain data. Its key PDE-based equations (Eq. 1-2) enable exponential filtering across factor graphs, with domain-specific diffusion times to mitigate over-smoothing. Experiments on standard spatiotemporal tasks show its advantage in both long-horizon accuracy and computational efficiency. For future research, we are curious to know the trade-off between the computation cost and the efficiency of CITRUS on five or more product graphs.

References

[1] A. Einizade, F. D. Malliaros, and J. H. Giraldo. Continuous product graph neural networks (CITRUS). *arXiv preprint arXiv:2405.18877*, 2024.

[2] M. Sabbaqi and E. Isufi. Graph-time convolutional neural networks: architecture and theoretical analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(12), 2023.