Graph Homomorphism Features: Why Not Sample?*

Paul Beaujean, Florian Sikora, and Florian Yger

LAMSADE, CNRS, Université Paris-Dauphine, PSL Research University
first.last@lamsade.dauphine.fr

Abstract. Recent research in the domain of computed graph embeddings has shown that graph homomorphism numbers constitute expressive features that are well-suited for machine learning tasks such as graph classification. In this work-in-progress paper, we attempt to make this methodology scalable by obtaining additive approximations to graph homomorphism densities via a simple sampling algorithm. We show in experiments that these approximate homomorphism densities perform as well as homomorphism numbers on standard graph classification datasets. Moreover, we show that, unlike algorithms that compute homomorphism numbers, our sampling algorithm is highly scalable to larger graphs.

Keywords: Graph embedding · Graph homomorphism · Graph classification.

1 Introduction

Graph embeddings are mappings from the set of all graphs to some well-behaved Euclidean space, which is the setting where most machine learning models operate. Approaches such as geometric deep learning [4] and more generally graph neural networks [17] attempt to learn these embeddings, but we focus instead on graph embeddings that are directly computed from the original graph data [10].

One particularly common approach in computed graph embeddings is the idea of representing graphs through statistics about particular substructures such as paths, trees, or walks. Graphlet kernels for example utilize the distribution of small induced subgraphs of size at most 5 to compare graphs [16].

In this work, we consider the notion of graph homomorphism, which can be seen as a relaxation of the notion of subgraph, that is easier to compute than induced subgraphs while being provably expressive. Statistics collected on these graph homomorphisms are then used as features to traditional machine learning models operating on Euclidean data. Novel ideas in computed embeddings have often led to advances in learned embeddings, see e.g. the Graph Substructure Network [3], and we are hopeful that graph homomorphisms may become a basis for new learned embeddings as well.

* Supported by Agence Nationale de la Recherche (ANR), projects STAP (ANR-17-CE23-0021) and ESIGMA (ANR-17-CE23-0010). F. Yger acknowledges the support of the ANR as part of the "Investissements d'avenir" program (ANR-19-P3IA-0001, PRAIRIE 3IA Institute).

2 Graph homomorphism numbers

We will often consider a target simple undirected graph $G = (V, E)$ with n nodes and m edges from which we would like to obtain homomorphism information relative to some, usually smaller, pattern graph F with k nodes and l edges.

A graph morphism from F to G is a mapping from the node set $V(F)$ to $V(G)$. Graph homomorphisms are morphisms that preserve adjacency, i.e. $uv \in E(F)$ implies that $f(u)f(v) \in E(G)$, see e.g. Figure 1. If a graph homomorphism is bijective then it also preserves non-adjacency and we call it a graph isomorphism. We write $G_1 \cong G_2$ when there exists an isomorphism between these two graphs.

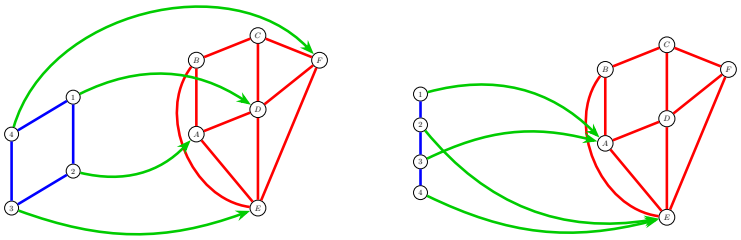

Fig. 1. Left: an homomorphism from a 4-cycle to a graph. Right: an homomorphism from a 4-path to a graph.

To see that a graph homomorphism from F to G is not the same notion as the fact that F is a subgraph of G notice that an homomorphism from a path graph on 4 nodes to G may represent a walk of size 4 in G and not only a path subgraph as can be seen in the right part of Figure 1.

The number of graph homomorphisms from F to G is written $\hom(F, G)$ and ranges from 0 to n^k. It is convenient to combine multiple homomorphism numbers into a vector as follows: $\hom_{\mathcal{F}}(G) = (\hom(F, G))_{F \in \mathcal{F}}$. A celebrated result by Lovász [12] states the following:

Theorem 1. *Given two undirected simple graphs G_1 and G_2, both with n nodes. Denoting by $\mathcal{G}_n$ the set of all simple graphs with at most n nodes, we have:*

$$G_1 \cong G_2 \iff \hom_{\mathcal{G}_n}(G_1) = \hom_{\mathcal{G}_n}(G_2).$$

In particular this means that as long as two graphs have different homomorphism numbers for a single pattern F then they are not isomorphic to each other. Extensions of this result have been obtained by Lovász [13] and show that graphs that have close homomorphism vectors are similar in the sense that they have cuts of similar value.

NT & Maehara have proposed to use homomorphism numbers as features to machine learning models, hoping that the first few components of the homomorphism vectors would suffice in practice to classify real-life graphs [15]. Their

approach was successful but remains limited in practice by the computational complexity of computing homomorphism numbers.

Indeed, the current best algorithm to compute $\hom(F, G)$ is an algorithm based on dynamic programming and tree decompositions given by Díaz et al. [8]. Its worst-case running time is $O(\text{poly}(k) \cdot n^{\text{tw}(F)+1})$ where $\text{tw}(F)$ is the treewidth of the pattern graph F. Hardness results in computational complexity show that there is no hope in finding a better algorithm, even if we settle for an approximate number that would be within $(1 \pm \varepsilon)$ of $\hom(F, G)$ [7, 5].

3 Graph homomorphism densities

Homomorphism densities are normalized versions of homomorphism numbers. Formally, $t(F, G) = \hom(F, G)/n^k$, which means that densities live in the $[0, 1]$ interval. These quantities carry most of the properties of homomorphism numbers and constitute the basis of the theory of graph limits developed by Lovász [13]. More concretely, the homomorphism density from F to G can be interpreted as the probability that a random morphism preserve adjacency for every edge of $E(F)$ into $E(G)$.

This simple scaling allows us to use sampling to estimate $t(F, G)$ in the same manner as a coin of unknown bias, i.e. a Bernoulli distribution with unknown parameter p. Standard Chernoff bound arguments give conservative estimates to the number of morphisms that need to be drawn uniformly at random from $V(F) \to V(G)$ to obtain an additive approximation within $\pm\varepsilon$ of $t(F, G)$.

Algorithm 1 starts with drawing N morphisms uniformly at random, which is equivalent to drawing k nodes in G at random as they are the images of the k nodes of F and then checking for each edge $uv \in E(F)$ if their image is indeed in $E(G)$. The first set membership query that fails will short-circuit the condition as the morphism is not going to be an homomorphism. The total running time of Algorithm 1 is $O\left((k \log n + l) \cdot N\right)$ where $O(N \cdot k \log n)$ corresponds to sampling N morphisms and $O(N \cdot l)$ corresponds to testing whether each morphism is an homomorphism. This leads to a running time of $O\left((k \log n + l) \cdot \varepsilon^{-2} \log \delta^{-1}\right)$ to obtain a value that is guaranteed to be $t(F, G) \pm \varepsilon$ with probability $1 - \delta$.

Algorithm 1 sGHD: Sample Graph Homomorphism Density

Require: G an undirected graph on n nodes, F a pattern graph on k nodes and l edges, $\varepsilon > 0$ the requested additive precision, $1 - \delta \in (0, 1)$ the desired confidence.
Ensure: $\bar{t}$ such that $\mathbb{P}(|\bar{t} - t(F, G)| > \varepsilon) \leq \delta$
1: $N \leftarrow O(\varepsilon^{-2} \log \delta^{-1})$
2: **for** $i = 1$ to N **do**
3: $\quad \mathbf{f}^{(i)} \sim (\mathcal{U}([1 \,..\, n])_{[k]}$
4: **end for**
5: $\bar{t} \leftarrow \frac{1}{N} \sum_{i=1}^{N} \prod_{uv \in E(F)} \mathbb{1}\left[\mathbf{f}_u^{(i)} \mathbf{f}_v^{(i)} \in E(G)\right]$
6: **return** $\bar{t}$

The existence of this algorithm does not contradict the hardness results presented in the previous section. Indeed, if we assume that there is a single K_k in G then $t(K_k, G) = 1/n^k$. A fixed precision of ε would not be able to distinguish between $t(K_k, G) = 1/n^k$ and $t(K_k, G) = 0$. To put it simply, a fixed additive approximation would give accurate values for high $t(F, G)$ and inaccurate estimates for very low $t(F, G)$. However we will see in the next section that this theoretical behavior has a moderate impact on classification accuracy.

For more details on the scalability of Algorithm 1 and details on its implementation using adequate data structures, we invite the reader to refer to our workshop paper [1].

4 Graph classification using homomorphism features

We compare the performance of different homomorphism features on standard graph classification datasets from the TUDataset collection [14]. We note that these standard datasets mostly contain small graphs. The datasets containing the largest graphs still contain graphs with an average of a few hundred nodes. On the other hand, as Algorithm 1 has logarithmic complexity in the size of G, we would ideally use it on classification datasets containing very large graphs. However, to the best of our knowledge, there is no graph classification dataset which contains very large graphs. For example, the recently introduced Open Graph Database [11] offers a graph classification dataset of 3.8M graphs containing 14.5 nodes on average. Noting the scalability issues of exact homomorphism number algorithms like those used by NT & Maehara, we settle for well-studied datasets.

In Table 1 we collect test accuracy scores of a common cross-validation procedure [9]. SGHD models are vanilla logistic regressions trained on approximate homomorphism densities obtained by Algorithm 1 by choosing a set of pattern graphs $\mathcal{A}_{10}$ that corresponds to the first 10 connected simple graphs. $\overline{\text{SGHD}}$ models are trained on features computed from the same pattern graphs but to complement graphs instead.[1] To analyze the impact of topological information, we remove all weights and labels to obtain completely unlabeled undirected graphs. GHC models are SVM models trained on exact homomorphism numbers obtained by the algorithm of Díaz et al. [8]. GHC models also come with a variety of scaling algorithms in the data ingestion pipeline while SGHD do not. Because of scalability issues, GHC models are given homomorphism numbers from specific classes of graphs of low treewidth such as trees for the $\mathcal{T}_{13}$ model collecting homomorphism numbers from the first 13 trees, and cycles for $\mathcal{C}_7$. Note that Algorithm 1 is not bound by these limitations even if its complexity, like in the case of homomorphism numbers, favors small pattern graphs.

We notice that on most datasets, the performance of SGHD models is on par with exact homomorphism numbers. On some datasets there is a notable

[1] Standard datasets contain relatively sparse graphs which means that the corresponding complement graphs are dense. Dense graphs have larger homomorphism densities which are easier to detect at fixed precision and more amenable for training machine learning models.

	MUTAG	NCI1	PROTEINS	DD	ENZYMES	REDDIT-B	COLLAB	IMDB-B
SGHD-$\mathcal{A}_{10}$ $\varepsilon = 0.1$	83.6 ± 8.7	62.6 ± 2.9	72.0 ± 4.1	$\mathbf{76.2 \pm 3.2}$	21.2 ± 4.1	73.6 ± 3.2	$\mathbf{68.1 \pm 2.0}$	63.3 ± 3.7
SGHD-$\mathcal{A}_{10}$ $\varepsilon = 0.01$	86.3 ± 7.9	62.7 ± 3.1	$\mathbf{72.3 \pm 3.6}$	76.1 ± 3.2	26.3 ± 4.5	73.8 ± 3.2	67.7 ± 2.1	68.7 ± 2.4
$\overline{\text{SGHD}}$-$\mathcal{A}_{10}$ $\varepsilon = 0.1$	83.8 ± 8.5	62.9 ± 2.9	72.1 ± 3.8	76.2 ± 3.4	23.3 ± 4.5	75.5 ± 2.7	67.3 ± 2.2	62.2 ± 3.5
GHC-$\mathcal{T}_{13}$ (NT&M.)	$\mathbf{88.2 \pm 7.4}$	$\mathbf{65.4 \pm 2.5}$	70.6 ± 4.7	75.3 ± 3.6	21.7 ± 3.6	$\mathbf{84.6 \pm 2.3}^*$	62.1 ± 1.9	$\mathbf{69.7 \pm 4.4}$
GHC-$\mathcal{C}_7$ (NT&M.)	-	-	-	76.1 ± 3.9	$\mathbf{29.5 \pm 3.2}$	-	-	-

Table 1. Test accuracy scores. "-": experiment not provided in the corresponding study. *: details of experiment in Section 4

loss of accuracy depending on the precision requested e.g. in the case of IMDB-BINARY where a lower ε is required to reach higher test accuracy. In some other cases, SGHD models beat GHC ones such as COLLAB or achieve similar performance like DD. We focus our attention on the case of REDDIT-BINARY where there is a massive difference in test accuracy between SGHD and GHC. When looking closely at the supplementary material of the study by NT & Maehara [15], we notice that depending on the cross-validated scaler, test accuracy is $73.8\% \pm 2.8\%$ with min/max or max/abs scaling while it's over 80% with other scaling techniques (standard, quantile, power). This shows that further preprocessing or models with higher capacity may extract more information than logistic regression or SVM models. We invite the reader to refer to our technical report for more details and comparisons to different machine learning models [2]. We conjecture that the cases where SGHD models beat GHC happen when approximate features operate as a regularizer which retaining most of the discriminating power of their exact counterparts. When the opposite happens, we suppose that information loss is responsible for most of the loss in accuracy.

5 Conclusion

We have proposed and implemented a simple randomized algorithm outputting an additive approximation of graph homomorphism densities. Our sampling algorithm is highly scalable and has a practically constant running time for a given fixed precision. This has to be contrasted with the theoretical and practical aspects of computing exact homomorphism numbers which rely on low treewidth pattern graphs and even then do not scale to large graphs.

We have shown in experiments on standard graph classification datasets that additively approximate homomorphism densities retain similar representational power compared to exact homomorphism numbers and sometimes to higher test accuracy even when they are used to train logistic regression models which are one of the simplest classifiers in the literature.

These preliminary results invite us to consider seeking larger datasets containing larger graphs to evaluate the performance of homomorphism densities in the setting that they were designed to excel. However, it remains to be seen how to train existing graph machine learning models on datasets of this scale. Another idea would be to consider learning the family of pattern graphs that best explains the information present in a given dataset.

Finally, we wish to compare the performance of models trained on homomorphism densities with alternative approaches such as the closely related graphlet kernels [6, 16] or popular graph neural network architectures [17]. Moreover, we are interested in studying weighted and/or labeled variants of homomorphisms which would allow us to extract richer information from graph datasets.

References

1. Beaujean, P., Sikora, F., Yger, F.: Scaling up graph homomorphism features with efficient data structures. In: ICLR 2021 Workshop on Geometrical and Topological Representation Learning (2021), https://openreview.net/forum?id=EwT8NpZIth8
2. Beaujean, P., Sikora, F., Yger, F.: Scaling up graph homomorphism for classification via sampling. CoRR **abs/2104.04040** (2021), https://arxiv.org/abs/2104.04040
3. Bouritsas, G., Frasca, F., Zafeiriou, S., Bronstein, M.M.: Improving graph neural network expressivity via subgraph isomorphism counting. CoRR **abs/2006.09252** (2020), https://arxiv.org/abs/2006.09252
4. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond Euclidean data. IEEE Signal Processing Magazine (2017)
5. Bulatov, A., Živný, S.: Approximate counting CSP seen from the other side. ACM Transactions on Computation Theory **12**, 1–19 (05 2020). https://doi.org/10.1145/3389390
6. Curticapean, R., Dell, H., Marx, D.: Homomorphisms are a good basis for counting small subgraphs. In: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing. pp. 210–223 (2017)
7. Dalmau, V., Jonsson, P.: The complexity of counting homomorphisms seen from the other side. Theoretical Computer Science **329**(1-3), 315–323 (2004)
8. Díaz, J., Serna, M., Thilikos, D.M.: Counting H-colorings of partial k-trees. Theor. Comput. Sci. (2002)
9. Errica, F., Podda, M., Bacciu, D., Micheli, A.: A fair comparison of graph neural networks for graph classification. In: ICLR (2019)
10. Grohe, M.: word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data. In: PODS (2020)
11. Hu, W., Fey, M., Ren, H., Nakata, M., Dong, Y., Leskovec, J.: OGB-LSC: a large-scale challenge for machine learning on graphs. arXiv preprint arXiv:2103.09430 (2021)
12. Lovász, L.: Operations with structures. Acta Mathematica Academiae Scientiarum Hungarica (1967)
13. Lovász, L.: Large networks and graph limits. American Mathematical Soc. (2012)
14. Morris, C., Kriege, N.M., Bause, F., Kersting, K., Mutzel, P., Neumann, M.: TUDataset: A collection of benchmark datasets for learning with graphs. In: ICML Workshop on Graph Representation Learning and Beyond (GRL+) (2020), www.graphlearning.io
15. NT, H., Maehara, T.: Graph homomorphism convolution. In: ICML (2020)
16. Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., Borgwardt, K.: Efficient graphlet kernels for large graph comparison. In: AIStat (2009)
17. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE Trans. Neural Networks Learn. Syst. (2020)