The Journal of Open Source Software

StructuralGT: A Python API for graph-based design of complex materials

Alain Kadar [1,2], **Sharon C. Glotzer** [1,2], **and Nicholas A. Kotov** [1,2¶]

1 Department of Chemical Engineering, University of Michigan, Ann Arbor, Michigan 48109, United States **2** NSF Center for Complex Particle Systems (COMPASS), Ann Arbor, Michigan 48109, United States ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

StructuralGT provides a set of tools for the quantitative analysis of complex materials combining order and disorder, relying extensively on methods from graph theory (GT). Many such materials are made from nanoscale components self-assembled into gels and other particle networks whose structure is particularly difficult to describe using the traditional toolbox of colloidal chemistry and soft matter. This release includes 3D capabilities, new descriptors, analysis of segmented tomography, and an optimized backend, all of which improve upon the *StructuralGT* GUI application released by Vecchio et al. (2021). Additionally, this contribution details the development of a customizable and extensible API, which exposes the entire data-to-network workflow, along with modular tools for analysis of the resulting graphs. The Python API makes calls to either fast C++ libraries or *StructuralGT* scripts, while the advanced user has the option to incorporate their own C++ scripts using the provided template wrapper. Finally, *StructuralGT* writes the analyzed graphs to a geometry-preserving filetype allowing for storage and easy visualization of the data with the OVITO desktop application (Stukowski, 2010). *StructuralGT* binaries are maintained on conda-forge (*StructuralGT - Conda-Forge*, 2025), and the open-source repository (along with a repository of *StructuralGT* examples) is hosted on the GitHub organization page for the Center for Complex Particle Systems (*COMPASS - GitHub*, 2025). Documentation is hosted on read-the-docs (*StructuralGT Documentation*, 2025).

Statement of Need

Many advanced materials essential for sustainability, exemplified by a wide range of self-assembled biomimetic nanostructures produced by self-assembly, exhibit long-range connectivity patterns, non-random disorder, and hierarchical complexity, and so a quantitative basis in which to represent their structure is needed. Particularly for nanomaterials, the analysis of network structures has repeatedly been shown to benefit from a GT-based representation of their 2D data (Vecchio et al., 2022; Zhang et al., 2020, 2021). As 3D imaging becomes increasingly accessible to experimental communities, tools for their larger and higher-dimensional analyses are needed. The utility of GT in the context of structural representation has already been demonstrated (Duan et al., 2023; Kalutantirige et al., 2024). The further extension of structural representation to property predictions requires tools specific to each system, motivating the development of a modular and extensible user-facing API. This also allows for bundling scripts with research publications, thus fulfilling the growing need for reproducible scientific computing. Collectively, these requirements necessitate the development of the present contribution, whose uses to date include the prediction of charge transport properties of silver nanowire networks (Wu et al., 2024); the prediction of stress distribution in strut-lattices (Reyes-Martinez et al., 2025); and the rationalization of non-monotonic chiroptical properties

of complex nanodendrimers (Kuznetsova et al., 2025).

Summary of Features

Graph Extraction and Descriptors

The structure of complex materials can be represented by a graph, i.e. a mathematical object containing a set of nodes and edges connecting them. Unlike most graphs, a GT-based description of complex materials is embedded in Cartesian 2 or 3D space. Besides the information about connectivity, the GT representations of materials have essential geometrical information associated with both nodes and edges. Both topological and geometrical information can be directly extracted from the experimental data, and specifically from the atomic force microscopy, electron microscopy, confocal microscopy, and other techniques, which dramatically increases accuracy and specificity of GT descriptions of materials. To most effectively represent the information encapsulated in GT descriptors of materials most often organized as a matrix, we developed the `NetworkMaterial` object. The `NetworkMaterial` object can be populated with a `graph` attribute that contains the connectivity information capturing much of the material's structure. Development of the `NetworkMaterial` object is motivated by the need to preserve information lost during abstractions of material structure as a graph. Computational methods of the `NetworkMaterial` object that allow the graph extraction involve image processing, skeletonization, and neighbor finding. Once the appropriate methods have been called, the `graph` attribute of the `NetworkMaterial` object is populated. The graph may be visualized and/or stored in a manner which preserves the underlying geometric and topological complexity of the material's structure. An example involving image processing and steps required to produce a simple heatmap of nodes colored by degree, is given in Figure 1.

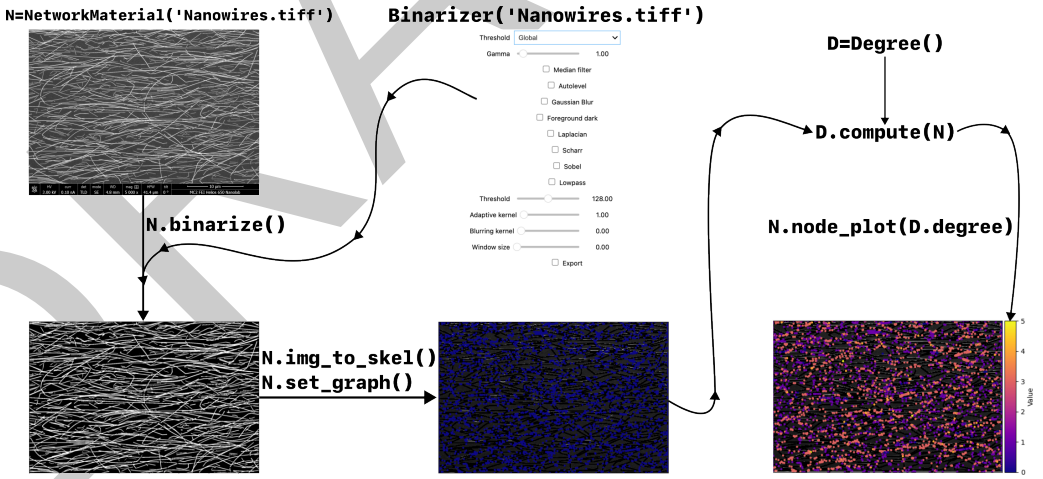

Figure 1: The workflow of complex material analysis with *StructuralGT* is exemplified using a network of nanowires. A `NetworkMaterial` object is first instantiated with a filepath (top-left). When analyzing experimental images, the `NetworkMaterial` object must binarize and skeletonize the image (bottom). Binarization is made possible by defining a set of image processing parameters. These can be determined with trial-and-error, using the `Binarizer` object. The subsequent skeletonization step is relatively parameter free (but special options are defined in the documentation). The graph can then be extracted by tracing the skeleton to classify pixels (or voxels) as belonging to nodes or edges. The `NetworkMaterial` object can then be combined with a `Compute` object (e.g. `Degree`) to calculate results. When both the `set_graph` method of the `NetworkMaterial` object and the `compute` method of the `Compute` object have been called, they can be combined to write annotated network files and plot heatmaps (bottom-right). Experimental micrograph taken from Wu et al. (2024).

Compute Objects

Once the graph attribute of a `NetworkMaterial` object has been populated (i.e. a GT-based representation of the complex material is created), it can be combined with the `Compute` objects to carry out analysis of the system. Alternatively, the user may generate a `graph` populated `NetworkMaterial` object by loading from the network filetype discussed in the next section. In either case, a loaded `NetworkMaterial` can be combined with `Compute` objects to perform different analyses. The current list of `Compute` objects is summarized below:

- "Structural": This object is mainly a wrapper for standard GT parameters, as calculated by igraph. It can be used to quantitatively establish structural differences between materials synthesized in different conditions (e.g. Kuznetsova et al., 2025).
- "Electronic": This object can be used for solving general networked linear transport problems in or 3D. Recently, it was used to simulate the four-point probe experiments carried out to assess electrical properties of 2D conductive films, and reproduced experimental results (Wu et al., 2024).
- "Geometric": This object is for assessing orientational order in networks, which often occurs when densely packed edges are forced into alignment, often resulting in anisotropic properties impossible to detect with traditional GT metrics (e.g. Wu et al., 2024).
- "Betweenness": This object provides betweenness parameters and variations that are designed to predict hotspots in stressed networks (e.g. Reyes-Martinez et al., 2025).
- "AverageNodalConnectivity": Due to the significant computational cost, computations for average nodal connectivity are relegated to their own object. This object provides fast calculation of the average nodal connectivity, which has been shown to correlate with the mechanical properties of gels (e.g. Vecchio et al., 2022).

Depiction of their combination with `NetworkMaterial` objects is given in Figure 2.

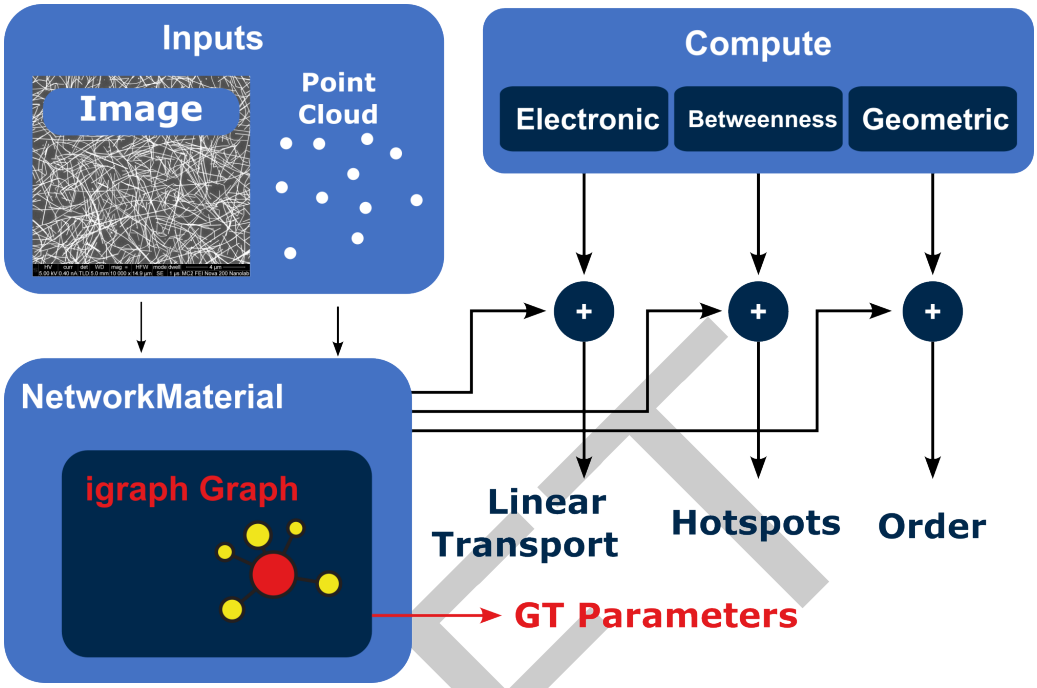

Figure 2: *StructuralGT* obtains results by combining an object that represents the system (`NetworkMaterial`) and an object that carries out some computation (`Compute`). NetworkMaterial objects are generated either from micrographs or (potentially dynamic) point cloud data. Compute objects are written to carry out computations not typically included in standard graph theory libraries. Additionally, once the `NetworkMaterial` object has been populated with the `graph` attribute, traditional GT parameters can be calculated via calls to the igraph library (e.g. degree, closeness, clustering coefficient). Experimental micrograph taken from Wu et al. (2024).

Network Material Filetype

While there are many filetypes designed for storing graphs, the requirement of preserving geometry in graphs extracted from material networks actually makes molecular simulation filetypes a more appropriate choice. For *StructuralGT*, we have opted for the `.gsd` format, because its compatibility with the OVITO toolkit allows us to make rich and dynamic visualizations with both the desktop application and their extensive Python API (Stukowski, 2010).

State of the field

While there are a few GT packages (*igraph* (Csardi & Nepusz, 2006), *NetworkX* (Hagberg et al., 2007), *graph-tool* (Peixoto, 2017)), there are fewer still that are compatible with materials science data. Even those that are (e.g. *crystal-torture* (O'Rourke & Morgan, 2019), *PoreBlazer* (Sarkisov et al., 2020)) are not compatible with image data, which is a crucial datatype used by experimentalists. Developing an image-to-graph workflow is a distinctly unique capability of *StructuralGT*, involving specialized image processing tools - such as `sknw` (*Sknw*, 2025) - which are not even part of standard image processing packages like `scikit-image`. The present contribution provides the necessary graph-extraction capabilities for unifying both experimental and computational materials under a common framework. Creating a new *StructuralGT* - as opposed to extending the existing codebase - was driven by the need for an extendable and modular object-oriented API to achieve the above objectives. This required completely rewriting the previous (now unmaintained) codebase of scripts and functions that could only be interacted with via a GUI. The API development has further enabled

- Bundling of Python scripts with research publications to ensure users' scientific computing is reproducible,

- Deployment of StructuralGT to HPC resources,
- Combination of StrcturalGT with numerous scientific computing Python packages.

Software Design

Tradeoffs considered in the design of this software include which GT package to use for calculations. While the initial prototype of *StructuralGT* used *NetworkX*, the implementation described here uses *igraph*. *igraph* was chosen over *NetworkX* because its C backend provides significantly faster calculations than the pure Python *NetworkX*. Although other GT packages with C backends exist (e.g. *graph-tool*), *igraph* is preferred because it offers easy access to C subroutines by exposing the pointer to the undelying C igraph object through the Python API. This allows us to provide a template for future *StructuralGT* developers to exploit simultaneously the accessibility of Python with the speed of C.

The design choice of partitioning analyses into system-like and compute-like objects was inspired by the *freud* API (Ramasubramani et al., 2020) (which, although designed for very different systems, takes a very similar approach). The list of possible analyses is then formed by a kind of Cartesian product between `NetworkMaterial` subclasses and `Compute` subclasses. The choice to employ a `NetworkMaterial` base class (as opposed to `graph`-like) is motivated by the conceptual difference between a network and a graph: Networks are real-world objects which exhibit a structure akin to points linked by connections. The mathematical objects which abstract these structures are graphs. A graph may be defined only by its nodes and connecting edges. While the nodes and edges may contain attributes, to give further detail, a large part of the network's description is lost as a result of its abstraction into a graph (most notably, its geometric features). Hence the `NetworkMaterial` class is used to contain a conventional graph, and all other information that would be otherwise lost.

Research Impact Statement

Research impact is evidenced by the peer-reviewed publications that would not have been possible without the present contribution (Kuznetsova et al., 2025; Reyes-Martinez et al., 2025; Wu et al., 2024). Furthermore, several in-progress, unpublished projects involve integration of *StructuralGT* with numerous additional computational resources for increased research impact. Pursuit of these projects by external researchers - with minimal input from developers - is made possible thanks to *StructuralGT's* strict adherence to best-practices software development for accessible open-source software, including versioning, testing, CI/CD, documented object-oriented API. Given that current benchmarks show that *StructuralGT* is at least 10 times faster than its initial release (even for graphs as small as 1000 nodes), it is well-poised for the research directions trending towards larger graphs and more compute-expensive data-driven methods.

As well as having been made accessible to computational scientists, training workshops for *StructuralGT* have been given to numerous experimentalists. This includes workshops for researchers at Addis Ababa Science and Technology, as part of efforts by the National Science Foundation's Center for Complex Particle Systems (COMPASS) to increase collaboration with research institutions in Africa (*StructuralGT - Workshop*, 2025). Tutorials, contribution invitations, and other resources are centralized on the COMPASS website (*StructuralGT - COMPASS*, 2025). Contributions to *StructuralGT* include the development of a new GUI that makes calls to the API.

AI Usage Disclosure

No AI was used in software creation, documentation generation, or paper authoring.

Acknowledgements

All authors are grateful for support from the U.S. National Science Foundation under Cooperative Agreement No. 2243104, "Center for Complex Particle Systems (COMPASS)" Science and Technology Center. Additionally, we would like to acknowledge Drew Vecchio for implementing the initial release of this software and answering any questions we had about it. We would like to acknowledge Joshua Anderson who mentored the implementation of C++/Python bindings, deployment to conda-forge, and software engineering best-practices. We would like to acknowledge the developers of the freud library, (Ramasubramani et al., 2020) whose choice to partition analysis into system-like and compute-like objects in their own library inspired the *StructuralGT* API. Finally we would like to acknowledge the Glotzer Group and Kotov Lab students who used initial prototypes of *StructuralGT* and gave feedback on how it could be improved, particularly Linlin Sun and Joshua Anderson who both gave feedback for this manuscript.

References

COMPASS - GitHub. (2025). https://github.com/compass-stc

Csardi, G., & Nepusz, T. (2006). The igraph software. *Complex Syst*, *1695*, 1–9. https://doi.org/10.5281/zenodo.3630268

Duan, S., Cattaruzza, M., Tu, V., Auenhammer, R. M., Jänicke, R., Johansson, M. K. G., Liu, F., & Asp, L. E. (2023). Three-dimensional reconstruction and computational analysis of a structural battery composite electrolyte. *Communications Materials*, *4*(1), 1–9. https://doi.org/10.1038/S43246-023-00377-0

Hagberg, A., Swart, P. J., & Schult, D. A. (2007). *Exploring network structure, dynamics, and function using NetworkX*. Los Alamos National Laboratory (LANL).

Kalutantirige, F. C., He, J., Yao, L., Cotty, S., Zhou, S., Smith, J. W., Tajkhorshid, E., Schroeder, C. M., Moore, J. S., An, H., Su, X., Li, Y., & Chen, Q. (2024). Beyond nothingness in the formation and functional relevance of voids in polymer films. *Nature Communications*, *15*(1), 1–16. https://doi.org/10.1038/s41467-024-46584-2

Kuznetsova, V., Kadar, A., Gaenko, A., Er, E., Ma, T., Whisnant, K. G., Ma, J., Ni, B., Mehta, N., Kim, J. Y., Gun'ko, Y. K., & Kotov, N. A. (2025). Graph-property relationships for complex chiral nanodendrimers. *ACS Nano*, *19*(6), 6095–6106. https://doi.org/10.1021/acsnano.4c12964

O'Rourke, C., & Morgan, B. J. (2019). Crystal-torture: A crystal tortuosity module. *Journal of Open Source Software*, *4*(38), 1306. https://doi.org/10.21105/joss.01306

Peixoto, T. P. (2017). The graph-tool python library. *(No Title)*. https://doi.org/10.1017/9781009118897

Ramasubramani, V., Dice, B. D., Harper, E. S., Spellings, M. P., Anderson, J. A., & Glotzer, S. C. (2020). Freud: A software suite for high throughput analysis of particle simulation data. *Computer Physics Communications*, *254*, 107275. https://doi.org/10.1016/J.CPC.2020.107275

Reyes-Martinez, M. A., Kadar, A., Dunne, S., Glotzer, S. C., Soles, C. L., & Kotov, N. A. (2025). Graph-theoretical description and continuity problems for stress propagation through complex strut lattices. *Npj Soft Matter 2025 1:1*, *1*, 1–8. https://doi.org/10.1038/S44431-025-00004-7

Sarkisov, L., Bueno-Perez, R., Sutharson, M., & Fairen-Jimenez, D. (2020). Materials informatics with PoreBlazer v4.0 and the CSD MOF database. *Chemistry of Materials*, *32*, 9849–9867. https://doi.org/10.1021/ACS.CHEMMATER.0C03575

Sknw. (2025). https://github.com/Image-Py/sknw

StructuralGT - COMPASS. (2025). https://compass.engin.umich.edu/structuralgt-software/

StructuralGT - conda-forge. (2025). https://anaconda.org/channels/conda-forge/packages/structuralgt/overview

StructuralGT - workshop. (2025). https://compass.engin.umich.edu/alain-kadars-experience-at-aastu-as-

StructuralGT documentation. (2025). https://structuralgt.readthedocs.io/en/latest/index.html

Stukowski, A. (2010). Visualization and analysis of atomistic simulation data with OVITO-the open visualization tool. *Modelling and Simulation in Materials Science and Engineering*, *18*(1). https://doi.org/10.1088/0965-0393/18/1/015012

Vecchio, D. A., Hammig, M. D., Xiao, X., Saha, A., Bogdan, P., & Kotov, N. A. (2022). Spanning network gels from nanoparticles and graph theoretical analysis of their structure and properties. *Advanced Materials*, *34*(23), 2201313. https://doi.org/10.1002/ADMA.202201313

Vecchio, D. A., Mahler, S. H., Hammig, M. D., & Kotov, N. A. (2021). Structural analysis of nanoscale network materials using graph theory. *ACS Nano*, acsnano.1c04711. https://doi.org/10.1021/ACSNANO.1C04711

Wu, W., Kadar, A., Lee, S. H., Glotzer, S. C., Goss, V., & Kotov, N. A. (2024). Layer-by-layer assembled nanowire networks enable graph-theoretical design of multifunctional coatings. *Matter*. https://doi.org/10.1016/j.matt.2024.09.014

Zhang, H., Vecchio, D., Emre, A., Rahmani, S., Cheng, C., Zhu, J., Misra, A. C., Lahann, J., & Kotov, N. A. (2020). Graph theoretical design of biomimetic aramid nanofiber nanocomposites as insulation coatings for implantable bioelectronics. In *bioRxiv* (p. 2020.12.28.424604). bioRxiv. https://doi.org/10.1101/2020.12.28.424604

Zhang, H., Vecchio, D., Emre, A., Rahmani, S., Cheng, C., Zhu, J., Misra, A. C., Lahann, J., & Kotov, N. A. (2021). Graph theoretical design of biomimetic aramid nanofiber composites as insulation coatings for implantable bioelectronics. *MRS Bulletin*, *46*(7), 576–587. https://doi.org/10.1557/S43577-021-00071-X