Balancing performance and complexity with adaptive graph coarsening

Marek Dědič[12][0000−0003−1021−8428], Lukáš Bajer[2][0000−0002−9402−6417], Pavel Procházka[2], and Martin Holeňa[3][0000−0002−2536−9328]

[1] Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University in Prague, Břehová 7, Prague, 110 00, Czech Republic
[2] Cisco Systems, Inc., Karlovo náměstí 10, Prague, 120 00, Czech Republic
{madedic,lubajer,paprocha}@cisco.com
[3] Institute of Computer Science, Czech Academy of Sciences, Pod vodárenskou věží 2, Prague, 182 07, Czech Republic martin@cs.cas.cz

Abstract Graph based models are used for tasks with increasing size and computational demands. We present a method for node classification that allows a user to precisely select the resolution at which the graph in question should be pretrained. Our method builds on an existing algorithm for pretraining on coarser graphs, HARP, which we extend in order to tune the effect of graph coarsening on the accuracy of node classification on a fine level. We present a novel way of refining the reduced graph in a targeted way based on the node classification confidence of particular nodes. This enhancement provides sufficient detail where needed, while collapsing structures where per-node information is not necessary for sufficient node classification accuracy. Hence, the method provides a meta-model for enhancing graph embedding models such as node2vec. We apply it to several datasets and discuss the differing behaviour on each of them in the context of their properties.

Keywords: Graph representation learning · Graph coarsening · Graph diffusion · Graph homophily · Performance-complexity trade-off · HARP

1 Introduction

Across a wide variety of applications and domains, graphs emerge as a domain-independent and ubiquitous way of organizing structured data. Consequently, machine learning on graphs has, in recent years, seen an explosion in popularity, breadth and depth of both research and applications. While there have been significant advances in algorithms for learning from graph data [15,30], the underlying graph topology has, until recent works [46,48], received much less attention. In the reported research, we investigate the interplay of graph coarsening and the quality of its learned embedding (as studied, for example, by [1,35]), which in turn entails an interplay between the coarsening and the performance of a downstream task, in our case, node classification.

The main aim of this work is to explore the performance-complexity characteristics in the context of graph learning, as introduced in [41]. Consider an undirected graph G with nodes $V(G)$ and edges $E(G)$. The result of a repeated application of graph coarsening is a sequence of graphs $G_0, G_1, G_2, \ldots, G_L$ where $G_0 = G$. Given a model M that operates on graphs, a performance metric, and a complexity metric, the sequence $G_0, G_1, \ldots, G_L$ corresponds to points in the performance-complexity plane, where advancing along the sequence generally hurts performance and decreases complexity.

This performance-complexity characteristic allows for a choice of a **working point** that is optimal for the particular use-case. The choice of the working point, suitable performance metric and complexity metric are subjective and depend on the particular use-case, downstream task and the environment in which the model is to be deployed. In this work, the transductive node classification accuracy on a testing dataset is chosen as the performance metric. For the complexity metric, the number of nodes in the graph was chosen as it constitutes a good proxy for real-world algorithmic complexity, as shown in [14].

While the methods proposed in the rest of this work may yield models and graphs with lower computational demands than models using the original graph, the algorithm for finding the optimal working point itself entails running the same complex models on multiple graphs, therefore potentially offsetting any gains from the lower complexity of the model itself. To overcome these potential shortcomings, the following options are considered:

- The optimal working point may generalize to datasets other than the one used for the performance-complexity analysis, for example when collecting data from the same source periodically.
- The whole performance-complexity curve is not needed to choose the optimal working point. In the context of this work, the graphs are evaluated in reverse order, i.e. starting with G_L. As such, the evaluation only needs to be run until reaching a working point that is acceptable for the intended use-case.

Further discussion of the performance-complexity trade-off problem is considered in [41].

2 Related work

The publication most relevant to our research is [11], in which the HARP approach is proposed. Because we directly extend and modify this method, it will be recalled in some detail in Section 3. Other important works concerning graph coarsening are [1,8,13], which survey numerous coarsening methods, [25], which presents results concerning scalability of graph coarsening, [10,22], which establish coarsening as a basis for partitioning, and [34], which shows relationships of graph coarsening to properties of the Laplacian. In view of the fact that the HARP approach is a multilevel approach, we paid attention also to the multilevel graph coarsening methods proposed in [5,33,49,51], among them [51] also being inspired by HARP.

In a broader context, our research is related to the more general topic of graph reduction, which apart from graph coarsening includes also graph sparsification and condensation. A general framework covering both coarsening and sparsification has been proposed in [7]. Graph condensation is a more recent approach [26,27], inspired by the gradient matching scheme for the construction of training datasets, called dataset condensation [52]: To a given original graph, it attempts to construct a much smaller synthetic graph such that the gradients of the parameters of a considered graph neural network with respect to both graphs match. Also of note is the recent work [28], presenting an alternative coarsening approach for planar graphs and [32], which sparsifies not only the graph topology, but simultaneously also the features of its nodes and weights of graph neural network used for its embedding. Elaboration of graph coarsening methods in machine learning can build on several decades of their successful application, such as pairwise aggregation, independent sets, or algebraic distance, in numerical linear algebra [13], including in particular multilevel graph coarsening [37,47].

More recently, a connection of graph coarsening with another more general topic has been addressed, namely with pooling in graph neural networks. In a framework presented in [20], pooling

is viewed as a composition of three subsequent transformations: selection, in which the nodes of the input graph are mapped to the nodes of the pooled one, reduction, in which the node attributes of the input graph are aggregated into the node attributes of the pooled one, and connection, in which the edges and possibly edge attributes of the input graph are mapped to the edges and possibly edge attributes of the pooled one. This sequence of transformations is perfectly relevant also to coarsening methods, as was in [20] demonstrated for the methods NMF [2] and top-K [9,19].

3 An overview of the HARP algorithm

Our work builds on the HARP method [11] for pretraining methods such as node2vec [21] on coarsened graphs. While HARP itself works with and modifies the graph structure, this is not the main interest of its authors, who focus more on the classification accuracy for the original graph. The sequence $G_0, G_1, G_2, \ldots, G_L$ is generated in HARP consecutively. Let φ_i denote this mapping $G_i = \varphi_i(G_{i-1})$. Following [43], we restrict the definition of such a coarsening φ_i to only consist of a series of edge contractions $\mathcal{C} \subseteq E(G)$. In an overview, the HARP algorithm first ahead-of-time consecutively coarsens the graph. The method itself can then be executed by repeating the following steps on the graphs from the coarsest to the finest (i.e., from G_L to G_0):

1. **Training on an intermediary graph**. The graph embedding model is trained on G_i, producing its embedding Φ_{G_i}.
2. **Embedding prolongation**. The embedding Φ_{G_i} is *prolonged* (i.e. refined) into $\Phi_{G_{i-1}}$ by copying embeddings of merged nodes. $\Phi_{G_{i-1}}$ is then used as the starting point for training on G_{i-1}.

The particular details of the coarsening and prolongation steps are further explained in [11].

4 HARP extension for flexible performance-complexity balancing

Graph representation learning methods such as node2vec typically have a large number of parameters – on the widely used OGBN-ArXiv dataset (see [23]), the state-of-the-art node2vec model has over 21 million parameters. At the same time, recent works in the domain of graph learning have started to focus more heavily on simpler methods as a competitive alternative to heavy-weight ones (see [17,24]). As the authors of [11] observed, HARP improves the performance of models when fewer labelled data are available. The proposed lower complexity models based on HARP could also improve performance in a setting where only low fidelity data are available for large parts of the graph. Coarser models could be trained on them, with a subsequent training of finer models using only a limited sample of high fidelity data.

While the prolongation used by HARP is sufficient when used only as a means of pre-training, the approach is far too crude when studying the relationship between graph complexity and the quality of graph embedding as a single coarsening iteration can reduce the number of nodes to less than half. In order to overcome this limitation, we present the adaptive prolongation approach, which aims to replace the fixed steps defined by the used coarsening algorithm (such as HARP) by a variable number of smaller "micro-steps", each of a predefined size that can be chosen independently from the underlying coarsening and its step size. The L coarsening steps are thus decoupled from K prolongation steps, where K is independent of L. The prolongation steps are driven by the interplay of the downstream task with the local properties of the underlying graph, enabling the

method to produce embeddings with different level of granularity in different parts of the graph, e.g. an embedding that is coarse inside clusters of similar nodes and at the same time fine at the border between such clusters.

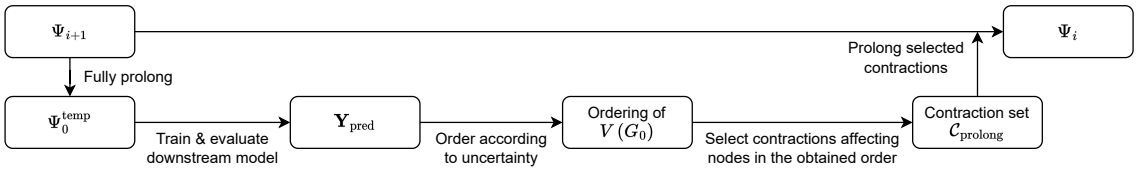

Figure 1. A schematic explanation of the adaptive prolongation algorithm for obtaining the embedding Ψ_i from Ψ_{i+1}.

Let us denote $\Psi_K, \ldots, \Psi_0$ the resulting embedding sequence. Similarly to standard HARP prolongation, the algorithm starts with the coarsest graph G_L, trains a graph model to compute its embedding Ψ_K and gradually refines it until reaching the embedding Ψ_0. These prolongation steps are interlaid with continued training of the graph model, as in standard HARP. A description of a single prolongation step from Ψ_{i+1} to Ψ_i follows, is schematically outlined in Figure 1 and described in detail in Algorithm 1.

The procedure keeps track of all the edge contractions that were made in the dataset augmentation part of the algorithm and gradually reverses them. To this end, apart from the embedding Ψ_i, the set of all contractions yet to be reversed as of step i is kept as $\mathcal{C}_L^{(i)}, \ldots, \mathcal{C}_0^{(i)}$, with the initial values $\mathcal{C}_j^{(K)}$ corresponding to the underlying coarsening φ_j as defined in Section 3.

In each prolongation step, the embedding Ψ_{i+1} is prolonged to Ψ_i by selecting a set of n_p contractions $\mathcal{C}_{\text{prolong}}$ and undoing them by copying and reusing the embedding of the node resulting from the contraction to both of the contracted nodes. To obtain $\mathcal{C}_{\text{prolong}}$, nodes of G_0 are first ordered in such a way that corresponds to the usefulness of prolonging them. Subsequently, the set $\mathcal{C}_L^{(i+1)}, \ldots, \mathcal{C}_0^{(i+1)}$ is ordered to match this node ordering by considering the nodes that the individual contractions affect. $\mathcal{C}_{\text{prolong}}$ is then selected by taking the first n_p contractions. If multiple contractions affecting the same node are available in the sequence $\mathcal{C}_L^{(i+1)}, \ldots, \mathcal{C}_0^{(i+1)}$, one is selected from $\mathcal{C}_j^{(i+1)}$ corresponding to the coarsest-level coarsening. The sequence $\mathcal{C}_L^{(i)}, \ldots, \mathcal{C}_0^{(i)}$ is produced from $\mathcal{C}_L^{(i+1)}, \ldots, \mathcal{C}_0^{(i+1)}$ by removing all of the edges contained in $\mathcal{C}_{\text{prolong}}$.

To obtain an ordering of nodes of G_0 based on the usefulness of their prolongation, the embedding Ψ_{i+1} is fully prolonged to a temporary embedding of the full graph, Ψ_0^{temp}. The downstream model is then trained using this temporary embedding to obtain $\mathbf{Y}_{\text{pred}}$, the predicted posterior distribution of classes for each node in G_0 (e.g. the output of the softmax layer of an MLP). The entropy of this distribution is measured, representing the amount of uncertainty in the classifier for each given node. The nodes are ordered based on the entropy from highest to lowest. This reflects the principle that it is most useful to prolong those nodes where the downstream classifier is the least certain. For downstream tasks other than node classification, the ordering would need to be defined in a different manner (for example using labels, which are not available for all nodes in our case), however the approach of prolonging the nodes about which the downstream model is the most uncertain can be extended to other tasks.

Algorithm 1 Adaptive prolongation

Require: G_0 ▷ The original graph
Require: $\boldsymbol{y}_{\text{train}}$ ▷ Training labels
Require: n_p ▷ The number of nodes to prolong
Require: Ψ_{i+1} ▷ The previous embedding
Require: $\mathcal{C}_L^{(i+1)}, \ldots, \mathcal{C}_0^{(i+1)}$ ▷ A list of all the contractions yet to be reversed
Ensure: Ψ_i ▷ The next embedding
Ensure: $\mathcal{C}_L^{(i)}, \ldots, \mathcal{C}_0^{(i)}$ ▷ Updated contraction list without the prolonged contractions

$node\_order \leftarrow \text{GET\_NODE\_ORDER}(G_0, \Psi_{i+1}, \boldsymbol{y}_{\text{train}}, \mathcal{C}_L^{(i+1)}, \ldots, \mathcal{C}_0^{(i+1)})$
$\mathcal{C}_{\text{prolong}} \leftarrow \text{SELECT\_CONTRACTIONS}(node\_order, n_p, \mathcal{C}_L^{(i+1)}, \ldots, \mathcal{C}_0^{(i+1)})$
$\Psi_i \leftarrow$ use $\mathcal{C}_{\text{prolong}}$ to prolong the embedding Ψ_{i+1}
$\mathcal{C}_L^{(i)}, \ldots, \mathcal{C}_0^{(i)} \leftarrow$ remove contractions in $\mathcal{C}_{\text{prolong}}$ from $\mathcal{C}_L^{(i+1)}, \ldots, \mathcal{C}_0^{(i+1)}$

function GET_NODE_ORDER($G_0, \Psi_{i+1}, \boldsymbol{y}_{\text{train}}, \mathcal{C}_L^{(i+1)}, \ldots, \mathcal{C}_0^{(i+1)}$)
  $\Psi_0^{\text{temp}} \leftarrow$ use $\mathcal{C}_L^{(i+1)}, \ldots, \mathcal{C}_0^{(i+1)}$ to fully prolong the current embedding Ψ_{i+1} to G_0
  $model \leftarrow \text{TRAIN\_DOWNSTREAM\_MODEL}(\Psi_0^{\text{temp}}, \boldsymbol{y}_{\text{train}})$
  $\mathbf{Y}_{\text{pred}} \leftarrow \text{PREDICT}(model, node)$ for each $node \in V(G_0)$
  $entropy\_per\_node \leftarrow H(\mathbf{Y}_{\text{pred}})$
  return $V(G_0)$, sorted in descending order by $entropy\_per\_node$
end function

function SELECT_CONTRACTIONS($ordered\_nodes, n_p, \mathcal{C}_L^{(i+1)}, \ldots, \mathcal{C}_0^{(i+1)}$)
  $\mathcal{C}_{\text{prolong}} \leftarrow \{\}$
  for $node \in ordered\_nodes$, **until** $|\mathcal{C}_{\text{prolong}}| = n_p$ **do**
    $contraction \leftarrow \text{RESOLVE\_CONTRACTION}(node, \mathcal{C}_{\text{prolong}}, \mathcal{C}_L^{(i+1)}, \ldots, \mathcal{C}_0^{(i+1)})$
    If $contraction \neq$ null, add $contraction$ to $\mathcal{C}_{\text{prolong}}$
  end for
  return $\mathcal{C}_{\text{prolong}}$
end function

function RESOLVE_CONTRACTION($node, \mathcal{C}_{\text{prolong}}, \mathcal{C}_L^{(i+1)}, \ldots, \mathcal{C}_0^{(i+1)}$)
  $contraction \leftarrow$ null
  for $j \in \{0, \ldots, L\}$ **do** ▷ I.e. all steps of the original coarsening from finest to coarsest
    $contraction\_candidate \leftarrow$ find in $\mathcal{C}_j^{(i+1)}$ a contraction that affects $node$, if not found, continue with $j+1$
    if $contraction\_candidate \in \mathcal{C}_{\text{prolong}}$ **then**
      return $contraction$
    end if
    $contraction \leftarrow contraction\_candidate$
    $node \leftarrow$ apply $contraction$ to $node$, so that in the next loop, a subsequent contraction may be selected
  end for
  return $contraction$
end function

5 Experimental evaluation

The proposed methods were experimentally verified on 10 publicly available datasets. The datasets Cora and CiteSeer [50] were used with the "full" train-test split as in [12]. In addition, 2 variants of the Twitch dataset [42] with the hignest node count (DE and EN) were used. Five medium sized datasets were also used, the PubMed dataset [50], the DBLP dataset [6], the IMDB dataset [18] and both variants of the Coauthor dataset [44]. Finally, one large dataset was used, the OGB ArXiv dataset [23].

The hyper-parameters for both the node2vec model used for the embedding training and the multi-layer perceptron used for downstream classification were initially set to values used in prior art (see [16,23]) and then manually fine-tuned for each dataset.

The achitecture of the algorithm was identical accross all datasets, with the only difference being in the values of the hyper-parameters, as listed in Table 1. For the Cora dataset, the node2vec model generated an embedding in $\mathbb{R}^{128}$ from 4 random walks of length 20 for each node with a context window of size 5. The optimizer ADAM [29] was used with a learning rate of 0.01 and batches of 128 samples. The model was trained for 5 epochs and in each step of the adaptive prolongation, 100 nodes were prolonged, until reaching the original graph (the value of n_p was calculated so that the total number of training epochs would match baseline model training). The MLP classifier using the embeddings featured 3 linear layers of 128 neurons with batch normalization after each layer. Each layer was normalized using dropout [45] with the rate of 0.5. Finally, a linear layer was used for the class prediction. For the classifier, ADAM with a learning rate of 0.01 was used for 30 epochs of training with the cross-entropy loss function. Dataset features weren't used for the classifier training as the aim of this work is to compare the embeddings. The experiment was run 10 times end-to-end and results averaged. The experiments were implemented using PyTorch [38] and PyTorch Geometric [16].

Table 1. Hyper-parameter values used for different datasets

Hyper-parameter	**Cora**	**CiteSeer**	**PubMed**	**DBLP**	**Twitch**	**IMDB**	**ArXiv**	**Coauthor**
Embedding dimension	128	32	64	32	128	128	128	128
# of random walks	4	5	3	2	10	40	10	40
Random walk length	20	20	40	20	80	100	80	10
Context window size	5	5	20	5	3	5	20	5
Node2vec learning rate	0.01	0.01	0.01	0.01	0.025	0.01	0.01	0.01
Node2vec batch size	128	128	128	128	128	256	128	256
Node2vec epochs	5	7	1	1	5	1	1	1
# of MLP layers	3	3	1	3	2	2	3	2
MLP hidden layer width	128	256	128	256	64	64	256	16
Dropout rate	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
MLP learning rate	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
MLP epochs	30	80	300	300	500	100	300	100

In order to study the effect of adaptive prolongation, the method was used to assess the performance of downstream transductive classification at different coarsening levels. For each prolongation step, the intermediary embedding was fully prolonged to obtain an embedding of the original graph

G. A classifier was then trained with this embedding as input. This setup allows us to compare classification accuracy at each step of the adaptive prolongation. Figure 2 shows the results of this experiment, compared with a baseline node2vec model (that is, without any coarsening or prolongation) that was trained for the same number of epochs as the total epochs of the adaptive model over all prolongation steps.

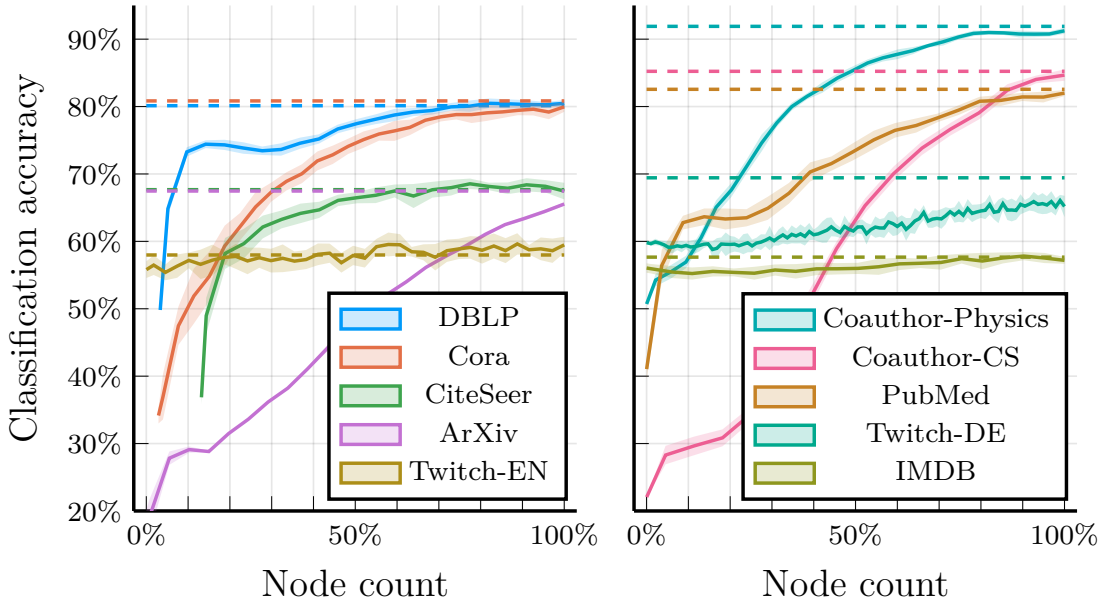

Figure 2. Downstream classifier test-set accuracies at different steps of adaptive prolongation. Dashed line shows the baseline node2vec model accuracy. The node count is taken relative to the total node count in each dataset. The results are averaged over multiple runs, with the solid line representing the mean and the shaded area denoting one standard deviation.

The behaviour of the model somewhat differs between the used datasets. For the Cora, CiteSeer, DBLP, PubMed, ArXiv and Coauthor datasets, the model starts from a very low performance, which quickly rises as the model trains for several prolongation steps. The model trained on CiteSeer attains performance comparable to the reference model when approximately half of all nodes are available to it. On the other hand, with e. g. Coauthor-CS, the model slowly approaches the reference model for the whole duration of training, only reaching comparable performance at a point where nearly the whole graph is available to it. Models trained on the two medium-sized datasets, DBLP and PubMed, exhibit a different behaviour in that they briefly reach a local maximum followed by a slight decrease in performance until finally approaching the performance of the reference model in a manner similar to the models trained on Cora and CiteSeer. The ArXiv dataset exhibits a similar behaviour to a lesser extent. A further discussion of this behaviour follows in the later part of this section. The Twitch and IMDB datasets are substantially different, with

very high initial performance and only small performance gains with increasing number of nodes available, with different variants of it exhibiting this effect to a different extent.

To study the distribution of model properties, the results were evaluated at k-th deciles of the node count of the full graph, for all possible values of k. At each decile, the performance of the model was compared to the baseline node2vec model using the Wilcoxon signed-rank test with the Holm-Bonferroni correction for multiple hypothesis testing. The hypotheses that the models are equivalent with the baseline were rejected by the test at the 5% level of significance for $k \in \{1, 2, 3, 4, 5\}$, suggesting that the adaptive prolongation approach is useful in situations where at least half of the nodes is available.

In view of recent proposals regarding statistical validation of results in machine learning [3] and to further compare the proposed model with the baseline, the results were also studied using the stronger assumptions of Bayesian estimation. The comparison was done using the Bayesian Wilcoxon signed-rank test [4] for 3 different widths of the region of practical equivalence (ROPE), 1%, 5% and 10%. The probabilities that the two models are practically equivalent are listed in Table 2. Of a particular note is the fact that at 60% complexity, the models have over a 99% probability of being within 10 percentage points of performance and at 80% complexity, they have over 99% probability of being withing 5 percentage points of performance. This shows that the proposed method may offer a significant complexity reduction in exchange for a relatively minor decrease in performance.

Table 2. The probabilities that the adaptive approach will be practically equivalent to node2vec when compared on different fractions of the full graph and with different widths of the region of practical equivalence.

Nodes	1% ROPE	5% ROPE	10% ROPE
10%	0%	0.3%	2.5%
20%	0%	0.8%	14.1%
30%	0%	1.7%	35.3%
40%	0%	5.3%	72.0%
50%	0.1%	35.3%	85.7%
60%	0.6%	62.2%	99.7%
70%	32.0%	84.7%	100.0%
80%	30.0%	99.9%	100.0%
90%	48.9%	100.0%	100.0%
100%	87.7%	100.0%	100.0%

When the models for DBLP and PubMed are studied more closely, both reach a local maximum at around 14% of the graph, followed by a slight decline and gradual approach to the baseline. This suggests a global structure in the data, which the model learns at the point of the local maxima. To investigate this hypothesis and try to explain the behaviour of the model in general, several graph measures were applied to the graphs generated during the adaptive prolongation algorithm run. All of the measures were applied to the graph G_i at each step in the prolongation process.

The measures used were edge homophily [53], node homophily [39], class homophily [31], adjusted homophily [40], balanced accuracy [40], adjusted accuracy [40], label informativeness [40], and global assortativity [36].

Table 3. The Pearson correlation coefficients of the relationship between 8 graph measures and the classification accuracy accross the prolongation procedure.

Measure	DBLP	PubMed
Edge homophily	0.91	0.99
Node homophily	0.94	0.96
Class homophily	0.89	0.99
Adjusted homophily	0.91	0.99
Balanced accuracy	0.89	0.99
Adjusted accuracy	0.89	0.99
Label informativeness	0.94	0.98
Global assortativity	-0.94	-0.38

The values of these measures were compared to the classification accuracy and a strong correlation was found for the majority of them, as listed in Table 3. This suggests the explanation of the graphs being heterophilic when very coarse (as could be expected), then reaching a point where the global structure of the graph is in place and is then only refined in a local sense. Such a behaviour may introduce nodes which have a different label than their neighbourhoods (a kind of noise in the data), which is a possible explanation for the local optima in performance.

6 Conclusion

In this work, an extension of the HARP algorithm was proposed, which generalizes it from a method for pretraining to a general graph reduction framework. A novel approach to prolonging graphs in the HARP setting was presented that selectively prolongs the graph in a way that maximizes performance of the considered downstream task under limited graph size. All of the proposed methods were experimentally verified, with the headline result being that at about 40% reduction in node count, the accuracy was still reasonably close to the accuracy on a full graph for most datasets.

In future work, a direct way of tackling the outlined problem may be explored, along the lines of our preliminary exploration in this direction [41]. The proposed approach constitutes of only a single coarsening pass, in contrast to the double procedure of coarsening followed by prolongation used in this work. As in this work, the coarsening procedure is viewed as a sequence of edge contractions. This sequence is, however, determined by the performance of an auxiliary linear regression model, providing a more realistic heuristic for the optimal way of gradually decreasing graph complexity.

References

1. Akyildiz, T.A., Alabsi Aljundi, A., Kaya, K.: Understanding Coarsening for Embedding Large-Scale Graphs. In: 2020 IEEE International Conference on Big Data (Big Data). pp. 2937–2946 (Dec 2020). https://doi.org/10.1109/BigData50022.2020.9377898
2. Bacciu, D., Di Sotto, L.: A Non-negative Factorization Approach to Node Pooling in Graph Convolutional Neural Networks. In: Alviano, M., Greco, G., Scarcello, F. (eds.) AI*IA 2019 – Advances in Artificial Intelligence. pp. 294–306. Lecture Notes in Computer Science, Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-35166-3_21

3. Benavoli, A., Corani, G., Demšar, J., Zaffalon, M.: Time for a Change: a Tutorial for Comparing Multiple Classifiers Through Bayesian Analysis. Journal of Machine Learning Research **18**(77), 1–36 (2017), http://jmlr.org/papers/v18/16-305.html
4. Benavoli, A., Corani, G., Mangili, F., Zaffalon, M., Ruggeri, F.: A Bayesian Wilcoxon signed-rank test based on the Dirichlet process. In: Proceedings of the 31st International Conference on Machine Learning. pp. 1026–1034. PMLR, Beijing, China (Jun 2014), https://proceedings.mlr.press/v32/benavoli14.html, iSSN: 1938-7228
5. Béthune, L., Kaloga, Y., Borgnat, P., Garivier, A., Habrard, A.: Hierarchical and Unsupervised Graph Representation Learning with Loukas's Coarsening. Algorithms **13**(9), 206 (Sep 2020). https://doi.org/10.3390/a13090206, https://www.mdpi.com/1999-4893/13/9/206, number: 9 Publisher: Multidisciplinary Digital Publishing Institute
6. Bojchevski, A., Günnemann, S.: Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In: 6th International Conference on Learning Representations (Feb 2018), https://openreview.net/forum?id=r1ZdKJ-0W
7. Bravo Hermsdorff, G., Gunderson, L.: A Unifying Framework for Spectrum-Preserving Graph Sparsification and Coarsening. In: Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc., Vancouver, Canada (2019), https://proceedings.neurips.cc/paper/2019/file/cd474f6341aeffd65f93084d0dae3453-Paper.pdf
8. Cai, C., Wang, D., Wang, Y.: Graph Coarsening with Neural Networks. In: International Conference on Learning Representations (Feb 2022), https://openreview.net/forum?id=uxpzitPEooJ
9. Cangea, C., Veličković, P., Jovanović, N., Kipf, T., Liò, P.: Towards Sparse Hierarchical Graph Classifiers (Nov 2018). https://doi.org/10.48550/arXiv.1811.01287, http://arxiv.org/abs/1811.01287, arXiv:1811.01287 [cs, stat]
10. Çatalyürek, Ü.V., Deveci, M., Kaya, K., Uçar, B.: Multithreaded Clustering for Multi-level Hypergraph Partitioning. In: 2012 IEEE 26th International Parallel and Distributed Processing Symposium. pp. 848–859 (May 2012). https://doi.org/10.1109/IPDPS.2012.81, iSSN: 1530-2075
11. Chen, H., Perozzi, B., Hu, Y., Skiena, S.: HARP: Hierarchical Representation Learning for Networks. Proceedings of the AAAI Conference on Artificial Intelligence **32**(1) (Apr 2018), https://ojs.aaai.org/index.php/AAAI/article/view/11849, number: 1
12. Chen, J., Ma, T., Xiao, C.: FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In: 6th International Conference on Learning Representations (Feb 2018), https://openreview.net/forum?id=rytstxWAW
13. Chen, J., Saad, Y., Zhang, Z.: Graph coarsening: from scientific computing to machine learning. SeMA Journal **79**(1), 187–223 (Mar 2022). https://doi.org/10.1007/s40324-021-00282-x
14. Chiang, W.L., Liu, X., Si, S., Li, Y., Bengio, S., Hsieh, C.J.: Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 257–266. KDD '19, Association for Computing Machinery, New York, NY, USA (Jul 2019). https://doi.org/10.1145/3292500.3330925, https://doi.org/10.1145/3292500.3330925
15. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In: Advances in Neural Information Processing Systems. vol. 29. Curran Associates, Inc., Barcelona, Spain (2016), https://proceedings.neurips.cc/paper/2016/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf
16. Fey, M., Lenssen, J.E.: Fast Graph Representation Learning with PyTorch Geometric (Apr 2019), arXiv: 1903.02428
17. Frasca, F., Rossi, E., Eynard, D., Chamberlain, B., Bronstein, M., Monti, F.: SIGN: Scalable Inception Graph Neural Networks (Nov 2020), arXiv: 2004.11198
18. Fu, X., Zhang, J., Meng, Z., King, I.: MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In: Proceedings of The Web Conference 2020. pp. 2331–2341 (Apr 2020). https://doi.org/10.1145/3366423.3380297, http://arxiv.org/abs/2002.01680, arXiv:2002.01680 [cs]

19. Gao, H., Ji, S.: Graph U-Nets. In: Proceedings of the 36th International Conference on Machine Learning. pp. 2083–2092. PMLR (May 2019), https://proceedings.mlr.press/v97/gao19a.html, iSSN: 2640-3498
20. Grattarola, D., Zambon, D., Bianchi, F.M., Alippi, C.: Understanding Pooling in Graph Neural Networks. IEEE Transactions on Neural Networks and Learning Systems pp. 1–11 (2022). https://doi.org/10.1109/TNNLS.2022.3190922, conference Name: IEEE Transactions on Neural Networks and Learning Systems
21. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 855–864 (2016)
22. Herrmann, J., Özkaya, M.Y., Uçar, B., Kaya, K., Çatalyürek, Ü.V.: Multilevel Algorithms for Acyclic Partitioning of Directed Acyclic Graphs. SIAM Journal on Scientific Computing **41**(4), A2117–A2145 (Jan 2019). https://doi.org/10.1137/18M1176865, https://epubs.siam.org/doi/abs/10.1137/18M1176865, publisher: Society for Industrial and Applied Mathematics
23. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J.: Open Graph Benchmark: Datasets for Machine Learning on Graphs (Feb 2021). https://doi.org/10.48550/arXiv.2005.00687, http://arxiv.org/abs/2005.00687, arXiv:2005.00687 [cs, stat]
24. Huang, Q., He, H., Singh, A., Lim, S.N., Benson, A.R.: Combining Label Propagation and Simple Models Out-performs Graph Neural Networks (Nov 2020), arXiv:2010.13993
25. Huang, Z., Zhang, S., Xi, C., Liu, T., Zhou, M.: Scaling Up Graph Neural Networks Via Graph Coarsening. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 675–684. KDD '21, Association for Computing Machinery, New York, NY, USA (Aug 2021). https://doi.org/10.1145/3447548.3467256
26. Jin, W., Tang, X., Jiang, H., Li, Z., Zhang, D., Tang, J., Yin, B.: Condensing Graphs via One-Step Gradient Matching. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 720–730. KDD '22, Association for Computing Machinery, New York, NY, USA (Aug 2022). https://doi.org/10.1145/3534678.3539429, https://doi.org/10.1145/3534678.3539429
27. Jin, W., Zhao, L., Zhang, S., Liu, Y., Tang, J., Shah, N.: Graph Condensation for Graph Neural Networks. In: The Tenth International Conference on Learning Representations (Jan 2022), https://openreview.net/forum?id=WLEx3Jo4QaB
28. Kammer, F., Meintrup, J.: Space-Efficient Graph Coarsening with Applications to Succinct Planar Encodings (Jun 2022), arXiv:2205.06128
29. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization (Jan 2017), arXiv:1412.6980
30. Kipf, T.N., Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks. In: 5th International Conference on Learning Representations, {ICLR} 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, Toulon, France (Apr 2017), https://openreview.net/forum?id=SJU4ayYgl
31. Lim, D., Hohne, F., Li, X., Huang, S.L., Gupta, V., Bhalerao, O., Lim, S.N.: Large Scale Learning on Non-Homophilous Graphs: New Benchmarks and Strong Simple Methods. In: Advances in Neural Information Processing Systems. vol. 34, pp. 20887–20902. Curran Associates, Inc., online (2021), https://proceedings.neurips.cc/paper/2021/file/ae816a80e4c1c56caa2eb4e1819cbb2f-Paper.pdf
32. Liu, C., Ma, X., Zhan, Y., Ding, L., Tao, D., Du, B., Hu, W., Mandic, D.: Comprehensive Graph Gradual Pruning for Sparse Training in Graph Neural Networks (Jul 2022), arXiv:2207.08629
33. Liu, N., Jian, S., Li, D., Zhang, Y., Lai, Z., Xu, H.: Hierarchical Adaptive Pooling by Capturing High-order Dependency for Graph Representation Learning. IEEE Transactions on Knowledge and Data Engineering pp. 1–1 (2021). https://doi.org/10.1109/TKDE.2021.3133646, conference Name: IEEE Transactions on Knowledge and Data Engineering
34. Loukas, A.: Graph Reduction with Spectral and Cut Guarantees. J. Mach. Learn. Res. **20**(116), 1–42 (2019)
35. Makarov, I., Kiselev, D., Nikitinsky, N., Subelj, L.: Survey on graph embeddings and their applications to machine learning problems on graphs. PeerJ Computer Science **7**, e357 (Feb 2021). https://doi.org/10.7717/peerj-cs.357, https://peerj.com/articles/cs-357, publisher: PeerJ Inc.

36. Newman, M.E.J.: Mixing patterns in networks. Physical Review E **67**(2), 026126 (Feb 2003). https://doi.org/10.1103/PhysRevE.67.026126, https://link.aps.org/doi/10.1103/PhysRevE.67.026126, publisher: American Physical Society
37. Osei-Kuffuor, D., Li, R., Saad, Y.: Matrix Reordering Using Multilevel Graph Coarsening for ILU Preconditioning. SIAM Journal on Scientific Computing **37**(1), A391–A419 (Jan 2015). https://doi.org/10.1137/130936610, https://epubs.siam.org/doi/abs/10.1137/130936610, publisher: Society for Industrial and Applied Mathematics
38. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F.d., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32. pp. 8024–8035. Curran Associates, Inc., Vancouver, Canada (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
39. Pei, H., Wei, B., Chang, K.C.C., Lei, Y., Yang, B.: Geom-GCN: Geometric Graph Convolutional Networks (Feb 2020). https://doi.org/10.48550/arXiv.2002.05287, http://arxiv.org/abs/2002.05287, arXiv:2002.05287 [cs, stat]
40. Platonov, O., Kuznedelev, D., Babenko, A., Prokhorenkova, L.: Characterizing Graph Datasets for Node Classification: Beyond Homophily-Heterophily Dichotomy (Sep 2022). https://doi.org/10.48550/arXiv.2209.06177, http://arxiv.org/abs/2209.06177, arXiv:2209.06177 [cs, math]
41. Procházka, P., Mareš, M., Dědič, M.: Downstream Task Aware Scalable Graph Size Reduction for Efficient GNN Application on Big Data. In: Information Technologies - Applications and Theory (ITAT 2022). Zuberec, Slovakia (2022)
42. Rozemberczki, B., Allen, C., Sarkar, R.: Multi-Scale attributed node embedding. Journal of Complex Networks **9**(2), cnab014 (Apr 2021). https://doi.org/10.1093/comnet/cnab014, https://doi.org/10.1093/comnet/cnab014
43. Schulz, T.H., Horváth, T., Welke, P., Wrobel, S.: Mining Tree Patterns with Partially Injective Homomorphisms. In: Berlingerio, M., Bonchi, F., Gärtner, T., Hurley, N., Ifrim, G. (eds.) Machine Learning and Knowledge Discovery in Databases. pp. 585–601. Lecture Notes in Computer Science, Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-10928-8_35
44. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of Graph Neural Network Evaluation (Jun 2019). https://doi.org/10.48550/arXiv.1811.05868, http://arxiv.org/abs/1811.05868, arXiv:1811.05868 [cs, stat]
45. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research **15**(56), 1929–1958 (2014), http://jmlr.org/papers/v15/srivastava14a.html
46. Topping, J., Giovanni, F.D., Chamberlain, B.P., Dong, X., Bronstein, M.M.: Understanding over-squashing and bottlenecks on graphs via curvature. In: The Tenth International Conference on Learning Representations (Sep 2021), https://openreview.net/forum?id=7UmjRGzp-A
47. Ubaru, S., Saad, Y.: Sampling and multilevel coarsening algorithms for fast matrix approximations. Numerical Linear Algebra with Applications **26**(3), e2234 (2019). https://doi.org/10.1002/nla.2234, https://onlinelibrary.wiley.com/doi/abs/10.1002/nla.2234
48. Veličković, P.: Geometric Deep Learning - Grids, Groups, Graphs, Geodesics, and Gauges (2021), https://geometricdeeplearning.com/
49. Xie, Y., Yao, C., Gong, M., Chen, C., Qin, A.K.: Graph convolutional networks with multi-level coarsening for graph classification. Knowledge-Based Systems **194**, 105578 (Apr 2020). https://doi.org/10.1016/j.knosys.2020.105578, https://www.sciencedirect.com/science/article/pii/S0950705120300629
50. Yang, Z., Cohen, W., Salakhudinov, R.: Revisiting Semi-Supervised Learning with Graph Embeddings. In: Proceedings of The 33rd International Conference on Machine Learning. pp. 40–48. PMLR, New York, NY, USA (Jun 2016), https://proceedings.mlr.press/v48/yanga16.html

51. Zhang, W., Yang, J., Shang, F.: HARP Pro: Hierarchical Representation Learning based on global and local features for social networks (2021)
52. Zhao, B., Mopuri, K.R., Bilen, H.: Dataset Condensation with Gradient Matching. In: International Conference on Learning Representations (Jan 2021), https://openreview.net/forum?id=mSAKhLYLSsl
53. Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., Koutra, D.: Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In: Advances in Neural Information Processing Systems. vol. 33, pp. 7793–7804. Curran Associates, Inc., online (2020), https://proceedings.neurips.cc/paper/2020/file/58ae23d878a47004366189884c2f8440-Paper.pdf