On the Use of Geometric Deep Learning Towards the Evaluation of Graph-Centric Engineering Systems

IDETC2023-114592

Anthony Sirico Jr. and Daniel R. Herber

Colorado State University — Department of Systems Engineering
{anthony.sirico, daniel.herber}@colostate.edu

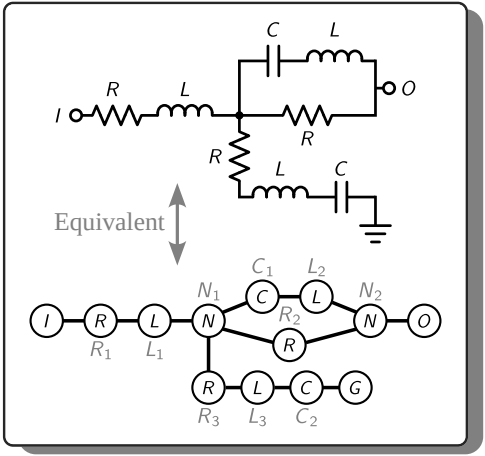
August 21, 2023
Boston, MA, USA

1

Introduction

Graph Representations of Engineered Systems

- **Graphs** can be used as a model for a variety of engineered systems
- Here we consider labeled graphs denoted by G



Design Situation of Interest

- Often, the performance (or value or utility) of a given graph $J(G_i)$ can be determined through analysis
- Consider the following three types of graph-centric design problems[1] with T is the amount of time allocated to complete the graph design study:

Type 0 All desired graphs can be generated, and so can their performance metric $J(G_i)$ within time T

Type 1 All desired graphs can be generated, but only some of the performance metrics $J(G_i)$ can be evaluated within time T; the analysis is too expensive

Type 2 All desired graphs cannot be generated within time T

- This **work focuses on methods for Type 1 problems** (using data from a large Type 0 study)

[1] Guo, Herber, and Allison 2018

Case Study Dataset (1): Analog Electric Circuit Synthesis

- The design case study here seeks to **identify the top set of best performing circuits** (each represented by graph G_i) assessed by the *optimal* value $J(G_i)$ of:

$$\underset{\mathbf{z}_i=\{\mathbf{R}_i,\mathbf{C}_i\}}{\text{minimize:}} \quad J = \sum_k \left(\log|H_i(j\omega_k,\mathbf{z}_i)| - \log|F(j\omega_k)|\right)^2 \tag{1a}$$

$$\text{subject to:} \quad 10^{-2} \leq R_j \leq 10^0 \quad \text{for all } R_j \text{ in } G_i \tag{1b}$$

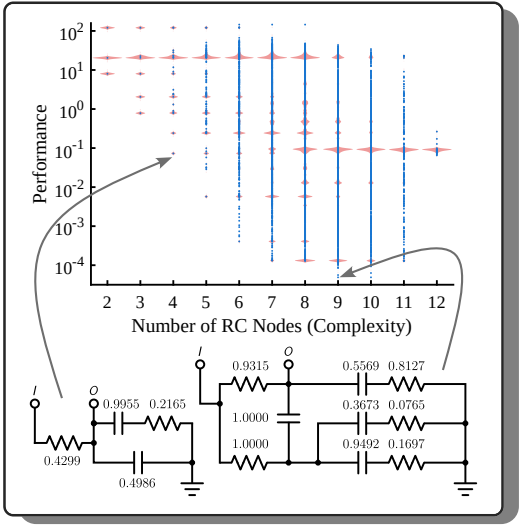
$$10^{-2} \leq C_j \leq 10^0 \quad \text{for all } C_j \text{ in } G_i \tag{1c}$$

$$\text{where:} \quad |F(j\omega)| = \sqrt{2\pi/10\omega} \quad 0.2 \leq \omega/2\pi \leq 5 \tag{1d}$$

  which represents minimizing the error between the desired frequency response $|F(j\omega)|$ and the one a candidate circuit G_i provides[1]
- Generating of *all unique (nonisomorphic) graphs* up to a certain size is possible using previous efficient graph enumeration techniques[2]
- However, the solving of Eq. (1) for all 43,249 graphs is expensive
- Previous work has incurred this expense[3] making this one suitable dataset for exploring case posed on Slide 3

[1] Grimbleby 1995; Herber 2017 [2] Herber 2020; Herber, Guo, and Allison 2017 [3] Herber 2017

Case Study Dataset (2): Performance vs. Complexity



Key Questions

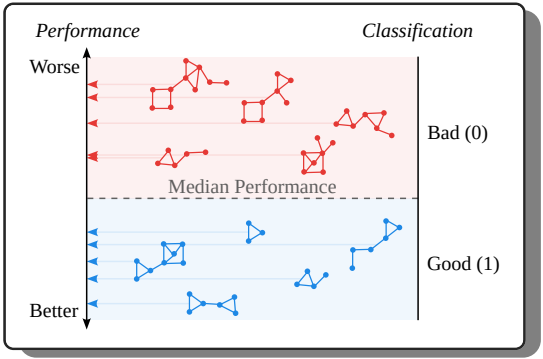
- Often the goal is not to narrow the potential graphs down to *one* particular graph, but rather a *group* of "good" or promising graphs that would be analyzed further
  - Sometimes at a higher fidelity due to assumptions made in modeling
  - To explore trade-offs (e.g., performance vs. complexity)

? Given a Type 1 problem, can we provide a reasonable likely set of "good" graphs without evaluating each of their performance $J(G_i)$?

? If so, how should we approach this challenge to reduce overall design study computational cost?

Graph Classification

- **Graph classification** approaches seek to assign graphs to a class based on a predetermined criteria — optimized assessment $J(G_i)$ from Eq. (1)
- Predictive models for classification are less concerned about absolute positioning than correct class assignment — aligned with the search for the top potential candidates



2

Methodology

→ Geometric Deep Learning (GDL) and Graph Neural Networks (GNNs)

- We consider **Geometric Deep Learning (GDL)** as a potential strategy for the classification goal
- GDL is an umbrella term encompassing a technique that generalizes neural networks to Euclidean and non-Euclidean domains, such as graphs, manifolds, meshes, or string representations[1]
- In essence, GDL encompasses approaches that incorporate information on the input variables' structure space and symmetry properties and leverage it to improve the quality of the data captured by the model
- GDL uses **Graph Neural Networks (GNNs)**[2], which have convolutional layers to determine node embeddings and pooling layers to average node embeddings
- GDL has been used in a variety of areas[3]

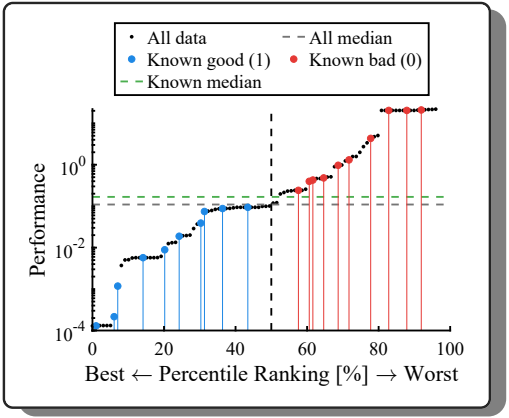
[1] Bronstein, Bruna, Cohen, et al. 2021; Bronstein, Bruna, LeCun, et al. 2017 [2] Lecun et al. 1998 [3] Wong et al. 2022; Pfaff et al. 2021; Park and Park 2019; Zhang, He, and Katabi 2019; Xiao, Ahmed, and Sha 2023; Ferrero et al. 2021; Atz, Grisoni, and Schneider 2021; Gainza et al. 2020; Krokos, Bordas, and Kerfriden 2022

→ Known and Unknown Graphs

- Based on the Type 1 problem classification, we will consider the case when only some of the performance values $J(G_i)$ for $G_i \subset \mathcal{G}$ are known
- This will divide the graphs into two sets as follows:

$$\mathcal{G} \equiv \mathcal{G}_{all} = \mathcal{G}_{known} \cup \mathcal{G}_{unknown} \qquad (2)$$

  - $\mathcal{G}_{known}$ is the set of graphs with known values for $J(G_i)$
  - $\mathcal{G}_{unknown}$ represents graphs with unknown $J(G_i)$ values (and this is what the GDL model is for)



→ Iterative Classification for Downselection

- Even if with perfect classification, the approach outlined so far would only result in determining the top 50% performing graphs in $\mathcal{G}_{all}$
- We seek a smaller, better median performance set of graphs from $\mathcal{G}_{all}$ by iteratively constructing GDL models with these steps:

Algorithm (Iterative Classification for Downselection)

1. *Set $k = 1$ and create an initial $\mathcal{G}^k_{known}$.*
2. *Create a GDL model $m^k(G_i)$ using $\mathcal{G}^k_{known}$, which is naturally broken into sets "Known 1" and "Known 0" based on the median J value of $\mathcal{G}^k_{known}$.*
3. *Predict the classes of the $\mathcal{G}^k_{unknown}$ using $m^k(G_i)$, creating "Predicted 1" and "Predicted 0", which are sets of graphs predicted to be good (1) or bad (0), respectively.*
4. *The goal is to identify good graphs, so we set $\mathcal{G}^{k+1}_{known}$ equal to "Known 1" and $\mathcal{G}^{k+1}_{unknown}$ equal to "Predicted 1" (and the remaining graphs are removed under the assumption that they are bad).*
5. *Set $k \to k + 1$ and repeat Step 2 until $k = n$.*

→ Tools and Code Availability

Table: List of Tools

Tool	Version
Python	3.9
PyTorch-Geometric[1]	2.1.0
PyTorch	1.12.1
Networkx	2.8.7
SciPy	1.9.1
Pandas	1.5.0

[1] Fey and Lenssen 2019

Code and dataset:
https://github.com/anthonysirico/GDL-for-Engineering-Design

3

Results

→ Experiments 1 & 2: Baseline and Additional Features

- *Experiment 1*: Baseline model had 79% accuracy and 77% precision on test set

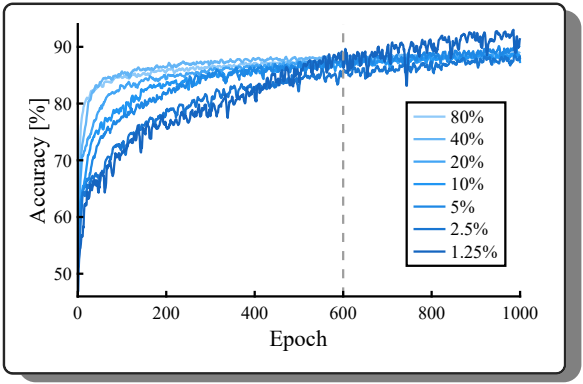
		Data	
		Actually Positive (1)	Actually Negative (0)
Model	Predicted Positive (1)	1,667	487
	Predicted Negative (0)	419	1,751

- *Experiment 2*: Model using **two additional features of eigenvector centrality and betweenness centrality** had 85% accuracy and 90% precision on test set

		Data	
		Actually Positive (1)	Actually Negative (0)
Model	Predicted Positive (1)	1,901	213
	Predicted Negative (0)	451	1,759

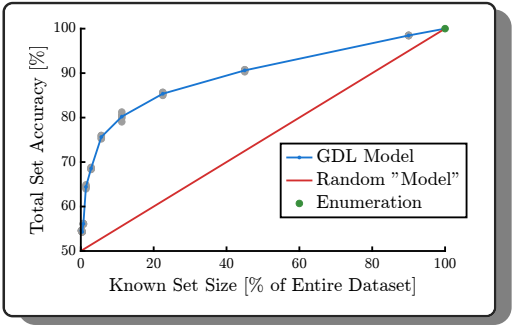
→ Experiment 3 (1): Require Epochs Determination

- What is an appropriate portion (%) of the dataset is needed to construct a *reasonable* GDL model?
- Several sizes of the known dataset (all graphs have $J(G_i)$ known) were selected
- Many epochs were also used to help determine a typical stopping condition



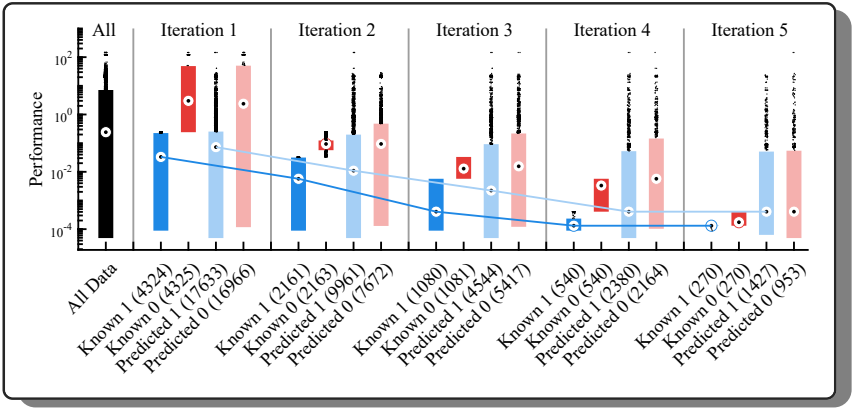
→ Experiment 3 (2): Total Set Accuracy vs. Known Set Size

- Since we are considering problems with a fixed number of graphs, we consider a **total set accuracy metric**
  - Allows comparisons to the worst-case random model and total enumeration
- Variability explored using 5 randomized runs for each known set size
- It was determined that 10–20% known set size was a reasonable balance between the cost of evaluating $J(G)$ and total set accuracy



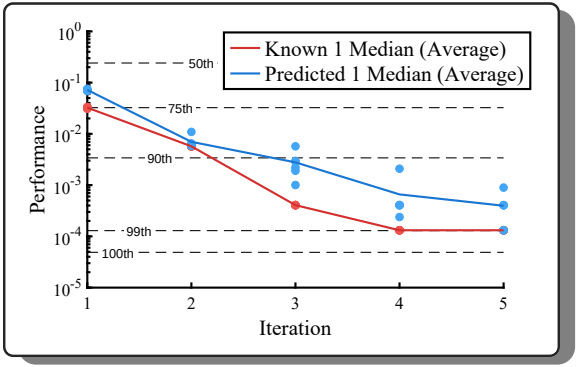
→ Experiment 4 (1): Iterative Classification Results

- Using the approach described on Slide 10, five iterations were performed
- Good separation occurs between the Predicted 1 and Predicted 0 sets until Iteration 5



→ Experiment 4 (2): Average Performance Change per Iteration

- Here we show the median values of the “Known 1” and “Predicted 1” sets averaged over six runs using the iterative GDL classification approach
- The previous slide is one of the six runs in this figure



→ Experiment 4 (3): Top X Graph Results

❓ Are the best-performing graphs still in our final set (after four iterations)?

❗ Important

For the top 100 graphs, 88.2 (2.5) remained compared to an expected value of 22.96. Finally, for the top 1000 graphs, 751.2 (39.5) remained compared to an expected value of 229.6.

- Over these runs, the average number of graphs that would be known (i.e., optimized) was 11282.2 graphs (492.8)
- Therefore, these outcomes are with about 25% of the computational cost of complete enumeration
- In another study with about 19% of the total potential cost, 9.2 (0.4) of the top 10 graphs remained and 85.2 (6.3) of the top 100 graphs remained

4

Conclusions & Future Work

→ Conclusions and Future Work

- We presented a Geometric Deep Learning (GDL) approach for classifying and down-selecting graph-based analog circuits toward sets of better-performing solutions
- Interesting, if not immediately valuable, trade-offs in accuracy and computational cost for this task were observed
- Potential future work items include:
  - Iteratively and intelligently adding new graphs to “Known” to combat model accuracy decreases
  - Transfer learning to similar problems
  - Explore this approach in other Type 1 problems, including ones with directed graphs and multiple graph-level performance metrics

→ References

- K. Atz, F. Grisoni, and G. Schneider (2021). *Geometric deep learning on molecular representations*. DOI: 10.48550/arXiv.2107.12375
- M. M. Bronstein, J. Bruna, T. Cohen, et al. (2021). *Geometric deep learning: grids, groups, graphs, geodesics, and gauges*. DOI: 10.48550/arXiv.2104.13478
- M. M. Bronstein, J. Bruna, Y. LeCun, et al. (2017). "Geometric deep learning: going beyond euclidean data". *IEEE Signal Process Mag.* 34.4. DOI: 10.1109/msp.2017.2693418
- V. Ferrero et al. (2021). "Classifying component function in product assemblies with graph neural networks". *ASME J. Mech. Design* 144.2. DOI: 10.1115/1.4052720
- M. Fey and J. E. Lenssen (2019). "Fast graph representation learning with PyTorch Geometric". *ICLR Workshop on Representation Learning on Graphs and Manifolds*
- P. Gainza et al. (2020). "Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning". *Nature* 17. DOI: 10.1038/s41592-019-0666-6
- J. B. Grimbleby (1995). "Automatic analogue network synthesis using genetic algorithms". *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*. DOI: 10.1049/cp:19951024
- T. Guo, D. R. Herber, and J. T. Allison (2018). "Reducing evaluation cost for circuit synthesis using active learning". *ASME International Design Engineering Technical Conferences*. DETC2018-85654. DOI: 10.1115/DETC2018-85654

→ References (Continued)

- D. R. Herber (2017). "Advances in combined architecture, plant, and control design". Ph.D. Dissertation. University of Illinois at Urbana-Champaign. URL: http://hdl.handle.net/2142/99394
- — (2020). "Enhancements to the perfect matching approach for graph enumeration-based engineering challenges". *ASME International Design Engineering Technical Conferences*. DOI: 10.1115/DETC2020-22774
- D. R. Herber, T. Guo, and J. T. Allison (2017). "Enumeration of architectures with perfect matchings". *ASME J. Mech. Design* 139.5. DOI: 10.1115/1.4036132
- G. Jurman, S. Riccadonna, and C. Furlanello (2012). "A comparison of MCC and CEN error measures in multi-class prediction". *PLOS One* 7.8. DOI: 10.1371/journal.pone.0041882
- V. Krokos, S. P. A. Bordas, and P. Kerfriden (2022). *A graph-based probabilistic geometric deep learning framework with online physics-based corrections to predict the criticality of defects in porous materials*. DOI: 10.48550/arXiv.2205.06562
- Y. Lecun et al. (1998). "Gradient-based learning applied to document recognition". *Proc. IEEE* 86.11. DOI: 10.1109/5.726791
- J. Park and J. Park (2019). "Physics-induced graph neural network: an application to wind-farm power estimation". *Energy* 187. DOI: 10.1016/j.energy.2019.115883
- T. Pfaff et al. (2021). *Learning mesh-based simulation with graph networks*. DOI: 10.48550/arXiv.2010.03409

→ References (Continued)

- J. C. Wong et al. (2022). "Graph neural network based surrogate model of physics simulations for geometry design". *IEEE Symposium Series on Computational Intelligence*. DOI: 10.1109/SSCI51031.2022.10022022
- Y. Xiao, F. Ahmed, and Z. Sha (2023). "Graph neural network-based design decision support for shared mobility systems". *ASME J. Mech. Design* 145.9. DOI: 10.1115/1.4062666
- G. Zhang, H. He, and D. Katabi (2019). "Circuit-GNN: graph neural networks for distributed circuit design". *International Conference on Machine Learning*

Questions?

On the Use of Geometric Deep Learning Towards the Evaluation of Graph-Centric Engineering Systems

IDETC2023-114592

Anthony Sirico Jr. and Daniel R. Herber
Colorado State University
{anthony.sirico, daniel.herber}@colostate.edu
www.engr.colostate.edu/~drherber

Links to the code on GitHub:
https://github.com/anthonysirico/GDL-for-Engineering-Design

doi: 10.48550/arXiv.2303.09770

→ Confusion Matrix

- A **Confusion Matrix (CM)**, the key metric representation used for a classification model, where the columns contain the samples of the model output, and the rows contain the samples *true* class
  - True Positives (T_P) — top left box
  - False Positives (F_P) — top right box
  - False Negatives (F_N) — bottom left box
  - True Negatives (T_N) — bottom right box

		Data	
		Actually Positive (1)	Actually Negative (0)
Model	Predicted Positive (1)	T_P	F_P
	Predicted Negative (0)	F_N	T_N

→ Performance Metrics

Metric	Definition	Equation
Accuracy	Overall accuracy of the model based on correct predictions	$Acc = \frac{T_P+T_N}{N}$ [b]
Precision	The proportion of positive classification predicted correctly	$Precision = \frac{T_P}{T_P+F_P}$
Recall	The proportion of actual positive classifications identified correctly	$Recall = \frac{T_P}{T_P+F_N}$
F1 Score	The harmonic mean between Precision and Recall	$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$
MCC[a]	Ranged from -1 to 1, takes into account all data from the CM	$MCC = \frac{T_P \cdot T_N - F_P \cdot F_N}{\sqrt{(T_P+F_P)(T_P+F_N)(T_N+F_P)}}$
Total Set Acc.	Accuracy of N_{all} using *known* and *unknown* when compared to N	$Total\ Set\ Accuracy = \frac{T_P^{(u)}+T_N^{(u)}+T_P^{(k)}+T_N^{(k)}}{N_{all}}$

[a] Matthews Correlation Coefficient[1]
[b] N is the total set size

[1] Jurman, Riccadonna, and Furlanello 2012