MOX-Report No. 79/2022

Agglomeration of Polygonal Grids using Graph Neural Networks with applications to Multigrid solvers

Antonietti, P. F.; Farenga, N.; Manuzzi, E.; Martinelli, G.; Saverio, L.

MOX, Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

mox-dmat@polimi.it https://mox.polimi.it

Agglomeration of Polygonal Grids using Graph Neural Networks with applications to Multigrid solvers

P. F. Antonietti[a,1], N. Farenga[a], E. Manuzzi[a,1,*], G. Martinelli[a], L. Saverio[a]

[a] *MOX, Department of Mathematics, Politecnico di Milano, p.zza Leonardo da Vinci, 32, I-20133 Milano, Italy*

Abstract

Agglomeration-based strategies are important both within adaptive refinement algorithms and to construct scalable multilevel algebraic solvers. In order to automatically perform agglomeration of polygonal grids, we propose the use of Graph Neural Networks (GNNs) to partition the connectivity graph of a computational mesh. GNNs have the advantage to process naturally and simultaneously both the graph structure of mesh and the geometrical information, such as the areas of the elements or their barycentric coordinates. This is not the case with other approaches such as METIS, a standard algorithm for graph partitioning which is meant to process only the graph information, or the k-means clustering algorithm, which can process only the geometrical information. Performance in terms of quality metrics is enhanced for Machine Learning (ML) strategies, with GNNs featuring a lower computational cost online. Such models also show a good degree of generalization when applied to more complex geometries, such as brain MRI scans, and the capability of preserving the quality of the grid. The effectiveness of these strategies is demonstrated also

*Abbreviations: Machine Learning (ML), Graph Neural Networks (GNNs), Polygonal Discontinuous Galerkin (PolyDG), Finite Element Methods (FEMs), MultiGrid (MG).

*Corresponding author

Email addresses: paola.antonietti@polimi.it (P. F. Antonietti), nicola.farenga@mail.polimi.it (N. Farenga), enrico.manuzzi@mail.polimi.it (E. Manuzzi), gabriele2.martinelli@mail.polimi.it (G. Martinelli), luca.saverio@mail.polimi.it (L. Saverio)

[1] P. F. Antonietti and E. Manuzzi are members of INDAM-GNCS. P.F. Antonietti has been partially supported by the Ministero dell'Università e della Ricerca [PRIN grant numbers 201744KLJL and 20204LN5N5].

when applied to MultiGrid (MG) solvers in a Polygonal Discontinuous Galerkin (PolyDG) framework.

Keywords: Agglomeration, Polygonal Grids, Graph Neural Networks, K-means, Multigrid solvers, Polygonal Discontinuous Galerkin.

1. Introduction

Many applications in the fields of Engineering and Applied Sciences, such as fluid-structure interaction problems, flow in fractured porous media, crack and wave propagation problems, are characterized by a strong complexity of the physical domain, possibly involving moving geometries, heterogeneous media, immersed interfaces and complex topographies. Whenever classical Finite Element Methods (FEMs) are employed to discretize the underlying differential model, the process of grid generation can be the bottleneck of the whole simulation, as computational meshes can be composed only of tetrahedral, hexahedral, or prismatic elements. To overcome this limitation, in the last years there has been a great interest in developing FEMs that can employ general polygons and polyhedra as grid elements for the numerical discretizations of partial differential equations. We mention the mimetic finite difference method [1, 2, 3, 4], the hybridizable discontinuous Galerkin method [5, 6, 7, 8], the Polyhedral Discontinuous Galerkin (PolyDG) method [9, 10, 11, 12, 13, 14, 15], the Virtual Element Method (VEM) [16, 17, 18, 19, 20, 21] and the Hybrid High-Order method [22, 23, 24, 25, 26]. This calls for the need to develop effective algorithms to handle polygonal and polyhedral (polytopal, for short) grids and to assess their quality (see e.g. [27]). For a comprehensive overview we refer to the monographs and special issues [4, 14, 25, 28, 20, 21] and the references therein. Among the open problems, there is the issue of efficiently handling polytopal mesh agglomeration, i.e., merging mesh elements to obtain coarser grids [29, 30, 10, 31, 32]. Mesh agglomeration can be used to obtain a coarser discretization of the differential problem at hand, in order to reduce the number of degrees of freedom where not needed and therefore also the computational

effort. This operations can be naturally performed in the context of polygonal and polyhedral grids, because of the flexibility in the definition of the shape of mesh elements. This approach has multiple applications in the numerical solution of partial differential equations, for example:

- it can be used, with adaptive procedures, to reduce the number of degrees of freedom where not needed because in certain parts of the domain the error is already under control;
- it can be used to generate a hierarchy of (nested) coarser grids starting from a fine mesh of a complex physical domain of interest, in order to employ them in multigrid solvers [10, 33, 34, 35, 36, 37, 38] to accelerate the converge of iterative algebraic;
- it can be employed together with domain decomposition techniques [39, 40, 41, 42] to obtain a meaningful decomposition of the domain, starting from a fine discretization.

Grid agglomeration is a topic quite unexplored, because it is not possible to develop such kind of strategies within the framework of classical FEMs. During this operation it is important to preserve the quality of the underlying mesh, since this might affect the overall performance of the method in terms of stability and accuracy. Indeed a suitable adapted mesh may allow to achieve the same accuracy with a much smaller number of degrees of freedom when solving the numerical problem, hence saving memory and computational power. However, since in such a general framework mesh elements may have any shape, there are no well established strategies to achieve effective agglomeration with an automatic, fast and simple approach.

In recent years there has been a great development of Machine Learning (ML) algorithms, a framework which allows to extract information automatically from data, to enhance and accelerate numerical methods for scientific computing [43, 44, 45, 46, 47, 48, 49, 50, 51, 52]. In this work, we propose to use ML-

based strategies to efficiently handle polygonal mesh agglomeration, in order to fully exploit all of the benefits of the above mentioned numerical methods, such as geometrical flexibility and convergence properties. The core concept lies in learning the "shape" of mesh elements in order to perform the desired operations accordingly. Such a learning needs to be performed in an automatic and flexible way, because of the too high variability of geometries of interest, tailoring the approach to a wide range of different possible situations. Rather than simply trying to decide a priori criteria to perform agglomeration, which would inevitably result in poor performance or high computational cost due to the impossibility of capturing all of the possible situations, ML strategies exploit and process automatically the huge amount of available data to learn only the distribution of the features of interest for the application, leading to high performance and computational efficiency. By combining the a priori approach of classical numerical methods, with the a posteriori approach of ML-strategies, it is possible to not only to boost existing algorithms but also to develop new algorithms capable to work in more general frameworks. In particular, we propose the use of Graph Neural Networks (GNNs) [53, 54, 55, 56, 57], that are deep learning architectures specifically meant to work with graph-structured data. The problem of mesh agglomeration can be re-framed as a graph partitioning problem, by exploiting the connectivity structure of the mesh. In particular, the graph representation of the mesh is obtained by assigning a node to each element of the mesh, and connecting with an edge the pair of nodes which are relative to adjacent elements in the original mesh. By exploiting such a representation, GNNs can be applied to solve a node classification problem, where each element is assigned to a cluster, which corresponds an element of the agglomerated mesh. GNNs can process naturally and simultaneously both the graph structure of mesh and the geometrical information that can be attached to the nodes, such the elements areas or their barycentric coordinates. This is not the case of other approaches such as METIS [58], a standard solver for graph partitioning which can process only the graph information, or k-means, which can process only the geometrical information. The proposed GNN-based

algorithms exploit an unsupervised training procedure, where parameters are set to minimize the expected value of the normalized cut, over a database of polygonal grids. To investigate the capabilities of the proposed approaches, we consider a second-order model problem discretized by the PolyDG method in a multigrid framework. We measure effectiveness through an analysis of quality metrics and number of iterations of the the algebraic iterative solver. We also consider the generalization capabilities over a human brain MRI scan section.

The paper is organized as follows. In Section 2 we present possible agglomeration criteria for polytopes. In Section 3 we propose a general framework to perform mesh agglomeration using GNNs. In Section 4 we measure the effectiveness of the proposed agglomeration strategies in terms of computational cost, quality metrics and generalization capabilities over unseen complex physical domains. In Section 5 we present some computations obtained by applying the considered agglomeration strategies to multigrid solvers in a PolyDG framework. In Section 6 we draw some conclusions.

2. Mesh agglomeration strategies

We recall that the problem of mesh agglomeration can be re-framed as a graph partitioning problem, by exploiting the connectivity structure of the mesh. In particular, the graph representation of the mesh is obtained by assigning a node to each element of the mesh, and connecting with an edge the pair of nodes which are relative to adjacent elements in the original mesh, i.e. polygons that share at least one edge. Moreover, features can be assigned to each node, storing geometrical information such as the area of the element or its barycentric coordinates. Finally, partitioning the nodes of the mesh into proper clusters using a suitable algorithm allows to obtain an agglomerated representation of the original mesh.

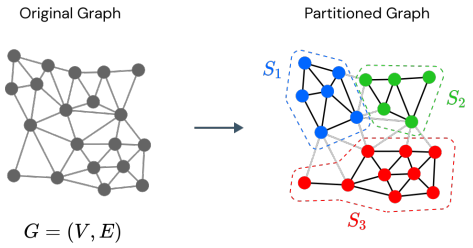

Figure 1: Example of graph partitioning into three sets. Left: original graph. Right: partitioned graph into three subsets S_1, S_2, S_3.

2.1. Graph Partitioning

Let $N > 0$ be the number of graph nodes. Given a graph $G = (V, E)$, where $V = \{v_i\}_{i=1}^{N}$ and $E = \{e(v_i, v_j) : v_i, v_j \in V\}$ are the sets of nodes and the set of edges, respectively. The problem of graph partitioning consists in finding M disjoint sets of nodes $S_1, ..., S_M$ such that $\cup_{i=1}^{M} S_i = V$ and $\cap_{i=1}^{M} S_i = \emptyset$, see, e.g., Figure 1. We restrict our framework to undirected graphs, i.e. $e(v_i, v_j) \in E \iff e(v_j, v_i) \in E$. We will use the following notation:

- $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix, i.e. $A_{i,j} = 1$ if $e(v_i, v_j) \in E$ and 0 otherwise;
- $X \in \mathbb{R}^{N \times F}$ is the features matrix of the nodes, which may represent any information such as coordinates, where F is the number of features;
- $\mathcal{N}(v_i) = \{v_j \in V : e(v_i, v_j) \in E\}$ denotes the neighborhood of the i-th node, containing the nodes v_j directly adjacent to v_i.

We can define the *cut* of a graph, representing the number of edges connecting the disjoint sets of nodes resulting from the paritioned graph. In the case of two partitions, it can be defined as:

$$\text{cut}(S_1, S_2) = |\{e(v_i, v_j) \in E : v_i \in S_1, v_j \in S_2\}|, \tag{1}$$

and can be easily generalized to the case of M partitions, as

$$\text{cut}(S_1, ..., S_M) = \frac{1}{2} \sum_{k=1}^{M} \text{cut}(S_k, S_k^C) \qquad \text{with} \quad S_k^C = V \setminus S_k. \tag{2}$$

The *normalized cut* is defined as

$$\mathrm{Ncut}(S_1, ..., S_M) = \sum_{k=1}^{M} \frac{\mathrm{cut}(S_k, S_k^C)}{\mathrm{vol}(S_k, V)}, \tag{3}$$

where the *volume* of the partition S_k is defined as

$$\mathrm{vol}(S_k, V) = |\{e(v_i, v_j) \in E : v_i \in S_k, v_j \in V\}|. \tag{4}$$

It represents the total degree of all nodes of the k-th partition. Depending on the application of interest, different notions of *volume* can be used. The *normalized cut*, contrary to the standard *cut*, allows to take into account partitions where the number of nodes is balanced between the different sets. Let Y_{ij} be the probability for node i of belonging to partition j. The *expected cut*, given two partitions S_k and its complement S_k^C, is defined as

$$\mathbb{E}[\mathrm{cut}(S_k, S_k^C)] = \sum_{i=1}^{N} \sum_{v_j \in \mathcal{N}(v_i)} Y_{ik}(1 - Y_{jk}) = \sum_{i=1}^{N} \sum_{j=1}^{N} Y_{ik}(1 - Y_{kj^T}) A_{ij}. \tag{5}$$

Let D be the column vector of the degrees of the nodes, i.e. $D_i = \mathrm{vol}(\{v_i\}, V)$. Then

$$\mathbb{E}[\mathrm{vol}(S_k, V)] = (Y^T D)_k = \Gamma_k. \tag{6}$$

The *expected normalized cut* can be defined as

$$\mathbb{E}[\mathrm{Ncut}(S_1, ..., S_M)] = \sum (Y \oslash \Gamma)(1 - Y)^T \odot A, \tag{7}$$

where $\oslash$ and $\odot$ denote the element-wise division and multiplication, respectively, and the summation is over all the entries of the resulting matrix.

2.2. METIS

A standard algorithm for partitioning large graphs or meshes and computing fill-reducing orderings of sparse matrices is METIS [58]. As depicted in Figure 2, METIS graph partitioning algorithm is based on a multi-level graph bisection procedure, consisting of the following main steps:

1. *Graph coarsening phase*: from the input graph successively smaller graphs are derived by collapsing adjacent pairs of vertices, until the size of the graph is sufficiently small.

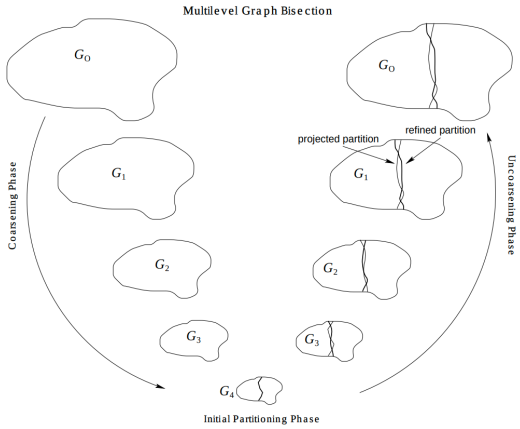

Figure 2: Scheme of the METIS graph bisection algorithm, taken from [58].

2. *Initial partitioning phase*: a partition of the coarsest graph is computed my minimizing the edge cut.
3. *Uncoarsening phase*: the partitioning of the smallest graph is projected back to the successively larger graphs, by assigning the pairs of vertices that were collapsed together to the same partition.
4. *Refinement phase*: after each uncoarsening step the partition is refined, adjusting nodes close to the interface of the two sets.

Once the graph has been divided into two sets, METIS applies the same algorithm recursively on the new sub-graphs, until the desired number of sets is reached.

2.3. Machine learning-based graph partitioning

We propose to employ ML-based bisection models of the form $\mathcal{M}(G, X) = Y$, that take as input a graph G together with a set of features X attached to each node, such as the barycentric coordinates or the area of the mesh elements, and output the vector of probabilities Y of each node belonging to cluster 1 or cluster 2. In order to apply such models for mesh agglomeration, we can use Algorithm 1, that recursively bisect the connectivity graph of the input mesh until the agglomerated elements have the desired size. We recall that the

diameter of a domain $\mathcal{D}$ is defined, as usual, as $\text{diam}(\mathcal{D}) := \sup\{|x-y|,\ x,y \in \mathcal{D}\}$. Given a polygonal mesh, i.e. a set of non-overlapping polygonal regions $\mathcal{T}_h = \{P_i\}_{i=1}^{N_P}$, $N_P \geq 1$, that covers a domain Ω, we can define the mesh size $h = \max_{i=1:N_P} \text{diam}(P_i)$. Algorithm 1 automatically generates a hierarchy of nested grids with different sizes, to be employed e.g. within multigrid solvers. If the model $\mathcal{M}$ does not return a valid partition Y, e.g., because within a set

Algorithm 1 General mesh agglomeration strategy

Input: mesh $\mathcal{T}_h$, target mesh size h^*, bisection model $\mathcal{M}$.

Output: agglomerated mesh $\mathcal{T}_{h^*}$.

Function AGGLOMERATE ($\mathcal{T}_h,\ h^*$)

if $\text{diam}(\mathcal{T}_h) \leq h^*$ **then**
  return $\mathcal{T}_h$
else
  Extract the connectivity graph G and features X from $\mathcal{T}_h$
  $Y \leftarrow \mathcal{M}(G, X)$
  Refine partition Y, by minimizing the length of the boundary of the agglomerated polygons.
  Partition $\mathcal{T}_h$ into sub-meshes $\mathcal{T}_h^{(1)}, \mathcal{T}_h^{(2)}$ according to Y.
  $\mathcal{T}_{h^*}^{(1)} \leftarrow$ AGGLOMERATE$(\mathcal{T}_h^{(1)}, h^*)$
  $\mathcal{T}_{h^*}^{(2)} \leftarrow$ AGGLOMERATE$(\mathcal{T}_h^{(2)}, h^*)$
  $\mathcal{T}_{h^*} \leftarrow$ merge $\mathcal{T}_{h^*}^{(1)}$, $\mathcal{T}_{h^*}^{(2)}$
end if

the graph is not connected, a suitable fixing procedure is required. This can be done, for example, by considering the largest suitable connected components as new clusters. Possible choices for the bisection model $\mathcal{M}$ are, but not limited to, the k-means clustering algorithm and GNNs, as we will see in the following.

3. The k-means clustering algorithm

The k-means clustering algorithm [59, 60, 61], is an iterative procedure for partitioning a set of input points in $\mathbb{R}^n$, where $n \geq 1$ is the number of features, into k parts, as described in Algorithm 2. It is an unsupervised learning algorithm, as no labels are provided together with the data. It relies on the definition of k centroids and each point is assigned to the closest centroid. The location of the centroids is chosen in such a way to minimize the euclidean distance between centroids and points within the clusters. In order to apply

Algorithm 2 k-means clustering

Input: set of points $\{x_i\}_{i=1}^N \subset \mathbb{R}^n$, number of clusters k.

Output: set of labels for each point $\{\ell_i\}_{i=1}^N$.

1: Randomly sample k points and label them centroids $c_1, c_2, ..., c_k$.
2: Compute labels $\{\ell_i\}_{i=1}^N$ by assigning each point to the cluster with the closest centroid in L^2 norm.
3: Compute the average of points in each cluster to obtain new centroids $c_1, c_2, ..., c_k$.
4: Repeat steps 3 and 4 until cluster assignments do not change, or the maximum number of iterations is reached.

the k-means algorithm for graph bisection within Algorithm 1, we simply use Algorithm 2 employing only two clusters and using as features the barycentric coordinates of the mesh elements. This strategy tends to produces "rounded" (star-shaped) agglomerated elements of similar size. Notice that no information on the connectivity graph is taken into account directly. However, for a regular mesh with elements of similar size, barycentric coordinates are usually strongly correlated with elements being adjacent.

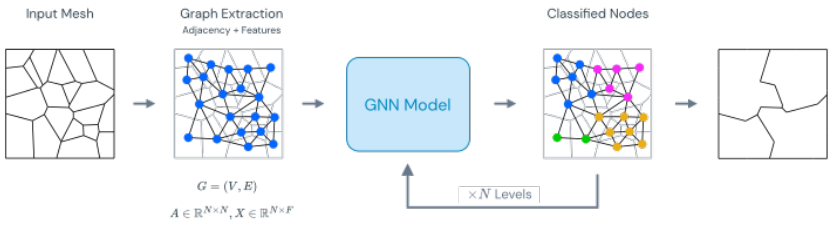

Figure 3: General framework for mesh agglomeration via GNNs.

4. Graph neural networks-based agglomeration strategies

In order to perform grid agglomeration by exploiting effectively both the graph representation and the geometrical features of meshes, we employ GNNs as bisection model in Algorithm 1. Examples of GNNs-based models directly applied to the original grid are [54, 55], which employ additional processing based on the graph spectrum to perform a preliminary embedding step, in order to extract features that can later be fed to a GNN-based partitioning module. In our case, the graph extraction of geometrical features such the elements area or their barycentric coordinates can be leveraged to perform a classification task, therefore avoiding the need for an additional spectral embedding module. The graph-bisection model performs a classification task, by taking as input the graph $G = (V, E)$ and the features related to its nodes $X \in \mathbb{R}^{N \times F}$, and outputting a probability tensor $Y \in \mathbb{R}^{N \times 2}$, where Y_{ij} represents the probability that the node $v_i \in V$ belongs to the partition S_j, with $j = 1, 2$. This approach can be obviously generalized to an arbitrary number of partitions. However, this would require a specific GNN model for each fixed number of classes, while the 2-classes case can be easily extended to a multi-class case by recursively calling the bisection model on the graphs of each partition. The general framework for mesh agglomeration via GNNs is shown in Figure 3.

4.1. Unsupervised learning for graph partitioning

In a unsupervised learning framework, we are given an unlabelled dataset, in our case of the form $\{(A^i, X^i)\}_{i=1}^{N_G}$ where $N_G \geq 1$ is the number of graphs

in the database, for which we want to find a different representation. For the case of mesh agglomeration, these graphs represent the elements connectivity of different computational grids. We consider then a node classifier F, which in our case will be a GNN, parameterized by a set of weights W, that takes as input the adjacency matrix A and the nodes features X of a graph and outputs the probability Y_{ij} for node i of belonging to partition j. Our goal is to tune W so that F minimizes the following *loss* function

$$\mathcal{L} = \sum_{i=1}^{N_G} \ell(A^i, Y^i; W), \tag{8}$$

where $Y^i = F(A^i, X^i; W)$ and

$$\ell(A, Y; W) = \sum_{k=1}^{M} \sum_{i,j=1}^{N} \frac{Y_{ik}(1 - Y_{kj}^T)A_{ij}}{\Gamma_k} \tag{9}$$

is the expected normalized cut of the graph.

4.2. Graph neural networks

GNNs are deep learning architectures specifically meant to work with graph-structured data within the framework of Geometrical Deep Learning, that concerns the application of neural networks to non-Euclidean data structures. Mapping, or layers, that can be combined to construct the GNN architecture are the following.

Graph convolutional layers. These layers take as input a graph $G = (V, E)$, consisting of an adjacency matrix A together with features attached to nodes, edges or the global graph, and returns a graph with the same connectivity structure while progressively transforming the information of the features, as shown in Figure 4. These mappings are permutation-invariant with respect to the order of the nodes. Let X denote the nodes features matrix, or the initial representations, related of the input graph G, and let H^k denote the hidden representation of those features after applying the k-th convolutional layer. At

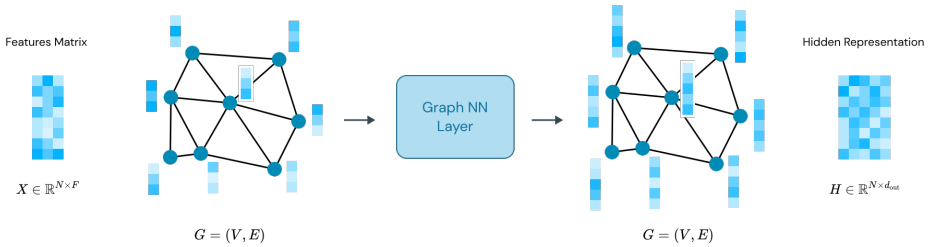

Figure 4: General GNN framework, consisting of an input graph G together with its features matrix X, and an output hidden representation H related to the same graph structure.

step k, the hidden representation H_i^k for the node v_i is computed as:

$$a_i^k = \Phi^k(\{H_j^{k-1} : v_j \in \mathcal{N}(v_i)\}), \tag{10}$$

$$H_i^k = \Psi^k(H_i^{k-1}, a_i^k), \tag{11}$$

where $H^0 = X$ for the initial layer, Φ is the *aggregation function* that defines how the information coming from the neighborhood $\mathcal{N}(v_i)$ of node v_i is aggregated, and Ψ is the *combination function* that defines how the aggregated information a_i^k is combined with the one stored in v_i. Different definitions of aggregation and combination functions leads to different GNN architectures. In particular, we re-frame equations (10) and (11) by considering the mean aggregation function, as follows:

$$H_i^{l+1} = \sigma(H_i^l W_1^l + (\text{mean}_{j \in \mathcal{N}(v_i)} H_j^l) W_2^l), \tag{12}$$

where $\sigma(\cdot)$ is a non-linear activation function, such as the REctified Linear Unit (ReLU) or the hyperbolic tangent (tanh), and $W_1^k, W_2^k \in \mathbb{R}^{F_k \times F_{k+1}}$ are weight matrices associated to the l-th layer, representing a trainable linear transformation, where F_k and F_{k+1} are the features dimensions for the current and next layers, respectively. We refer to (12) as SAmpling-and-aggreGatE Convolutional (SAGEConv) layer [53] followed by activation function σ. Such layers also account for the possibility of sub-sampling the neighbourhood of a node during the aggregation process, leveraging the information coming from features to better generalize to unseen nodes, while keeping the computational cost under control.

Input normalization layer. We consider a mapping of the form INORM : $\mathbb{R}^{N\times F} \to \mathbb{R}^{N\times F}$, where N is the number of nodes and F is the number of features, that normalizes the input feature matrix $X = [x_1|\dots|x_F]$. The normalization is performed differently for each type of feature, i.e. column-wise: for strictly positive features, such as the mesh elements areas, we simply rescale to $[0, 1]$

$$\tilde{x}_i = \frac{x_i}{\max(x_i)}, \quad i = 1, ..., F,$$

while for other features, such as barycentric coordinates, we first center them, by subtracting the mean, and then rescale to $[-1, 1]$

$$\tilde{x}_i = \frac{y_i}{\max(|y_i|)}, \quad y_i = x_i - \text{mean}(x_i), \quad i = 1, ..., F.$$

Dense layers. Dense layers are generic linear maps of the form LINEAR : $\mathbb{R}^m \to \mathbb{R}^\ell$, $m, \ell \geq 1$ defined by parameters to be tuned. They are used to separate graph features extracted in the previous layers.

Softmax. We define the function SOFTMAX : $\mathbb{R}^\ell \to (0, 1)^\ell$, where $\ell \geq 2$ is the number output classes,

$$[\text{SOFTMAX}(x)]_i = \frac{e^{x_i}}{\sum_{j=1}^{\ell} e^{x_j}}.$$

They are used to assign a probability to each class.

4.3. Graph neural network training

The input features matrix is $X \in \mathbb{R}^{N\times 3}$, where the first columns contains the area of each mesh element followed by the two coordinates of its barycenter. The GNN architecture we employed first applies a INORM layer, then four SAGECONV layer with features dimensions 64, then three LINEAR layer with progressively decreasing output dimensions (32, 8 and 2) and finally a SOFTMAX layer. Each SAGECONV layer is followed by a tanh activation function to keep the features inside the interval $[-1, 1]$, so that the geometrical information

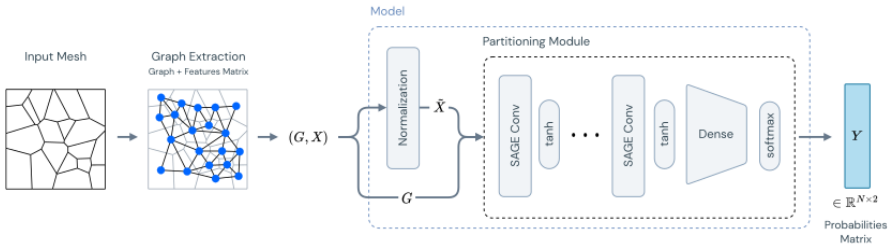

Figure 5: GNN model structure, consisting of a normalization module, responsible of rescaling the features extracted from the input mesh, and of a partitioning module made of a stack of graph convolutions and a dense classifier. The output is a matrix of probabilities $Y \in \mathbb{R}^{N\times 2}$, containing the probabilities of belonging either to one of the two classes for each node.

regarding the re-scaled coordinates will be kept in the same domain, as information flows through the layers. To further simplify the classification process, the INORM layer also rotates of 90 degrees the barycentric coordinates if the input mesh is more stretched along the y-axis rather than the x-axis. The resulting model consists of approximately 28k parameters, where roughly 25k resulting from the SAGECONV layers and the remaining from the LINEAR layers. A scheme of the described GNN model is shown in Figure 5.

The training of the models has been performed by employing an unsupervised approach by minimizing the expected normalized cut, as described in Section 2. To perform the training a train and a validation datasets were generated. The training dataset consists of meshes of the following type: grids of regular squares, grids of regular triangles, grids of triangles with random perturbation of the vertices, grids of Voronoi with random location of the seeds. The database is composed of 800 meshes, with 200 meshes per type, while the validation dataset consists of 200 meshes, 50 per type. The cardinality of the datasets has been chosen to keep a 80-20 split ratio between train and validation respectively. Training has been performed using the Adam optimizer [62] with a learning rate 1e-5, L^2 regularization coefficient 1e-5 and mini-batch size 4. Training was performed for 300 epochs in approximately 35 minutes on Google

Colab cloud platform, using a 2.20GHz Intel Xeon processor, with 12GB of RAM memory and NVIDIA Tesla T4 GPU with 16GB of GDDR6 memory.

5. Validation on a set of polyhedral grids

In this section we compare the performance of the proposed algorithms. To evaluate the quality of the refined grids, we employ the following quality metrics introduced in [27]:

- *Uniformity Factor* (UF): ratio between the diameter of an element P and the mesh size
$$\mathrm{UF}(P) = \frac{\mathrm{diam}(P)}{h}.$$
This metric takes values in $[0, 1]$. The higher its value is the more mesh elements have comparable sizes.

- *Circle Ratio* (CR): ratio between the radius of the inscribed circle and the radius of the circumscribed circle of a polyhedron P
$$\mathrm{CR}(P) = \frac{\max_{\{B(r)\subset P\}} r}{\min_{\{P\subset B(r)\}} r},$$
where $B(r)$ is a circle of radius r. For the practical purpose of measuring the roundness of an element the radius of the circumscribed circle has been approximated with $\mathrm{diam}(P)/2$. This metric takes values in $[0, 1]$. The higher its value is the more rounded mesh elements are.

We consider four different grids of domain $(0, 1)^2$: a grid of regular triangles, a grid of triangles with random location of the vertices, a Voronoi grid with random location of the seeds, and a grid of regular squares. In Figure 6 these grids have been agglomerated using METIS, k-means and GNNs strategies. For METIS the target number of mesh elements is $N_0/16$, where N_0 is the initial number of elements, while for k-means and GNN the target mesh size in Algorithm 1 is $h_0/4$, where h_0 is the initial mesh size. As we can see, the k-means and the GNN algorithms are capable to recover a regular grid of squares when starting from regular meshes (triangles and squares), while this is not the case

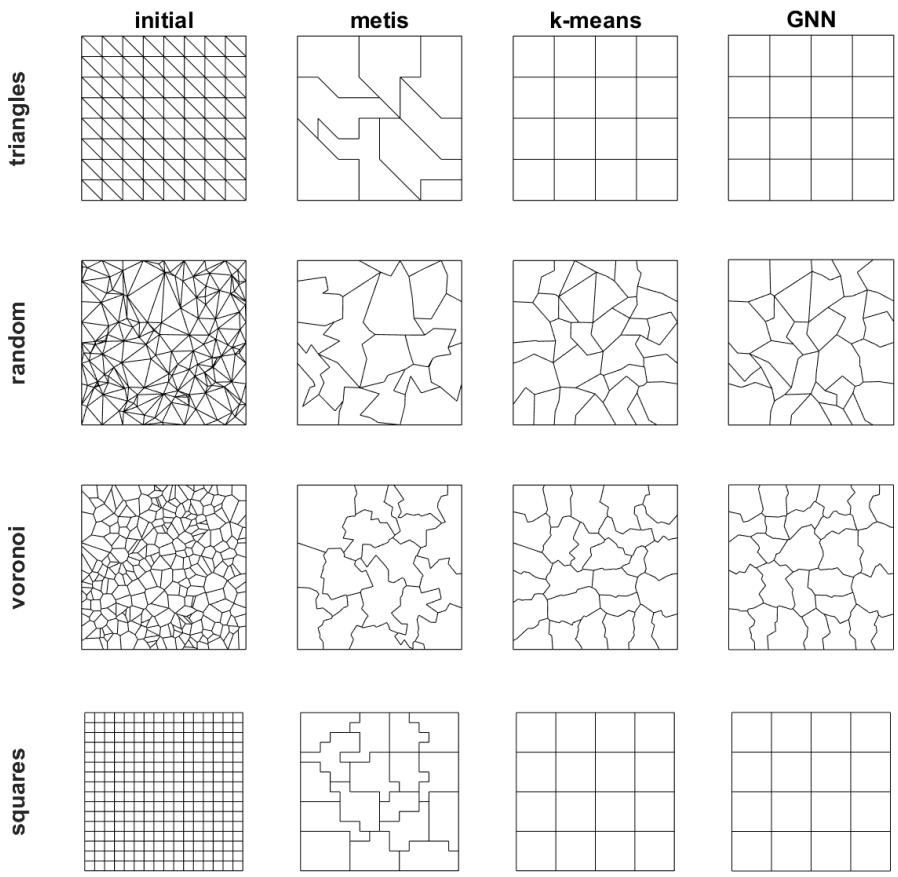

Figure 6: Initial grids (first column) and corresponding agglomerated versions (second to fourth column) obtained based on employing different strategies. Each row corresponds to the same initial grid (from top to bottom: regular triangles, random triangles, Voronoi, squares) while each column corresponds to the same agglomeration strategy (from left to right: initial grids, METIS, k-means, GNN).

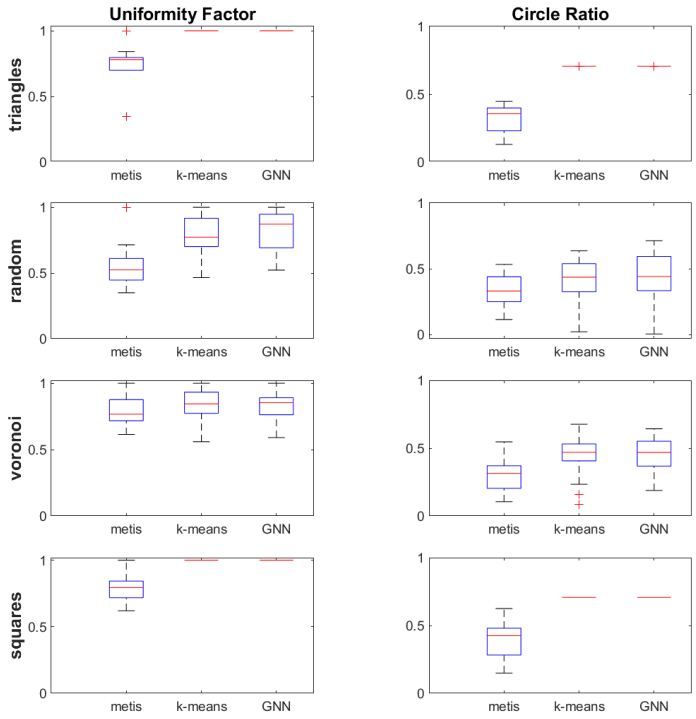

Figure 7: Box plots of the computed quality metrics (UF and CR) for the agglomerated grids reported in Figure 6 (second to fourth column), obtained based on employing different agglomeration strategies (METIS, k-means, GNN). Some box plots collapse into a single line because the corresponding metric has the same value for all mesh elements.

for METIS. In Figure 7 we show the box plots of the computed quality metrics for the selected grids. In general, quality metrics are lower for the METIS algorithm, while k-means and GNNs have comparable performance. This is further confirmed by Tables 1 and 2, where we report the average values of UF and CR. In particular, the performance difference is much more evident for regular grids. In Tables 3 and 4 we also report the relative performance with respect to METIS, i.e., the ratio between the average UF and CR metrics of the ML-strategies (k-means and GNN) and METIS. This further highlights the gain is using ML-based strategies. In general, ML-based strategies (either the

UF	metis	k-means	GNN
triangles	0.7648	1.0000	1.0000
random	0.6115	0.7003	0.7418
Voronoi	0.7605	0.8105	0.8225
squares	0.7725	1.0000	1.0000

Table 1: Average values of the UF for the agglomerated grids reported in Figure 6, obtained based on employing different agglomeration strategies (METIS, k-means, GNN).

CR	metis	k-means	GNN
triangles	0.3447	0.7071	0.7071
random	0.3399	0.4190	0.3928
Voronoi	0.3056	0.4509	0.4845
squares	0.3733	0.7071	0.7071

Table 2: Average values of the CR for the agglomerated grids reported in Figure 6, obtained based on employing different agglomeration strategies (METIS, k-means, GNN).

UF relative	k-means/metis	GNN/metis
triangles	1.3076	1.3076
random	1.1452	1.2130
Voronoi	1.0658	1.0815
squares	1.2945	1.2945

Table 3: Relative UF: performance improvement with respect to METIS, i.e., ratio between the average UF of the ML-strategies (k-means and GNN) and METIS, for the agglomerated grids reported in Figure 6.

CR relative	k-means/metis	GNN/metis
triangles	2.0512	2.0512
random	1.2327	1.1554
Voronoi	1.4755	1.5853
squares	1.8940	1.8940

Table 4: Relative CR: performance improvement with respect to METIS, i.e., ratio between the average CR of the ML-strategies (k-means and GNN) and METIS, for the agglomerated grids reported in Figure 6.

k-means and the GNN algorithms), seem to preserve the initial geometry and quality of the grids, because the geometric information attached to the nodes is taken into account, making them suitable for adaptive mesh coarsening. This is not the case for the METIS algorithm, because it processes only the information coming from the graph topology of the mesh.

5.1. Application to a computational mesh stemming from a human brain MRI-scan

In order to further test the generalization capabilities of our models, we apply them on a much more complex domain with respect to the meshes considered so far. In particular, we consider the mesh of a section of a human brain coming from an MRI-scan, consisting of 14372 triangular elements. The domain is highly non-convex and presents many constrictions and narrowed sections. We agglomerated such a grid using METIS, k-means and GNN algorithms: the result is reported in Figure 8. For k-means and GNN the target mesh size in Algorithm 1 is $0.2D$, where D is the maximum distance between any two vertices of the initial mesh. For METIS the target number of mesh elements is 50, which corresponds approximately to the same mesh size. In Figure 9 we show also the corresponding box plots of the quality metrics and in Table 5 we report the average values. As we can see, performance are comparable in terms of UF, while the ML-based strategies achieve on average a higher "roundness", measured in terms of CR. This indicates good generalization capabilities of

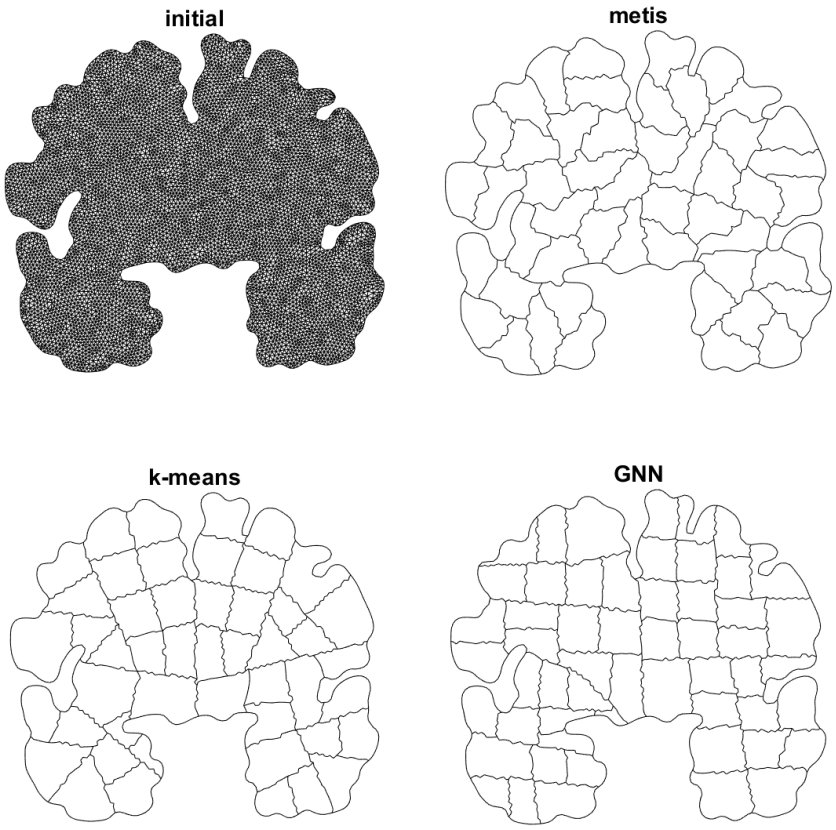

Figure 8: Agglomerated meshes using different strategies (METIS, k-means, GNN), starting from an initial grid of a human brain MRI-scan, consisting of 14372 triangular elements as shown in the top-left figure.

metric	metis	k-means	GNN
UF	0.7808	0.7875	0.7672
CR	0.4134	0.5146	0.4841
UF relative	1	1.0085	0.9826
CR relative	1	1.2449	1.1712

Table 5: Average and relative values (with respect to METIS) of the UF and the CR for different agglomeration methods (METIS, k-means, GNN), applied to the section of the MRI brain scan, for the agglomerated MRI brain scan meshes reported in Figure 8.

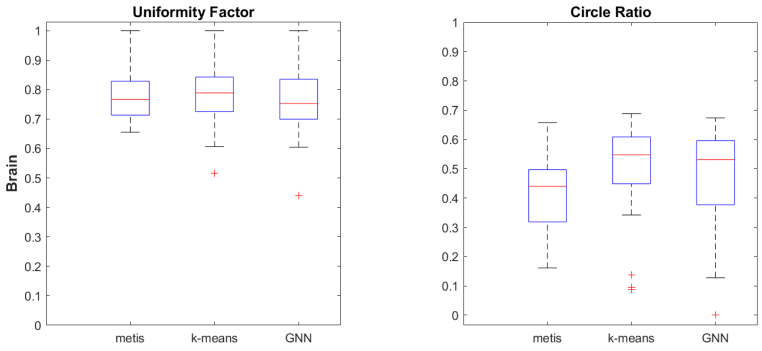

Figure 9: Box plots of the computed quality metrics (UF and CR) for the agglomerated MRI brain scan meshes reported in Figure 8, obtained based on employing different agglomeration strategies (METIS, k-means, GNN).

the GNN algorithm, as the considered mesh was very different from the ones included in the training set, both in terms of shape, dimensions and number of elements. The k-means reasonably performs slightly better than the GNN, because it does not require prior information coming from a database at the cost of having a higher online computational cost, as we will see in the next section.

5.2. Runtime performance

The main advantage in using Neural Network-based solutions is the low computational cost for online inference, with respect to k-means or METIS. We measured the computational cost of the different graph partitioning algorithms on 21 random Voronoi meshes with an increasing number of elements from 25 to 5000. Since the runtimes are not deterministic, we performed a sampling of 20 runtimes for each mesh. Results are shown in Figure 10. As we can see, the GNN algorithm outperforms the METIS and the k-means ones. For a mesh with 5000 elements we have that GNN is 4.4836 times faster than Metis and 3.5670 times faster than k-means. Such a performance improvement is expected to grow asymptotically, due to the fact that the computational complexity of the Neural Network mainly scales with the dimension of the data manifold [63],

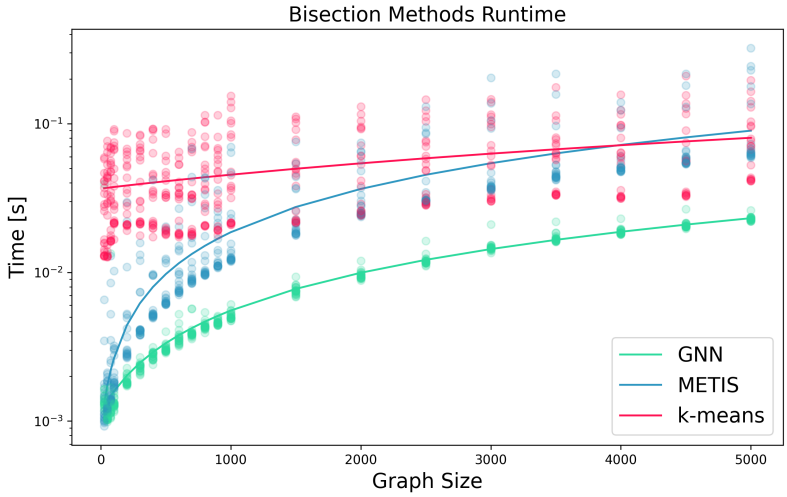

Figure 10: Runtime performance for different graph bisection models (METIS, k-means, GNN) as a function of the number of nodes in the connectivity graph of Voronoi meshes. The y-axis is in logarithmic scale.

while METIS and k-means mainly scale with the dimension of the graph. Being able to compute the solution in fast manner, even with a slight loss of accuracy, can be particularly beneficial when using multigrid scheme as preconditioners.

6. Applications to agglomeration-based multigrid methods

In this section we test the effectiveness of the proposed agglomeration strategies, to be used in combination with polygonal finite element discretizations within a MultiGrid (MG) framework [10, 33, 34, 35, 36, 37, 38]. We consider the following model problem: find $u \in V = \mathrm{H}^2(\Omega) \cap \mathrm{H}_0^1(\Omega)$ such that

$$\int_\Omega \nabla u \cdot \nabla v = \int_\Omega f v \quad \forall v \in V, \tag{13}$$

with Ω and $f \in L^2(\Omega)$ the forcing term, selected in such a way that the exact solution is given by $u(x, y) = \sin(\pi x)\sin(\pi y)$. We consider the V-cycle MG algorithm with additive Schwarz smoothing within a PolyDG discretization framework, as described in [38]. We employ the initial grids shown in the first column

of Figure 6 (triangles, random, Voronoi, squares), agglomerated using the proposed strategies (METIS, k-means, GNNs). In particular, for each initial mesh we construct three agglomerated grids of increasing size: for METIS the target numbers of mesh elements are $N_0/64$, $N_0/16$, $N_0/4$, where N_0 is the initial number of elements, while for k-means and GNN the target mesh sizes in Algorithm 1 are $h_0/8$, $h_0/4$, $h_0/2$, where h_0 is the initial mesh size. These correspond to the different levels of the V-cycle algorithm, where level 1 corresponds to the initial grid (level 3 grids are the ones reported in Figure 6). As performance metric, we consider the Iteration count of the MG algorithm to reduce the (relative) residual below 10^{-6} in solving the algebraic formulation of problem (13). We vary the following parameters: number of levels employed ℓ, polynomial degree p, number of smoothing steps m. In Table 6 we report the iteration count when varying the number of levels $\ell = 2, 3, 4$ with $m = 3$ and $p = 1$.

We can observe the following:

- The iteration count of the MG methods are significantly lower than the ones the of the classical Conjugate Gradient (CG) method, meaning the considered MG implementation is indeed effective.
- The iteration count of the k-means and GNN algorithms are lower than the ones of METIS, meaning the higher grid quality provided by the ML-based agglomeration strategies can help accelerating the convergence of the numerical method, with GNN having the best performance.
- The iteration count required when varying the number of levels ℓ is constant, meaning it is independent of the granularity of the underlying grid and therefore scalable in terms of mesh size.

In Table 7 we also consider the case for $m = 1$. Despite using such a low value, the MG method still converges in all cases. As expected, the iteration count increases but the iteration count of the ML strategies is still lower with respect to the ones of the CG, while this is not the case for the METIS algorithm.
In Table 8 we report the iteration count when varying the number of smoothing

grids	ℓ	Agglomeration-based MG			CG
		metis	k-means	GNN	
triangles	2	21	9	9	114
	3	21	9	9	
	4	21	9	9	
random	2	46	41	32	655
	3	46	41	32	
	4	46	41	32	
Voronoi	2	19	16	15	348
	3	19	16	15	
	4	19	16	15	
squares	2	20	17	17	109
	3	20	17	17	
	4	20	17	17	

Table 6: Iteration count of the MG algorithm to reduce the (relative) residual below 10^{-6} employing different initial grids (triangles, random, Voronoi, squares) agglomerated with different strategies (METIS, k-means, GNN) with $\ell = 2, 3, 4$, $p = 1$, $m = 3$. As a comparison, the iteration count of the Conjugate Gradient (CG) method are reported in the last column.

grids	ℓ	Agglomeration-based MG			CG
		metis	k-means	GNN	
triangles	3	230	67	67	114
	4	227	66	66	
random	3	706	551	405	655
	4	708	577	407	
Voronoi	3	154	143	129	348
	4	154	143	131	
squares	3	178	162	162	109
	4	180	162	162	

Table 7: Iteration count of the MG algorithm to reduce the (relative) residual below 10^{-6} employing different initial grids (triangles, random, Voronoi, squares) agglomerated with different strategies (METIS, k-means, GNN) with $\ell = 3, 4$, $p = 1$, $m = 1$. As a comparison, the iteration count of the Conjugate Gradient (CG) method are reported in the last column.

steps $m = 3, 5$ with $\ell = 3$ and $p = 1$. As we can see, when the number of steps increases the iteration count decreases, as well as the performance difference between the different methods. In Table 9 we report the iteration count when varying the polynomial dgree $p = 1, 2, 3$ with $\ell = 3$ and $m = 3$. As expected, since m is fixed, when p increases also the iteration count increases, because more degrees of freedom are involved in the formulation of the problem [38]. In general, the considered experiments highlights that when employing ML-based agglomeration strategies a lower iteration count is required by the MG solver with respect to METIS, thanks to the higher quality of the produced grids, with GNN having the best performance.

7. Conclusions

We presented a new ML-based framework to efficiently handle the open problem of mesh agglomeration for polygonal grids. It is based on the concept of learning the geometrical information contained in the shape of mesh elements,

grids	m	Agglomeration-based MG			CG
		metis	k-means	GNN	
triangles	3	21	9	9	114
	5	10	5	5	
random	3	46	41	32	655
	5	20	18	15	
Voronoi	3	19	16	15	348
	5	10	8	8	
squares	3	20	17	17	109
	5	9	8	8	

Table 8: Iteration count of the MG algorithm to reduce the (relative) residual below 10^{-6} employing different initial grids (triangles, random, Voronoi, squares) agglomerated with different strategies (METIS, k-means, GNN) with $\ell = 3$, $p = 1$, $m = 3, 5$. As a comparison, the iteration count of the Conjugate Gradient (CG) method are reported in the last column.

in an automatic and cost effective manner in order to tailor the approach a wide range of possible situations, which would not be possible using classical strategies. The novelty of the proposed method consists in improving classical methods for mesh agglomeration using GNN architectures and the k-means clustering algorithm. Advantages include computational efficiency, handling a wider range of data variability, independence from the differential model and the numerical method at hand, flexibility in exploiting additional geometrical information and higher grid quality.

We first re-framed the problem of mesh agglomeration as a graph partitioning problem, by exploiting the graph representation of the connectivity structure of mesh elements. We then developed in this context a framework to employ ML-based strategies, such as k-means and GNNs which can exploit the geometrical information of the grid. We trained GNNs to perform graph partitioning over a suitably constructed database of meshes. We compared the performance of the ML-based strategies with METIS, a classical algorithm for graph partitioning,

grids	p	Agglomeration-based MG			CG
		metis	k-means	GNN	
triangles	1	21	9	9	114
	2	49	10	10	355
	3	63	17	17	1101
random	1	46	41	32	655
	2	89	76	53	4425
	3	115	102	64	9893
Voronoi	1	19	16	15	348
	2	36	25	25	987
	3	39	33	27	3235
squares	1	20	17	17	109
	2	45	34	34	365
	3	46	52	52	703

Table 9: Iteration count of the MG algorithm to reduce the (relative) residual below 10^{-6} employing different initial grids (triangles, random, Voronoi, squares) agglomerated with different strategies (METIS, k-means, GNN) with $\ell = 3$, $p = 1, 2, 3$, $m = 3$. As a comparison, the iteration count of the Conjugate Gradient (CG) method are reported in the last column.

over a set of different grids, featuring also complex real domains such a brain MRI-scan. Results show that ML-based strategies are more robust and can better preserve mesh quality, making them suitable for adaptive mesh coarsening. We also employed the proposed algorithms in the context MG methods in PolyDG framework, showing a lower iteration count for ML-based strategies, with GNN having the best performance. Indeed, GNNs have the advantage to process naturally and simultaneously both the graph structure of mesh and the geometrical information, such the elements areas or their barycentric coordinates. This is not the case of METIS [58], which is meant to process only the graph information, or k-means, which can process only the geometrical information. Moreover, GNNs have a significantly lower online computational cost. Being able to compute the solution in fast manner, even with a slight loss of accuracy, can be particularly beneficial when using multigrid scheme as preconditioners.

Future developments certainly include fostering the generalization capabilities of the GNNs, by generating a larger database of grids or by considering different geometrical features, such as quality metrics or error estimators. Another possibility is to reframe the approach in a reinforcement learning framework, as proposed by [55]. The extension to three dimensional agglomeration strategies should certainly be explored, where the low online computational cost of GNNs will play a key role. It is worth noticing that all of the proposed agglomeration strategies can be applied to the three dimensional case with little or no modification, as they mainly rely on the connectivity graph structure of the mesh, which is a dimensionless entity, or on geometrical quantities with natural extensions such the area/volume of polytopes of their barycentric coordinates. From a point of view of numerical applications several options can be explored: adaptive mesh coarsening, domain decomposition, using MG schemes as preconditioners, considering discretization frameworks other than the PolyDG ones such as the one of Virtual Element Methods. Finally it would be interesting to consider more complex scenarios, for example geophysical numerical simula-

tions, including fluid-structure interaction with complex and moving geometries. Preliminary results of agglomeration of the brain obtained from medical images are encouraging.

CRediT authorship contribution statement

P. F. Antonietti: Conceptualization, Methodology, Resources, Writing - review and editing, Supervision, Project administration, Funding acquisition.

N. Farenga, G. Martinelli, L. Saverio: Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Visualization.

E. Manuzzi: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Visualization, Supervision, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Funding: P.F. Antonietti has been partially supported by the Ministero dell'Università e della Ricerca [PRIN grant numbers 201744KLJL and 20204LN5N5]. P. F. Antonietti and E. Manuzzi are members of INDAM-GNCS.

References

[1] J. Hyman, M. Shashkov, S. Steinberg, The numerical solution of diffusion problems in strongly heterogeneous non-isotropic materials, Journal of Computational Physics 132 (1) (1997) 130–148.

[2] F. Brezzi, K. Lipnikov, V. Simoncini, A family of mimetic finite difference methods on polygonal and polyhedral meshes, Mathematical Models and Methods in Applied Sciences 15 (10) (2005) 1533–1551.

[3] F. Brezzi, K. Lipnikov, M. Shashkov, Convergence of the mimetic finite difference method for diffusion problems on polyhedral meshes, SIAM Journal on Numerical Analysis 43 (5) (2005) 1872–1896.

[4] L. Beirao da Veiga, K. Lipnikov, G. Manzini, The mimetic finite difference method for elliptic problems, Vol. 11, Springer, 2014.

[5] B. Cockburn, B. Dong, J. Guzmán, A superconvergent ldg-hybridizable galerkin method for second-order elliptic problems, Mathematics of Computation 77 (264) (2008) 1887–1916.

[6] B. Cockburn, J. Guzmán, H. Wang, Superconvergent discontinuous galerkin methods for second-order elliptic problems, Mathematics of Computation 78 (265) (2009) 1–24.

[7] B. Cockburn, J. Gopalakrishnan, R. Lazarov, Unified hybridization of discontinuous galerkin, mixed, and continuous galerkin methods for second order elliptic problems, SIAM Journal on Numerical Analysis 47 (2) (2009) 1319–1365.

[8] B. Cockburn, J. Gopalakrishnan, F.-J. Sayas, A projection-based error analysis of hdg methods, Mathematics of Computation 79 (271) (2010) 1351–1367.

[9] J. S. Hesthaven, T. Warburton, Nodal discontinuous Galerkin methods: algorithms, analysis, and applications, Springer Science & Business Media, 2007.

[10] F. Bassi, L. Botti, A. Colombo, D. A. Di Pietro, P. Tesini, On the flexibility of agglomeration based physical space discontinuous galerkin discretizations, Journal of Computational Physics 231 (1) (2012) 45–65.

[11] P. F. Antonietti, S. Giani, P. Houston, hp-version composite discontinuous galerkin methods for elliptic problems on complicated domains, SIAM Journal on Scientific Computing 35 (3) (2013) A1417–A1439.

[12] A. Cangiani, E. H. Georgoulis, P. Houston, hp-version discontinuous galerkin methods on polygonal and polyhedral meshes, Mathematical Models and Methods in Applied Sciences 24 (10) (2014) 2009–2041.

[13] P. F. Antonietti, A. Cangiani, J. Collis, Z. Dong, E. H. Georgoulis, S. Giani, P. Houston, Review of discontinuous galerkin finite element methods for partial differential equations on complicated domains, in: Building bridges: connections and challenges in modern approaches to numerical partial differential equations, Springer, 2016, pp. 281–310.

[14] A. Cangiani, Z. Dong, E. H. Georgoulis, P. Houston, hp-Version discontinuous Galerkin methods on polygonal and polyhedral meshes, Springer, 2017.

[15] P. F. Antonietti, C. Facciolà, P. Houston, I. Mazzieri, G. Pennesi, M. Verani, High–order discontinuous galerkin methods on polyhedral grids for geophysical applications: seismic wave propagation and fractured reservoir simulations, Polyhedral Methods in Geosciences (2021) 159–225.

[16] L. Beirão da Veiga, F. Brezzi, A. Cangiani, L. D. Manzini, GianM.and Marini, A. Russo, Basic principles of virtual element methods, Mathematical Models and Methods in Applied Sciences 23 (01) (2013) 199–214.

[17] L. Beirão da Veiga, F. Brezzi, L. D. Marini, A. Russo, The hitchhiker's guide to the virtual element method, Mathematical models and methods in applied sciences 24 (08) (2014) 1541–1573.

[18] L. Beirão da Veiga, F. Brezzi, L. Marini, A. Russo, Virtual element method for general second-order elliptic problems on polygonal meshes, Mathematical Models and Methods in Applied Sciences 26 (04) (2016) 729–750.

[19] L. Beirao da Veiga, F. Brezzi, L. D. Marini, A. Russo, Mixed virtual element methods for general second order elliptic problems on polygonal meshes, ESAIM: Mathematical Modelling and Numerical Analysis 50 (3) (2016) 727–747.

[20] L. Beirao da Veiga, N. Bellomo, F. Brezzi, L. Marini, Recent results and perspectives for virtual element methods, Mathematical Models and Methods in Applied Sciences (2021) 1–6.

[21] P. F. Antonietti, L. Beirão da Veiga, G. Manzini, The Virtual Element Method and its Applications, Springer International Publishing, 2022.

[22] D. A. Di Pietro, A. Ern, S. Lemaire, An arbitrary-order and compact-stencil discretization of diffusion on general meshes based on local reconstruction operators, Computational Methods in Applied Mathematics 14 (4) (2014) 461–472.

[23] D. A. Di Pietro, A. Ern, A hybrid high-order locking-free method for linear elasticity on general meshes, Computer Methods in Applied Mechanics and Engineering 283 (2015) 1–21.

[24] D. A. Di Pietro, A. Ern, Hybrid high-order methods for variable-diffusion problems on general meshes, Comptes Rendus Mathématique 353 (1) (2015) 31–34.

[25] D. A. Di Pietro, A. Ern, S. Lemaire, A review of hybrid high-order methods: formulations, computational aspects, comparison with other methods, in: Building bridges: connections and challenges in modern approaches to numerical partial differential equations, Springer, 2016, pp. 205–236.

[26] D. A. Di Pietro, J. Droniou, The Hybrid High-Order method for polytopal meshes, Vol. 19, Springer, 2019.

[27] M. Attene, S. Biasotti, S. Bertoluzza, D. Cabiddu, M. Livesu, G. Patanè, M. Pennacchio, D. Prada, M. Spagnuolo, Benchmark of polygon quality metrics for polytopal element methods, arXiv preprint arXiv:1906.01627.

[28] D. A. Di Pietro, L. Formaggia, R. Masson, et al., Polyhedral methods in geosciences.

[29] T. F. Chan, J. Xu, L. Zikatanov, An agglomeration multigrid method for unstructured grids, Contemporary Mathematics 218 (1998) 67–81.

[30] P. F. Antonietti, P. Houston, G. Pennesi, E. Süli, An agglomeration-based massively parallel non-overlapping additive schwarz preconditioner for high-order discontinuous galerkin methods on polytopic grids, Mathematics of Computation.

[31] Y. Pan, P.-O. Persson, Agglomeration-based geometric multigrid solvers for compact discontinuous galerkin discretizations on unstructured meshes, Journal of Computational Physics 449 (2022) 110775.

[32] J. R. Gilbert, G. L. Miller, S.-H. Teng, Geometric mesh partitioning: Implementation and experiments, SIAM Journal on Scientific Computing 19 (6) (1998) 2091–2110.

[33] F. Bassi, L. Botti, A. Colombo, S. Rebay, Agglomeration based discontinuous galerkin discretization of the euler and navier–stokes equations, Computers & fluids 61 (2012) 77–85.

[34] P. F. Antonietti, M. Sarti, M. Verani, Multigrid algorithms for hp-discontinuous galerkin discretizations of elliptic problems, SIAM Journal on Numerical Analysis 53 (1) (2015) 598–618.

[35] P. F. Antonietti, P. Houston, X. Hu, M. Sarti, M.and Verani, Multigrid algorithms for hp-version interior penalty discontinuous galerkin methods on polygonal and polyhedral meshes, Calcolo 54 (4) (2017) 1169–1198.

[36] J. Xu, L. Zikatanov, Algebraic multigrid methods, Acta Numerica 26 (2017) 591–721.

[37] T. F. Chan, S. Go, L. Zikatanov, Multilevel elliptic solvers on unstructured grids, in: Computational Fluid Dynamics Review 1998: (In 2 Volumes), World Scientific, 1998, pp. 488–511.

[38] P. Antonietti, G. Pennesi, V-cycle multigrid algorithms for discontinuous galerkinmethods on non-nested polytopic meshes, Journal of Scientific Computing 78 (2019) 625–652.

[39] P. F. Antonietti, B. Ayuso, Schwarz domain decomposition preconditioners for discontinuous galerkin approximations of elliptic problems: non-overlapping case, ESAIM: Mathematical Modelling and Numerical Analysis 41 (1) (2007) 21–54.

[40] P. F. Antonietti, S. Giani, P. Houston, Domain decomposition preconditioners for discontinuous galerkin methods for elliptic problems on complicated domains, Journal of Scientific Computing 60 (1) (2014) 203–227.

[41] A. Toselli, O. Widlund, Domain decomposition methods-algorithms and theory, Vol. 34, Springer Science & Business Media, 2004.

[42] X. Feng, O. A. Karakashian, Two-level additive schwarz methods for a discontinuous galerkin approximation of second order elliptic problems, SIAM Journal on Numerical Analysis 39 (4) (2001) 1343–1365.

[43] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707.

[44] M. Raissi, G. E. Karniadakis, Hidden physics models: Machine learning of nonlinear partial differential equations, Journal of Computational Physics 357 (2018) 125–141.

[45] F. Regazzoni, L. Dedè, A. Quarteroni, Machine learning for fast and reliable solution of time-dependent differential equations, Journal of Computational Physics 397 (2019) 108852.

[46] F. Regazzoni, L. Dedè, A. Quarteroni, Machine learning of multiscale active force generation models for the efficient simulation of cardiac electromechanics, Computer Methods in Applied Mechanics and Engineering 370 (2020) 113268.

[47] J. S. Hesthaven, S. Ubbiali, Non-intrusive reduced order modeling of nonlinear problems using neural networks, Journal of Computational Physics 363 (2018) 55–78.

[48] D. Ray, J. S. Hesthaven, An artificial neural network as a troubled-cell indicator, Journal of Computational Physics 367 (2018) 166–191.

[49] P. F. Antonietti, M. Caldana, L. Dede, Accelerating algebraic multigrid methods via artificial neural networks, arXiv preprint arXiv:2111.01629.

[50] F. Regazzoni, M. Salvador, L. Dedè, A. Quarteroni, A machine learning method for real-time numerical simulations of cardiac electromechanics, arXiv preprint arXiv:2110.13212.

[51] P. Antonietti, E. Manuzzi, Refinement of polygonal grids using convolutional neural networks with applications to polygonal discontinuous galerkin and virtual element methods, Journal of Computational Physics 452 (2022) 110900.

[52] P. Antonietti, F. Dassi, E. Manuzzi, Machine learning based refinement strategies for polyhedral grids with applications to virtual element and polyhedral discontinuous galerkin methods, Journal of Computational Physics (2022) 111531.

[53] W. L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs (2017). arXiv:arXiv:1706.02216.

[54] A. Gatti, Z. Hu, T. Smidt, E. G. Ng, P. Ghysels, Deep learning and spectral embedding for graph partitioning (2021). arXiv:arXiv:2110.08614.

[55] A. Gatti, Z. Hu, T. Smidt, E. G. Ng, P. Ghysels, Graph partitioning and sparse matrix ordering using reinforcement learning and graph neural networks (2021). arXiv:arXiv:2104.03546.

[56] H. Xu, Z. Duan, Y. Wang, J. Feng, R. Chen, Q. Zhang, Z. Xu, Graph partitioning and graph neural network based hierarchical graph matching for graph similarity computation, Neurocomputing 439 (2021) 348–362.

[57] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (7553) (2015) 436–444.

[58] G. Karypis, V. Kumar, Kumar, v.: A fast and high quality multilevel scheme for partitioning irregular graphs. siam journal on scientific computing 20(1), 359-392, Siam Journal on Scientific Computing 20. doi:10.1137/S1064827595287997.

[59] J. Macqueen, Some methods for classification and analysis of multivariate observations (1967).

[60] J. A. Hartigan, M. A. Wong, Algorithm as 136: A k-means clustering algorithm, Journal of the royal statistical society. series c (applied statistics) 28 (1) (1979) 100–108.

[61] A. Likas, N. Vlassis, J. J. Verbeek, The global k-means clustering algorithm, Pattern recognition 36 (2) (2003) 451–461.

[62] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[63] P. C. Petersen, Neural network theory, University of Vienna.

MOX Technical Reports, last issues

Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

78/2022 Bucelli, M.; Gabriel, M. G.; Gigante, G.; Quarteroni, A.; Vergara, C.
A stable loosely-coupled scheme for cardiac electro-fluid-structure interaction

77/2022 Ziarelli, G.; Dede', L.; Parolini, N.; Verani, M.; Quarteroni, A.
Optimized numerical solutions of SIRDVW multiage model controlling SARS-CoV-2 vaccine roll out: an application to the Italian scenario.

76/2022 Spreafico, M.; Ieva, F.; Fiocco, M.
Longitudinal Latent Overall Toxicity (LOTox) profiles in osteosarcoma: a new taxonomy based on latent Markov models

71/2022 Calabrò, D.; Lupo Pasini, M.; Ferro, N.; Perotto, S.
A deep learning approach for detection and localization of leaf anomalies

74/2022 Salvador, M.; Regazzoni, F.; Dede', L.; Quarteroni, A.
Fast and robust parameter estimation with uncertainty quantification for the cardiac function

70/2022 Andrini, D.; Balbi, V.; Bevilacqua, G.; Lucci, G.; Pozzi, G.; Riccobelli, D.
Mathematical modelling of axonal cortex contractility

69/2022 Franco, N.R; Manzoni, A.; Zunino, P.
Learning Operators with Mesh-Informed Neural Networks

68/2022 Orlando, G.; Benacchio, T.; Bonaventura, L.
An IMEX-DG solver for atmospheric dynamics simulations with adaptive mesh refinement

72/2022 Spreafico, M.; Ieva, F.; Arlati, F.; Capello, F.; Fatone, F.; Fedeli, F.; Genalti, G.; Anninga, J.; Gelderblom, H.; Fiocco, M.
Novel longitudinal multiple overall toxicity score to quantify adverse events experienced by patients during chemotherapy treatment: a retrospective analysis of the MRC BO06 trial in osteosarcoma

73/2022 Spreafico, M.; Gasperoni, F.; Barbati, G.; Ieva, F.; Scagnetto, A.; Zanier, L.; Iorio, A.; Sinagra, G.; Di Lenarda, A.
Adherence to disease-modifying therapy in patients hospitalized for HF: findings from a community-based study