GraphMesh: Geometrically Generalized Mesh Refinement using GNNs

Ainulla Khan[1 [0009−0007−4062−2049]], Moyuru Yamada[1 [0009−0009−1907−7503]], Abhishek Chikane[1], and Manohar Kaul[1 [0000−0003−1871−1620]]

Fujitsu Research of India, {ainulla.khan, yamada.moyuru, abhishek.chikane, manohar.kaul}@fujitsu.com

Abstract. Optimal mesh refinement is important for finite element simulations, facilitating the generation of non-uniform meshes. While existing neural network-based approaches have successfully generated high quality meshes, they can only handle a fixed number of vertices seen during training. We introduce GraphMesh, a novel mesh refinement method designed for geometric generalization across meshes with varying vertex counts. Our method employs a two-step process, initially learning a unified embedding for each node within an input coarse mesh, and subsequently propagating this embedding based on mesh connectivity to predict error distributions. By learning a node-wise embedding, our method achieves superior simulation accuracy with reduced computational costs compared to current state-of-the-art methods. Through experimentation and comparisons, we showcase the effectiveness of our approach across various scenarios, including geometries with different vertex counts. We validated our approach by predicting the local error estimates for the solution of Poisson's equation.

Keywords: Finite Element Methods · Mesh Refinement · Graph Neural Networks.

1 Introduction

In the fields of structural mechanics [2], aerospace [3], geophysics [4], and acoustics [5] physics simulation is a crucial tool, allowing for the exploration of phenomena that are otherwise challenging to study directly. For these applications, simulations rely on mathematical models represented mainly using partial differential equations (PDEs) [1]. These PDEs are solved on a specific geometric domain using Finite Element Method (FEM) [18]. Meshing is the initial step in FEM, wherein a physical structure is discretized into a mesh of finite elements. Each element represents a segment of the overall behavior of the structure. The method solves PDEs to provide the Finite Element (FE) solution by utilizing mesh of the structure. The effectiveness of FEM highly depends on the mesh quality. The mesh needs to be sufficiently detailed to capture critical structural behaviors, however finer meshes yield longer computations. Therefore, mesh refinement methods focus on generating an optimal mesh, where only specific

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_9

regions with simulation errors are refined for improving the simulation accuracy. The traditional mesh refinement method involves a multi-step process: initially generating coarse meshes for the geometry, followed by computing the FE solution on these initial meshes. Subsequently, a posteriori error estimation is conducted to refine the mesh elements in high-error regions. These steps repeat until the mesh meets the user's specified tolerance. Despite its effectiveness in generating non-uniform meshes, this method is computationally intensive because of the repeated execution of complex error estimation step [7]. Significant advancements have been witnessed through introduction of deep learning techniques in the area of mesh refinement to overcome the limitations of the traditional methods.

MeshingNet [9] is a significant work in the domain of mesh refinement using deep learning techniques [8]. It introduces a novel approach to mesh generation, employing deep neural networks to automate the mesh refinement. The key innovation in MeshingNet is its ability to generate high-quality meshes for geometries by learning from existing mesh datasets. This approach significantly reduces the reliance on manual meshing and expert knowledge. This approach uses Triangle software [6] for dataset generation, which involves Delaunay-based mesh refinement. The reported results outperform a non deep learning based method, ZZ estimation [11]. However, a crucial limitation of MeshingNet is the lack of the generalization capability to handle a number of edges that are different from those in training data, further the method accommodates only a single dimension for learning the boundary condition with other limited features.

There are other works that use graph neural networks for learning mesh-based simulations. But, they focus on simultaneously approximating physical quantities for time-dependent PDEs and perform Adaptive Mesh Refinement (AMR) on geometries across multiple time steps [12]. The method particularly excels in handling External Dynamic Lagrangian Systems, where mesh refinement occurs progressively across different time steps adapting progressively to external (time dependent) conditions. Our method is different from this work as our method can be used for various PDEs (e.g. Poisson's Equation, Linear Elasticity) while researchers in [12] consider only time-dependent PDEs. On the other-hand, GraphMesh deals with automatic mesh refinement on domains where the approach of AMR may not be applicable.

A recent work known as GMR-Net [13] has introduced an effective mesh refinement approach using Graph Convolutional Networks (GCNs) [17]. The above work leverages supervised learning specifically for elliptic PDEs. In this method, the model exhibits the capability to predict local error density at each nodal region of the coarse mesh with advanced input features based on polygon vertices to enhance prediction accuracy. However, a major limitation of this method lies in its inability to generalize over geometries characterized by a number of vertices beyond a predefined range. This limitation occurs due the use of a fixed input embedding size for features, with padding employed for accommodating a restricted range of polygonal geometries based on the number of vertices.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_9

The current landscape of mesh refinement methodologies exhibits limitations in their performance, particularly when faced with tested geometries that deviate from the training set. Addressing this challenge necessitates the development of a model capable of accommodating input polygonal geometries with an arbitrary number of vertices. Such a model should not only demonstrate adaptability to diverse geometries but also excel in delivering accurate refinements while simultaneously optimizing simulation time.

In response to the existing limitations in mesh refinement methods, we present GraphMesh, a geometrically generalized mesh refinement model to handle polygonal geometries with any given numbers of vertices. Enclosed within a generalized Embedding-Decoder based two-stage framework GraphMesh is designed to refine meshes in two sequential steps, leading to the generation of high-quality optimal meshes. In the first step, GCN-based network is employed to embed the node-wise input features to a latent embedding. Subsequently, a second GCN employs a residual connections based architecture on the obtained node-wise latent embeddings to predict simulation errors for the solution of Poisson's equation. This two-stage process enhances the adaptability of our framework, ensuring effective performance across any shaped geometries.

Leveraging the capabilities of Graph Neural Networks, our proposed framework handles polygonal structures with varying numbers of edges, addressing a crucial limitation observed in existing methodologies. The essence of this approach is not only the model's adaptability to diverse polygonal configurations but also its significant impact on enhancing simulation accuracy while concurrently reducing simulation time. Our work distinguishes itself through the following contributions:

1. We propose a novel architecture that performs mesh refinement on any given geometric shape irrespective of its number of vertices, without re-training.
2. Our method shows superior results in the trade-off between simulation accuracy and simulation cost over the state of the art existing method.

The following section provides background details, followed by Section 3, which discusses our methodology. Section 4 outlines the experimental setup, while section 5 delves into the results. The main findings from our experiments are then discussed in section 6, and the conclusion, along with potential future applications of our research, is presented in section 7.

2 Background

2.1 Mesh Generation

Mesh representations of shapes involve discretization of the domain leading to the generation of both CoM and FiM of the given geometry. In this work, we generated meshes by initially creating diverse input geometries through the methodology employed in [13] and subsequently utilizing the Gmsh software on the generated geometries. The geometry generation process first, entails the use of two

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_9

concentric circles with proper spacing. Then the internal area formed by these circles is subsequently divided into N sections, where N represents the number of vertices required for the specific geometry under generation. Subsequently, within each section, an interior point is randomly generated which forms the polygon vertex, lastly joining of the points in each section leads to the formation of a unique geometric shape.

2.2 FE Solution and Simulation Error

The generated meshes are used for simulations, executed with the aid of an open-source FE solver FreeFEM [15]. The problem we address in this paper involves the solution of Poisson's equation $\bigtriangledown u + \varphi = 0$ within generated meshes with the boundary condition of $u = \alpha$ imposed on the polygon boundaries. Here, φ and α represents the PDE parameter and boundary value of the input meshes respectively. Following the methodology outlined by Minseong et al. [13], we estimated the simulation error values e at each nodal location of input CoM. The error estimation process utilizes the simulation results u_{CoM} and u_{FiM} corresponding to both CoM and FiM, respectively.

2.3 Characteristic Length

We facilitated the mesh refinement using refinement module of Gmsh. This refiner operates on the basis of an element's l, which can be dynamically adjusted to enhance accuracy. Specifically, the refinement process focuses on regions where the e around a given node is notably large. As the characteristic length decreases, meshes are locally refined in the vicinity of nodal region, optimizing the resolution in areas with significant error deviations. The optimal length around a node, is precisely defined by a user-defined parameter β as:

$$l = \frac{\beta}{e}. \tag{1}$$

Here, β plays a role in governing the refinement density of the meshes, offering users the flexibility to tailor the refinement process based on specific considerations and desired levels of accuracy.

3 Methodology

3.1 Overview

The core of our approach involves training a GNN model to learn a mapping between parameters, including geometry G, boundary conditions (BC), and the corresponding region-specific simulation error e. To enhance geometric generalization, we leverage the inherent properties of GNNs, which are independent of the number of nodes within a given graph. Specifically, we employ a two-step approach, consisting of the Vertex-Graph G^{nv} for each node n in the input CoM

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_9

and the Main-Graph G^m representing the input CoM itself (Figure 1). The graph G^{nv} captures embeddings for each CoM node n using geometric features with respect to polygon vertices and boundary, PDE features, regardless of the number of edges or vertices. The nodal features between the vertex x and node n of G^{nv} is represented by f_{nx}. Consequently, the graph G^m predicts e at each nodal region of the input CoM. The predicted simulation errors (denoted as $\hat{e}$) are then subjected to an established mathematical function (Equation (1)), yielding an optimal length l. These optimal lengths are subsequently employed to guide the mesh refinement process.

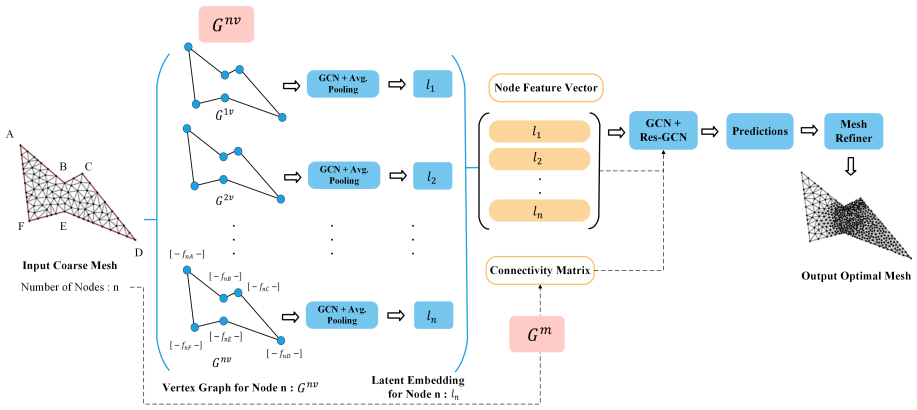

Fig. 1. Architectural Framework with the Vertex-Graph G^{nv} containing node features f_{nx} and Main-Graph G^m, demonstrated through an example input mesh.

By predicting e our proposed method provides control of mesh refinement through adjustments in mesh refinement parameters in equation 1. This presents a versatile and adaptable strategy for refining meshes at the user's discretion (by the choice of β), without necessitating re-training for each choice of l (see equation 1). Moreover, this approach also facilitates a rigorous evaluation methodology that enables the observation of the relationship between simulation accuracy and simulation time across varying refinement levels during inference. This not only provides insights into the trade-offs between accuracy and computational efficiency but also contributes to a more better understanding of the model's performance under different refinement configurations.

We employ a non-neural network based conventional method using the Gmsh software [14] for the generation of ground-truths, encompassing an input geometry, CoM and FiM of the geometry. Simulations are conducted on both the coarse and fine meshes, generating region-wise error distributions based on the L1-norm of simulation results. Subsequently, acknowledging the time-consuming and resource-intensive nature of fine mesh generation, the automation strategy

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_9

is to use GNN-based supervised learning to predict the region-wise simulation errors.

3.2 Data Preparation

For training we use the geometric and PDE features extracted from the CoM. The geometric features, for defining node attributes, are mean value coordinates (MVC), vertex distance, and depth. These features contribute to a representation of nodal attributes, for extraction of the mesh characteristics during the error estimation process.

MVC. The MVC feature computes the weight of a polygon vertex p for the node n in the given input CoM. Considering an input CoM with k vertices the weight η_v for vertex v is computed as [16] :

$$\begin{cases} \eta_v & = \dfrac{q_v}{\sum_{j=1}^{k} q_j} \\ q_v & = \dfrac{tan(\theta_{v-1}/2)+tan(\theta_v/2)}{||p_v - v||} \end{cases}, \tag{2}$$

with θ_v = angle $\angle p_v v p_{v+1}$.

This feature, employed in learning the node's position relative to polygon vertices, demonstrates robustness against variations in polygon geometry [16].

Vertex Distance and Depth. The Vertex Distance feature provides insights into the number of hops separating each polygon vertex p from the node n in the given input CoM, offering information on spatial information. The Depth feature gives the the closest polygon vertex p to the node n within the CoM [13].

PDE and Boundary Condition. The PDE features relates to the PDE parameter φ of the Poisson's equation and the boundary values α on all the boundary edges. For the boundary condition corresponding α value is used for the polygon vertex p.

Our model takes e, as the ground truth feature in our dataset. This unique approach aims to make our model independent of the refinement parameter β.

3.3 Architectural Details

3.3.1 Vertex Graph for Nodal Representations. The formation of the vertex graph G^{nv} is central to our methodology, aiming to obtain a node-wise representation of the overall polygon shape. The graph for a node n of input CoM is given by $G^{nv} = (V^{nv}, E^{nv})$ where G^{nv}represent the external geometry of the polygon. Here, vertices of the polygon represents nodes V^{nv} (A,B,C,D,E and F as given in figure 1), and polygon edges represent the links E^{nv} (Figure 1).

The input node features f_{nx} for a given node n depends on polygon vertex x of the given input CoM. (see section 3.2). Previous state-of-the-art method

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_9

(GMR-Net), resorted to padding to handle these features only up to polygons with pre-defined 9 polygon vertices. Each node n in the graph used in GMR-Net comprised a 29-dimensional node feature vector, with 9, 9 and 9 dimensions allocated to MVC, Vertex Distance, and BC, respectively. Since for the node n these features are different with respect to each polygon vertex p. The remaining two dimensions were dedicated to depth and φ. Notably, a padding mechanism using -1 was employed to address polygons with fewer than 9 edges [13].

Our model introduces a novel concept by constructing a vertex graph for each node n of input CoM, aiming to accommodate features for any number of polygon vertices. This results in a fixed dimensional feature vector for the nodes of the vertex graph. In this configuration, the features f_{nx} of a node n concerning the polygon vertex x are specifically assigned to the corresponding node in the vertex graph. Consequently, a given input CoM consists of sub graphs equivalent to the number of nodes in the mesh. Furthermore, each sub graph contains nodes equivalent to the number of polygon vertices with a fixed dimensional node feature vector irrespective of the polygon geometry. This arrangement makes the model adaptable to diverse polygonal shapes, independent of number of edges without re-training. After the training phase, the latent embedding of the nodes in the input coarse mesh stores information corresponding to the polygon corners.

The vertex graph G^{nv} is designed to encode the polygon's distinct geometry into a representative embedding. Within this graph, the message-passing step within the GCN [17] gives an updated node feature in G^{nv} vector $x_i^{'}$ using the function given by:

$$x_i^{'} = f^v \bigoplus_{k \in N(i)} (x_k W). \tag{3}$$

Where f^v denotes the implementation of MLP and $\bigoplus_{k \in N(i)}$ denote averaging over the one-hop neighbours $N(i)$ of node feature vector x_i with learnable parameter W. This function aggregates the neighborhood's information for each node via its connected edges. Further, to obtain a graph level latent representation for the nodes in main graph G^m, a subsequent average pooling layer is introduced over the nodes of the corresponding vertex graph G^{nv}. For information aggregation, we experimentally opted for a single-layered GCN with 128 hidden-dimensions on the vertex graph, hence integrating one-hop neighborhood-based information.

3.3.2 Main Graph for Prediction of Simulation Error. The latent representations of nodes within the CoM, learned from the vertex graph, act as the node features for the main graph G^m and the edge information is obtained using the linkages between nodes in CoM. Subsequently, the graph G^m is passed through a GCN architecture based on Residual Network [13]. The architecture employs information aggregation over a 6-hop neighborhood, featuring skip-connections every two layers of GCN, and a hidden dimension size of 128. This ResNets-based GCN enables the extraction of node-level e within the CoM. The proposed two-step process enables an initial compression of polygon vertex-based

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_9

information into a fixed-sized latent embedding followed by leveraging this embedding for predicting e values. This introduces a generalization ability to our approach, making it universally applicable to various polygon geometries without any limitations concerning the number of polygon vertices.

3.4 Training and Evaluation

During the training phase, both of our graphs are trained using a supervised approach, with ground truth error values associated with each node in G_m. We train GraphMesh with the L1 loss function between the ground truth e and model predicted $\hat{e}$ across N nodes, as outlined in equation 4. To evaluate the model's performance, we calculate the average simulation error estimates A_e across nodes in the refined mesh. This involves, first obtaining node-wise l values using the model-predicted $\hat{e}$ with the help of equation 1. The Gmsh mesh refiner is then employed to generate refined meshes based on these l values. Subsequently simulations are conducted on these refined meshes, to compute A_e by comparing the simulation results of the model-predicted refined mesh with the corresponding FiM (section 2.2) followed by node-wise averaging.

Furthermore, we assess methods for achieving optimal simulation accuracy within minimal time. In this evaluation, we generate multiple refined meshes with varying refinement levels by adjusting β values based on a single model-predicted $\hat{e}$. For each refined mesh, we compute A_e values while recording the simulation time. The results are visually represented, illustrating the variation in A_e and simulation time under this evaluation.

$$L = \frac{1}{N} \sum_{i=1}^{N} |\hat{e}_i - e_i|. \tag{4}$$

4 Experimental Setup

4.1 Dataset

The CoM in our study contained elements within the range of 250-550, with corresponding fine meshes FiM containing 15-20 times the number of elements present in their coarse mesh counterparts. The positional coordinates of each node in both the CoM and FiM were constrained within the 0 to 1 range. For GraphMesh, we generated train-validation dataset (section 2.2) featuring 8000 training CoM and FiM, alongside 1500 validation CoM and FiM.

4.2 Experiment Details

To assess the performance of our proposed method, our experiments were performed using one Nvidia A30 24GB GPU. We implemented our neural network and training scripts using PyTorch [19] and PyTorch Geometric [20]. The batch size was set to 32 and Adam optimizer [21] with a learning rate of 0.0005 was used for optimization. The training process spanned 100 epochs, and the corresponding loss curve is depicted in figure 2.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_9

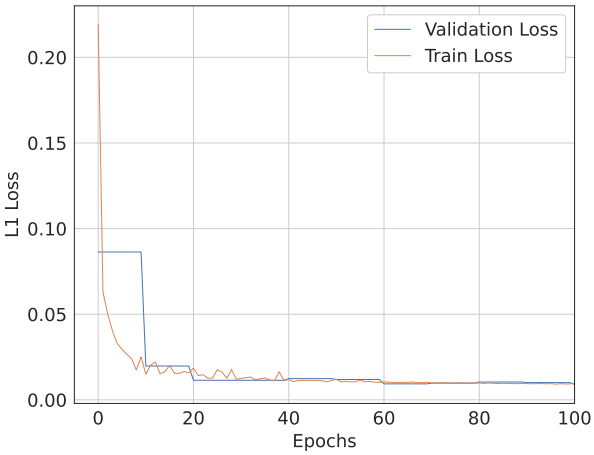

Fig. 2. Training and validation loss curves for GraphMesh

4.3 Governing Equation and Boundary Condition

The displacement field u represented the FE solution. The PDE parameter φ was randomly assigned values within the range of 0 to 1. Additionally, each boundary (α) in the mesh was subject to a random displacement ranging from 1 to 100. Further, the boundary values were normalized between 0 and 1 during training.

4.4 Evaluation Details

We conduct evaluation of our model across two distinct data settings. (1) In-distribution data setting, where both the training and validation sets comprise meshes with polygons having 6, 7, 8, and 9 vertices (E). (2) Out-of-distribution data setting, where the training set includes meshes of polygons with E equal to 6, 7, 8, and 9, and the validation set introduces meshes of polygons with E equal to 10 and 12. We generated refined meshes with model predicted $\hat{e}$ using Gmsh's l, with β set to $0.0084 * max(\hat{e})$, where $max(\hat{e})$ is the maximum of predicted simulation error values among the nodes of the corresponding mesh. The authors of [13] assessed their technique, GMR-Net, across various methods and proved its superiority. Therefore, we conduct a performance comparison between our model and the current state-of-the-art method, GMR-Net, both quantitatively and qualitatively, to encompass all existing methods. Qualitative assessments involve comparing mesh refinements, while quantitative evaluations employ simulations on output refined meshes from GraphMesh to obtain A_e across nodes of the refined mesh. Notably, while GraphMesh inherently handles out-of-distribution data settings, GMR-Net required feature down-sampling for inferencing. In the case of GMR-Net, features corresponding to vertices exceeding

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_9

the 9-vertex limit are averaged out and equally distributed among the nine vertex features to facilitate fair comparisons without re-training.

Leveraging our method's independence from the refinement parameter β, we also conducted evaluations on a case with varying refinement levels. We set 10 values of β between $0.0084 * max(e)$ and $0.01 * max(e)$. The corresponding simulation error versus simulation time plots are observed across varying refinement level-based meshes.

5 Results

5.1 Mesh Comparison

In the evaluation of refinement results, we conducted a comparison using the data from both the in-distribution (Figure 3) and out-of-distribution (Figure 4) datasets, considering polygons with all polygon edges subjected to displacements as boundary conditions during training. The figures present refined meshes from GraphMesh and GMR-Net, along with an error distribution plot depicting the distribution of e on the evaluated geometries. The error distribution plot is obtained based on the comparison of simulation results between the CoM and FiM, as explained in Section 2.2. Ideally, the model should refine regions with simulation errors. Figures 3 and 4 highlight GraphMesh's accurate capture of refinement regions, resulting in the generation of improved non-uniform meshes.

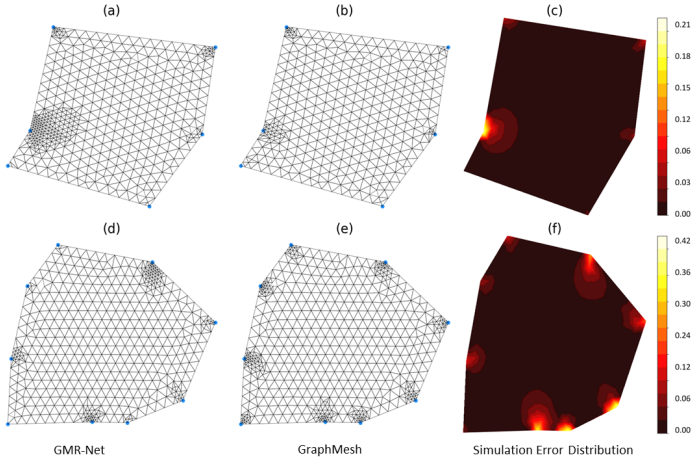

Fig. 3. Output refined meshes obtained for in-distribution dataset ($E = 6$ in (a),(b) and (c); $E = 9$ in (d),(e) and (f)) using the methods GMR-Net ((a) and (d)) and GraphMesh ((b) and (e)) with error distribution of the corresponding CoM ((c) and (f))

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_9

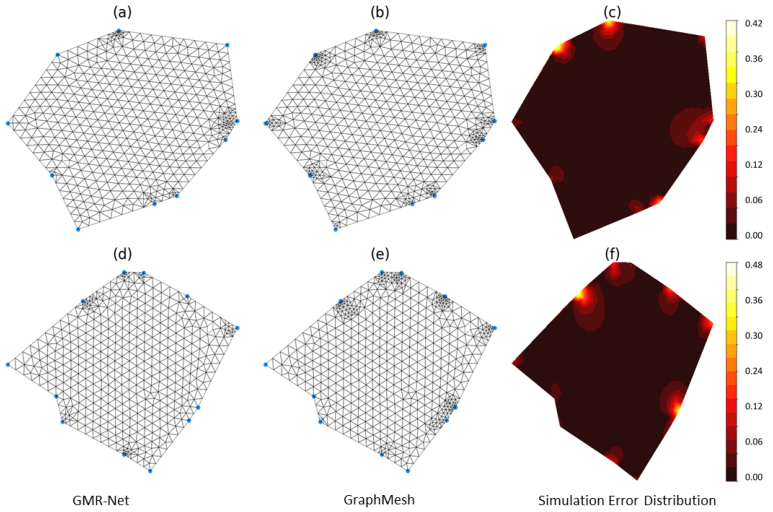

Fig. 4. Output refined meshes obtained for out-of-distribution dataset (E = 10 in (a),(b) and (c); E = 12 in (d),(e) and (f)) using the methods GMR-Net ((a) and (d)) and GraphMesh ((b) and (e)) with error distribution of the corresponding CoM ((c) and (f))

5.2 Simulation Error Comparison

In figure 5, we present quantitative comparisons of simulation errors across a set of 100 unseen meshes, with refined meshes obtained using refinement parameter equal to $0.0084 * max(e)$. The focus here is on polygons with E ranging from 6 to 9, constituting an in-distribution dataset.

The plots deomstrate the distribution of simulation errors, providing insights into the effectiveness of GraphMesh. Specifically, for $E = 6$ the average simulation error on GraphMesh refined meshes was approximately 21% smaller than that of GMR-Net refined meshes. Hence, the results highlight GraphMesh's capability to yield accurate refinements while efficiently identifying regions of errors, as also observed in the refinement plots depicted in figure 3 and table 1.

Table 1. Average Simulation error and time comparison for in-distribution validation meshes

Metric	Method	E = 6	E = 7	E = 8	E = 9
Average Simulation Error	GMR-Net	0.0063	0.0051	0.0055	0.0062
	GraphMesh	**0.005**	**0.0044**	**0.0046**	**0.0051**
Average Simulation Time (s)	GMR-Net	0.0065	0.0073	0.0075	0.0076
	GraphMesh	**0.0047**	**0.0055**	**0.0059**	**0.0060**

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_9

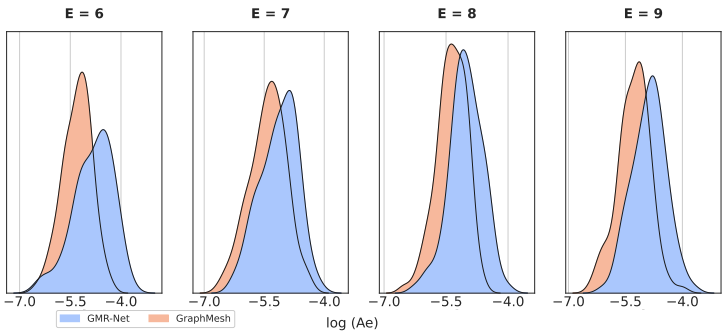

Fig. 5. Comparison of simulation errors for in-distribution dataset

In Figure 6, we present quantitative results for the out-of-distribution dataset, involving polygons with 10 and 12 edges during inference. These results, obtained using the GMR-Net method (as shown in Figure 6), were acquired through a down-sampling technique explained in Section 4.4. Figure 4 displays the qualitative evaluations, which further support the findings in Figure 6. The proposed method accurately captures refinement regions, whereas GMR-Net fails to refine certain regions, leading to outcomes closer to a uniformly coarse mesh.

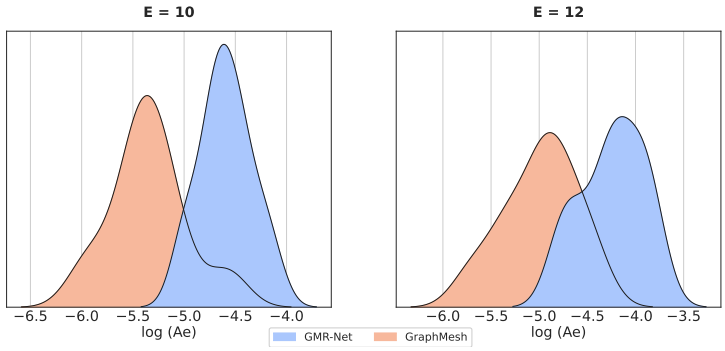

Fig. 6. Comparison of simulation errors for out-of-distribution dataset

5.3 Comparison for various simulation times

Since mesh size depends on the refinement level parameter (β), the simulation time will also depend on β. Therefore, to plot errors at different times, we vary β between 0.0084 * max(e) and 0.01 * max(e) and record the corresponding simulation errors. Figures 7 and 8 compare GraphMesh and GMR-Net for both

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_9

in-distribution and out-of-distribution datasets, in terms of a specific simulation accuracy for varying refinement levels. Each data point on the plot is obtained using the averaged values of simulation time and errors. The results clearly demonstrate that, under each of the 10 refinement levels, GraphMesh outperforms GMR-Net, achieving superior simulation accuracy within a shorter time frame.

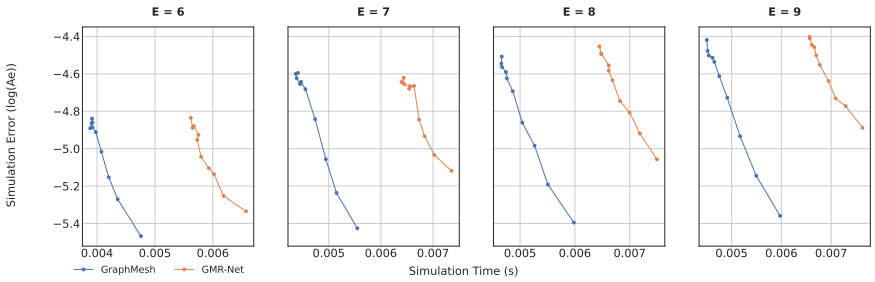

Fig. 7. Comparative evaluation of mesh optimality between GraphMesh and GMR-Net for in-distribution dataset

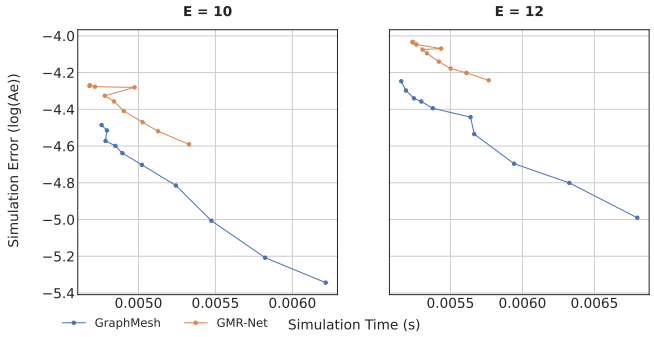

Fig. 8. Comparative evaluation of mesh optimality between GraphMesh and GMR-Net for out-of-distribution dataset

6 Discussion

We focus on several key aspects of GraphMesh that addresses limitations encountered in previous approaches. GMR-Net's ablation study had emphasized the significance of geometric node features based on input polygon vertices, restricting its adaptability to polygons within a fixed range of vertex counts.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_9

GraphMesh addresses this limitation by leveraging GNNs. The introduction of the vertex graph G^{nv} in GraphMesh allows for handling features from any input geometry, proving especially advantageous for intricate shapes where determining the precise number of edges might be challenging.Beyond generalization, the vertex graph favours learning of the inherent topology of a given polygon. The latent embeddings for each node, obtained through learnable aggregations, provide information about the node's positioning relative to polygon corners, boundaries, and PDE-defined conditions. This information is then used in the main graph, which utilizes the mesh connectivity information to predict node-wise simulation errors effectively. The generalization capabilities of GraphMesh across various datasets, both in-distribution and out-of-distribution, highlight the GraphMesh's improvement compared to GMR-Net under the considered scenarios.

7 Conclusion

This paper introduces GraphMesh, a novel methodology designed to enhance geometric generalization in the domain of automatic mesh refinement, specifically emphasizing its adaptability to out-of-distribution geometries. Our approach was tested through the solution of Poisson's equation across diverse scenarios, with comparative assessments against GMR-Net's output refined meshes. We presented comparisons that demonstrated the generalization ability and effectiveness of GraphMesh. Approximately, GraphMesh achieved an improvement in simulation accuracy of 21% with a 27% reduction in simulation time on comparison with existing state of the art method. Potential avenues for improvement include the incorporation of advanced topological features to address complexities in geometries and extensions to accommodate additional boundary conditions and diverse governing equations in various domains.

7.0.1 Disclosure of Interests. The authors do not have any competing interests that are applicable to the content presented in this article.

References

1. Zienkiewicz, O.C., Taylor, R.L., David, F.: The finite element method for solid and structural mechanics. 7th edn. Elsevier (2014)
2. Panthi S.K, Ramakrishnan N., Pathak K.K., Chouhan J.S.: An analysis of springback in sheet metal bending using finite element method (fem). Journal of Materials Processing Technology **186**, 120–124 (2007)
3. Economon T.D., Palacios F., Copeland S.R., Lukaczyk T.W., Alonso J.J.: Su2: An open-source suite for multiphysics simulation and design. Aiaa Journal **54**, 828–846 (2016)
4. Zhengyong R., Jingtian T.: 3d direct current resistivity modeling with unstructured mesh by adaptive finite-element method. Geophysics **75**, H7–H17 (2010)
5. Steffen M.,Bodo N.: Computational acoustics of noise propagation in fluids: finite and boundary element methods. Springer **578** (2008)

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_9

6. Shewchuk, J.R.: Delaunay refinement algorithms for triangular mesh generation. Computational geometry **22**(1-3), 21–74 (2002)
7. Ainsworth M., Oden J.T.: A posteriori error estimation in fnite element analysis. Computational Methods in Applied Mechanics and Engineering **142**, 1–88 (1997)
8. Bank R.E., Weiser A.: Some a posteriori error estimators for elliptic partial diferential equations. Mathematics of Computation **44**, 283–301 (1985)
9. Zhang Z., Wang Y., Jimack P.K., Wang H.: MeshingNet: a new mesh generation method based on deep learning. In: International conference on computational science, pp. 186–198. Springer, Berlin (2020)
10. Zhang Z., Jimack P.K., Wang H.: MeshingNet3D: efficient generation of adapted tetrahedral meshes for computational mechanics. Advances in Engineering Software **157**, 103021 (2021)
11. Zienkiewicz O., Zhu J.: Adaptivity and mesh generation. International Journal for Numerical Methods in Engineering **32**, 783–810 (1991)
12. Pfaff T., Fortunatoet M., Sanchez-Gonzalez A., Battaglia P.: Learning mesh-based simulation with graph networks. In: International conference on learning representations. Vienna (2020)
13. Minseong K., Jaeseung L., Jibum K.: GMR-Net: GCN-based mesh refinement framework for elliptic PDE problems. Engineering with Computers **39**, 3721–3737 (2023)
14. Geuzaine C., Remacle F.: Gmsh: a three dimensional finite element mesh generator with built-in pre- and post-processing facilities. International Journal for Numerical Methods in Engineering **79**, 1309–1331 (2009)
15. Hecht F.: New development in FreeFem++. Journal of Numerical Mathematics **20**, 251–266 (2012)
16. Floater M.S.: Mean value coordinates. Computer Aided Geometric Design **20**, 19–27 (2003)
17. Kipf T.N., Welling M.: Semi-supervised classifcation with graph convolutional networks. In: International conference on learning representations (2017)
18. Abdelaziz Y., Nabbou A., Hamouine A.: A state-of-the art review of the x-fem for computational fracture mechanics. Applied Mathematical Modelling **33**, 4269–4282 (2009)
19. Paszke A., et al.: PyTorch: an imperative style, high performance deep learning library. In: Advances in neural information processing systems. (2019)
20. Fey M., Lenssen J.E.: Fast graph representation learning with PyTorch. In: ICLR workshop on representation learning on graphs and manifolds. (2019)
21. Kingma D.P., Ba J.: Adam: a method for stochastic optimization. In: International conference on learning representations. (2015)

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_9