Adaptive graph coarsening in the context of local graph quality

Marek Dědič[1,2,*], Lukáš Bajer[2], Jakub Repický[2], Pavel Procházka[2] and Martin Holeňa[3]

[1]Czech Technical University in Prague, Břehová 7, Prague, Czech Republic
[2]Cisco Systems, Inc., Karlovo náměstí 10, Prague, Czech Republic
[3]Institute of Computer Science, Czech Academy of Sciences, Pod vodárenskou věží 2, Prague, Czech Republic

Abstract

Graph based models are used for tasks with increasing size and computational demands. We present a method for studying graph properties from the point of view of a downstream task. More precisely, the method allows a user to precisely select the resolution at which the graph in question should be coarsened. Our method builds on an existing algorithm for pretraining on coarser graphs, HARP. We extend both main parts of the algorithm in order to observe the effect of graph coarsening to model quality on a fine level. We present a general framework for graph coarsenings, providing two alternative algorithms based on graph diffusion convolution and evolutionary algorithms. Additionally, we present a novel way for un-coarsening the reduced graph in a targeted way based on the confidence of downstream classification for particular nodes. Together, these enhancements provide sufficient detail where needed, while collapsing structures where per-node information is not necessary for high model performance. Our method is a general meta-model for enhancing graph embedding models such as node2vec. We apply the method to several datasets and discuss the differing behaviour on each of them. Furthermore, we compare the proposed coarsening schemas.

Keywords

Graph representation learning, Graph coarsening, Graph diffusion, 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 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 [1, 2], the structure of the underlying data has, until recent works [3, 4], received much less attention. In this work, we investigate graph structure, in particular its granularity, in the context of quality of graph

Data and Model Quality for Mining and Learning with Graphs: Methods and Open Challenges @ECML-PKDD 2022, September 23, 2022, Grenoble, France

*Corresponding author.

EMAIL: marek@dedic.eu (M. Dědič); lubajer@cisco.com (L. Bajer); jrepicky@cisco.com (J. Repický); paprocha@cisco.com (P. Procházka); martin@cs.cas.cz (M. Holeňa)

URL: https://dedic.eu (M. Dědič); http://cs.cas.cz/~martin (M. Holeňa)

ORCID: 0000-0003-1021-8428 (M. Dědič); 0000-0002-9402-6417 (L. Bajer); 0000-0002-3695-1727 (J. Repický); 0000-0002-2536-9328 (M. Holeňa)

© 2022 Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

embedding, subsequently reflected in the quality of the downstream task relying on that embedding. The importance of such an investigation follows from the importance of the quality aspect for the downstream tasks.

Typically, an application of machine learning to graphs has two phases: representation learning, which maps the graph into a Euclidean space, and a downstream task, such as classification, regression, or clustering. The first phase has very high computational demands, which can be substantially decreased with graph coarsening. However, it is known that there is an interplay between coarsening and the quality of embedding [5, 6], which in turn entails an interplay between coarsening and the quality of the downstream task.

Our work builds on the HARP [7] method for pretraining on coarsened graphs. In HARP, a graph is repeatedly coarsened and the coarser graphs are then used in reverse order (from coarsest to finest) to pre-train a graph representation learning algorithm. While HARP itself works with and modifies the graph structure, this is not the main interest of its authors and it does not include any adaptation to the quality of the obtained result as the authors focus more on the representation and classification accuracy for the original graph. In our work, we modify and generalize the HARP framework to closely study the relationship between graph coarsenings and graph quality in terms of the performance of a downstream task. In our particular case, we chose transductive node classification as the final task, however, the presented algorithms are general graph representation learners that can be utilized for a wide variety of tasks.

The main contributions of this paper are the general framework for graph coarsening and extensions of the HARP algorithm. We extend both main parts of the algorithm in order to observe the effect of graph coarsening to model quality on a fine level. We present two alternative graph coarsening schemes based on graph diffusion convolution and evolutionary algorithms. Additionally, we present a novel way for un-coarsening the reduced graph in a targeted way, based on the confidence of downstream classification for particular nodes. This method maximizes performance under limited graph size.

In the next section, related work is discussed, with the exception of HARP, which is the basis of the reported research, therefore is addressed in detail in Section 3, together with our proposal how to extend it into a general framework for graph coarsening. Section 4 is the core of this work, presenting our extension of the prolongation step, as well as several proposed alternative graph coarsening schemes. Finally, all proposals are experimentally verified and compared in Section 5.

2. Related work

The publications most relevant to our research are [7], in which the HARP approach is proposed, and [8], in which graph coarsening is performed by means of graph diffusion. Because we directly extend, modify or combine those methods, they will be explained in detail in Sections 3.1 and 4.2.2. Other important works concerning graph coarsening are [5, 9, 10], which survey numerous coarsening methods, [11], which presents results concerning scalability or graph coarsening, and [12] which shows relationships of graph coarsening to properties of the Laplacian. In view of the fact that the HARP approach, which we extend and modify, is a multilevel approach, we paid attention also to the multilevel graph coarsening methods proposed in [13] and [14], the latter

being also inspired by HARP. Also of note is the recent work [15] which presents an alternative coarsening approach for planar graphs and [16], which studies a trade-off similar to the one in this work from a different point of view and with different models.

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. A general framework covering both coarsening and sparsification has been proposed in [17]. Elaboration of graph coarsening methods in machine learning can built on several decades of succesful application of their main kinds, such as pairwise aggregation, independent sets, or algebraic distance, in numerical linear algebra [9].

3. The HARP framework

In this Section, first, a brief overview of the HARP framework for graph coarsening [7] is presented, followed by our analysis of the properties of graph coarsenings in general.

3.1. HARP

HARP is a method for improving the performance of graph representation learning (i.e. graph embedding) algorithms such as DeepWalk [18], node2vec [19], or, in general, any algorithm that produces embeddings as a distinct output. The method is a combination of dataset augmentation and pre-training based on the general principle that graph-based models train more efficiently on smaller graphs and can thus be pre-trained on a coarsened representation of the graph at hand. Moreover, the coarsened graphs are able to approximate the global properties of the original data, enabling the representations to better encapsulate such a global structure. In an overview, the method consists of the following steps:

Consider an undirected graph G with nodes $V(G)$ and edges $E(G)$. The aim of the graph coarsening part of the algorithm is to generate a sequence of graphs $G_0, G_1, G_2, \ldots, G_L$ where $G_0 = G$ and $L \in \mathbb{N}$ is a hyper-parameter of the method. In this sequence, each graph G_i is generated from the graph G_{i-1} by coarsening it – lowering the number of nodes and edges while preserving in some sense the general structure of the graph. Following [7], let φ_i denote the mapping of G_{i-1} such that $G_i = \varphi_i(G_{i-1})$.

1. **Dataset augmentation**. The graph G is consecutively reduced in size by the application of several graph coarsening schemas. In each step, the coarsened graph can be viewed as an ever coarser representation of the graph data and its global structure. This step can be run ahead-of-time to produce the resulting coarsened graphs $G_0, G_1, \ldots, G_L$.

After all the coarsened graphs are pre-computed, the method itself can be executed by repeating the following steps on the graphs from the coarsest to the finest (i.e. from G_L to G_0):

2. **Training on an intermediary graph**. The graph embedding model is trained on G_i, producing Φ_{G_i}, an embedding of the graph in a Euclidean space.
3. **Embedding prolongation**. The embedding Φ_{G_i} is prolonged from the current graph to one that is one step closer to G_0, yielding $\Phi_{G_{i-1}}$. This embedding is used as the starting state for training on G_{i-1}.

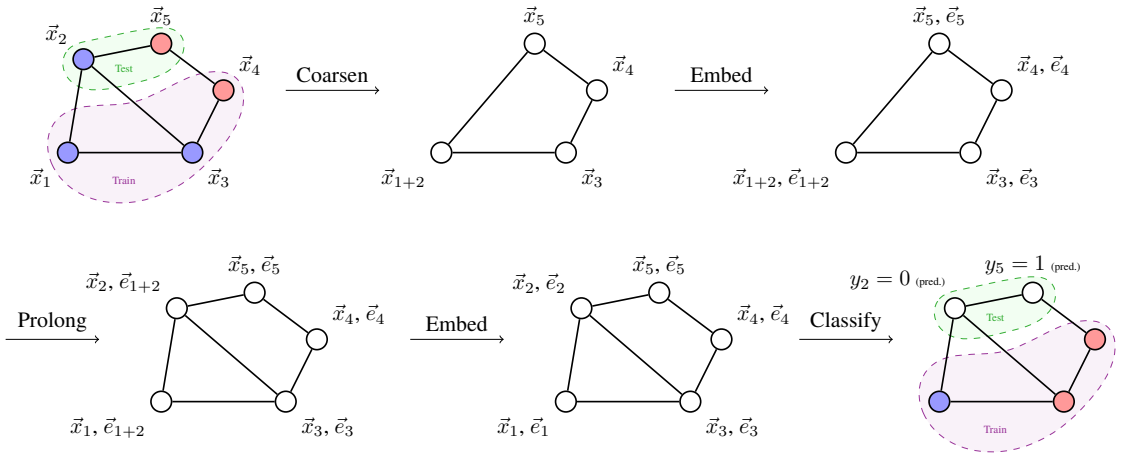

Figure 1: An overview of the HARP processing pipeline with one level of coarsening

These steps are repeated until Φ_{G_0} is computed. While the training step of this schema is a straightforward application of one of the aforementioned embedding algorithms to G_i, the particular details of the coarsening and prolongation steps are further explained in Sections 4.1 and 4.2.1.

The first step is independent of the rest of the computation and can be done ahead of time. The last two steps can be seen as a form of pre-training for the model that is to be learnt on the original graph. The whole HARP pipeline is demonstrated in Figure 1.

3.2. Properties of graph coarsenings

The HARP framework presents a particular way of producing coarsened graphs. In general, however, we would like to describe any coarsening of a graph in order to be able to explore the set of all coarsened graphs to study the properties of both the applied algorithms, as well as the underlying data.

To properly describe the set of all coarsenings, some assumptions are made about what constitutes a coarsening. In our work, we define a coarsening as a set $\mathcal{C} \subseteq E(G)$, i.e. a subset of edges of the original graph. The graph $H = \varphi_{\mathcal{C}}(G)$ obtained by applying this coarsening to G is constructed from G by contracting all edges in $\mathcal{C}$[1]. This description of a graph coarsening is inspired and builds upon the work [20], which is why it differs from the weaker definition of a graph reduction as selecting a subset of nodes and edges [11, 12].

Under such a definition of a graph coarsening, the set of all coarsenings of a graph G forms a complete bounded lattice with the partial order

$$\varphi_{\mathcal{C}_1} \preceq \varphi_{\mathcal{C}_2} \iff \mathcal{C}_1 \subseteq \mathcal{C}_2$$

The least element (bottom) of this lattice is the coarsening $\perp = \varphi_{\emptyset}$ where $\perp(G) = G$. The greatest element (top) of this lattice is the coarsening $\top = \varphi_{E(G)}$, where $\top(G)$ is a graph in

[1]Whether to allow parallel edges to arise from this operation is left to the user.

which there is a single node for each connected component in G. In ideal conditions, to study the effect of graph coarsenings on its quality (viewed through the lens of a downstream task), one would exhaustively study this lattice. In practice, such an exhaustive search is computationally infeasible. With a method such as HARP, one can view the coarsened graphs produced by the method as a kind of "path" through the set of all coarsenings - the individual coarsenings represent one way of navigating the lattice with such a path forming a chain between $\top$ and $\bot$.

4. HARP extension for flexible performance-complexity balancing

Graph-based methods such as node2vec typically have a large number of parameters – on the widely used OGBN-ArXiv dataset (see [21]), 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 [22, 23]). As the authors of [7] 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.

In this work, we extend the general HARP framework to study the preformance-complexity characteristics of graph data. To this end, we propose alternatives to both the coarsening as well as the prolongation step of HARP. First, in Section 4.1, we replace the simple prolongation approach by an adaptive prolongation algorithm. Second, in Section 4.2, we study two alternative ways of coarsening the graph.

4.1. The adaptive prolongation approach

In standard HARP, once the coarsened graphs are obtained, the way to train the graph embedding is fairly straightforward. Starting with the coarsest graph, an embedding model such as node2vec is trained for a given amount of training epochs. Following that, a step to a graph that is one level finer is made. The embedding learned on the immediately preceding coarser graph is *prolonged* to the embedding of the following finer graph, in which the representations of merged nodes are copied and reused. Then, with this prolonged embedding as the starting state, the embedding algorithm continues training and this process is repeated until reaching the original graph.

While this style of prolongation is fine when HARP is used only as a means of pre-training, this approach is far too crude when studying the relationship between graph complexity and the quality of graph embedding and subsequent downstream applications. For example, the widely-used Cora dataset [24] has in its original form 2708 nodes, while the graph resulting from one application of the HARP coarsening schema has only about 1100 nodes (exact numbers may differ run-by-run). Such a relatively high reduction ratio effectively prevents any sufficient understanding of the relationship between graph reduction and changes in the quality of its embedding.

In order to offer a more fine-grained observation of the graph complexity and its effect on the downstream task, we present the adaptive prolongation approach. This algorithm works with the

pre-coarsened graphs produced for example by HARP, however, the embedding is learned in a different manner.

The adaptive prolongation approach (Algorithm 1) uses the pre-computed coarsened graphs as a way to progressively increase the number of nodes with which the embedding is trained. However, its iterations of embedding training and prolongation are decoupled from the pre-computed coarsened graphs. Instead, in each step of the training, the current embedding is used to train a node classifier that guides which nodes should be prolonged. The node classifier is equivalent to the one which is to be eventually used for the downstream task and is trained on the same training subset of graph nodes. A measure guiding the prolongation is to be produced using this classifier – ideally, only the nodes where the classifier performs the worst should be prolonged. To this end, the confidence of the classifier for each node is used. To measure such confidence, entropy of the output of the softmax layer for each node is used – this represents the amount of uncertainty in the classifier for each node. For each node, starting with nodes with the highest entropy, the pre-computed coarsenings are searched for edge contractions involving the node, with preference for contractions from later steps of the repeated coarsening (corresponding to coarser graphs). A given number of such edge contractions is selected and undone in each prolongation step, gradually advancing from the coarsest graph to the original, finest one.

4.2. More general approaches to coarsening

While the adaptive prolongation approach substantially generalizes the original method into a powerful tool for studying and leveraging graph structure and its properties under a coarsening, it still relies on the pre-computed coarsenings to guide the prolongation process. Moreover, the general coarsening schema described in Section 3.2 and its variants described in this Section are useful even beyond the specific methods proposed in this work. In this Section, we first present a brief overview of the coarsening algorithm as proposed by [7], followed by two alternative proposals for coarsening construction.

4.2.1. HARP coarsening

The authors of [7] introduce two particular coarsening methods that together realize the function φ_i from Section 3.1 – *edge collapsing* and *star collapsing*. Edge collapsing is a very simple method – out of all the edges $E\,(G)$, a maximal subset E' is selected randomly such that no two edges from E' are incident on the same node. Then, each edge in E' is contracted.

The edge collapsing algorithm is a good general way of lowering the number of nodes in a graph, however, some structures are not easily collapsed by it. An example of such a structure is a "star" – a single node connected to many other nodes. To coarsen graphs with such structures effectively, the star collapsing algorithm is proposed. For each such *hub* node u in order of decreasing degree, its unconnected neighbouring nodes are taken and merged pairwise. All edges incident on such nodes are replaced with edges incident on the corresponding newly created nodes. As in edge collapsing, nodes to be merged are selected in such a way that no node is merged twice.

These two approaches are combined, with each HARP coarsening step being a star collapsing step followed by an edge collapsing step. Of a particular note is the fact such a coarsening scheme is not in agreement with the definition presented in Section 3.2. The star collapsing algorithm

Algorithm 1 Adaptive prolongation

Require: G ▷ The original graph
Require: $\boldsymbol{e}_{i+1}$ ▷ The previous embedding
Require: $\boldsymbol{y}_{\text{train}}$ ▷ Training labels
Require: $replacement\_maps$ ▷ A list of records of all the original graph coarsenings
Require: n_p ▷ The number of nodes to prolong
Ensure: $merges\_to\_prolong$ ▷ A list of node merges to be prolonged ("undone")
Ensure: $new\_replacement\_maps$ ▷ Updated coarsening records without the prolonged merges

$node\_order \leftarrow$ GET_NODE_ORDER$(G, \boldsymbol{e}_{i+1}, \boldsymbol{y}_{\text{train}}, replacement_maps)$
$merges\_to\_prolong \leftarrow$ SELECT_NODES$(node_order, n_p, replacement_maps)$
$new\_replacement\_maps \leftarrow$ UNDO_MERGES$(replacement_maps, merges_to_prolong)$

function GET_NODE_ORDER$(G, \boldsymbol{e}_{i+1}, \boldsymbol{y}_{\text{train}}, replacement_maps)$
$\boldsymbol{e}_0^{\text{temp}} \leftarrow$ use $replacement\_maps$ to fully prolong the current embedding $\boldsymbol{e}_{i+1}$ to G
$model \leftarrow$ TRAIN_DOWNSTREAM_MODEL$(\boldsymbol{e}_0^{\text{temp}}, \boldsymbol{y}_{\text{train}})$
$entropy_per_node \leftarrow H\,($PREDICT$(model, node))$ for each $node \in V\,(G)$
return $V\,(G)$, sorted in descending order by $entropy\_per\_node$
end function

function SELECT_NODES$(ordered_nodes, n_p, replacement_maps)$
$selected\_merges \leftarrow \{\}$
for $node \in ordered\_nodes$, **until** $|selected\_merges| = n_p$ **do**
$merge \leftarrow$ RESOLVE_MERGE$(node, selected_merges, replacement_maps)$
If $merge \neq$ null, add $merge$ to $selected\_merges$
end for
return $selected\_merges$
end function

function RESOLVE_MERGE$(node, already_selected_merges, replacement_maps)$
$merge \leftarrow$ null
for $replacement\_map \in replacement\_maps$ from finest graph to coarsest **do**
$merge\_candidate \leftarrow$ find in $replacement\_map$ a merge that affects $node$, if not found, continue with next $replacement\_map$
if $merge\_candidate \in already\_selected\_merges$ **then**
return $merge$
end if
$merge \leftarrow merge\_candidate$
Apply $merge$ to $node$, so that in the next loop, a subsequent merge may be selected
end for
return $merge$
end function

merges nodes that are adjacent to a common hub node, however, these nodes need not be connected by an edge. In our previous work [25], we experimentally verified that the star collapsing algorithm can be replaced by a similar algorithm that merges nodes adjacent on a hub node with the hub node itself. Such a replacement modifies the HARP coarsening scheme to be in line with the definition presented in Section 3.2.

4.2.2. Graph diffusion coarsening

Our definition of a graph coarsening requires choosing some edges from the original graph. Intuitively, one way of constructing a graph coarsening would be to merge nodes which are similar and therefore no significant amount of information is lost due to such a coarsening. Following both of these premises, we propose a coarsening based on Graph Diffusion Convolution (GDC) [8] algorithm. GDC is a general graph transformation which, in an overview, constructs for a given graph a new edge set, represented by a so-called sparsified generalized graph diffusion matrix $\hat{\mathbf{S}}$, which would, in the original setting of the method, be used as a replacement for the adjacency matrix of the graph. To apply this algorithm as a way of coarsening the graph, the edge set obtained by GDC is intersected with the edges of the original graph and the resulting edges contracted in the graph, i.e.

$$\mathcal{C} = E\left(G_{\hat{\mathbf{S}}}\right) \cap E\left(G\right)$$

using the notation of a general graph coarsening presented in Section 3.2.

In principle, the authors of GDC define the generalized graph diffusion matrix

$$\mathbf{S} = \sum_{k=1}^{\infty} \theta_k \mathbf{T}^k \tag{1}$$

such that the power series converges. The parameters θ_k together with the generalized transition matrix $\mathbf{T}$ define the exact way in which the diffusion is achieved. Among the choices for $\mathbf{T}$ is the random walk transition matrix $\mathbf{T}_{\text{rw}} = \mathbf{A}\mathbf{D}^{-1}$ and the symmetric transition matrix $\mathbf{T}_{\text{sym}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ where $\mathbf{D}$ is the diagonal matrix of node degrees. Two special cases of this general schema are the Personalized PageRank algorithm (PPR) [26] and the heat kernel [27]. The result of most diffusion processes (including PPR and the heat kernel) is a dense matrix $\mathbf{S}$, which then needs to be sparsified. In GDC, two methods of sparsification are considered – thresholding the matrix values and selecting top-k entries for each column of the matrix. After sparsification, the transition matrix is normalized in a similar way as the input adjacency matrix, i.e. by rows, columns or symmetrically, thus finally producing $\hat{\mathbf{S}}$.

4.2.3. Evolved coarsening

The idea behind the evolutionary coarsening algorithm is to find a coarsening providing a good trade-off between the performance and complexity of the downstream task by searching through a space of graph coarsenings composed of elementary (atomic) coarsenings, each of which contracts $O(1)$ edges. This search is concluded using a simple evolutionary algorithm.

Let be $\mathcal{AC}$ a basis of *atomic coarsenings*, $\mathcal{AC} = \{\psi_i, i \in \{1, \ldots, K\}\}$ for some $K \in \mathbb{N}$, where each ψ_i is a function of a hyper-parameter vector θ_i and a graph G. For the sake of simplicity,

we will assume that in the course of an optimization run, each ψ_i is already partially applied to a particular fixed value of its hyper-parameters θ_i and thus is a function of a graph only.

Each candidate solution $\boldsymbol{x} = (j_i)_{i=1}^{L}, j_i \in \{1, \ldots, K\}$ is a sequence of L integers (genes), where the length L depends only on the graph size. The genome encodes a transformation $\varphi_{\boldsymbol{x}} = \psi_{j_0} \cdot \psi_{j_1} \cdot \ldots \cdot \psi_{j_L}$ of the input graph G, where $\cdot$ denotes function composition.

The fitness $f(\boldsymbol{x})$, i.e. the objective function, is the classification accuracy of the downstream task on the graph $\varphi_{\boldsymbol{x}}(G)$.

The evolutionary coarsening algorithm is a simple genetic algorithm. The offspring population is generated by the mutation and crossover operators. The crossover operator is a two-point crossover. The mutation of a single gene is a uniform choice from the index set of $\mathcal{AC}$. Both the mutation and the crossover operators preserve the candidate solution length. To compute the fitness, the graph is passed through the composite coarsening defined by each candidate solution. The selection is a standard tournament selection.

5. Experimental evaluation

5.1. Experiment setup

5.1.1. Datasets

The proposed methods were experimentally verified on several datasets. The datasets Cora and CiteSeer [24] were used with the "full" train-test split as in [28], as well as the Enzymes dataset [29], where only the 100 largest datsets were selected and used for training, validation and testing with a random 60:20:20 split. Two larger datasets were also used, the PubMed dataset [24] containing 19 717 nodes and 88 648 edges, and the DBLP dataset [30] containing 17 716 nodes and 105 734 edges.

5.1.2. Methodology of experiments

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 [21, 31]) and then manually fine-tuned for each dataset.

For the Cora dataset, the node2vec model generated an embedding into $\mathbb{R}^{128}$ from 4 random walks of length 20 for each node with a context window of size 5. The optimizer ADAM [32] 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 MLP classifier using the embeddings featured 3 linear layers of 128 neurons with batch normalization after each layer. Each layer was normalized using dropout [33] with the rate of 0.5. Finally, a linear layer was used for the class prediction. 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. For other datasets, the overall design of the experiment was identical, with the only difference being in the values of some hyper-parameters. Additionally, for the Enzymes datasets, the results were not averaged across

Table 1
Hyper-parameter values used for different datasets

Hyper-parameter	Cora	CiteSeer	Enzymes	PubMed	DBLP
Embedding dimension	128	32	16	64	32
# of random walks	4	5	40	3	2
Random walk length	20	20	10	40	20
Context window size	5	5	5	20	5
Node2vec learning rate	0.01	0.01	0.01	0.01	0.01
Node2vec batch size	128	128	8	128	128
Node2vec epochs	5	7	2	1	1
# of prolonged nodes	100	150	300	1000	800
# of MLP layers	3	3	3	1	3
MLP hidden layer width	128	256	32	128	256
Dropout rate	0.5	0.5	0.5	0.5	0.5
MLP learning rate	0.01	0.01	0.01	0.01	0.01
MLP epochs	30	80	100	300	300
# of runs	10	10	1	10	10

multiple runs, but across testing graphs instead. The hyper-parameter values for all datasets are listed in Table 1. The experiments were implemented using PyTorch [34] and PyTorch Geometric [31].

5.2. Evaluation of the adaptive approach

In order to study the effect of the adaptive prolongation, the adaptive prolongation method was used to assess the performance of downstream transductive classification at different coarsening levels. A node2vec model as described in the previous Section was trained with adaptive prolongation based on coarsenings pre-computed by the HARP coarsening algorithm as described in Section 4.2.1. For each prolongation step, the intermediary embedding was afterwards fully prolonged to obtain an embedding of the original graph G (as that is the only graph for which authoritative labels are available). 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.

The behaviour of the model somewhat differs between the used datasets. For each dataset, 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 Cora, 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 larger datasets, DBLP and PubMed, exhibit a substantially different behaviour. Both models 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

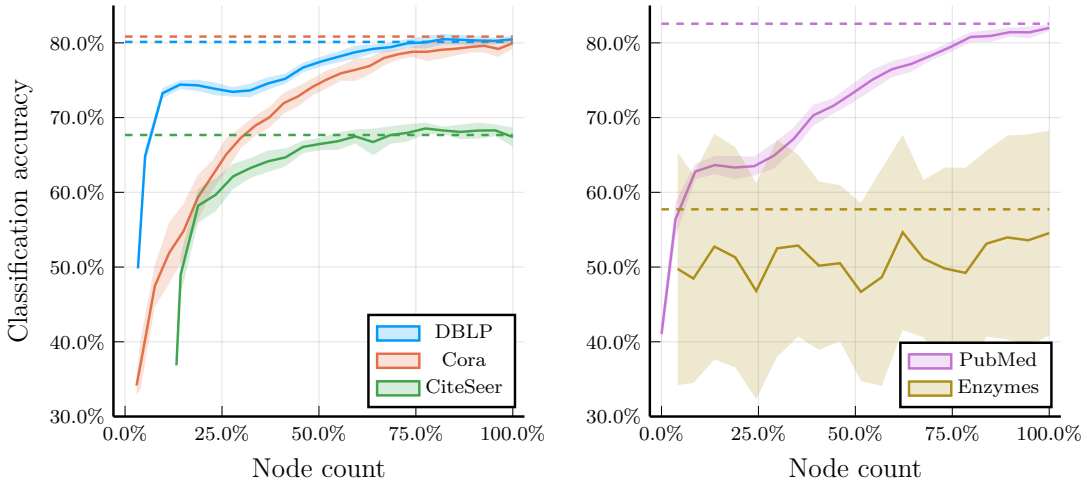

Figure 2: Downstream classifier accuracies at different steps of adaptive prolongation using the basic HARP coarsening algorithm. Dashed line shows the baseline node2vec model accuracy. The node count is taken relative to the total node count in each dataset. The results are averaged over multiple runs,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

learns at the point of the local maxima. After that point, this global structure may be obscured due to noise in the added data. The Enzymes dataset is substantially different yet again, with only small performance gains with increasing number of nodes available, however, the results contain a lot of noise, as the algorithm performace varies greatly over the different testing datasets for both the baseline as well as the adaptive prolongation approach. Of a particular note is the relatively decent accuracy even after the first prolongation step. Given the properties of the graph coarsening, this may suggest correlation between node labels of the individual graphs, which would indeed make even the coarsest step have a decent performance.

To further study the model properties, the Friedman two-way analysis of ranks was used, with the Holm-Bonferroni correction for multiple hypotheses testing. The hypotheses tested were that the resulting accuracy is the same from the k-th decile of the node count to the full graph, with tests for all possible values of k. The hypotheses were tested against the 20 testing graphs from the Enzymes dataset, as well as Cora, CiteSeer, DBLP and PubMed. None of the hypotheses were rejected by the test at the 5% level of significance. We attribute this mainly to the Enzymes dataset, which introduces a lot of noise into the data.

5.3. Comparison of coarsening approaches

To compare the proposed coarsening algorithms, they were trained on the Enzymes, Cora and CiteSeer datasets. This limitation is due to the GDC algorithm, which constructs a dense matrix S as part of the process, which is computationally demanding. There exists an approximate version of the algorithm, however, it is currently not implemented for the chosen combination of diffusion, normalization and sparsification methods. An extension of this approximate solution to cover all

Table 2
The atomic coarsening functions

Set $\mathcal{C}$ of edges to contract	Parameters
A random edge	
A random edge incident to the highest degree node	
Edges of a random triangle	
The edge whose incident nodes share the highest number of common neighbours	
The edge whose incident nodes share the lowest number of common neighbours	
A random edge whose incident nodes are adjacent to the highest degree node	
A random node of degree at least d_{lower} and its random neighbour	d_{lower}
A random node of degree at most d_{upper} and its random neighbour	d_{upper}
The edge whose incident nodes have the most neighbours in total	
The edge whose incident nodes have the least neighbours in total	
The edge whose incident nodes have the most similar k-th feature	k
The edge whose incident nodes have the least similar l-th feature	l
The edge whose incident nodes have the most similar features under the L^2 distance	
The edge whose incident nodes have the least similar features under the L^2 distance	
The edge whose incident nodes have the most similar set of features of 1-hop neighbours (under the Hausdorff distance)	
The edge whose incident nodes have the least similar set of features of 1-hop neighbours (under the Hausdorff distance)	

combinations of these parameters is possible, however, outside of the scope of this work.

For GDC coarsening, only the top-k sparsification method produces reliable results as thresholding leads to instability in the coarsening process, either collapsing the whole graph almost immediately, or not collapsing it at all. As for the other parameters, we followed [8] and have chosen symmetric normalization for the input matrix and column normalization for the output matrix. Both of the diffusion methods propsed in [8] were implemented with the recommended parameter values, that is the heat kernel was used with the diffusion time $t = 5$ and the Personalized PageRank algorithm with the teleport probability $\alpha = 0.15$.

For the evolutionary coarsening, the experimental evaluation was conducted with crossover probability $p_{\text{cx}} = 0.9$, individual and gene mutation probabilities $p_{\text{mut}} = p_{\text{gene}} = 0.1$, parent population size $\lambda = 3$, tournament size $n_{\text{tourn}} = 3$, and the individual length L set to 5% of the input graph node count. The set $\mathcal{AC}$ consisted of 16 atomic coarsenings (Table 2). Only four of them had hyper-parameters. These were initialized by random sampling where d_{lower} and d_{upper} were sampled from a Poisson distribution with $\mu = 30$ and k and l were sampled uniformly from the set of all features. The evolutionary coarsening algorithm was implemented in the DEAP framework [35].

The behaviour of the models (Figure 3) again differs substantially between the datasets. As in the previous experiment, the Enzymes dataset contains a lot of noise and generally, the methods behave in a similar manner, retaining decent performance even with at the coarsest levels and only improving slightly as more data is available. At the same time, for both the Cora and the CiteSeer dataset, a clear trend emerges where the GDC coarsening quickly outperforms the original HARP coarsening, with PPR producing better results on Cora under heavy coarsening and both having

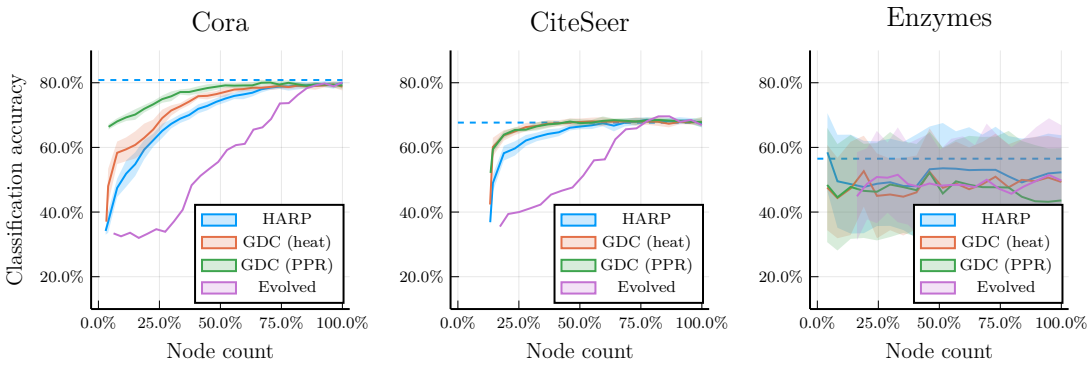

Figure 3: Downstream classifier accuracies at different steps of adaptive prolongation for different coarsening algorithms. Dashed line shows the baseline node2vec model accuracy. The node count is taken relative to the total node count in each dataset. The results are averaged over multiple runs, with the solid line representing the mean and the shaded area denoting one standard deviation.

similar behaviour on CiteSeer. The larger portions of the data the algorithms see, the more the margin shrinks until it vanishes completely. This suggests that the GDC algorithm is able to better preserve the global structure of the graph over successive coarsenings. On the other hand, the evolved coarsening clearly performs the worst, being outperformed by all other methods.

An identical statistical test to the one described in Section 5.2 was carried out for all of the coarsening algorithms. Only the GDC coarsening with the heat kernel rejected the hypothesis that the accuracy is the same from the k-th decile onwards at the 5% significance level, for $k \in \{1, 2, 3\}$. The Holm-corrected familywise p-values for $k \in \{1, 2, 3, 4, 5\}$ were $0.019, 0.023, 0.027, 0.071, 0.8$, respectively. For other algorithms, the hypotheses weren't rejected for any value of k.

6. Conclusion

In this work, the HARP algorithm was presented and generalized into an all-purpose graph coarsening schema. A novel way of prolonging graphs in the HARP setting was presented, yielding an adaptive algorithm that selectively prolongs the graph in a way that maximizes performance under limited graph size. Additionaly, 3 alternatives to HARP coarsening were presented, two based on graph diffusion and one based on evolutionary algorithms. Together, these two improvements substantially increase the versatility of HARP, turning it from a method for pre-training into a framework for graph reduction. Such a framework enables the study of properties of particular graphs, making it possible to reveal global structures. The framework may be used for lowering computational demands while preserving downstream task performance as well.

All of the proposed methods were experimentally verified. While the behaviour differs between the graphs studied, in general, our experiments reveal that at about 50% reduction in node count, the accuracy was still reasonably close to the accuracy on a full graph on most datasets. Additionally, the coarsenings based on graph diffusion were shown to outperform the original coarsening, again

with the exact difference depending on the particular dataset.

In future work, some parts of the proposed method may be substantially simplified. This applies mostly to the adaptive coarsening schema, where a surrogate metric could be used to guide the prolongation, instead of evaluating the downstream task, which may be prohibitively computationally expensive. Our work-in-progress [36] explores this direction in the setting of direct adaptive coarsening, instead of a fixed coarsening followed by adaptive prolongation. A similar simplification may be applied to the evolved coarsening, where a surrogate fitness function may also substantially lower computational costs. Regarding the proposed alternative coarsening schemas, hyper-parameter optimization techniques such as grid-search could be used to fine-tune the methods. For the evolved coarsening, encoding the coarsening parameters into the population will make the algorithm less sensitive to the initial settings with the added cost of increasing the dimension of the optimization task. Of interest is also comparison of different choices of genetic operators as well as a multi-criterial optimization with an additional criterion defined, e.g., as the reduction ratio of the input graph size.

Acknowledgments

The research reported in this paper has been supported by the German Research Foundation (DFG) funded project 467401796.

References

[1] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, in: Advances in Neural Information Processing Systems, volume 29, Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/hash/04df4d434d481c5bb723be1b6df1ee65-Abstract.html.

[2] T. N. Kipf, M. Welling, 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, 2017. URL: https://openreview.net/forum?id=SJU4ayYgl.

[3] J. Topping, F. D. Giovanni, B. P. Chamberlain, X. Dong, M. M. Bronstein, Understanding over-squashing and bottlenecks on graphs via curvature, in: The Tenth International Conference on Learning Representations, 2021. URL: https://openreview.net/forum?id=7UmjRGzp-A.

[4] P. Veličković, Geometric Deep Learning - Grids, Groups, Graphs, Geodesics, and Gauges, 2021. URL: https://geometricdeeplearning.com/geometricdeeplearning.com/.

[5] T. A. Akyildiz, A. Alabsi Aljundi, K. Kaya, Understanding Coarsening for Embedding Large-Scale Graphs, in: 2020 IEEE International Conference on Big Data (Big Data), 2020, pp. 2937–2946. doi:10.1109/BigData50022.2020.9377898.

[6] I. Makarov, D. Kiselev, N. Nikitinsky, L. Subelj, Survey on graph embeddings and their applications to machine learning problems on graphs, PeerJ Computer Science 7 (2021) e357. URL: https://peerj.com/articles/cs-357. doi:10.7717/peerj-cs.357, publisher: PeerJ Inc.

[7] H. Chen, B. Perozzi, Y. Hu, S. Skiena, HARP: Hierarchical Representation Learning for Networks, Proceedings of the AAAI Conference on Artificial Intelligence 32 (2018). URL: https://ojs.aaai.org/index.php/AAAI/article/view/11849, number: 1.

[8] J. Gasteiger, S. Weiß enberger, S. Günnemann, Diffusion Improves Graph Learning, in: Advances in Neural Information Processing Systems, volume 32, Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/hash/23c894276a2c5a16470e6a31f4618d73-Abstract.html.

[9] J. Chen, Y. Saad, Z. Zhang, Graph coarsening: from scientific computing to machine learning, SeMA Journal 79 (2022) 187–223. doi:10.1007/s40324-021-00282-x.

[10] C. Cai, D. Wang, Y. Wang, Graph Coarsening with Neural Networks, in: International Conference on Learning Representations, 2022. URL: https://openreview.net/forum?id=uxpzitPEooJ.

[11] Z. Huang, S. Zhang, C. Xi, T. Liu, M. Zhou, Scaling Up Graph Neural Networks Via Graph Coarsening, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 675–684. doi:10.1145/3447548.3467256.

[12] A. Loukas, Graph Reduction with Spectral and Cut Guarantees., J. Mach. Learn. Res. 20 (2019) 1–42.

[13] Y. Xie, C. Yao, M. Gong, C. Chen, A. K. Qin, Graph convolutional networks with multi-level coarsening for graph classification, Knowledge-Based Systems 194 (2020) 105578. URL: https://www.sciencedirect.com/science/article/pii/S0950705120300629. doi:10.1016/j.knosys.2020.105578.

[14] W. Zhang, J. Yang, F. Shang, HARP Pro: Hierarchical Representation Learning based on global and local features for social networks (2021).

[15] F. Kammer, J. Meintrup, Space-Efficient Graph Coarsening with Applications to Succinct Planar Encodings, 2022. ArXiv:2205.06128.

[16] C. Liu, X. Ma, Y. Zhan, L. Ding, D. Tao, B. Du, W. Hu, D. Mandic, Comprehensive Graph Gradual Pruning for Sparse Training in Graph Neural Networks, 2022. ArXiv:2207.08629.

[17] G. Bravo Hermsdorff, L. Gunderson, A Unifying Framework for Spectrum-Preserving Graph Sparsification and Coarsening, in: Advances in Neural Information Processing Systems, volume 32, Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/hash/cd474f6341aeffd65f93084d0dae3453-Abstract.html.

[18] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701–710.

[19] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.

[20] T. H. Schulz, T. Horváth, P. Welke, S. Wrobel, Mining Tree Patterns with Partially Injective Homomorphisms, in: M. Berlingerio, F. Bonchi, T. Gärtner, N. Hurley, G. Ifrim (Eds.), Machine Learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2019, pp. 585–601. doi:10.1007/978-3-030-10928-8_35.

[21] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, J. Leskovec, Open Graph

Benchmark: Datasets for Machine Learning on Graphs, 2021. ArXiv: 2005.00687.
[22] F. Frasca, E. Rossi, D. Eynard, B. Chamberlain, M. Bronstein, F. Monti, SIGN: Scalable Inception Graph Neural Networks, 2020. ArXiv: 2004.11198.
[23] Q. Huang, H. He, A. Singh, S.-N. Lim, A. R. Benson, Combining Label Propagation and Simple Models Out-performs Graph Neural Networks, 2020. ArXiv:2010.13993.
[24] Z. Yang, W. Cohen, R. Salakhudinov, Revisiting Semi-Supervised Learning with Graph Embeddings, in: Proceedings of The 33rd International Conference on Machine Learning, PMLR, 2016, pp. 40–48. URL: https://proceedings.mlr.press/v48/yanga16.html.
[25] M. Dědič, Graph Coarsening Can Increase Learning Efficiency, Technical Report, Czech Technical University in Prague, 2021. URL: http://kmwww.fjfi.cvut.cz/ddny.
[26] L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank Citation Ranking: Bringing Order to the Web., 1999. URL: http://ilpubs.stanford.edu:8090/422/, publisher: Stanford InfoLab.
[27] R. I. Kondor, J. Lafferty, Diffusion kernels on graphs and other discrete structures, in: Proceedings of the 19th international conference on machine learning, volume 2002, 2002, pp. 315–322.
[28] J. Chen, T. Ma, C. Xiao, FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling, in: 6th International Conference on Learning Representations, 2018. URL: https://openreview.net/forum?id=rytstxWAW.
[29] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, M. Neumann, TUDataset: A collection of benchmark datasets for learning with graphs, in: ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020), 2020. URL: www.graphlearning.io.
[30] A. Bojchevski, S. Günnemann, Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking, in: 6th International Conference on Learning Representations, 2018. URL: https://openreview.net/forum?id=r1ZdKJ-0W.
[31] M. Fey, J. E. Lenssen, Fast Graph Representation Learning with PyTorch Geometric, 2019. ArXiv: 1903.02428.
[32] D. P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, 2017. ArXiv:1412.6980.
[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research 15 (2014) 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.
[34] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An Imperative Style, High-Performance Deep Learning Library, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
[35] F.-A. Fortin, F.-M. D. Rainville, M.-A. Gardner, M. Parizeau, C. Gagné, DEAP: Evolutionary Algorithms Made Easy, Journal of Machine Learning Research 13 (2012) 2171–2175.
[36] P. Procházka, M. Mareš, M. Dědič, 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.