The Journal of Open Source Software

GradGraph: Gradient-based Parameter Optimization on Graph-Structured Data in Python

Nicolas E. Fricker[2¶], **Laurent Monasse**[3], **and Claire Guerrier**[1,2]

1 Centre de Recherches Mathématiques (CRM), Université de Montréal, CNRS, Montréal, Canada **2** Université Côte d'Azur, CNRS, Laboratoire J. A. Dieudonné (LJAD, UMR 7351), Nice, France **3** Université Côte d'Azur, Inria, CNRS, Laboratoire J. A. Dieudonné (LJAD, UMR 7351), EPC ACUMES, Nice, France ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 25 September 2025
Published: unpublished

Summary

Many scientific systems can be modeled as **dynamical processes on graphs**, such as the spread of disease in populations, flows in infrastructure networks, or transport phenomena across irregular domains (Enright & Kao, 2018; Neri et al., 2013). To fit such models to observed data, researchers need both (1) a way to transform irregular graph-structured data into optimization-ready arrays, and (2) a framework for efficient parameter estimation using gradient descent. This work was motivated by challenges we encountered when modeling fungal growth, which required efficient parameter optimization in a graph-based framework.

GradGraph provides this functionality. It preprocesses graphs by extracting **linear paths** and applying a **moving-window (overlapping span) approach** to generate standardized arrays of equal length. This converts irregular, graph-based observations into a structured dataset suitable for machine learning pipelines. On top of this preprocessing, GradGraph supplies **TensorFlow templates** for simulating systems of ODEs or PDEs on these arrays and optimizing parameters via gradient-based methods.

Statement of need

While frameworks like `TensorFlow` (Abadi et al., 2016) and `PyTorch` (Paszke et al., 2019) provide robust automatic differentiation, they do not directly support the **graph-to-array preprocessing pipeline** required to model dynamical systems on networks. Users must typically:

- Manually traverse graphs to extract data sequences,
- Implement custom windowing schemes to standardize sequence lengths,
- Build training loops for ODE/PDE parameter optimization.

`GradGraph` addresses these challenges by:

- Extracting **linear paths** from graphs using `networkx` and related tools,
- Applying a **sliding-window transformation** to produce many equal-length arrays from each path,
- Providing **ready-to-use TensorFlow templates** for ODE/PDE models,
- Enabling **gradient-based optimization** of model parameters with minimal boilerplate,

- Remaining compatible with the broader Python scientific stack (`numpy`, `scipy`, `networkx`).

This combination makes GradGraph a powerful and accessible framework for researchers and practitioners who want to calibrate dynamical models on graph-structured data.

Framework

Let $G = (V, E)$ be a graph with data $D : V \to \mathbb{R}^m$. For a linear path

$$P = (v_1, v_2, \dots, v_L) \subseteq V,$$

GradGraph constructs overlapping windows of fixed length w,

$$W_j = (v_j, v_{j+1}, \dots, v_{j+w-1}), \quad j = 1, \dots, L - w + 1.$$

From each window W_j, an array

$$X_{W_j} \in \mathbb{R}^{w \times m}$$

is created. This ensures all arrays have the same shape, making them suitable for TensorFlow-based optimization.

For a model M_θ (ODE or PDE system) parameterized by θ, GradGraph defines a loss

$$\mathcal{L}(\theta) = \sum_W \mathcal{L}_W(M_\theta(X_W), D_W)\,,$$

where D_W are the observed data restricted to window W. TensorFlow's autodiff then computes

$$\nabla_\theta \mathcal{L},$$

enabling optimization via standard methods (SGD, Adam, etc.).

The intended user community for GradGraph includes researchers and practitioners in **network science, computational biology, applied mathematics, and machine learning** who are interested in fitting dynamical models to graph-structured data.
Typical applications include the study of **biological growth processes, epidemiological spread, transport phenomena, and infrastructure networks**, where processes evolve on irregular domains and efficient **gradient-based parameter estimation** is required.

Citations

GradGraph builds on the Python ecosystem: `networkx` (Hagberg et al., 2008) for graph handling, `numpy` (Harris et al., 2020) and `scipy` (Virtanen et al., 2020) for numerical routines, and TensorFlow (Abadi et al., 2016) for autodiff and optimization.

Acknowledgements

This work was supported by EUR SPECTRUM at Université Côte d'Azur (50%) and by the French National Research Agency (ANR) through the project NEMATIC (50%), grant number ANR-21ANR08Z6RCHX.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2016). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv Preprint arXiv:1603.04467*. https://arxiv.org/abs/1603.04467

Enright, J., & Kao, R. R. (2018). Epidemics on dynamic networks. *Epidemics*, *24*, 88–97. https://doi.org/10.1016/j.epidem.2018.04.003

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). NetworkX: Network analysis in python. In T. Vaught & J. Millman (Eds.), *Proceedings of the 7th python in science conference (SciPy2008)* (pp. 11–15). https://networkx.org/

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2

Neri, I., Kern, N., & Parmeggiani, A. (2013). Exclusion processes on networks as models for cytoskeletal transport. *New Journal of Physics*, *15*(8), 085005. https://doi.org/10.1088/1367-2630/15/8/085005

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, *32*. https://papers.nips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S. J. van der, Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2