ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

SEMESTER PROJECT

Niels R. Holm & Povl Á. Klarlund
École Polytechnique Fédérale de Lausanne, Switzerland

10th of January 2025

1 INTRODUCTION

1.1 THE MAXIMUM INDEPENDENT SET PROBLEM

To make our later problem definitions clear, we will start out with some definitions from graph theory.

A graph $G = (V, E)$ consists of a set of nodes V and a set of edges E.
Edges are dyads or pairs of nodes (v_i, v_j), that signify connections between these nodes. Nodes that are connected are called **adjacent,** and the set of edges that include a node v is denoted as $E[v]$.
An induced subgraph $F = (S, E[S]) \subseteq G$ is a graph formed from a subset of nodes $S \subseteq V$.
A neighborhood of v is the subgraph induced by nodes adjacent to v, and the set of nodes in the neighborhood is called the **neighbors** - we will denote the set of neighbors $N(v) \subseteq V$. we consider the *open* neighborhood, i.e. $v \notin N(v)$.

Given a graph $G = (V, E)$, an **independent set** is a set of nodes S s.t. the subgraph it induces on G has no edges, i.e. $(S, G[S]) = (S, \emptyset) \subseteq G$.

A **maxim*al* independent set** S of G is an independent set to which no new nodes from V can be added without breaking independence, i.e. for all $v \in V$, either $v \in S$ or $N(v) \cap S \neq \emptyset$..

The **maxim*um* independent set** of G is its largest maxim*al* independent set.

For practical problems, the terms maximum and maximal independent set are often used *interchangeably*. This is because the (maxim*um*) problem is NP-hard, meaning it is infeasible to check if we have actually found the true optimum. Thus, for real networks, we only ever find maximal independent sets and empirically compare them with baseline methods and results from the literature, as this is the best proxy for the maximum independent set problem we currently have available.

Thus, the abbreviation **MIS** can refer to both maxim*um* and maxim*al*.

1.2 APPLYING LEARNING METHODS TO THE MIS PROBLEM

Representation learning, transfer learning and pre-training are methods that have yielded massive success in domains such as computer vision and natural language processing. We aim to apply these methods in the domain of graphs, and in particular to the problem of finding the MIS.

This leaves several open-ended questions to be answered, such as which graph structures encode a signal of the MIS size, and how do we learn to represent these structures? How do we decode an MIS from a full graph?

In this project, we attempt to leverage synthetically generated datasets of contrastive examples, where we know the true MIS sizes of all graphs. We train a model - the **Comparator** - that takes a pair of graphs, embeds each of them separately, and then tries to classify which of them have the higher MIS. This model is then applied to out-of-distribution real networks, where we can use it for a decoding process - namely, we want to recursively compare pairs of induced subgraphs of the full graph, and then select the subgraph with the highest MIS size until we arrive at an independent set - i.e. no edges remaining.

2 METHODS

The main task of our approach is to train a model that can determine which graph has a larger MIS given a pair of graphs. This is done using contrastive training examples inspired by the algorithm constructing an MIS.

2.1 CONSTRUCTING AN MIS

Consider a function CMP that takes two graphs A, B and is defined by

1. If MIS$(A) \geq$ MIS(B) , then CMP$(A, B) = 0$
2. If MIS$(A) <$ MIS(B) , then CMP$(A, B) = 1$

This function is what we refer to as a 'Comparator'. Given such a function, we may systematically construct an MIS. Following the work of previous semester projects, we initially used the following algorithm for this.

Algorithm 1 Comparator induced algorithm $\mathcal{A}^{\text{CMP}}$

Input: Graph $G = (V, E)$, Comparator CMP
Output: Maximal Independent Set.
while $V \neq \emptyset$ **do**
  Select vertex $v \in V$ with the smallest degree.
  Let $N(v) \subseteq V$ be the neighbors of v, and $E[v]$ be the edges incident to v.
  Define two subgraphs:
  - $A = (V \setminus \{v\}, E \setminus E[v])$ ▷ Remove just v.
  - $B = (V \setminus (\{v\} \cup N(v)), E \setminus (E[v] \cup E[(N(v)])$ ▷ Remove v and its neighbors.

  if CMP$(A, B) = 0$ **then** ▷ MIS$(A) >$ MIS(B)
    Set $G = A$.
  else
    Set $G = B$ and add v to the Maximal Independent Set.
  end if
end while
Return: The recovered Maximal Independent Set.

The idea of the algorithm is that if B has a larger MIS, then this indicates that v should be part of the MIS. However, we realized that this logic is flawed because in general $B \subset A$ and so it will always hold that MIS$(A) \geq$MIS(B). This invalidates the algorithm as an optimal CMP function will always output 0.

In Brusca et al. 2023, they pose the problem as a Dynamic Programming problem. Central to this is that they proof the following theorem:

Theorem 1: Let a graph $G(V, E) \in \mathcal{G}$. Then for any vertex $v \in V$ with $d(v) \geq 1$,

$$\text{MIS}(G) = \max(\text{MIS}(G/\mathcal{N}(v)), \text{MIS}(G/\{v\})).$$

This theorem suggests that graph B in algorithm 1 should actually be $B = (V \setminus N(v), E \setminus E(N(v))$. The paper states a very similar algorithm to algorithm 1, but with the proposed B.

2.2 Contrastive Data Generation Using Synthetic Graphs

Synthetic graphs propose a way to construct a labeled dataset. This is a key difference to Brusca et al. 2023 where real world graphs are used as training material. This difference is important and we discuss it in the Discussion section. In this project, we work with very specific types of graphs with known MIS size. These are listed in Table 2.1.

From the previous semester projects, two different generation approaches were available to us from the start. Both methods first construct a base dataset, $\mathcal{D}$, of graphs of the basic types and then build on top of that.

- **Heterogeneous dataset** For each graph in $\mathcal{D}$ extend the dataset by either (i) add a connected clique (ii) add an isolated node (iii) union with another random graph from $\mathcal{D}$. Then construct d pairs of graphs by sampling randomly from the extended dataset
- **Transformation dataset** Generate d graph pairs by selecting a random node and then generate two subgraphs by (i) deleting a node and (ii) a node and its neighbourhood. The result of these operations are described in Table 2.1

In this project, we mostly worked with the transformation-based dataset as this approach closely resembles the downstream comparator task.

Table 2.1
Maximum Independent Set (MIS) sizes for various graph types, along with the effect of graph modifications. Multiple entries means that the result depends on which node the operation is performed on

Graph Type	Initial	Node Deletion	Neighborhood Deletion
Clique (N)	1	1	1
Star (N)	$N-1$	$N-2$ (leaf), $N-1$ (center)	1
Empty (N)	N	$N-1$	N
Line (N)	$\lceil N/2 \rceil$	$\lfloor N/2 \rfloor$ (endpoint), Split (middle)	$\lceil (N-1)/2 \rceil + 1$
Anti-Communities (A, B)	B	B ($A > 1$), $B-1$ ($A = 1$)	B
Complete Bipartite (A, B)	$\max(A, B)$	$\max(A-1, B)$, $\max(A, B-1)$	A (left), B (right)

Balancing the dataset: Since our pre-training optimization problem is a binary classification problem, it is essential that the dataset distribution does not induce bias in the model.

Concretely, consider our fixed dataset of N contrastive pairs $\mathcal{D} := \{\big((G_i, y_i),(G_i', y_i')\big)\}_{i=1}^{N}$, where $y_k = \text{MIS}(G_k)$. Here, we want to make sure that we have an approximately equal number of pairs where $y_i < y_i'$ and $y_i > y_i'$. Practically, we see that the MIS classification problem is hard to learn, and it is easy for the models (in particular GCNN, GAT and Exphormer) to simply learn to predict the maximum cardinality class.

To mitigate this, we devised a procedure during data generation that ensures an approximately equal distribution between these cases. For each graph pair ($\underbrace{G_k}_{\text{index 0}}$, $\underbrace{G_k'}_{\text{index 1}}$) and label pair $(y_k^{(a)}, y_k^{(b)})$, where $y_k^{(a)} < y_k^{(b)}$, we flip a fair coin (draw a sample $b \sim \text{Ber}\left(p = \frac{1}{2}\right)$) and use the result to choose which index - 0 or 1 - should have the higher MIS label $y_k^{(b)}$. Then, we check if the labels match the graph MISes at the chosen indices, and otherwise swap the graphs. This yields a dataset that is close to balanced.

Ensuring good contrastive examples: Contrastive training aims to utilize pairs of examples that are clearly separated in the data space, to obtain a good separation in the latent space, leading to the model having a better internal representation for downstream tasks.

To achieve this, we thus want our synthetic graph pairs to be different from each other in terms of MIS, such that we maximize the signal of whether a graph is a "good" and "bad" choice for our model. This means that examples where the graphs in the sampled pair have equal MIS provide a poor signal, and depending on how we choose to handle the equal case, might even essentially be pure noise.

During data generation, we simply opt to drop pairs with equal MISes. We continue data generation until we have the desired number of pairs, as specified in the configuration.

For the data generation setting used predominantly in the project, as much as 46% of the generated pairs are dropped.

2.3 Graph embedding models

As stated when introducing the problem, we have an open-ended question regarding how to learn a representation of graphs that allows us to predict - or rather, compare - MIS sizes reliably.

One design choice is the contrastive training, but another is in choosing an expressive model that can adequately capture the relevant graph structures. To this end, we will employ the following 3 models, ranked from least to most complex: Graph Convolutional Neural Networks (**GCNN**, Kipf and Welling 2016), Graph Attention Network (**GAT**, Veličković et al. 2018) and the **Exphormer** (Shirzad et al. 2023). We want to quantify their performance, firstly in distribution to ensure that they learn a useful representation at all, next out of distribution on synthetic data to monitor overfitting, and finally, we want to analyze the behavior when using the models in decoding algorithms on real networks.

GCNN: A basic GCN layer is defined as $\mathbf{H}^{(k)} = \sigma\left(\tilde{\mathbf{A}}\mathbf{H}^{(k-1)}\mathbf{W}^{(k)}\right)$, where $\tilde{\mathbf{A}} = (\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}(\mathbf{I}+\mathbf{A})(\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}$ is a normalized adjacency matrix - here, $\mathbf{A}$ is the standard adjacency matrix, $\mathbf{D} = \text{diag}(d_1, d_2, \ldots, d_{|V|})$ is the matrix with each node's degree in its diagonal and $\mathbf{I}$ is the identity matrix. Further, σ is a nonlinear function, and $\mathbf{W}^{(k)}$ is a learnable weight matrix.

Given a matrix $\mathbf{X} \in \mathbb{R}^{|V|\times m}$ of node features, a graph convolutional filter is $\mathbf{Q}_h\mathbf{X} = \alpha_0\mathbf{I}\mathbf{X} + \alpha_1\mathbf{A}\mathbf{X} + \alpha_2\mathbf{A}^2\mathbf{X} + \cdots + \alpha_{|V|}\mathbf{A}^{|V|}\mathbf{X}$. If we consider the simple convolutional filter $\mathbf{Q}_h = \mathbf{I} + \mathbf{A}$, this is equivalent to the basic message passing GNN, defined as $\mathbf{H}^{(k)} = \sigma\left(\mathbf{A}\mathbf{H}^{(k-1)}\mathbf{W}^{(k)}_{\text{neigh}} + \mathbf{H}^{(k-1)}\mathbf{W}^{(k)}_{\text{self}}\right)$. We can thus view the basic GNN as performing a very simple graph convolution, and the GCN as a variation on this.

GAT: "Attention", in the context of ML, refers to a family of mechanisms for assigning relative relevance scores to different pieces of interconnected data, e.g. sequences. It has been applied in many different ML contexts, prominently in Transformers (Vaswani et al. 2017). For GATs, we use a variation of attention that is applied on the neighborhood of each node. We pass in vectors of node features (denoted $\mathbf{h}_v$ for node v) obtain a vector of attention scores $\mathbf{m}_{N(u)} = \sum_{v\in N(u)} \alpha_{u,v}\mathbf{h}_v$, where $\alpha_{u,v} = \frac{\exp(\mathbf{a}^\top[\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_v])}{\sum_{v'\in N(u)}\exp(\mathbf{a}^\top[\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_{v'}])}$. Here, $\oplus$ is the concatenation operator. Similarly to other attention schemes, $\mathbf{W}$ is a trainable weight matrix, and $\mathbf{a}$ is a trainable attention vector. The idea here is that the weight matrix is a shared linear transformation, and the attention vector parametrizes a single layer neural network, which computes coefficients for each node.

According to Graph Representation Learning (book, Hamilton 2020), this type of attention is known to work well with graph data.

Exphormer: The Exphormer is a variation of the Transformer adapted to graph data. Similarly to the GAT, an attention mechanism is also central component. However, Exphormer attention varies a bit from GAT attention.

The main idea is to use an approach similar to the scaled dot product used in standard transformers, but adapt it to the structure of graph data. Concretely, they consider attention patterns, which dictate which elements the attention should be computed between. For graphs, an attention pattern could naturally be their edges. In this view, attention for standard sequence modeling can be viewed as interactions in a fully connected graph, i.e. why it is referred to as dense attention.

In the paper, they define the attention mechanism for l attention heads (i.e. parallel attention operations) of size m with attention pattern $\mathcal{H}$ as

$$\text{ATTN}_{\mathcal{H}}(\mathbf{X})_{:,\boldsymbol{u}} = \mathbf{x}_{\boldsymbol{u}} + \sum_{j=1}^{l} \mathbf{W}_O^j \mathbf{W}_V^j \mathbf{X}_{N_{\mathcal{H}}(\boldsymbol{u})} \cdot \sigma\left(\left(\mathbf{W}_K^j \mathbf{X}_{N_{\mathcal{H}}(\boldsymbol{u})}\right)^\top \left(\mathbf{W}_Q^j \mathbf{x}_{\boldsymbol{u}}\right)\right),$$

where $\mathbf{X} = \bigoplus_{v \in V} \mathbf{h}_{\boldsymbol{v}} \in \mathbb{R}^{d \times |V|}$ are the concatenated initial node feature vectors, $\mathbf{W}_K^j, \mathbf{W}_Q^j, \mathbf{W}_V^j \in \mathbb{R}^{m \times d}$ are the canonical "key", "query" and "value" trainable weight matrices, and $\mathbf{W}_O^j \in \mathbb{R}^{d \times m}$ is an additional trainable weight matrix, which is referred to as the "output" matrix. Note that in the Exphormer paper, they extend the attention function to take edge features into account, but we have not used this in the project, hence why we present this simplified formulation.

Sparse attention is then achieved by considering 3 different attention patterns that are hypothesized to work well for graphs, namely local neighborhood attention (essentially the same as the GAT), expander graph attention and global attention with a single virtual node.

Expander graph attention applies attention by ad hoc creating new virtual edges for the attention computation. These edges induce a subgraph with desirable connectivity properties.

Global attention consists of creating a virtual node that connects to all real nodes, which can then aggregate some information from the entire graph.

These sparse attention patterns constitute the main contribution of the Exphormer paper, which we refer to for more details. Figure 2.1 reproduces the visualizations of the 3 sparse attention patterns, as well as the combination.

FIGURE 2.1
(a) Local neighborhood attention using the real edges. (b) Degree 3 expander graph attention. (c) Global attention with a virtual node. (d) Combination of all 3. All credits to Shirzad et al. 2023.

Finally, as a practical note, we primarily use only Local Neighborhood Attention, since our initial pilot tests showed little to no performance increase for the two others, while especially the Expander Graph Attention is very computationally expensive, due to it running on CPU only.

3 Results

3.1 In-distribution Results

Even though the ultimate goal of the project is to compare large, real-world graphs, we do not actually train the comparator on real-world data, but on synthetic data. Before investigating whether this strategy generalizes to real-world graphs, we must first ensure that the comparator works well for in-distribution data. Below is a plot showing the performance of the comparator on the graph types used in the training.

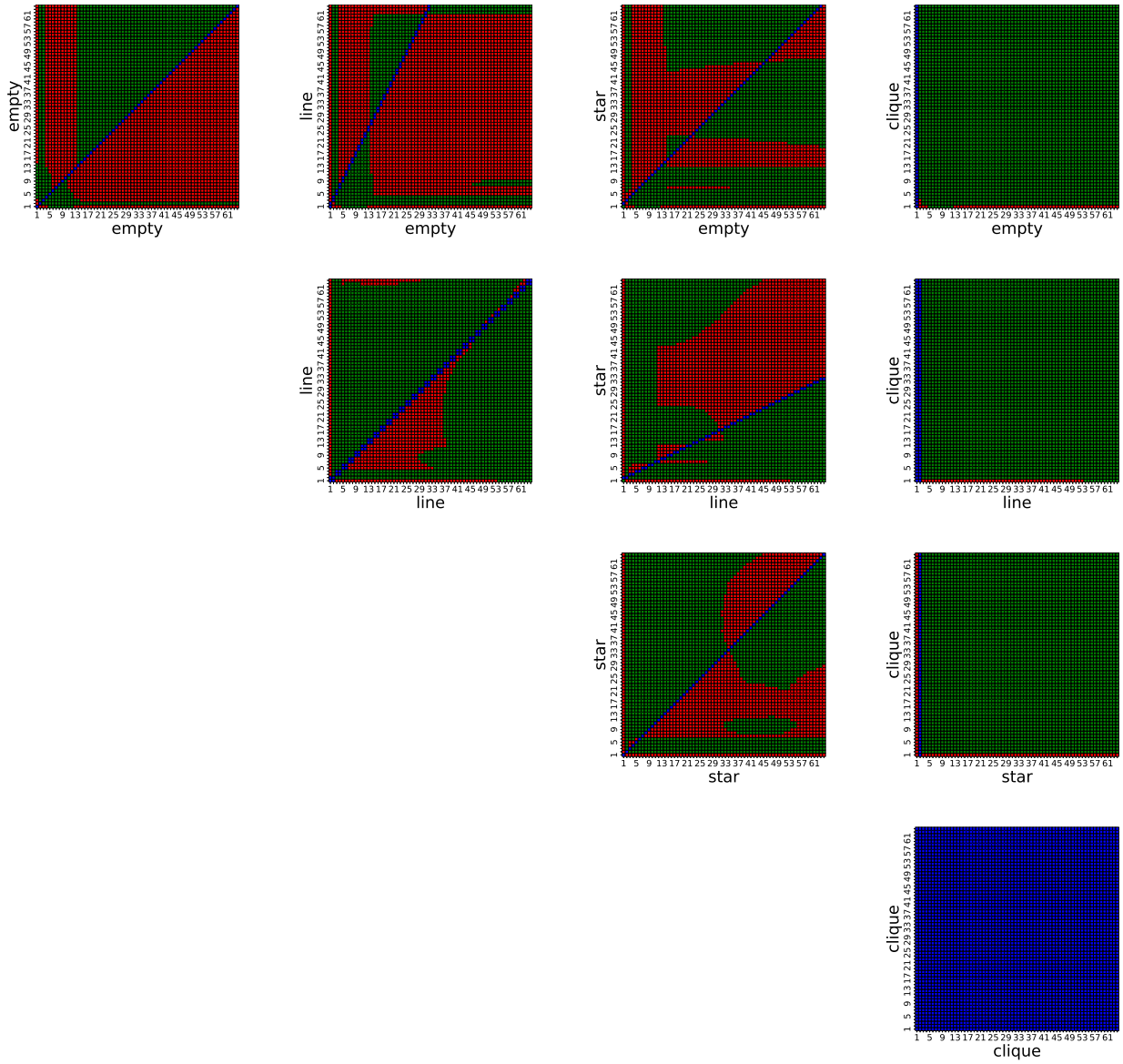


Figure 3.1
The plot compares every combination of two of the graph types used in training. The axes show the size of the graph type and the colors indicate whether the comparator correctly predicted the larger MIS. Green indicates correct prediction, red indicates wrong prediction and blue indicates cases where the MIS was of equal size for the two graph types.

The plots in Figure 3.1 might suggest a systematic behavior in the model. Looking, for example, at the empty-empty plot, we see that when the empty graphs reach a certain size (from 14 nodes), the model will opt to always predict that the empty graph on the second axis has larger MIS. This is correct for cases above the blue diagonal, but wrong for cases below.

In plots involving cliques, the model exhibits almost perfect behavior. This should be expected, as a clique trivially always has a MIS size of 1, regardless of its size.

3.2 Out-of-distribution Results

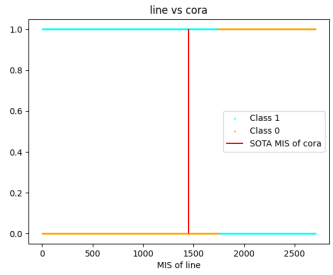
Below is a table showing the obtained MIS for some selected public datasets.

Table 3.1
Results of running the comparator-induced algorithm on some selected datasets. Brando is the best results from previous project students. Unbalanced means the comparator trained on a dataset prior to balancing as described in section 2.1and balanced means after balancing the training. The SOTA results are gathered from Alkhouri, Atia and Velasquez 2022. Note that for the balanced case, the best results were obtained with a GCNN as opposed to an Exphormer as for the unbalanced case. For this reason, we were not able to run on the larger datasets due to memory constraints.

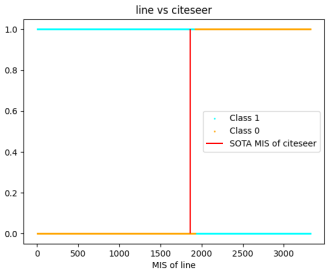
Name	**MIS** (Brando)	**MIS** (unbalanced)	**MIS** (balanced)	**MIS** (SOTA)
cora	1446	1448	1426	**1451**
citeseer	1865	1863	1844	**1867**
pubmed	15910	**15912**	–	**15912**
wiki-Vote	**4866**	**4866**	4827	**4866**
soc-sign-bitcoinalpha	2717	**2718**	2697	**2718**
soc-sign-bitcoinotc	4339	4342	4346	**4347**
soc-Epinions1	-	53594	–	**53599**
soc-Slashdot0811	-	53311	–	**53314**
soc-Slashdot0902	-	56340	–	**56398**

Figure 3.2 attempts to see how well the comparator can tell when a public dataset has a higher MIS than, for example, an empty graph of a given size.

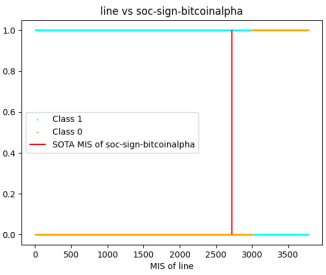
What we see is that the comparator can tell pretty accurately when the public dataset has a higher MIS compared to a line graph. When comparing to an empty graph, the comparator doesn't recognize when to interchange the class assignments.



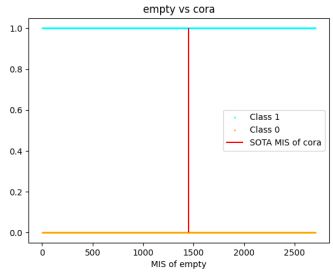
(A) Line vs Cora



(B) Line vs Citeseer

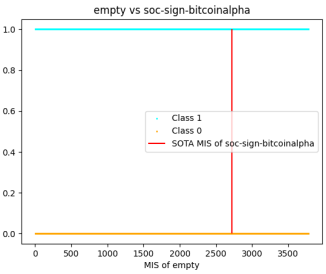


(C) Line vs soc-sign-bitcoinalpha



(D) Empty vs Cora

(E) Empty vs Citeseer



(F) Empty vs soc-sign-bitcoinalpha

FIGURE 3.2

Comparing a graph type with known MIS vs a public graph. The line at $y = 0$ is the class assigned to the line/empty graph (indicated by color) and the line at $y = 1$ is the class assigned to the public dataset. One should expect the classes to interchage around the red line (the SOTA result)

4 Discussion

The paper Brusca et al. 2023 (from LIONS) tackles the problem of finding the MIS of public datasets with a similar approach. The approach of this project differs in at least three ways:

1. **Training strategy:** In the paper, a self-annotating approach is followed to train their comparator. First, a set of subgraphs is constructed by running Algorithm 1, but using a random vertex $v \in V$ in each iteration. Then for each of these subgraphs, their MIS is estimated by running the comparator induced algorithm m times. From this, a number of pairs is constructed with self-annotated labels. In contrast, our approach is based on the idea of learning fundamental graph structures which serve as building blocks for more complicated graph structures. These fundamental graph structures have known MIS sizes, and certain transformations can be performed for which the resulting MIS is also known.

2. **Generalizes to unseen data:** The datasets used for training and testing in the paper are subsamples of public datasets. This approach may work better for testing in-distribution, but there is no saying how well it will generalize to unseen graph datasets. Our approach attempts to understand fundamental graph structures as explained above. Admittedly, our work is not currently able to show that this idea works.

3. **Scales better with graph size:** The training process of the paper, to our understanding, scales badly with the size of the graphs used. This is due to the way the MIS is estimated as explained in point 1. Indeed, the graph sizes used in the paper is relatively low (a maximum of ~ 500 nodes), whereas the smallest public dataset that we test on is 2708 nodes (Cora).

Looking at Table 3.1 we can see that using balanced training data, we obtain worse results than by using the unbalanced data. This is what led us, and likely previous project students to believe that the model was doing very well. This turned out to be a red herring, as the underlying reason is that the unbalanced model always predicts label 1 (i.e. include a node in the MIS) due to the unbalanced training set. The result is essentially a greedy version of Algorithm 1, where the node is always included. It is worth noting that the greedy algorithm is a very good heuristic in fact hard to beat for many natural graphs; and it really highlights that this problem isn't easily solved. Also worth noting is that you can of course construct examples which the greedy algorithm will always get wrong.

A lesson of this project is thus to be wary of assuming that a good result means that the fundamentals are working. For a much of the project's duration, we, and likely also previous semester project students, falsely believed that the comparator model was the cause of the seemingly great results as indicated by Table 3.1. In reality, these results were nothing other than the greedy MIS algorithm in disguise. Unfortunately, this forced us to shift our focus from improving on previous results to a critical investigation of the existing comparator model.

5 Conclusion

This project was originally intended to investigate extensions of the current MIS decoding approach, for example using reinforcement learning, to select nodes in the construction of a MIS based on a pre-trained contrastive model developed in previous semester projects.

Substantial code clean-up and restructuring, as well as a thorough investigation indicated that the pre-trained model was not working as intended. Instead, the good results obtained were due to the model making choices equivalent to those of a greedy algorithm.

Our investigation of the pre-trained model showed that the model was heavily biased to predict label 1. This explained why we obtained results equal to the greedy algorithm. We alleviated this bias by making sure that the training was more balanced.

The results show that the model performs poorly even on simple in-distribution graph data. Our results thus indicate that this approach - in its current form - does not show promise in solving the MIS problem. However, many smaller choices and details in the training process and model selection could be made differently, and we cannot refute that this could improve performance. Further work on this approach should look into improving the learned representations, first in-distribution, and then on extending to real networks, i.e. what we are investigating in Figure 3.1 and Figure 3.2, respectively.

BIBLIOGRAPHY

Brusca, Lorenzo et al. (2023). *Maximum Independent Set: Self-Training through Dynamic Programming*. arXiv: 2310.18672 [cs.LG]. URL: https://arxiv.org/abs/2310.18672.

Kipf, Thomas N. and Max Welling (2016). 'Semi-Supervised Classification with Graph Convolutional Networks'. In: *CoRR* abs/1609.02907. arXiv: 1609.02907. URL: http://arxiv.org/abs/1609.02907.

Veličković, Petar et al. (2018). *Graph Attention Networks*. arXiv: 1710.10903 [stat.ML]. URL: https://arxiv.org/abs/1710.10903.

Shirzad, Hamed et al. (2023). *Exphormer: Sparse Transformers for Graphs*. arXiv: 2303.06147 [cs.LG]. URL: https://arxiv.org/abs/2303.06147.

Vaswani, Ashish et al. (2017). *Attention Is All You Need*. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

Hamilton, William L. (2020). *Graph Representation Learning*. Vol. 14. Synthesis Lectures on Artificial Intelligence and Machine Learning 3. Morgan & Claypool Publishers. ISBN: 9781681739632. DOI: 10.2200/S01045ED1V01Y202009AIM046. URL: https://doi.org/10.2200/S01045ED1V01Y202009AIM046.

Alkhouri, Ismail R., George K. Atia and Alvaro Velasquez (2022). 'A differentiable approach to the maximum independent set problem using dataless neural networks'. In: *Neural Networks* 155, pp. 168–176. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2022.08.008. URL: https://www.sciencedirect.com/science/article/pii/S0893608022003082.