Towards a Theory of Machine Learning on Graphs and its Applications in Combinatorial Optimization

Christopher Morris
RWTH Aachen University

Abstract

Machine learning on graphs, especially using graph neural networks (GNNs), has seen a surge in interest due to the wide availability of graph data across many disciplines, from life and physical to social and engineering sciences. Despite their practical success, our theoretical understanding of the properties of GNNs remains incomplete. Here, we survey the author’s and his collaborators’ progress in developing a deeper theoretical understanding of GNNs’ expressive power and generalization abilities. In addition, we overview recent progress in using GNNs to speed up solvers for hard combinatorial optimization tasks.

1 Introduction

Graphs serve as powerful mathematical representations, capturing intricate interactions among entities across a spectrum of disciplines, spanning from life and physical to social and engineering sciences [Easley and Kleinberg, 2010] and optimization [Cappart et al., 2021]. This diversity underlines the critical demand for developing principled machine-learning methods that extract valuable patterns from complex graph data.

Hence, in recent years, neural networks capable of handling graph-structured data received a lot of attention in the machine learning community, especially message-passing graph neural networks (MPNNs) [Gilmer et al., 2017; Scarselli et al., 2009], or more generally, graph neural networks (GNNs). Nowadays, MPNNs and GNNs are among the most prominent topics at top-tier machine learning conferences,2 and have showcased promising outcomes across diverse domains, including breakthroughs in discovering new antibiotics [Stokes et al., 2020; Wong et al., 2023].

While GNNs are successful in practice and are making real-world impact, their theoretical properties are understood to a lesser extent. Here, we outline the author’s and his collaborator’s recent progress in understanding GNNs’ expressive power and generalization abilities. In addition, we outline how the principled design of GNN architectures can aid in speeding up exact solvers for hard combinatorial optimization problems.

1.1 Background

In the following, we introduce notation and provide the necessary background.

Let \( \mathbb{N} := \{1, 2, 3, \ldots\} \). For \( n \geq 1 \), let \( [n] := \{1, \ldots, n\} \subset \mathbb{N} \). We use \( \mathcal{L} \) to denote multisets, i.e., the generalization of sets allowing for multiple instances for each of its elements.

Graphs. An (undirected) graph \( G \) is a pair \((V(G), E(G))\) with finite sets of vertices or vertices \( V(G) \) and edges \( E(G) \subseteq \{\{u, v\} \mid u \neq v\} \). For ease of notation, we denote an edge \( \{u, v\} \in E(G) \) by \( (u, v) \) or \( (v, u) \). The neighborhood of \( v \in V(G) \) is denoted by \( N(v) := \{u \in V(G) \mid (v, u) \in E(G)\} \) and the degree of a vertex \( v \) is \( |N(v)| \). A (vertex-labeled graph \( G \) is a triple \((V(G), E(G), \ell)\) with a (vertex-)label function \( \ell: V(G) \to \mathbb{N} \). Then \( \ell(v) \) is a label of \( v \), for \( v \in V(G) \). Two graphs \( G \) and \( H \) are isomorphic, and we write \( G \cong H \) if there exists a bijection \( \varphi: V(G) \to V(H) \) preserving the adjacency relation, i.e., \((u, v)\) is in \( E(G) \) if, and only if, \((\varphi(u), \varphi(v))\) is in \( E(H) \). Then \( \varphi \) is an isomorphism between \( G \) and \( H \). In the case of labeled graphs, we additionally require that \( \ell(v) = \ell(\varphi(v)) \) for all \( v \in V(G) \).

The 1-dimensional Weisfeiler–Leman algorithm. The 1-WL or color refinement is a well-studied heuristic for the graph isomorphism problem, originally proposed by Weisfeiler and Leman [1968]. Intuitively, the algorithm determines if two graphs are non-isomorphic by iteratively coloring or labeling vertices. Formally, let \( G = (V(G), E(G), \ell) \) be a labeled graph. In each iteration, \( t > 0 \), the 1-WL computes a vertex coloring \( C_t^1: V(G) \to \mathbb{N} \), depending on the coloring of the neighbors. That is, in iteration \( t > 0 \), we set

\[
C_t^1(v) := \text{RELABEL}\left(\left(C_{t-1}^1(v), \mathcal{L}(C_{t-1}^1(u) \mid u \in N(v))\right)\right),
\]

for all vertices \( v \in V(G) \), where \( \text{RELABEL} \) injectively maps the above pair to a unique natural number, which has not been used in previous iterations. In iteration 0, the coloring \( C_0^1 := \ell \) is used. To test whether two graphs \( G \) and \( H \) are non-isomorphic, we run the above algorithm in “parallel” on both graphs. If the two graphs have a different number of vertices colored \( c \in \mathbb{N} \) at some iteration, the 1-WL distinguishes the graphs as non-isomorphic. Moreover, if the number of colors

---

1We use the term MPNNs to refer to graph-machine learning architectures that fit into the framework of Gilmer et al. [2017], see also Section 1.1, and use the term GNNs in a broader sense, i.e., all neural network architectures capable of handling graph-structured inputs.

2http://tinyurl.com/mpn89vju
between two iterations, \(t\) and \((t + 1)\), does not change, i.e., the cardinalities of the images of \(C^1_t\) and \(C^1_{t+1}\) are equal, the algorithm terminates.

The 1-WL has clear limitations in distinguishing non-isomorphic graphs [Arvind et al., 2015; Cai et al., 1992; Morris et al., 2023a]. However, it cannot distinguish between a pair of non-isomorphic \(d\)-regular graphs. Hence, to overcome 1-WL’s limitations, the \(k\)-dimensional Weisfeiler–Leman algorithm (\(k\)-WL) operates on \(k\)-tuples over the set of vertices of a given graph instead of vertices, leading to a strictly more powerful algorithm with increasing \(k\) [Cai et al., 1992].

**Message-passing graph neural networks.** Intuitively, MPNNs learn a vectorial representation, i.e., a \(d\)-dimensional real-valued vector, representing each vertex in a graph by aggregating information from neighboring vertices. Formally, let \(G = (V(G), E(G), \ell)\) be a labeled graph with initial vertex features \(h^{(0)}_v \in \mathbb{R}^d\) for all \(v \in V(G)\), that are consistent with \(\ell\).

That is, each vertex \(v\) is annotated with a feature \(v^{(0)} \in \mathbb{R}^d\) such that \(h^{(0)}_v = h^{(0)}_u\) if, and only, if \(\ell(v) = \ell(u)\). An example is a one-hot encoding of the labels \(\ell(u)\) and \(\ell(v)\). An MPNN architecture consists of a stack of neural network layers, i.e., a composition of permutation-equivariant parameterized functions. Following, Scarselli et al. [2009] and Gilmer et al. [2017], in each layer, \(t > 0\), we compute vertex features

\[
\mathbf{h}^{(t)}_v := \text{UPD}^{(t)}(h_v^{(t-1)}, \text{AGG}^{(t)}(\|v^{(t-1)}_u\| \mid u \in \mathcal{N}(v))) \in \mathbb{R}^d,
\]

for \(v \in V(G)\), where \(\text{UPD}^{(t)}\) and \(\text{AGG}^{(t)}\) may be parameterized functions, e.g., neural networks. In the case of graph-level tasks, e.g., graph classification, one uses

\[
\mathbf{h}_G := \text{READOUT}(\|h^{(L)}_v\| \mid v \in V(G)) \in \mathbb{R}^d,
\]

to compute a single vectorial representation based on learned vertex features after iteration \(L\). Again, \(\text{READOUT}\) may be a parameterized function. To adapt the parameters of the above three functions, they are optimized end-to-end, usually through a variant of stochastic gradient descent, e.g., Kingma and Ba [2015], together with the parameters of a neural network used for classification or regression.

**2 The Expressive Power of GNNs**

The expressivity of a GNN is the architecture’s ability to express or approximate different functions over a domain, e.g., graphs. High expressivity means the neural network can represent many functions over this domain. In the literature, the expressivity of GNNs or MPNNs is modeled mathematically based on two main approaches: algorithmic alignment with graph isomorphism test [Morris et al., 2023b] and universal approximation theorems [Azizian and Lelarge, 2021; Geerts and Reutter, 2022]. Works following the first approach study if a GNN, by choosing appropriate weights, can distinguish the same pairs of non-isomorphic graphs as the 1-WL or the \(k\)-WL. Here, a GNN distinguishes two non-isomorphic graphs if it can compute different vectorial representations for the two graphs.

Concurrently with Xu et al. [2019], Morris et al. [2019] showed that any MPNN’s expressive power is upper bounded by the 1-WL in distinguishing non-isomorphic graphs. That is, given two non-isomorphic graphs, for any choice of functions \(\text{UPD}^{(t)}\) and \(\text{AGG}^{(t)}\) and all possible parameter choices, the MPNN is not able to learn vertex features distinguishing two graphs if the 1-WL cannot distinguish them. Let \(W^{(t)}\) denote the set of parameters up to layer \(t\). We can write the above down as follows.

**Theorem 1.** Let \(G = (V(G), E(G), \ell)\) be a labeled graph. Then for all \(t \geq 0\), choices of initial colorings consistent with \(\ell\), choices of \(\text{UPD}^{(t)}\) and \(\text{AGG}^{(t)}\), and weights \(W^{(t)}\),

\[
C^1_t(u) = C^1_t(v) \text{ implies } h^{(t)}_u = h^{(t)}_v,
\]

for all vertices \(u\) and \(v\) in \(V(G)\).

The above results can easily be extended to distinguishing graphs instead of vertices. On the positive side, Morris et al. [2019] proved that there exists a sequence of parameter matrices \(W^{(t)}\) such that MPNNs have the same expressive power as the 1-WL algorithm by deriving injective variants of the functions \(\text{UPD}^{(t)}\) and \(\text{AGG}^{(t)}\).

**Theorem 2.** Let \(G = (V(G), E(G), \ell)\) be a labeled graph. Then for all \(t \geq 0\), there exists an initial coloring consistent with \(\ell\), a sequence of weights \(W^{(t)}\), choices of \(\text{UPD}^{(t)}\) and \(\text{AGG}^{(t)}\), such that

\[
C^1_t(u) = C^1_t(v) \text{ if, and only, if } h^{(t)}_u = h^{(t)}_v,
\]

for all vertices \(u\) and \(v\) in \(V(G)\).

We note here that the result of Morris et al. [2019] is stronger than the one by Xu et al. [2019] as the former results shows a polynomial number of parameters in the number of vertices of the input graph is sufficient. See Grohe [2021] for an in-depth discussion of both approaches and Aamand et al. [2022]; Amir et al. [2023]; Bravo et al. [2024] for further refinements of the above result.

Subsequently, the authors developed GNN architectures overcoming the limitations of MPNNs, e.g., by devising GNN architectures aligned with the \(k\)-WL [Morris et al., 2019] or utilizing subgraph graph information [Qian et al., 2022]. In Morris et al. [2020, 2023b], he developed scalable variants of the \(k\)-WL and corresponding neural architectures that take the sparsity of the underlying graph into account while allowing for a fine-grained tradeoff between expressivity and scalability, also resulting in good predictive performance on real-world benchmarks, often outperforming standard MPNNs.

**Refined notions of expressive power.** While the graph isomorphism perspective has helped the community understand MPNNs’ ultimate limitations in capturing graph structure, it is inherently binary. For example, it does not give insights into the degree of similarity between two given graphs, prohibiting a more fine-grained analysis. Hence, in some recent work, Boiker et al. [2023] developed a more fine-grained analysis based on graph distances or pseudo-metrics on the set of graphs, or more generally, graphons. Intuitively, we leveraged known graph distances or, more precisely, pseudo-metrics on the set of graphs and showed that two graphs are close regarding tree distance [Boiker, 2021] if, and only, if MPNNs’ outputs on the graphs \(G\) and \(H\) are close regarding the 2-norm, resulting in the following result.
Theorem 3 (Informal). The following are equivalent for all graphs $G$ and $H$.
1. The graphs $G$ and $H$ are close in the tree distance.
2. MPNN outputs on the graphs $G$ and $H$ are close for all MPNNs with Lipschitz constant $C$ and $L$ layers.

Understanding the expressive power of graph transformers. Other recent works of the author and his collaborators include studying the expressive power of graph transformer architectures [Müller et al., 2024]. Intuitively, graph transformer architectures can be viewed as an MPNN operating on a complete graph using an attention mechanism. However, to capture any non-trivial graph structures, graph transformers need to be equipped with so-called positional or structural encodings [Müller et al., 2024], complicating their analysis of expressive power. To address this, we show that the recently proposed edge transformer [Bergen et al., 2021], a global attention model operating on vertex pairs instead of vertices, has the expressive power of the 3-WL. Empirically, we demonstrated that the edge transformer surpasses other theoretically aligned architectures regarding predictive performance and is competitive with state-of-the-art models on algorithmic reasoning [Velickovic et al., 2022] and molecular regression tasks while not relying on positional or structural encodings. Additionally, leveraging a known connection between fragments of first-order logic and the 3-WL, we explain why the edge transformer is capable of solving systematic generalization problems [Bergen et al., 2021]. Systematic (or compositional) generalization refers to the ability of a model to generalize to complex novel concepts by combining primitive concepts observed during training, posing a challenge to even the most advanced models such as GPT-4 [Dziri et al., 2023]. In addition, in Müller and Morris [2024], we devised general techniques to design graph transformers that have the same expressive power of the $k$-WL, improving over initial attempts [Kim et al., 2021, 2022] in terms of running time and also showing promising predictive performance.

3 The Generalization Abilities of GNNs

While understanding MPNNs’ and related architectures’ expressive power is essential, understanding when such architectures generalize to unseen graphs and how to find parameter assignments that allow so is equally important. However, in MPNNs and GNNs, the essential aspects of generalization are severely understudied. Here, generalization refers to a trained GNN’s ability to make predictions for data not seen during the training accurately.

In an attempt to understand the generalization properties of MPNNs, in a recent paper, Morris et al. [2023a] studied MPNNs’ generalization abilities via classical VC dimension theory [Vapnik and Chervonenkis, 1964; Vapnik, 1998]. Here, for a class $C$ of MPNNs and a set of $X$ of graphs, $VC-dim_X(C)$ is the maximal number $m$ of graphs $G_1, \ldots, G_m$ in $X$ that can be shattered by $C$. Here, $G_1, \ldots, G_m$ are shattered if for any $\tau$ in $\{0, 1\}^m$ there exists an MPNN MPNN in $C$ such that for all $i$ in $[m]$: 
$$MPNN(G_i) = \begin{cases} \geq 2/3 & \text{if } \tau_i = 1, \\ \leq 1/3 & \text{if } \tau_i = 0. \end{cases}$$

The paper showed that MPNNs’ generalization abilities are tightly linked to their expressive power, summarized in the following result.

Theorem 4 (Informal). When considering input graphs of a given size, the VC dimension of MPNNs is lower and upper bounded by the number of graphs the 1-WL can distinguish.

Hence, by classical learning theoretic results [Vapnik, 1995], the above result directly links an architecture’s expressive power and its generalization power. Unlike previous results, e.g. Garg et al. [2020]; Scarselli et al. [2018], it accounts for non-trivial graph structure. Moreover, the paper also studies how MPNNs’ generalization properties are connected to the bit length of MPNNs’ parameters, i.e., the maximum number of bits needed to represent each parameter assignment. Lastly, the paper also shows a novel connection between the number of colors 1-WL induces over each graph in a given set of graphs and the VC dimension of MPNNs.

In a more recent work [Franks et al., 2024], we studied the question of when more expressive GNN architectures, e.g., based on injecting subgraph information into MPNNs [Bouritsas et al., 2020], improve generalization performance. In practice, it is observed that more expressive architectures lead to better generalization. However, Theorem 4 does not explain this. That is, a more expressive GNN results in a higher VC dimension, leading to worse generalization. Hence, we employ classical margin theory [Mohri et al., 2012] to investigate the conditions under which an architecture’s increased expressivity aligns with improved generalization performance. In addition, we show that gradient flow, gradient descent continuous variant, pushes the MPNN’s weights toward the maximum margin solution. Further, we introduce variations of expressive MPNN architectures with provable generalization properties.

4 MPNNs for Combinatorial Optimization

Machine learning, particularly MPNNs, has recently gained traction in enhancing exact optimization algorithms; see Cappart et al. [2021]. For example, MPNNs can speed up exact solvers for mixed-integer linear programs (MIPs) by encoding the MIP as a bipartite graph; see below; and learn to imitate computational intensive heuristics like strong branching, which entails solving multiple linear optimization problems (LPs) [Gasse et al., 2019].

In Khalil et al. [2022], we generalize the approach of [Gasse et al., 2019] to work for a large set of essential heuristics for solving MIPs. That is, we introduce the MIP-GNN architecture, a general MPNN-based framework for guiding state-of-the-art branch-and-cut solvers on (binary) MIPs. Specifically, we encode the variable-constraint interactions of MIPs as a bipartite graph where a pair of variable/constraint vertices share an edge if the variable has a non-zero coefficient in the constraint. To guide a solver in finding a solution or certifying optimality faster for a given instance, we train the MPNN to predict variable biases [Hsu et al., 2008], which are computed by component-wise averaging over a set of near-optimal solutions of a given MIP. Intuitively, these biases encode how likely it is for a variable to take a value of 1 in near-optimal solutions. To tailor the MPNN more closely to the task of variable bias prediction, we propagate a “residual error,” indicating how much
the current assignment violates the constraints.

We integrated such trained MPNNs into a state-of-the-art MIP solver, namely CPLEX IBM [2020], by using the MPNN’s variable bias prediction in crucial tasks within the branch-and-cut algorithm, such as node selection and warm-starting. On a large set of diverse, real-world binary MIPs, modeling the generalized independent set problem [Colombi et al., 2017] and a fixed-charge multi-commodity network flow problem [Hewitt et al., 2010], we show significant improvements over default CPLEX for the task of node selection, while also reporting promising performance for warm-starting and branching variable selection. This is achieved without feature engineering, i.e., by relying purely on the graph information induced by the given MILP.

Crucially, for the first time in this line of research, we used a single, once-trained model for bias prediction to speed up multiple components of the CPLEX simultaneously. In other words, we show that learning the bias associated with sets of near-optimal solutions is empirically beneficial to multiple crucial MIP ingredients.

Despite the empirical success, the reasons behind MPNNs’ effectiveness in, e.g., predicting strong branching scores, remain unclear. Hence, in an attempt to explain the empirical findings in [Gasse et al., 2019; Khalil et al., 2022], in Qian et al. [2024], we show that MPNNs can simulate standard interior-point methods [Gondzio, 2012; Nocedal and Wright, 2006] (IPMs) for LPs, explaining their practical success in the above two works. IPMs are algorithms for solving constrained optimization problems. They are particularly efficient for linear optimization, where they were first developed as a (polynomial-time) alternative to the Simplex methods. Variants of the algorithm differ in theoretical guarantees and empirical performance but revolve around the same core approach [Shanno, 2012]. In short, the LP to solve is replaced by a perturbed family of problems where a barrier penalty has replaced hard constraints with a parameter \( \mu > 0 \). IPMs alternate between taking a Newton step to solve this perturbed problem and decreasing this parameter \( \mu > 0 \), eventually converging to the optimal solution of the original problem.

In Qian et al. [2024], we show that there exists an MPNN and corresponding parameter assignments such that \( O(m) \) message-passing steps, where \( m \) denotes the number of constraints of the LP instance, reproduce an iteration of an IPM method for solving LPs. Furthermore, we highlight how MPNNs can serve as a lightweight proxy for solving LPs, adapting to a given problem instance distribution. Empirically, we show that MPNNs solve LP relaxations of standard combinatorial optimization problems close to optimality, often surpassing conventional solvers and competing approaches in solving time.

While the above-described works have deepened our understanding of machine learning on graphs, many challenges persist. Regarding expressive power, as outlined above, most current approaches heavily rely on combinatorial practices, such as 1-WL. While the graph isomorphism perspective has helped the community understand GNNs’ ultimate limitations in capturing graph structure, it is inherently binary. For example, it does not give insights into the degree of similarity between two given graphs, prohibiting a more fine-grained analysis. While our recent work [Böker et al., 2023] and others [Chen et al., 2022] aimed at a more fine-grained analysis, they still have substantial limitations, such as large constants.

A second limitation of current GNN expressivity results in their lack of specificity. They are often tailored to particular classes of GNNs, overlooking practical and relevant architectural choices, and are not aligned with architectures and engineering tricks used in practice. It is crucial to address the need for more specific and expressive GNNs, especially in light of competing approaches’ expressive power, e.g., graph transformer architectures [Müller et al., 2024]. Thirdly, in the literature [Morris et al., 2023b], there is a large set of more expressive architectures overcoming the limitations of MPNNs. However, it is largely unclear when the expressivity of specific architectures is needed for a given learning task.

Regarding generalization, all current works, including the authors’, are based on variants of classical uniform generalization bounds. This entails very large and unrealistic constants in the bound and a description of a classical bias-variance curve that does not describe the reality of machine learning on graphs, in which higher complexity and more expressive models often generalize better. In addition, how to find a set of generalizing parameters using stochastic gradient descent is mostly unclear. The few works studying MPNNs’ optimization aspects are limited and often based on unrealistic assumptions, such as the use of linear activation functions [Xu et al., 2021; Franks et al., 2024], unrealistic learning scenarios [Du et al., 2019], or neglecting the influence of graph structure [Tang and Liu, 2023]; see also [Bechler-Speicher et al., 2023].

In terms of applying MPNNs in combinatorial optimization, we still need to understand better when MPNNs for predicting, e.g., strong branching scores, generalize to larger problem instances than trained ones. Moreover, while Qian et al. [2024] showed that MPNN could solve LPs, it remains to be seen if such MPNNs trained with stochastic gradient descent learn to execute IPMs.

6 Conclusion

Here, we outlined the authors’ and his collaborators’ recent progress in developing a theoretical understanding of machine learning with graphs. We outlined his recent progress in understanding the expressive power of MPNNs and related architectures and their ability to generalize beyond the training dataset. Moreover, we outlined some progress in leveraging MPNNs to replace costly heuristics within exact solvers for hard combinatorial optimization problems.
Acknowledgments

The works presented in this overview article were mostly done in collaboration with colleagues and students. The authors sincerely thanks them for the fun collaborations and interesting discussions. Christopher Morris is partially funded by a DFG Emmy Noether grant (468502433) and RWTH Junior Principal Investigator Fellowship under Germany’s Excellence Strategy.

References


J. Böker. Graph similarity and homomorphism densities. In ICALP, 2021.


