Graph Pooling with Maximum-Weight $k$-Independent Sets

Davide Bacciu  
bacciud@di.unipi.it  
Università di Pisa

Alessio Conte  
conte@di.unipi.it  
Università di Pisa

Francesco Landolfi  
francesco.landolfi@phd.unipi.it  
Università di Pisa

ABSTRACT

Graph reductions are fundamental when dealing with large scale networks and relational data. They allow to downsize tasks of high computational impact by solving them in coarsened structures. At the same time, graph reductions play the role of pooling layers in graph neural networks, to extract multi-resolution representations from structures. In these contexts, the ability of the reduction mechanism to preserve distance relationships and topological properties appears fundamental, along with a scalability enabling its application to real-world sized problems. In this paper, we introduce a graph coarsening mechanism based on the graph-theoretic concept of maximum-weight $k$-independent sets, providing a greedy algorithm that allows efficient parallel implementation on GPUs. Our method is the first graph-structured counterpart of controllable equispaced coarsening mechanisms in regular data (images, sequences). We prove theoretical guarantees for distortion bounds on path lengths, as well as the ability to preserve key topological properties in the coarsened graphs. We leverage these concepts to define a graph pooling mechanism that we empirically assess in graph classification tasks, showing that it compares favorably against pooling methods in literature.

CCS CONCEPTS

- Mathematics of computing → Graph algorithms; Approximation algorithms; Computing methodologies → Parallel algorithms; Machine learning.

KEYWORDS

Maximal Independent Sets, Pooling, Graph Neural Networks

1 INTRODUCTION

The concept of information coarsening is fundamental in the adaptive processing of data, as it provides a simple, yet effective, means to obtain multi-resolution representations of information at different levels of abstraction. In large scale problems coarsening also serves to provide computational speed-ups by solving tasks on the reduced representation, ideally with a contained loss in precision with respect to solving the original problem.

Coarsening is key in Convolutional Neural Networks (CNNs), where pooling is often used to repeatedly subsample an image to extract visual feature detectors at increasing levels of abstraction (e.g., blobs, edges, parts, objects, etc). Downsampling is also popular in the adaptive processing of timeseries where, for instance, it is used in clockwork-type Recurrent Neural Networks (RNNs) [43, 21] to store information extracted at different frequencies and timescales. More recently, the Graph Convolutional Networks (GCNs) community popularized graph reduction mechanisms as a structured counterpart of the image pooling mechanism in classical CNNs [7].

The definition of a reduction mechanism that downsamples information while preserving the distances between data points (e.g., a sample, a pixel, a timestamped observation, etc) is straightforward when working with images and timeseries. It can be achieved simply by picking up a data point every $k$ ones, where $k$ is a given reduction factor defining the distance between the sampled points in the original data. The same approach cannot be straightforwardly applied to graphs, which lack regularity and a consistent ordering among their constituent data points, i.e., the nodes. Therefore, defining a well-formed notion of downsampling for graphs becomes non-trivial. The research community has been tackling this issue by a number of approaches, including differentiable clustering of node embeddings [69, 14], graph reductions [63, 47], and node ranking [20, 33, 41, 45, 71, 58, 51]. Notably, approaches like the latter select important nodes in a graph and simply discard the rest without protecting the linked structure of the network, while reduction methods typically focus on preserving structure without accounting for the role or relevance of nodes involved.

What is yet an open problem is how to define a controllable graph coarsening method, which reduces the size while preserving the overall structure by sampling representative yet evenly spaced elements, similarly to the approaches discussed above for image and timeseries reduction. This paper provides a first attempt at introducing such a relevance- and distance-preserving graph coarsening mechanism and its use in graph pooling.

We define a mechanism to find a set of nodes that are approximately equally spaced (at distance no less than $k$) in the original graph that builds on the graph-theoretic concept of Maximal $k$-Independent Sets ($k$-MIS), that simultaneously allows enough freedom to pin-point important nodes in each area of the graph. The selected nodes are then used as vertices of the reduced graph whose topology is defined in such a way that key structural properties of the original graph are well preserved. To this end, we provide theoretical guarantees regarding distance distortions between a graph and its reduction. In particular, we prove that the path length between two nodes in the original graph can be estimated on the coarsened graph with a bounded distortion. Additionally, we prove the reduced graph has the same number of connected components as the original.

Such properties are fundamental to ensure that the original graph is downsampled evenly throughout its structure, preserving distances and sparseness of the key focal points in the graph. By this means, the reduced graph can be used as an accurate fast estimator of the distances between nodes in the original graph, where the amount of compression from the original graph can be easily regulated through the choice of the $k$ reduction factor.

Concurrently, we borrow from node-ranking methods [33] to produce a weighted graph and generalize our goal to finding a maximum weight $k$-independent set ($k$-MWIS), to use as a skeleton...
for coarsening in order to preserve relevant nodes without compromising structure. The latter point is particularly relevant for a graph pooling mechanisms as it guarantees that the structure is not broken in disconnected fragments, which can deplete performance of the neural message passing in the GCN layers. Such guarantees would typically come at a cost: classical greedy heuristics for the maximum-weight independent set problem \cite{775,52} require dynamic computational steps that do not scale well for \textsc{k-MWIS} on large graphs. As an additional contribution, we introduce a relaxed heuristic for \textsc{k-MWIS} that allows efficient parallel execution on GPU. We provide a full characterization of the computational complexity of the parallel algorithm and an empirical analysis of its costs, showing that it does not incur in any significant loss of quality (compared to the classical heuristics) on the datasets considered.

In summary, the contributions of this paper are the following:

- We introduce a graph coarsening method leveraging \textsc{k-MWIS} that is the graph-structured counterpart of equispaced sampling in flat data (Section 3.1). We provide evidence that the algorithm is deterministic and the conditions under which its results are permutation invariant (Section 3.2).
- We provide a greedy parallel algorithm to efficiently compute the \textsc{k-MWIS} reduction, which is well suited to use in GPU accelerators (Section 4.1).
- We prove theoretical guarantees on the distance distortions between a graph and its reduction (Sections 4.2 and 4.3).
- We integrate \textsc{k-MWIS} reduction as a pooling layer in GCNs, providing an empirical confirmation of its advantages over literature approaches on graph classification benchmarks (Section 6).

Notation is provided in Section 2, related works in Section 5.

2. NOTATION AND DEFINITIONS

We represent a graph \( G \) as a pair of disjoint sets \( (V, E) \), where \( V = \{1, \ldots, n\} \) is its node set and \( E \subset V \times V \) its edge set, with \( |E| = m \).

A graph can also be represented as a symmetric matrix \( A \in \mathbb{R}^{n \times n} \), such that \( A_{uv} = A_{vu} \) is equal to a weight associated to the edge \( uv \in E \) or zero if \( uv \notin E \). The neighborhood \( N(v) \) of \( v \) is the set of nodes adjacent to it (denoted \( N[v] \) if includes \( v \) itself), and the degree \( \deg(v) \) of \( v \) is defined as the number of its neighbors, i.e., \( \deg(v) = |N(v)| \). The average degree of a graph is denoted as \( \deg(G) \).

A path \( u \sim v \) is a sequence of distinct nodes starting in \( u \) and ending in \( v \) where every node is adjacent to the following one. The length of a path \( u \sim v \) is the number of its edges (hops), which we denote as \( \ell(u, v) \). The \( k \)-hop neighborhood \( N_k(v) \) of \( v \) (\( N_k[v] \) if inclusive) is the set of nodes that can be reached by a path in \( G \) of length at most \( k \).

The \( k \)-th power of a graph \( G^k \) is the graph where each node of \( G \) is connected to its \( k \)-hop neighbors. If there is no path between the two nodes, then \( \ell(u, v) = \infty \). To avoid confusion, any function may be denoted with a subscript to specify the graph on which is defined (e.g., \( \ell_G \)). An independent set is a set of nodes \( S \subseteq V \) such that no two of which are adjacent in \( G \). An independent set is maximum if \( G \) contains no larger independent set, while a maximal independent set (MIS) is an independent set that is not a subset of another one in \( G \) \cite{775}; observe that a non-maximal independent set always allows adding some node to it. Given a vector of positive weights \( x \in \mathbb{R}^n \) associated to every node in \( G \), a maximum-weight independent set (MWIS) is the independent set \( S \subseteq V \) that maximizes \( \sum_{v \in S} x_v \).

\[
\forall = [17]; \text{observe that a MWIS is always maximal, as adding nodes does adds weight. The sum of the weights in the MWIS is denoted as } a(G). \text{ A (maximal, maximum, or maximum-weight) } k \text{-independent set is a (maximal, maximum, maximum-weight) independent set of } G^k \text{.}
\]

3. GRAPH COARSENING WITH \textsc{k-MWIS}

When dealing with signals, images, or other kinds of Euclidean data, downsampling often amounts to keeping every \( k \)-th data point, where \( k \) is a given reduction factor. This means, for a generic discrete \( n \)-dimensional Euclidean datum, keeping a subset of its points such that every two of them are exactly \( k \) points far from each other on every of its dimensions. On graph-structured data, we lose this regularity along with the concept of dimensionality, and hence defining a new notion of downsampling that applies to graphs becomes non-trivial.

In Section 3.1, we define a graph coarsening method that, similarly to classical downsampling, reduces the size of a graph \( G \) by a given "factor", by finding a set of almost evenly spaced nodes within \( G \). These nodes will form the node set of the reduced graph, while its topology will be constructed starting from \( G \) in a way in which some of its key properties will be preserved, such as connectivity, or approximated, such as pairwise node distances.

Section 3.2 then elaborates on how the general formulation of our method allows, by a proper choice of the node ranking function, to approximate a \textsc{k-MWIS}, where this aspect becomes fundamental for its later use as a graph pooling mechanism.

3.1. Coarsening algorithm

Given a graph \( G = (V, E) \) and a distance \( k \), we want to obtain a coarsened representation of \( G \) by first selecting a set of nodes \( S \subseteq V \), that we refer to as centroids, such that every two centroids are more than \( k \) hops distant from each other, and such that no area of the graph remains unsampled; in other words, a maximal \( k \)-independent sets (\( k \)-MIS) of \( G \): this way, each centroid will be more than \( k \) hops from every other, while the maximality ensures every node of \( G \) is within \( k \) hops from a centroid.

Any MIS of a graph \( G \) is a \( k \)-MIS of \( G \), thus a \( k \)-MIS could be naively computed by known MIS algorithms, such as Luby \cite{48} or Blelloch et al. \cite{15}, on the \( k \)-th power of the adjacency matrix of \( G \). Using this approach will require \( O(n^2) \) space since the density of \( G^k \) increases rapidly with \( k \), becoming rapidly impractical for real world graphs with millions or billions of nodes. To overcome this problem, in Algorithm 1 we propose a variation of Blelloch’s algorithm \cite{15} that efficiently computes a \( k \)-MIS of \( G \) without explicitly computing its \( k \)-th power.

Once the \( k \)-MIS \( S \subseteq V \) is computed with Algorithm 1, we construct the coarsened graph \( H = (S, E') \) as follows:

(1) using Algorithm 2, we compute a partition \( P \) of \( V \) of size \(|S|\), such that

(a) every \( P \in P \) contains exactly one centroid and (a subset of) its \( k \)-hop neighbors, and
(b) for every node in \( P \) there is a centroid in \( P \) at distance at most \( k \)-hops;

(2) for every edge in \( E \) we add an edge in \( E' \) joining the two nearest centroids in the partitions containing the source and destination nodes. If this generates multiple edges, we coalesce them into...
Algorithm 1 Parallel Greedy k-MIS algorithm, adapted from Blelloch et al. [15]. Given a graph $G$, a subset of its nodes $V \subseteq V(G)$, and a node ranking $\pi$, returns a maximal $k$-independent set in $G$, with $k \in \mathbb{N}$.

1: function $k$-MIS($G, V, \pi$) 
2: if $|V| = 0$ then return $\emptyset$
3: $\pi_0 \leftarrow \pi$
4: for $i = 1, \ldots, k$ do
5:  for $v \in V$ do in parallel
6:    $S \leftarrow \{v \in V \mid \pi(v) = \pi_k(v)\}$
7:    $N_0 \leftarrow S$
8:  for $i = 1, \ldots, k$ do
9:    $N_i \leftarrow \bigcup_{v \in N_{i-1}} N[v]$
10:   $R \leftarrow V \setminus N_k$
11:  return $S \cup k$-MIS($G, R, \pi$)

Algorithm 2 Graph partitioning algorithm based on $k$-MIS. Given a graph $G$, $k \in \mathbb{N}$, and a node ranking $\pi$, returns a partition of $G$.

1: function Cluster($G = (V, E); k, \pi$) 
2: $S \leftarrow k$-MIS($G, V, \pi$)
3: $\pi_0 \leftarrow \pi$
4: for $v \in V \setminus S$ do in parallel
5:    $\pi_0(v) \leftarrow +\infty$
6: for $i = 1, \ldots, k$ do
7:    for $v \in V$ do in parallel
8:      $\pi_i(v) \leftarrow \min_{u \in N(v)} \pi_{i-1}(u)$
9: return $\{\{u \in V \mid \pi_k(u) = \pi(v)\}\}_{v \in S}$

(3) (optional) in case of weights/labels associated to the nodes, these can also be aggregated according to the partitioning $P$. Unless otherwise stated, for the rest of the paper we will apply no weight reduction to any node, and simply keep the weights associated to the nodes in $S$ (the centroids).

3.2 Node ordering and $k$-MWIs

A key property of Blelloch’s algorithm [15], that also applies to our $k$-MIS extension, is that it is deterministic: given a graph $G$ and a ranking of the nodes $\pi$ (which we always assume have no ties), Algorithm 1 will always produce the same $k$-MIS, for any $k \in \mathbb{N}$. This property has some interesting consequences:

- If the ranking $\pi$ can be uniquely determined by the nodes themselves (e.g., by some of their attributes or by their neighbors), Algorithms 1 and 2 become permutation invariant.
- Ranking can be used to lead Algorithm 1 to greedily include nodes having a higher rank under a given order of importance, such as a centrality measure or a task-dependent relevance.
- Algorithm 1 can also be used to approximate a maximum-weight $k$-independent set ($k$-MWIS) by setting an ordering that simulates any node-selecting rule proposed by Sakai et al. [62]. This aspect will be discussed in Section 4.2.

We now provide two examples on how we can change the ranking of the nodes to prioritize salient aspects according to a specific preference. Examples are conducted on the graph defined by the first sample of the MNIST dataset [44], a $28 \times 28$ monochromatic image (first row of Fig. 1) where every pixel is connected to the ones in the same pixel row, column or diagonal.

First, we simulate the typical downsampling on images (also known as average pooling [32, 34]), where squared partitions of $p \times p$ pixels are averaged together (first column of Fig. 1). To do this, we set the ranking $\pi$ of Algorithm 2 as the row-major ordering: given $(i, j)$ the coordinate of a pixel, we rank the nodes in decreasing order of $28i + j$. The resulting reduction is in the second column of Fig. 1: averaging intensities of pixels in the same partition produces a coarsened graph which is identical to classical downsampling. Note that this result is partly due to the fact that Algorithm 2 also makes use of $\pi$ to define the clustering, such that the nodes in a partition have always a lower rank w.r.t. the centroid in the same partition.

Secondly, we rank nodes in decreasing order of intensity, thus prioritizing the pixels belonging to the digit. Here we show two different results: the first, where we average the lightness and coordinates of the nodes in the same clusters (third column of Fig. 1), and a second one, where we just keep the ones belonging to the nodes in the $k$-MIS (fourth column). We see that the reduced graphs indeed...
prioritized the digit against other pixels, producing a coarsened representation where the digit is also remarkably recognizable.

It is important to note that the computation of the ranking can impact the complexity of the algorithm. In Section 4.2 we will provide two heuristic functions that generalize the ones proposed by Sakai et al. [62] to compute larger (weighted) independent sets, that can be computed in parallel and in limited space, while preserving a specific theoretical bound on the final size of the weights of the selected nodes.

4 THEORETICAL ANALYSIS AND RESULTS

4.1 Algorithm discussion and complexity

In order to avoid computing the $k$-th graph power of a possibly large-scale graph, Algorithm 1 modifies the one by Blelloch et al. [15, also reported in Appendix A] to compute the $k$-MIS without explicitly generating every $k$-hop neighborhood. Given a graph $G$, a subset of its nodes $V \subseteq V(G)$, and a (injective) node mapping $\pi : V \rightarrow [n]$ (that we can consider as a ranking of the nodes under a given permutation), Algorithm 1 works as follows:

1. If $V$ is not empty, in Lines 3 to 7 we find the set of nodes $S$ with minimum rank among their $k$-hop neighbors (i.e., their neighbors in $G^k$). This is done with $k$ steps of label propagation such that, at each step, every node takes the minimum label found within their inclusive ($1$-hop) neighbors. We only propagate labels belonging to nodes still in $V$.
2. In Lines 8 to 11 we remove from $V$ all the nodes that are at most $k$-hops from a node in $S$ (i.e., all their neighbors in $G^k$). This is also done with $k$ steps of label propagation starting from the nodes in $S$, where this time the propagated label is a flag signaling that the node shall be removed;
3. Finally, the algorithm makes a recursive call in Line 12 using only the remaining nodes. The resulting set is merged with $S$ and returned.

It is easy to see that, if $k = 1$, steps 1 to 3 become exactly Blelloch’s algorithm (i.e., Lines 2 to 5 of Algorithm 3), whereas by taking a general $k$ every step is extended to consider $k$-hop neighborhoods of $G$, thus efficiently emulating Blelloch’s algorithm on $G^k$.

As for complexity, Blelloch et al. [15] propose several trade-offs between work and depth on a concurrent-read/concurrent-write PRAM model (CRCW, with minimum priority concurrent write). Here, we consider one version (restituted in Appendix A as Algorithm 3) which allows an efficient parallel implementation with $O(m)$ work and $O(\log^3 n)$ depth w.h.p. [see 15, Lemma 4.2], and most closely resembles the structure of Algorithm 1. Our algorithm introduces a factor $k$ (compared to Algorithm 3) on the operations performed on lines Lines 3 to 7 and Lines 8 to 11 to compute the $k$-hop neighborhood. It follows that the work and depth of Algorithm 1 are bounded by $k$ times that of Blelloch’s algorithm, i.e., $O(k(n + m))$ work and $O(k \log^3 n)$ depth w.h.p., where an extra $O(n)$ work is needed to generate the additional vector of labels, which is modified every $k$ iterations. We remark that our algorithm can be modified to obtain the other trade-offs presented in Blelloch et al. [15], but we omit this for space reasons. Regarding Algorithm 2, after computing the $k$-MIS, the algorithm performs $k$ steps of label propagation, which add $O(k(n + m))$ work and $O(k \log n)$ depth to the total computation. Total space consumption is $O(n + m)$, comprising input and $O(1)$ label vectors of size $O(n)$.

**Proposition 4.1.** Given a graph $G$, an integer $k \in \mathbb{N}$, and a random ranking of the nodes $\pi$, both Algorithms 1 and 2 can be implemented to run on a CRCW PRAM using $O(k(n + m))$ work, $O(k \log^3 n)$ depth, and $O(n + m)$ space. The depth bound holds w.h.p.

4.2 Bounds on the $k$-MISs

In any greedy MIS algorithm, whenever we add a node to the independent set we have to remove all of its neighbors from the graph. Having observed this, a typical heuristic to compute larger-weight independent sets is to select nodes with high weight and low degree. Following this intuition, Sakai et al. [62] proposed the following rules: given $x \in \mathbb{R}^n_+$ a vector of positive weights associated to each node, add to the independent set the node maximizing either (i) $x_v/\text{deg}(v) + 1$, or (ii) $x_v/\sum_{u \in N[v]} x_u$. Both rules can be trivially extended to $k$-hop neighborhoods by computing $G^k$, which would however require $O(n^2)$ space, unless done sequentially. Computation of the neighborhood function $\text{deg}_k(v) = |N_k(v)|$ in limited space can be achieved only by resorting to approximations, e.g. using Monte Carlo methods [22] or approximate sets representations [56, 16], and still this would not extend to approximate rule (ii).

To overcome these limitations, we propose to overestimate the sums in the $k$-hop neighborhood of each node, by computing instead $c_k = (A + 1)^k x \in \mathbb{R}^n_+$, where $A, I \in \{0, 1\}^{|\mathbb{N}|^n}$ are, respectively, the adjacency and the identity matrices. The $k$-MIS is computed by Algorithm 1 with $c_k$ explicitly, as $c_k$ can be obtained with a sequence of $k$ matrix-vector products, that can be computed in $O(n + m)$ space, $O(k(n + m))$ work and $O(k \log n)$ depth.

In the following, we provide a generalization of the bounds of Sakai et al. [62] when a $k$-MIS is computed by Algorithm 1 with the ranking defined by rules (i)-(ii) approximated by the $k$-walk matrix $c_k$. We remark that, for $k = 1$, the following theorems are providing the same bounds as the one given by Sakai et al. [62]. The full proofs can be found in Appendix B.1.

**Theorem 4.2.** Let $G = (V, E)$ be a graph, with (unweighted) adjacency matrix $A \in \{0, 1\}^{|\mathbb{N}|^n}$ and with $x \in \mathbb{R}^n_+$ representing a vector of positive node weights. Let $k \in \mathbb{N}$ be an integer, then define $w : V \rightarrow \mathbb{R}_+$ as

$$w(v) = \frac{x_v}{(A + 1)^k x_v},$$

and $\pi_w$ as the ranking of the nodes in decreasing order of $w$. Then, $\text{k-MIS}(G, V, \pi_w)$ outputs a maximal $k$-independent set $S$ such that $\sum_{u \in S} x_u \geq \sum_{v \in V} w(v)$.

**Theorem 4.3.** Let $G = (V, E)$ be a graph, with (unweighted) adjacency matrix $A \in \{0, 1\}^{|\mathbb{N}|^n}$ and with $x \in \mathbb{R}^n_+$ representing a vector of positive node weights. Let $k \in \mathbb{N}$ be an integer, then define
For any \( w(\cdot) \in \mathbb{R}_+^n \) as
\[
w(v) = \frac{x_v}{(A + I)^k x_v},
\]
and \( \pi_w \) as the ranking of the nodes in decreasing order of \( w \). Then, \( k\text{-MIS}(G, V, \pi_w) \) outputs a maximal \( k \)-independent set \( S \) such that \( \sum_{v \in S} x_v \geq \sum_{v \in V} w(v) \cdot x_v \).

**Theorem 4.4.** Let \( G = (V, E) \) be a non-empty graph with positive node weights \( x \in \mathbb{R}_+^n \), and let \( \pi_w \) be a ranking defined as in Theorem 4.2 or 4.3 for any given \( k \in \mathbb{N} \). Then
\[
\sum_{v \in S} x_v \geq \frac{\alpha(G)}{\Delta_k},
\]
where \( S = k\text{-MIS}(G, V, \pi_w) \) and \( \Delta_k = \max_{r \in V} \left\{ \{(A + I)^k 1\}_r \right\} \).

Recalling that \( \alpha(G) \) is the optimal solution, Theorem 4.4 shows that our heuristics guarantee a \( \Delta_k \)-approximation.

### 4.3 Connectivity of the reduced graph

For the sake of conciseness, hereafter we denote with \((H, \rho) = \mathcal{R}(G, k)\) the function reducing a graph \( G \) by contracting the clusters obtained with Algorithm 2, as described in Section 3.1. The term \( H = (S, E') \) denotes the reduced graph, with \( S \) the \( k \)-MIS of \( G \), while \( \rho : V \to S \) is the function mapping every node to the (exactly one) centroid in its cluster. The following results are invariant w.r.t. the ranking parameter and the aggregation function used to reduce the edges or the nodes.

We follow a simple observation: for every edge in \((u, v) \in E'\) with \( u \neq v \), the nodes \( u \) and \( v \) are within \( 2k + 1 \) hops in \( G \), since two nodes in \( S \) are connected in \( H \) only if an edge in \( G \) crosses their two clusters. This property, combined with the lower bound implicitly defined by the \( k \)-MIS, yields the following bounds.

**Remark 1.** For any \( w(\cdot) \in \mathbb{R}_+^n \) such that \( u \neq v \), we have that \( k + 1 \leq \ell_G(u, v) \leq 2k + 1 \).

An example of this property is shown in Fig. 2, where bounds in Remark 1 apply for the Minnesota road network [23] reduced with different values of \( k \). From the above observation, we can obtain the two following properties.

**Proposition 4.5.** Let \( G \) be a connected graph and \((H, \rho) = \mathcal{R}(G; k)\), with \( k \in \mathbb{N} \). Then, \( \forall u, v \in V(G) \)
\[
\ell_H(\rho(u), \rho(v)) \leq \ell_G(u, v) \leq (2k + 1) \ell_H(\rho(u), \rho(v)) + 2k.
\]

**Corollary 4.6.** For any \( k \in \mathbb{N} \), \( G \) and \( H = \mathcal{R}(G; k) \) have the same number of connected components.

The full proofs are provided in Appendix B.2. Both Proposition 4.5 and Corollary 4.6 are fundamental in our proposal of using \( k \)-MIS reduction as a pooling method in Graph Neural Networks. In particular: (i) differently from several other pooling techniques [20, 33, 41, 45, 71, 58, 51], we can guarantee that the input graph is not divided in multiple components, and that, if applied repeatedly, our method will eventually produce a single representation node for the whole graph; (ii) when training with batches of graphs at a time, our method guarantees also that different graphs are not joined together.

### 5 RELATED WORKS

**Maximal \( k \)-Independent Sets.** Computing a maximum \( k \)-independent set (\( k \)-IS), or even its size, is known to be NP-hard [1], while computing a maximal \( k \)-IS can be trivially done in (super-linear) polynomial time and space using matrix powers [5] and any greedy MIS algorithm [e.g., 48, 15]. Koerts [42] proposed a formulation of the problem both as an integer linear program and as a semi-definite program, but still relying on the \( k \)-th power of the input graph. Several papers propose efficient algorithms to solve the maximum (weighted or unweighted) \( k \)-IS problem on specific classes of graphs \([4, 5, 28, 12, 27]\), which fall beyond the scope of this article. To the best of our knowledge the only other parallel algorithm for computing a maximal \( k \)-independent set was proposed by Bell et al. [11] as a generalization of the one of Luby [48] for \( k > 1 \). This algorithm is essentially the same as Algorithm 1, but without the ranking argument, making the algorithm non-deterministic, as the nodes are always extracted in a random order. Bounds on the size of the maximum \( k \)-independent sets are instead an active research topic in discrete mathematics and algebraic graph theory, with results both in terms of average distance between the nodes \([31, 54]\) and in terms of the graph spectrum \([3, 1, 2, 30, 66]\). As for \( k \)-MIS, most of the works on maximum-weight \( k \)-independent sets regard specific classes of graphs \([55, 37, 36, 61]\). Bounds on the classical MWIS problem have been widely studied in literature \([62, 38, 24, 35]\), and could also be applied to the \( k \)-MWIS problem if obtained via the \( k \)-th power of a graph.

**Graph Coarsening and Reduction.** The most relevant coarsening method is the one proposed by Bell et al. [11], which has been successfully used to accelerate algebraic multi-grid methods to solve sparse linear systems. Their method computes a \( k \)-MIS \( S \subseteq V \) and then collapses every node in the graph to the nearest node in \( S \). As already mentioned, the main difference between their methods w.r.t. the one proposed in this paper is that their algorithm is non-deterministic, as the nodes in the \( k \)-MIS are extracted at random. Classical MIS (i.e., with \( k = 1 \)) was proposed as a first sampling step in Barnard and Simon [10], although their final reduction step may not preserve the connectivity of the graph. Using MIS was also suggested by Shuman et al. [63] as an alternative sampling step for their graph reduction method. The spectral reduction proposed by Loukas [47, neighborhood variant] does not use sampling as a first reduction step, but sequentially contracts node neighborhoods until a halting condition is reached, performing similar steps to the classical greedy algorithm for maximum-weight independent sets.

**Graph Pooling.** Apart from a few cases \([49, 50, 65, 6, 8]\), pooling in Graph Neural Networks (GNNs) usually entails an adaptive approach, typically realized by means of another neural network. These pooling methods can be divided in two types: *dense* and *sparse*. Dense methods, such as DiffPool [69], MinCutPool [14, 13] and StructPool [70], compute for each node a soft-assignment to a fixed number of clusters defined by a reduction factor \( r \in (0, 1) \), thus generating a matrix requiring \( O(rn^2) \) space. Sparse methods, such as gPool/TopKPool [33, 20], SAGPool [45, 41], GSAPool [71], ASAPool [58], and PANPool [51], instead, compute a score for each node (requiring \( O(n) \) space), and reduce the graph by keeping only the top \([rn]\) scoring ones and dropping the rest. Although
Figure 2: Minnesota road network [23] reduced with different values of $k$. For $k = 0$, the two bounds coincide, as the graph is not reduced at all. For $k = 1$, the real distance covered by an edge is polarized (is either 2 or 3). For greater values of $k$, the edges’ real distance span over all the range $[k + 1, 2k + 1] \cap \mathbb{N}$.

scalable, these methods provide no theoretical guarantees regarding the preservation of connectivity of the reduced graph, as the $n - \lceil r n \rceil$ dropped nodes may disconnect the graph. Mesquita et al. [52] questioned the effectiveness of pooling in GNNs, by empirically showing that dense methods produce clusterings no more effective that random assignments, while a spectral clustering method (GrAClus [25]) showed no significant changes in performance if computed on the graph complement. To gauge the effectiveness of our proposed ranking functions we thus added a baseline with randomized ranking; indeed this proved powerful on the smallest dataset, but was not competitive with our method in general.

6 EXPERIMENTAL ANALYSIS

Our empirical analysis centers on demonstrating both the scalability of our $k$-MWIS coarsening to large graphs, and its effectiveness as a graph pooling mechanism. In the following experiments we considered real world networks of four different types: social networks, traceroute graphs, co-authorship networks, and road maps. Full details are in Appendix C.1, while in Table 1 we briefly report their sizes (both in terms of number of nodes $n$ and edges $m$), average degree (deg), and effective diameter ($\ell_{90}$), i.e., the 90th percentile of the distances between all pairs of nodes. In Section 6.1 we will show the performance of the coarsening algorithm in itself, showing its running time, the reduction factors achieved by varying $k$, and the weight of $k$-MISs computed as compared to the classical (sequential) greedy algorithm. In Section 6.2 we will then provide an experimental comparison against top-$k$-based pooling methods on selected graph classification datasets, whose information is synthesized in Table 1, showing the effectiveness of our approach.

Our method has been implemented using PyTorch [57] and PyTorch Geometric [29], to allow high-level scripting of CUDA code. Every label propagation required by the algorithms has been implemented in form of message-passing (i.e., gather-scatter) to run exclusively on GPU. All the experiments have been executed on a machine running Ubuntu Linux with an AMD EPYC 7742 64-Core processor with 1TB of RAM, and a NVIDIA A100 with 40GB of on-board memory.

6.1 Running time and $k$-MWIS approximations

Figure 3 reports the average reduction ratio (top row) and average running time (bottom row, in user seconds) of ten runs of our method on selected benchmark graphs, using different values of $k$ and a ranking induced by Eq. (2) with constant weights. Times refer to the ones needed respectively to compute the ranking (as described in Section 4.2), the $k$-MIS (Algorithm 1), and then reducing the graph (Algorithm 2 and edge reduction, as described in Section 3.1). We can clearly observe the linearity of the time complexity of our algorithm, since the execution time increases linearly with $k$. We can also see that, for the graphs with small diameter (see Table 1), the running time decreases once $k$ approaches their effective diameter, since most of the nodes will be assigned to the same centroid during the first recursive call of Algorithm 1, thus decreasing the expected depth of the algorithm and also its execution time.

Table 2 reports the average total weight obtained by computing the greedy sequential MWIS algorithm on $G^k$ (Greedy), compared to Algorithm 1 on $G$ (Ours). For both algorithms we used Eqs. (1) and (2), fixing $k = 1$ for the greedy one (thus applying the original rules of Sakai et al. [62] on $G^k$). Results are averaged on ten runs with node weights extracted uniformly at random in the interval $[1, 100]$. The rationale behind these results is two-fold. First, differently from Algorithm 1, in sequential greedy algorithms (GW [see, e.g., 62, 38]) those nodes maximizing Eq. (1) or (2) are iteratively added to the independent set and their neighbors are removed from

Table 1: Benchmark graphs and dataset information

| Graph        | Type         | Ref.  | $n$   | $m$   | deg   | $\ell_{90}$ |
|--------------|--------------|-------|-------|-------|--------|-------------|
| Orkut        | Social       | [46]  | 3072441 | 117185083 | 76.3 | 4.8 |
| LiveJournal  | Social       | [46]  | 3997962 | 69362378  | 34.7 | 6.5 |
| Youtube      | Social       | [46]  | 1134890 | 5975248   | 10.5 | 6.5 |
| Brightkite   | Social       | [46]  | 58228  | 214078   | 7.4  | 6.0 |
| Skitter      | Web          | [46]  | 1696415 | 22190596  | 26.2 | 6.0 |
| Enron        | Email        | [46]  | 36692  | 367662   | 20.0 | 4.8 |
| AstroPh      | Auth.        | [46]  | 18772  | 396160   | 42.2 | 4.8 |
| DBLP         | Auth.        | [9]   | 540486 | 30491458 | 112.8 | 4.8 |
| Europe       | Road         | [9]   | 50912018 | 108109320 | 4.2  | $>10^3$ |
| Luxembourg   | Road         | [9]   | 114599 | 239332   | 4.2  | $>10^3$ |

| Dataset      | Type         | Ref. | $n$ (avg.) | $m$ (avg.) | Size | Class |
|--------------|--------------|------|------------|------------|------|-------|
| DD           | Protein      | [26] | 284.32     | 715.66     | 1178 | 2     |
| REDDIT-B     | Social       | [68] | 429.63     | 497.75     | 2000 | 2     |
| REDDIT-5K    | Social       | [68] | 508.52     | 594.87     | 4999 | 5     |
| REDDIT-12K   | Social       | [68] | 391.41     | 456.89     | 11929| 11    |
| GITHUB       | Social       | [59] | 113.79     | 234.64     | 12725| 2     |
Table 2: Weight comparison of \( k \)-MISs obtained with our relaxation and the classical greedy algorithm.

| Graph   | \( k \) | Ranking with Eq. (1) | Ranking with Eq. (2) |
|---------|--------|----------------------|----------------------|
|         |        | Greedy   | Ours    | Greedy   | Ours    |
| AstroPh | 1      | 392347.4 | 392355.1 | 392333.8 | 392528.4 |
|         | 2      | 149701.3 | 148257.2 | 149625.2 | 147953.5 |
|         | 3      | 72923.3  | 71929.1  | 72857.1  | 71847.4  |
| Enron   | 1      | 1220157.4 | 120194.0 | 1219789.6 | 1219789.3 |
|         | 2      | 148203.3  | 147460.3 | 148175.6 | 147392.6 |
| Brightkite | 1      | 1931765.8 | 1931884.3 | 1932375.7 | 1932378.2 |
|         | 2      | 824662.6  | 821815.2 | 824654.3 | 820564.1 |
|         | 3      | 456965.4  | 457761.3 | 456838.0 | 457368.3 |
| Luxembourg | 1      | 3286607.7 | 3286587.3 | 3309810.9 | 3309836.2 |
|         | 2      | 2267010.7  | 2265353.6 | 2290198.3 | 2284708.0 |
|         | 3      | 1721185.9  | 1717789.7 | 1740735.1 | 1725635.3 |
|         | 4      | 1379199.0  | 137352.9  | 1393494.6 | 1371500.4 |
|         | 5      | 1147810.9  | 1141633.0 | 1159119.1 | 1133255.1 |
|         | 6      | 978606.1   | 970727.6  | 988133.8 | 959172.7 |
|         | 7      | 851061.6   | 840502.3  | 858769.6 | 830110.7 |
|         | 8      | 751680.4   | 739632.1  | 757523.8 | 728799.9 |

Figure 3: Reduction ratio (top) and running time (bottom, in log scale) of our approach for varying \( k \) (also in log scale).

6.2 Graph pooling with \( k \)-MWIS

Table 3 summarizes the average classification accuracy obtained on selected classification benchmarks from the TUDataset collection [53] using the same underlying Graph Neural Networks (GNNs) and different kinds of pooling mechanisms. For classification tasks, we chose those datasets having the highest number of nodes, where pooling layers may prove more useful. All datasets were divided in training (70%), validation (10%), and test (20%) sets using a random stratified split with fixed seed (42). All models have the same general architecture: 3 GNNs (optionally) interleaved by 2 layers of pooling, a global pooling method (sum and max), and a final MLP with dropout [64] as classifier. All models were trained using Adam optimizer [39]. We performed a model selection using the training and validation split, and then we computed the average test set classification accuracy obtained by the best configuration, on 10 inference runs using different seed values. A detailed description of the models and the experimental setting are provided in Appendix C.2. The hyper-parameter concerning the reduction factor (\( k \) in our case, or \( r \) for other methods) has been chosen among the other parameters during the model selection phase.

We compared our reduction method against different kinds of pooling readily available on the PyG library (we avoided the dense methods as DiffPool [69] and MinCutPool [14], as they do not scale well on the selected datasets). Apart from the baseline (with no pooling) and Graclus [25] (which performs spectral clustering), all the remaining compared methods perform a selection of the top scoring nodes according to an importance value computed adaptively, that is, by means of (another) parametric model. For our method, we computed the node scores in four different ways: at random (rand); using a constant value (const); by computing the norm of the feature vectors associated to the nodes (norm); and adaptively, by means of a linear layer having as input the features of the nodes, followed by a sigmoid function, similarly to TorKPool [33] (linear). In the latter case, as in the other parametric methods, the final score is then multiplied by the feature vector of the selected nodes, to make the scoring function end-to-end learnable. In all cases, computed scores constitute node weights and, as described in Section 4.2, the resulting ranking function is obtained according to Eq. (2).

Looking at Table 3 (where the top 2 accuracy scores for each dataset are in boldface) it is immediately evident how—regardless of the weighting strategy—the proposed \( k \)-MWIS-based approaches obtain consistently high accuracy, suggesting that the evenly-spaced centroid selection is indeed able to capture essential properties of each graph. This is particularly noteworthy as even the simpler...
We introduced a new general graph coarsening approach that leverages nodes according to a (learned) notion of relevance, seem non-parametric pooling variants (k-MWIS \( \text{rand, const, and norm} \)) outperform the parametric competitors in several instances.

On the other hand, the more refined k-MWIS (linear), which also weights nodes according to a (learned) notion of relevance, seem to overall perform better than the other k-MWIS-based strategies, as well as outperforming the other parametric pooling methods in all experiments. This suggests that exploiting the ranking function of Algorithm 1 to select relevant nodes is indeed able to improve the representativeness of the downsampled graph.

We observe a remarkable performance of the baseline algorithm (no pooling) on the REDDIT-12K and GITHUB datasets: we may speculate that the graphs are simple enough to not require pooling, yet at the same time k-MWIS (linear) obtains competitive accuracy, suggesting it is a reliable and versatile choice.

Finally, we also observe how the standard deviation in the experiments of k-MWIS-based approaches, and in particular of k-MWIS (linear) appears consistently and significantly lower than all other considered approaches.

### 7 CONCLUSIONS

We introduced a new general graph coarsening approach that leverages on k-MWISs to preserve fundamental topological properties of the original graph as well as distances, which are guaranteed to maintain a bounded stretch. The coarsening reduction can be regulated by the parameter \( k \), going from the original graph, when \( k = 0 \), to up to a single node as \( k \) approaches the graph’s effective diameter. We showed empirically that our approach is general enough to provide efficient approximations of k-MWIS, yet it also offers theoretical guarantees on the quality of such an approximation.

The algorithm is designed to provide distance distortion guarantees while at the same time allowing a scalable parallel implementation, which processes graphs with up to 100 million edges in just a few seconds on a single GPU.

The empirical analysis provided evidence of effectiveness of our k-MWIS pooling in several graph classification benchmarks, showing superior performance with respect to related parametric and non-parametric methods from the literature.

Overall, given the generality and scalability of our approach, it has great potential of positively impacting a plethora of computationally-intensive applications for large scale networks, such as graph visualization, 3D mesh simplification, and classification. In the future we plan to explore specializations of our algorithm to more of these applications, as well as assessing its scalability in multi-GPU scenarios.

### REFERENCES

[1] A. Abiad, G. Coutinho, and M. A. Fiol. 2019. On the k-independence number of graphs. Discrete Mathematics. Algebraic and Extremal Graph Theory 342, 10 (2019), 2875–2885.

[2] A. Abiad, G. Coutinho, M. A. Fiol, B. D. Nogueira, and S. Zeijlemaker. 2022. Optimization of eigenvalue bounds for the independence and chromatic number of graph powers. Discrete Mathematics. 345, 3, (2022), 112706.

[3] A. Abiad, S. M. Cioabă, and M. Tait. 2016. Spectral bounds for the k-independence number of a graph. Linear Algebra and its Applications. 510, (2016), 160–170.

[4] G. Agnarsson, R. Greenlaw, and M. Halldórsson. 2000. On Powers of Chordal Graphs And Their Colorings. In Powers of Chordal Graphs and Their Colorings, Congressus Numerantium, 144, 41–65.

[5] G. Agnarsson, P. Damaschke, and M. Halldórsson. 2003. Powers of geometric intersection graphs and dispersion algorithms. Discrete Applied Mathematics. Stability in Graphs and Related Topics 132, 1, (2003), 3–16.

[6] D. Bacciu and L. Di Sotto. 2019. A Non-negative Factorization Approach to Node Pooling in Graph Convolutional Neural Networks. In A*IA 2019 – Advances in Artificial Intelligence (Lecture Notes in Computer Science). Cham, 294–306.

[7] D. Bacciu, E. Etrica, A. Micheli, and M. Poddà. 2020. A gentle introduction to deep learning for graphs. Neural Networks. 129, (2020), 203–221.

[8] D. Bacciu, A. Conte, R. Grossi, F. Landolfi, and A. Marino. 2021. K-plex cover pooling for graph neural networks. Data Mining and Knowledge Discovery. 35, 5, (2021), 1–21.

[9] D. A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner. 2011. 10th DIMACS Implementation Challenge-Graph Partitioning and Graph Clustering.

[10] S. T. Barnard and H. D. Simon. 1994. Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. Concurrency: Practice and Experience. 6, 2, 101–117.

[11] N. Bell, S. Dalton, and L. N. Olson. 2012. Exposing Fine-Grained Parallelism in Algebraic Multigrid Methods. SIAM Journal on Scientific Computing. 34, 4, (2012), C123–C152.

[12] B. K. Bhattacharya and M. E. Houle. 1999. Generalized Maximum Independent Sets for Trees in Subquadratic Time. In Algorithms and Computation. (1999), 435–445.

[13] F. M. Bianchi, D. Grattarola, and C. Alippi. 2020. Spectral Clustering with Graph Neural Networks for Graph Pooling. Proceedings of the International Conference on Machine Learning, 1.

[14] G. E. Blelloch, J. T. Fineman, and J. Shun. 2012. Greedy sequential maximal independent set and matching are parallel on average. In Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures (SPAA ’12). New York, NY, USA. (2012), 308–317.

[15] P. Boldi, M. Rosa, and S. Vigna. 2011. HyperANF: approximating the neighborhood function of very large graphs on a budget. In Proceedings of the 20th international conference on World wide web (WWW ’11). New York, NY, USA, (2011), 625–634.

[16] A. Bondy and M. R. Mury. 2008. Graph Theory. Graduate Texts in Mathematics. Springer-Verlag, London.
Algorithm 3 Parallel Greedy MIS algorithm, from Blelloch et al. [15]. Given a graph $G$ and a node ranking $\pi$, returns a maximal independent set in $G$.

1: function MIS($G = (V, E)$, $\pi$)
2: if $|V| = 0$ then return $\emptyset$
3: $S \leftarrow \{v \in V \mid \forall u \in N(v), \pi(v) < \pi(u)\}$
4: $R \leftarrow V \setminus \bigcup_{v \in S} N[v]$
5: return $S \cup$ MIS($G[R]$, $\pi$)

Remark 2. Let $S = k$-MIS($G, V, \pi$). Then, $\mathcal{P} = \{\hat{N}_k[v]\}_{v \in S}$ forms a partition of $V$.

Remark 3. Let $G = (V, E)$ a graph, $A \in \{0, 1\}^{n \times n}$ its (unweighted) adjacency matrix, and $x \in \mathbb{R}^n$ a vector of non-negative values. Then, for any $v \in V$ and $k \in \mathbb{N}$,

$$\sum_{u \in \hat{N}_k[v]} x_u \leq [(A + I)^k]_v x_v.$$

Proof of Theorem 4.2.

$$\sum_{v \in S} x_v = \sum_{v \in S} w(v) \cdot [(A + I)^k]_v$$

$$\geq \sum_{v \in S} w(v) \cdot |N_k[v]|$$

$$\geq \sum_{v \in S} w(v) \cdot |\hat{N}_k[v]|$$

$$\geq \sum_{v \in S} \sum_{u \in \hat{N}_k[v]} w(u)$$

$$\geq \sum_{v \in V} w(v) \cdot x_v.$$  \hspace{1cm} (Remark 2)

Proof of Theorem 4.3.

$$\sum_{v \in S} x_v = \sum_{v \in S} w(v) \cdot [(A + I)^k]_v$$

$$\geq \sum_{v \in S} w(v) \cdot \sum_{u \in N_k[v]} x_u$$

$$\geq \sum_{v \in S} w(v) \cdot \sum_{u \in \hat{N}_k[v]} x_u$$

$$\geq \sum_{v \in S} \sum_{u \in \hat{N}_k[v]} w(u) \cdot x_u$$

$$\geq \sum_{v \in V} w(v) \cdot x_v.$$  \hspace{1cm} (Remark 2)

Remark 4. Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix and $x \in \mathbb{R}^n$ be a vector. Then

$$\sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} \cdot x_j = \sum_{i=1}^{n} x_i \cdot \sum_{j=1}^{n} A_{ij}.$$

Proposition B.2 (Kako et al. [38]). Assume that $a_i > 0$ and $b_i > 0$ for $1 \leq i \leq n$. Then

$$\sum_{i=1}^{n} \frac{b_i^2}{a_i} \geq \left( \sum_{i=1}^{n} b_i \right)^2 \frac{1}{\sum_{i=1}^{n} a_i}.$$

Proof of Theorem 4.4. Let $\Delta_k = \max_{v \in V} [(A + I)^k]_v$. When using a ranking induced by Eq. (1) we have that

$$\sum_{v \in S} x_v \geq \sum_{v \in V} w(v) \cdot [((A + I)^k]_v \geq \frac{\Delta_k \sum_{v \in V} x_v}{\alpha(G)}.$$  \hspace{1cm} (Theorem 4.2)

$$\geq \frac{\sum_{v \in V} x_v}{\Delta_k}.$$  \hspace{1cm} (Remark 4)

Remark 4. When using the rank induced by Eq. (2), instead, we have

$$\sum_{v \in S} x_v \geq \sum_{v \in V} \frac{w(v)}{[(A + I)^k]_v} \geq \frac{\Delta_k \sum_{v \in V} x_v}{\alpha(G)}.$$  \hspace{1cm} (Remark 4)

Proof of Proposition 4.5. The first inequality holds by the same arguments used for the fist inequality of Lemma B.3. The
second inequality follows from Lemma B.3 and triangle inequality. Namely,
\[
\ell_G(u, v) \leq \ell_G(u, \rho(u)) + \ell_G(\rho(u), \rho(v)) + \ell_G(\rho(v), v).
\]
Finally, \(\forall x \in V, \ell_G(\rho(x), x) \leq k\) by construction of \(H\).

Proof of Corollary 4.6. If \(G\) is connected, then also \(H\) is connected, by Lemma B.3. Otherwise, let \(C \subseteq V(G)\) the nodes of a connected component of \(G\). Since \(V(H)\) is a \(k\)-MIS of \(G\), \(C \cap V(H)\) is a \(k\)-MIS of \(G[C]\). Applying Algorithm 2 to \(G[C]\) (with the same ordering used on \(G\)) will produce the reduced graph \(H' = H[C \cap V(H)]\), which by Lemma B.3 is connected. Finally, Algorithm 2 joins two nodes in \(V(H)\) only if there exists and edge in \(G\) intersecting their \(k\)-hop neighborhood, hence Algorithm 2 does not connect different components of \(G\).

c
C EXPERIMENTAL SETTING

C.1 Description of the Datasets

We run benchmark experiments on real-world undirected graphs from different domains, namely

- Orkut, LiveJournal, Youtube, and Brightkite, are social networks from the SNAP dataset [46], where every node represents a user and every edge a friendship relation. Like most social networks, those networks have an small effective diameter, that in this case amounts at most to 6.5.
- Skitter, which is an Internet topology graph built from traceroutes, where every autonomous system (AS) is represented as a node which is connected by an edge to other ASs if there has been reported an exchange of information between two of them. This graph was also retrieved from the SNAP dataset [46]. Its diameter is 6.
- Enron is an email communication network from SNAP [46], where two nodes are email addresses and there is an edge between two of them if they exchanged at least an email (sent or received). It has an effective diameter of 4.8.
- DBLP and AstroPh are two co-authorship networks, respectively from the 10th DIMACS Challenge [9] and from the SNAP [46] dataset. In DBLP, every node in the networks represents a paper and there is an edge if two papers share at least an author, while in AstroPh every node represents an author and there is an edge between two of them if they co-authored a paper. Their effective diameters are 6.8, and 4.8, respectively.
- Europe and Luxembourg are two road networks from the 10th DIMACS Challenge dataset [9]. In these networks every edge represents a road (of some kind) and a node a crossing. These are the only weighted graphs, where every weight represents the Euclidean distance between the coordinates of the two endpoints. Differently from the previous networks, street maps are (almost) planar graphs and, as such, can be expected to have a high diameter.

All the graphs were retrieved from the University of Florida Sparse Matrix Collection [23].

For the classification tasks, we used the following benchmark datasets:

- DD [26], a dataset of graphs representing protein structures, in which the nodes are (labeled) amino acids and two nodes are connected by an edge if they are less than 6 Angstroms apart. The task consists in classifying enzymes and non-enzymes.
- REDDIT (-BINARY, -MULTI-5K, and -MULTI-5K) [68], are social networks where there is an edge between two users if there was reported an interaction between them (in the form of comments in a discussion thread). The task consists in classifying different kind of communities (respectively, 2, 5, and 11 of them).
- GITHUB-STARGAZERS [59], a social network of developers, where every edge is a “following” relation. The task is to decide if a community belongs to web or machine-learning developers.