Benchmark First: Defining Tasks for Graph Transformation Learning

Adam Machowczyk and Reiko Heckel

University of Leicester, UK
amm106,rh122@le.ac.uk

Abstract. Learning graph transformations from examples requires datasets for training and benchmarking. To establish which machine learning architectures are suitable for which kinds of problems, we need to experiment on a range of graph-computation tasks. In this paper, we propose a framework for defining such tasks and categorising them along four dimensions: complexity, input-output relation, graph type, and the structural changes they require. We apply the framework to 16 tasks for which we have implemented data generators or provided data sets in a unified interface. Our aim is to work towards a benchmark that supports the training and evaluation of graph transformation models in a framework that can be used by the wider community to support their own research, define more tasks, thereby extending and refining the framework.

1 Introduction

Research in machine learning models to perform graph transformations requires data sets for training and evaluation. Graph transformation tasks vary, so we need different model architectures to address different types of tasks. Benchmarks assessing such models should offer a range of tasks of varying complexity, working on different types of graphs, involving different graph manipulations and input-output relations. We are not aware that such benchmarks exist, and more fundamentally, that the different types of graph transformation tasks are well understood and agreed.

In graph machine learning, a *graph-to-graph transformation*, or a *Deep Graph Transformation* (DGT), is a mapping of graphs from the input domain to the output domain [22]. The three most common types of DGT tasks are *Node-level Transformation (NT)*, *Edge-level Transformation (ET)* and *Node-Edge Co-Transformation* (NECT). NTs generate node attributes or classifications, including the classification of nodes as deleted, created or preserved. Similarly, ETs update edges and their attributes, and NECTs combine both kinds of changes [22].

DGT models are typically purpose-trained for specific domains. For example, the Diffusion Convolutional Recurrent Neural Network—a node-level DGT model—was built for traffic forecasting and evaluated exclusively on the METR-LA and PEMS-BAY datasets [32,41]. Because these models are developed in

isolation to solve custom problems, the discipline lacks standardised engineering and evaluation frameworks. This situation mirrors well-documented reproducibility crises in other areas of machine learning, such as Deep Reinforcement Learning [25] and Generative Adversarial Networks [34], where inconsistent evaluation metrics and hidden hyperparameter optimisation frequently generated false impressions of architectural superiority. Within graph learning specifically, there is growing evidence that this lack of standards is a critical vulnerability. Recent findings highlight the fragility of experimental setups that rely on a single train, validation, and test split: When training procedures and hyperparameter selections are applied robustly over multiple data splits, simple models can surprisingly outperform supposedly sophisticated GNN architectures [42]. Consequently, researchers are forced into expensive, time-consuming replication efforts, sometimes deriving entirely different findings than the original studies [17]. Ultimately, lacking unified benchmarks, structured comparison between tasks and models remains difficult, imprecise, and prone to the same pitfalls that have historically plagued broader deep learning research.

To address these challenges, we propose the first version of a conceptual framework for defining and comparing graph-to-graph transformation tasks. While specialised benchmarks seek to establish competitive models in a given domain, our goal is to propose an extensible framework to support reliable and reproducible training and evaluation across a range of tasks. To demonstrate and populate the approach, we introduce 16 tasks from existing literature selected to cover a broad spectrum of complexities, input-output relations, graph types, and structural changes. By outlining these dimensions, we motivate the development of a unified benchmark interface to support clear task specification, implementation, and evaluation across diverse problems. This is intended as an evolving community resource as we invite others to contribute tasks and expand the taxonomy.

2 Task Categories

To provide a common framework for graph transformation tasks, we propose a categorisation based on four factors: *complexity*, *input-output relation*, *graph type* and *structural changes*.

Complexity is defined using standard classes such as linear, polynomial-time (P), NP-complete or NP-hard. Using established transformation tasks, their complexity is usually known from the literature. It is tempting, given our interest in rule-based graph transformations, to define transformation complexity in terms of the number of rule applications, but not all transformation tasks have established rule-based solutions.

The relation between input and output graphs can be *deterministic*, *nondeterministic*, or *probabilistic*. In a deterministic relation, the same input always produces the same output. In the nondeterministic case, a single input admits multiple valid outputs. Such tasks can be further divided: In unconstrained tasks, any valid output is acceptable (e.g., any spanning tree of a connected graph).

In *optimisation* tasks, the goal is to find a valid output that minimises or maximises a given objective function, such as removing a minimal number of edges to eliminate all cycles. In *heuristic optimisation*, the task is to approximate such an optimum without guaranteeing it. In a *probabilistic* relation, each input graph induces a probability distribution over output graphs.

Graph machine learning architectures differ in which graph types are supported. Tasks can require *directed* or *undirected simple*, *multi-* or *hypergraphs*. Support for *homogeneous* graphs is more usual than for *bipartite* or *heterogeneous* (i.e., explicitly typed) graphs. However, *labelled* graphs are common, as are *attributed* graphs with numerical and textual features that models rely on to learn.

Structural changes are classified as *node- or edge-level transformation (NT or ET) or node-edge co-transformation (NECT)* [22]. Edge-level tasks include edge *addition* and *deletion*, and *edge attribute computation*. Node-level tasks involving *adding or deleting nodes* are challenging for neural networks since they change the shape of adjacency matrices [48], while *node attribute computation* is common in graph neural networks. Node- and edge-level attribute computation can be numerical (regression) or textual (classification). NECT tasks support node and edge-level transformation, and are especially useful for graph optimisation, error detection, and correction. An example is HOPPITY, a tool for detecting and correcting errors in JavaScript programs, which converts source code to graphs, learns a sequence of graph transformations to fix syntactically and semantically invalid code, and parses the results back to their original form [16].

3 Task Selection and Definitions

Existing graph machine learning models are evaluated against narrow, domain-specific datasets that do not comprehensively test topological generalisation. To address this gap, we conducted an exploratory search to curate a broad spectrum of problems. Guided by the four criteria above, we identified tasks that populate this multi-dimensional space, spanning the required structural changes, complexities, graph types, and input-output relations.

None of the selected tasks have as yet been solved using graph transformation learning. We specifically sought novel tasks because our objective is to evaluate the feasibility of different approaches. Several of the selected tasks represent problems for which efficient algorithms already exist. Applying machine learning to well-defined algorithmic problems is a choice in line with *neural algorithmic reasoning* (NAR) [45]. While NAR is motivated by practical considerations, such as improved efficiency and processing speed, we expect tasks with well-understood algorithmic solutions to provide insights into the nature of the challenges posed, such as their worst-case or mean complexity.

Due to the limitations of current models for NT tasks with dynamic sets of nodes, 13 out of 16 tasks are edge classifications, including edge creation and deletion, with the remainder covering edge regression, node regression, and

node classification. This shared formulation across otherwise distinct problems underscores the feasibility of a unified benchmark interface.

Next, we define the 16 tasks, ordered by increasing complexity, except where subsequent problems build on their predecessors.

Symmetric Closure: Given datasets of directed simple graphs, the objective is to transform the network by adding an inverse edge for every existing connection, provided one does not already exist. Requiring a model to learn a logical OR operation purely from topology makes symmetric closure a strict test of structure-based learning [10]. This poses a specific difficulty for graph transformation models, as they will inherently fail if they rely on non-structural node or edge features rather than recognising the actual connectivity patterns.

Transitive Closure: The ability to uncover hidden, multi-hop relationships is a fundamental challenge in network analysis. Given datasets of directed simple graphs, the objective is to transform the network by adding a direct edge between any two nodes connected by a path of length greater than one [38]. This presents a twofold graph learning challenge: the model must learn to compute all missing closures simultaneously across the network, while also generalising to recognise and bridge paths of arbitrary lengths that exceed those observed during training.

Shortcut Elimination: The inverse of transitive closure, shortcut elimination forces a model to prioritise multi-hop routing over direct links. Given datasets of directed simple graphs, the objective is to transform the network by deleting any direct edge between two nodes if an alternative path of arbitrary length also connects them. As a learning problem, this is difficult because the model must accurately scan and verify the existence of longer, secondary routes within the broader topological neighbourhood before safely pruning a connection [23].

Shape Completion: Reconstructing localised geometric patterns tests a model's acute spatial awareness within a larger network. Given datasets of undirected simple graphs containing 3- to 6-cycles (triangles to hexagons) with specific cycle edges removed, the objective is to transform the network by predicting and restoring these missing links. The primary graph learning difficulty lies in context recognition—a model might misinterpret the surrounding structure, hallucinate a missing edge in a perceived square, and erroneously connect unrelated nodes to force a shape that geometrically does not belong [37].

Molecular Bond Inference: Determining the exact nature of chemical connections tests a model's ability to perform complex multi-class discrimination within structured data. Given datasets of undirected, attributed multigraphs representing molecules generated from predefined fragments, the objective is to predict and assign the correct bond type—single, double, triple, or aromatic—between pairs of atoms [13]. While compensating the model with access to domain-specific structural and numerical atom features, the core learning difficulty lies in effectively mapping these rich local node attributes to the correct categorical edge formations across diverse molecular topologies. A standard use of this inference

is reconstructing a chemically valid molecular graph from 3D atomic coordinates, including bond orders, hybridisations, and formal charges [29].

Node Core Number Computation: Evaluating the hierarchical density of a network requires isolating heavily interconnected sub-structures from peripheral nodes. Given datasets of undirected simple graphs with node attributes, the objective is to compute and assign the core number for every node, defined as the largest k for which the node belongs to a k-core (a maximal subgraph where every internal node has a degree of at least k). This is challenging as a graph learning problem because the model must recursively evaluate degrees within dynamically shifting subgraphs rather than relying on static global connectivity, demanding a deep understanding of local neighbourhood density [5]. A common use case is finding influential spreaders in online social networks, since high-coreness nodes often sit in dense, diffusion-relevant parts of the graph [3].

Intra-Community Edge Prediction: Distinguishing whether structural links bridge distinct social groups or connect internal members is a key practical challenge in social network analysis. Given [30], the SNAP Facebook Combined dataset of undirected simple graphs with edge attributes, the objective is to evaluate each edge and predict whether it connects users from the same or different communities, as originally identified by the Louvain algorithm. Rather than allocating users to specific groups, the primary graph learning difficulty lies in recognising the topological signatures of intra- versus inter-community ties, compounded by the computational challenge of efficiently training the model to scale across a medium-sized, real-world dataset. A direct application of this task is predicting future links between users in the same community in a dynamic social network [40].

Edge Betweenness Centrality Computation: Identifying critical bottlenecks in network flow patterns requires a deep understanding of global routing dynamics. Given datasets of undirected simple graphs with edge attributes, the objective is to compute the Edge Betweenness Centrality for every edge, a continuous metric quantifying how frequently an edge lies on the shortest paths between all possible node pairs. As a graph learning problem, predicting this global property is particularly difficult because it rigorously tests a model's long-distance regression capabilities; the algorithm must accurately aggregate topological information from across the entire network rather than relying on immediate local neighbourhoods [9]. This computation is highly useful in community detection, where edges with high betweenness are treated as bridges between groups and removed to reveal isolated communities [21].

Feedback Edge Set Identification: Minimising edge deletions to break all network cycles is a notorious optimisation challenge where local decisions compound into immense global complexity. Given datasets of undirected simple graphs with edge attributes, the objective is to transform the network by deleting the fewest edges required to eliminate every cycle. As a graph learning problem, this is exceptionally difficult because individual edges frequently participate in multiple overlapping cycles; the model must therefore navigate an exponentially vast

search space, dynamically optimising its deletions to achieve the exact global minimum without indiscriminately pruning essential connections [18]. A practical application of this problem is reducing scan-register overhead in circuit testing [19].

Spanning Tree Identification: Extracting a foundational skeletal structure from a dense network poses a unique challenge due to non-differentiable operations and non-unique solutions. Given datasets of connected, undirected simple graphs, the objective is to transform the network by identifying and extracting a valid spanning tree that connects all nodes without forming any cycles. This poses a significant graph learning challenge because thresholding a reconstructed adjacency matrix is non-differentiable, forcing the model to optimise a surrogate objective function that assesses the structural validity of the candidate tree rather than directly evaluating exact-match correctness. Furthermore, since dense graphs possess numerous valid spanning trees, carefully selecting consistent target structures for the training data is a critical requirement to prevent the model from being penalised for finding alternative correct solutions [43]. A practical application of this identification is configuring data aggregation and broadcasting pathways in distributed systems, where an arbitrary, loop-free topology ensures that messages reach all network nodes without creating redundant broadcast storms [35].

Graph Colouring: Finding the optimal chromatic assignment in a network is a classic, computationally demanding problem that tests an algorithm's ability to balance competing constraints. Using the DIMACS dataset (specifically the DSJC125.1 subset of undirected simple graphs with node attributes), the objective is to assign colours to nodes such that no two adjacent nodes share the same colour, while strictly minimising the total number of colours used to reach the graph's known chromatic number of 5 [26]. Training a model for this task is inherently difficult because it requires simultaneously optimising three distinct objectives: Potts and Entropy to statistically minimise conflicting adjacent assignments, alongside Usage to penalise excess colour allocation [24]. Even on this relatively simple dataset subset, achieving the absolute minimum number of colours under a standardised evaluation interface remains a formidable graph learning challenge. In real-world applications, graph colouring is used for frequency or channel assignment in cellular phone networks, where colours model frequencies and edges model interference constraints [8].

Minimum Chordal Graph Completion: Enforcing a triangulated structure across a network is a strictly constrained, NP-hard optimisation problem. Given datasets of undirected simple graphs, the objective is to transform the network into a chordal graph—where every cycle of four or more vertices contains a chord bridging non-consecutive nodes—by adding the absolute minimum number of new edges [20]. As a graph learning challenge, this is particularly formidable because the model must navigate an exponentially vast search space where a single network often possesses multiple valid minimal chordal completions, forcing the algorithm to learn an optimal global strategy for edge addition rather

than relying on a single deterministic pattern. Practically, this completion finds use in triangulating Bayesian networks for exact junction-tree inference, where better triangulations significantly reduce inference costs [31].

Estimated Minimum Chordal Graph Completion: Approximating computationally intractable network triangulations provides a practical alternative to true minimum-fill problems. Given datasets of undirected simple graphs, the objective is to transform the network into a chordal structure by predicting the edge additions generated by a Maximum-Cardinality-Search-based triangulation (such as [7] implemented in the Python NetworkX library). This presents a unique graph learning dynamic: rather than hunting for the absolute, NP-hard minimum-cardinality completion, the model must learn to approximate a specific computational heuristic. The primary difficulty lies in accurately replicating an algorithmic process that produces a minimal set of non-redundant edges, rather than the absolute smallest possible fill, meaning the ground-truth objective itself is inherently heuristic rather than an exact mathematical optimum.

Algebraic Connectivity Maximisation: Addressing the chronic under-capacity of UK power grids poses a critical, highly complex global optimisation challenge. By modelling the grid as an edge-attributed directed multigraph, where edges represent electrical capacity. The core task is to apply specific graph transformations—adding new edges—to maximise the objective function of algebraic connectivity λ_2, the second smallest eigenvalue of the Laplacian. As a graph transformation learning problem, this is exceptionally difficult; maximising λ_2 is NP-hard, and models struggle to learn the global objective because training on too few edges fails to fully exploit optimisation capabilities, while training on too many leads to arbitrary selections due to the diminishing returns from later edge additions. This challenge is further compounded by extremely limited real-world data from the Power Graph Cascades dataset [44], which provides just four small UK power grid networks (three for training, one for evaluation), ranging from fewer than 100 to slightly over 400 nodes, culminating in the strict evaluation requirement to optimise the network's capacity by adding exactly 5 edges.

Graph Denoising: Filtering unwanted noise from structured datasets is essential for reliable data imputation, standardisation, and stability. Given the medium-sized ConceptNet dataset of directed simple graphs, the objective is to transform the network by identifying and removing erroneous edges, replicating the GOLD model's denoising process without relying on the original pre-mined rules [15]. As a graph learning problem, this tests both topological reasoning and system extensibility. The primary difficulty lies not just in predicting structural anomalies based on connectivity, but in successfully integrating external, pre-trained BERT embeddings within a standardised framework to evaluate the model's capacity for multimodal feature processing. An example application is predicting whether a user will retweet a post in a microblogging network based on social influence patterns within the user's ego network [49].

Retweet Prediction: Anticipating user interactions is a foundational mechanism for driving recommender systems in large-scale social networks. Given [12], the SNAP Higgs Twitter Retweet Network dataset—a massive directed simple graph with edge features—the objective is to predict the existence of a directed edge between two users, indicating whether one actively retweeted the other. The primary graph learning difficulty here stems directly from computational complexity: efficiently processing a continuous, real-world network with over 250,000 nodes demands rigorous memory management and highly scalable message passing, severely testing a model's ability to maintain predictive accuracy at scale.

4 Task Categorisation

Having defined all 16 tasks, we now categorise them in the following table according to the criteria outlined in Section 2.

Table 1: Summary of Task Complexities and Structural Requirements

Task	*Complexity*	*I-O Relation*	*Graph Type*	*Structural Changes*
Symmetric Closure Completion	Polynomial [39]	Deterministic	*Directed* simple graphs	*Edge* Structure Completion – Add Only
Transitive Closure Completion	Polynomial [11]	Deterministic	*Directed* simple graphs	*Edge* Structure Completion – Add Only
Shortcut Elimination	Polynomial [2]	Deterministic	*Directed* simple graphs	*Edge* Structure Completion – Delete Only
Shape Completion	Polynomial [4]	Deterministic	Undirected simple graphs	*Edge* Structure Completion – Add Only
Molecular Bond Inference	Polynomial [11]	Deterministic	*Undirected, attributed multigraphs*	*Edge* Attribute Computation – Classification
Node Core Number Computation	Polynomial [6]	Deterministic	Undirected simple graphs with *node attributes*	*Node* Attribute Computation – Regression
Intra-Community Edge Prediction	Polynomial [14]	Deterministic	Undirected simple graphs with *edge attributes*	*Edge* Attribute Computation – Classification
Edge Betweenness Centrality Computation	Polynomial [9]	Deterministic	Undirected simple graphs with *edge attributes*	*Edge* Attribute Computation – Regression
Feedback Edge Set Identification	Polynomial [1]	Optimisation	Undirected simple graphs with *edge attributes*	*Edge* Attribute Computation – Classification
Spanning Tree Identification	Polynomial [28]	Nondeterministic	Undirected simple graphs	*Edge* Structure Completion – Delete Only
Graph Colouring	NP-hard [27]	Nondeterministic; can be deterministic	Undirected simple graphs with *node attributes*	*Node* Attribute Computation – Classification

Continued on next page

Table 1 – continued from previous page

Task	*Complexity*	*I-O Relation*	*Graph Type*	*Structural Changes*
Minimum Chordal Graph Completion	NP-hard [46]	Optimisation	Undirected simple graphs	*Edge* Structure Completion – Add Only
Estimated Minimum Chordal Graph Completion	Polynomial [7]	Heuristic optimisation	Undirected simple graphs	*Edge* Structure Completion – Add Only
Algebraic Connectivity Maximisation	NP-hard [36]	Optimisation	Undirected *multi-graphs*	*Edge* Structure Completion – Add Only
Graph Denoising	NP-hard [47]	Deterministic	*Directed* simple graphs	*Edge* Structure Completion – Delete Only
Retweet Prediction	Polynomial [33]	Deterministic	*Directed* simple graphs with *edge features*	*Edge* Attribute Computation – Classification

5 Discussion and Further Research

Analysing the distribution of this initial task set reveals specific areas where the first version of the taxonomy requires future expansion. The categorisation results show that we have not defined any tasks with a probabilistic input-output relation, nor any that require hypergraphs. Only one task—Estimated Minimum Chordal Graph Completion—performs heuristic optimisation. There is only one task each for node classification, node regression, and edge regression; all other tasks fall under edge classification. With the partial exception of Molecular Bond Inference, which addresses edge typing via edge-attribute classification, all tasks operate on homogeneous graphs. Spanning Tree Identification is one of two nondeterministic tasks; however, optimisation tasks can also be viewed as constrained instances of a nondeterministic input-output relation.

By categorising 16 diverse, illustrative tasks, we aim to lay the groundwork for a standardised, extensible foundation that facilitates reliable and reproducible research. Our idea is to develop a unified ML interface that accepts task specifications defined by input graphs and corresponding output graphs. The pipeline is currently in the prototyping stage and supports both static graph access and dataset generation, covering all defined tasks. Pending work involves, but is not limited to, custom feature registration, automatic standardised feature computation, support for all complexity classes, required graph types, input-output relationships, and required structural changes outlined in Table 1.

This is where we are looking to involve the wider community. If you are currently working on or have in mind any tasks that align with our outlined categorisation criteria, please email us your suggestions and any limitations you encounter or are aware of. Edge classification and node attribute computation tasks are likely to be more interesting to us in the near future, whereas node-level transformation and NECT tasks will be of interest in the long term.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Agrawal, A., Panolan, F., Saurabh, S., Zehavi, M.: Simultaneous feedback edge set: A parameterized perspective. Algorithmica **83**(2), 753–774 (Feb 2021). https://doi.org/10.1007/s00453-020-00773-9, https://doi.org/10.1007/s00453-020-00773-9
2. Aho, A.V., Garey, M.R., Ullman, J.D.: The transitive reduction of a directed graph. SIAM Journal on Computing **1**(2), 131–137 (1972). https://doi.org/10.1137/0201008, https://doi.org/10.1137/0201008
3. Al-garadi, M.A., Varathan, K.D., Ravana, S.D.: Identification of influential spreaders in online social networks using interaction weighted k-core decomposition method. Physica A: Statistical Mechanics and its Applications **468**, 278–288 (2017). https://doi.org/https://doi.org/10.1016/j.physa.2016.11.002, https://www.sciencedirect.com/science/article/pii/S0378437116308068
4. Alon, N., Yuster, R., Zwick, U.: Finding and counting given length cycles. Algorithmica (New York) **17**, 209–223 (03 1997). https://doi.org/10.1007/BF02523189
5. Batagelj, V., Zaversnik, M.: An o(m) algorithm for cores decomposition of networks (2003), https://arxiv.org/abs/cs/0310049
6. Batagelj, V., Zaversnik, M.: An o(m) algorithm for cores decomposition of networks. CoRR **cs.DS/0310049** (2003), http://arxiv.org/abs/cs/0310049
7. Berry, A., Blair, J., Heggernes, P., Peyton, B.: Maximum cardinality search for computing minimal triangulations of graphs. Algorithmica **39**, 287–298 (08 2004). https://doi.org/10.1007/s00453-004-1084-3
8. Borndörfer, R., Eisenblätter, A., Grötschel, M., Martin, A.: Frequency assignment in cellular phone networks. Annals of Operations Research **76**, 73–93 (Jan 1998). https://doi.org/https://doi.org/10.1023/a:1018908907763
9. Brandes, U.: A faster algorithm for betweenness centrality*. The Journal of Mathematical Sociology **25**(2), 163–177 (2001). https://doi.org/10.1080/0022250X.2001.9990249, https://doi.org/10.1080/0022250X.2001.9990249
10. Campbell, G.: Efficient graph rewriting (2021), https://arxiv.org/abs/1906.05170
11. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, Third Edition. The MIT Press, 3rd edn. (2009), [Molecular Bond Inference: Sect. 22.2, Transitive Closure: Sect. 25.2]
12. De Domenico, M., Lima, A., Mougel, P., Musolesi, M.: The anatomy of a scientific rumor. Scientific Reports **3**(1) (Oct 2013). https://doi.org/10.1038/srep02980, http://dx.doi.org/10.1038/srep02980
13. Dehof, A.K., Rurainski, A., Bui, Q.B.A., Böcker, S., Lenhof, H.P., Hildebrandt, A.: Automated bond order assignment as an optimization problem. Bioinformatics **27**(5), 619–625 (01 2011). https://doi.org/10.1093/bioinformatics/btq718, https://doi.org/10.1093/bioinformatics/btq718
14. Delling, D., G?rke, R., Nikoloski, Z., Gaertler, M., Brandes, U., Wagner, D., Hoefer, M.: On Modularity Clustering . IEEE Transactions on Knowledge & Data Engineering **20**(02), 172–188 (Feb 2008), https://doi.ieeecomputersociety.org/10.1109/TKDE.2007.190689

15. Deng, Z., Wang, W., Wang, Z., Liu, X., Song, Y.: Gold: A global and local-aware denoising framework for commonsense knowledge graph noise detection (2023), https://arxiv.org/abs/2310.12011
16. Dinella, E., Dai, H., Li, Z., Naik, M., Song, L., Wang, K.: Hoppity: Learning graph transformations to detect and fix bugs in programs. In: International Conference on Learning Representations (2020), https://openreview.net/pdf?id=SJeqs6EFvB
17. Errica, F., Podda, M., Bacciu, D., Micheli, A.: A fair comparison of graph neural networks for graph classification. CoRR **abs/1912.09893** (2019), http://arxiv.org/abs/1912.09893
18. Even, G., Naor, J.S., Schieber, B., Sudan, M.: Approximating minimum feedback sets and multi-cuts in directed graphs. In: Balas, E., Clausen, J. (eds.) Integer Programming and Combinatorial Optimization. pp. 14–28. Springer Berlin Heidelberg, Berlin, Heidelberg (1995)
19. Even, G., Naor, J.S., Schieber, B., Sudan, M.: Approximating minimum feedback sets and multi-cuts in directed graphs. In: Balas, E., Clausen, J. (eds.) Integer Programming and Combinatorial Optimization. pp. 14–28. Springer Berlin Heidelberg, Berlin, Heidelberg (1995)
20. Fomin, F.V., Villanger, Y.: Subexponential parameterized algorithm for minimum fill-in (2011), https://arxiv.org/abs/1104.2230
21. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proceedings of the National Academy of Sciences **99**(12), 7821–7826 (2002). https://doi.org/10.1073/pnas.122653799, https://www.pnas.org/doi/abs/10.1073/pnas.122653799
22. Guo, X., Wang, S., Zhao, L.: Graph neural networks: Graph transformation. In: Wu, L., Cui, P., Pei, J., Zhao, L. (eds.) Graph Neural Networks: Foundations, Frontiers, and Applications, pp. 251–275. Springer Singapore, Singapore (2022)
23. Heckel, R., Taentzer, G.: Graph Transformation for Software Engineers. Springer International Publishing (2020). https://doi.org/https://doi.org/10.1007/978-3-030-43916-3, http://graph-transformation-for-software-engineers.org/
24. Helaly, A., Sakr, N., Madkour, K., Torunoglu, I.: Unsupervised graph neural network framework for balanced multipatterning in advanced electronic design automation layouts (2025), https://arxiv.org/abs/2511.16374
25. Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D.: Deep reinforcement learning that matters. CoRR **abs/1709.06560** (2017), http://arxiv.org/abs/1709.06560
26. Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by simulated annealing: An experimental evaluation; part ii, graph coloring and number partitioning. Operations Research **39**(3), 378–406 (1991), http://www.jstor.org/stable/171393
27. Karp, R.M.: Reducibility among Combinatorial Problems, pp. 85–103. Springer US, Boston, MA (1972). https://doi.org/10.1007/978-1-4684-2001-2_9, https://doi.org/10.1007/978-1-4684-2001-2_9
28. Kruskal, J.B.: On the shortest spanning subtree of a graph and the traveling salesman problem (1956), https://api.semanticscholar.org/CorpusID:120068278
29. Labute, P.: On the perception of molecules from 3d atomic coordinates. Journal of Chemical Information and Modeling **45**(2), 215–221 (Jan 2005). https://doi.org/https://doi.org/10.1021/ci049915d
30. Leskovec, J., Mcauley, J.: Learning to discover social circles in ego networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural

Information Processing Systems. vol. 25. Curran Associates, Inc. (2012), https://proceedings.neurips.cc/paper_files/paper/2012/file/7a614fd06c325499f1680b9896beedeb-Paper.pdf
31. Li, C., Ueno, M.: An extended depth-first search algorithm for optimal triangulation of bayesian networks. International Journal of Approximate Reasoning **80**, 294–312 (2017). https://doi.org/https://doi.org/10.1016/j.ijar.2016.09.012, https://www.sciencedirect.com/science/article/pii/S0888613X16301645
32. Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: Data-driven traffic forecasting (2018), https://arxiv.org/abs/1707.01926
33. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the American Society for Information Science and Technology **58**(7), 1019–1031 (2007). https://doi.org/https://doi.org/10.1002/asi.20591, https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.20591
34. Lucic, M., Kurach, K., Michalski, M., Gelly, S., Bousquet, O.: Are gans created equal? a large-scale study (2018), https://arxiv.org/abs/1711.10337
35. Lynch, N.A.: Distributed Algorithms. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1996)
36. Mosk-Aoyama, D.: Maximum algebraic connectivity augmentation is np-hard. Oper. Res. Lett. **36**(6), 677–679 (Nov 2008). https://doi.org/10.1016/j.orl.2008.09.001, https://doi.org/10.1016/j.orl.2008.09.001
37. Pascual, R., Le Gall, P., Arnould, A., Belhaouari, H.: Topological consistency preservation with graph transformation schemes. Science of Computer Programming **214**, 102728 (2022). https://doi.org/https://doi.org/10.1016/j.scico.2021.102728, https://www.sciencedirect.com/science/article/pii/S0167642321001210
38. Plump, D.: Reasoning about graph programs (2016). https://doi.org/10.4204/EPTCS.225.6, https://doi.org/10.4204/EPTCS.225.6, publisher Copyright: © D. Plump.
39. Rosen, K.H.: Discrete mathematics and its applications. WCB/McGraw-Hill, 7th edn. (2012), [Symmetric Closure: Sect. 9.3, 9.4]
40. Rossetti, G., Guidotti, R., Miliou, I., Pedreschi, D., Giannotti, F.: A supervised approach for intra-/inter-community interaction prediction in dynamic social networks. Social Network Analysis and Mining **6**(1), 86 (09/2016 2016). https://doi.org/10.1007/s13278-016-0397-y, http://dx.doi.org/10.1007/s13278-016-0397-y
41. Shao, Z., Zhang, Z., Wang, F., Xu, Y.: Pre-training enhanced spatial-temporal graph neural network for multivariate time series forecasting. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. p. 1567–1577. KDD '22, ACM (Aug 2022). https://doi.org/10.1145/3534678.3539396, http://dx.doi.org/10.1145/3534678.3539396
42. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of graph neural network evaluation. CoRR **abs/1811.05868** (2018), http://arxiv.org/abs/1811.05868
43. Tarjan, R.: Depth-first search and linear graph algorithms. SIAM Journal on Computing **1**(2), 146–160 (1972). https://doi.org/10.1137/0201010, https://doi.org/10.1137/0201010
44. Varbella, A., Amara, K., Gjorgiev, B., El-Assady, M., Sansavini, G.: Powergraph: A power grid benchmark dataset for graph neural networks (2024), https://arxiv.org/abs/2402.02827

45. Velickovic, P., Blundell, C.: Neural algorithmic reasoning. CoRR **abs/2105.02761** (2021), https://arxiv.org/abs/2105.02761
46. Yannakakis, M.: Computing the minimum fill-in is np-complete. SIAM Journal on Algebraic Discrete Methods **2**(1), 77–79 (1981). https://doi.org/10.1137/0602010, https://doi.org/10.1137/0602010
47. Yannakakis, M.: Edge-deletion problems. SIAM Journal on Computing **10**(2), 297–309 (1981). https://doi.org/10.1137/0210021, https://doi.org/10.1137/0210021
48. You, J., Ying, R., Ren, X., Hamilton, W.L., Leskovec, J.: Graphrnn: Generating realistic graphs with deep auto-regressive models (2018), https://arxiv.org/abs/1802.08773
49. Zhang, J., Tang, J., Li, J., Liu, Y., Xing, C.: Who influenced you? predicting retweet via social influence locality. ACM Trans. Knowl. Discov. Data **9**(3) (Apr 2015). https://doi.org/10.1145/2700398, https://doi.org/10.1145/2700398