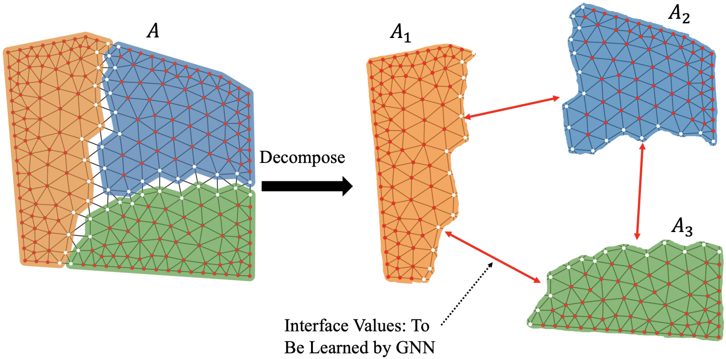
Learning Multilevel Optimized Domain Decomposition

Ali Taghibakhshi, Nasim Gholizadeh

Fall 2022

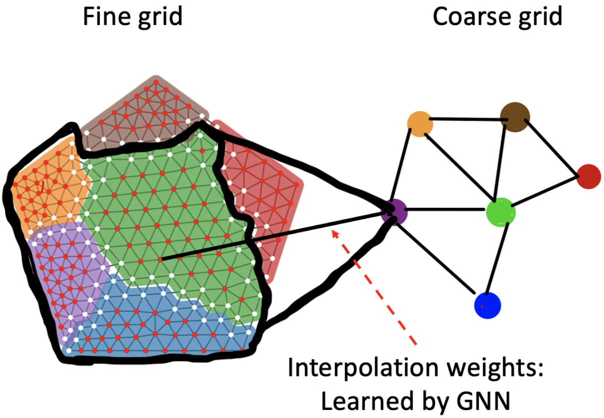
Statement of the problem

- Domain Decomposition Methods (DDMs) among the most popular in solving PDEs
- Discretized PDE system: $Ax = b$, but A^{-1} is expensive
- Decompose $A \to A_1, A_2, ..., A_s$ and solve A_i^{-1} instead



Statement of the problem (continued)

- For multilevel, a coarse level is concerned
- The interpolation operator can also be optimized
- Interpolation weights = edges between coarse and fine grids



Our approach

Learning interface values and the prolongation operator (R_0 and L_i).

$$C = R_0^T \left(R_0 A R_0^T\right)^{-1} R_0$$

$$M = \sum_{i=1}^{S} (\tilde{R}_i)^T (\tilde{A}_i^\delta)^{-1} R_i^\delta$$

$$\text{Error propagation operator :} \quad T = (I - CA)(I - MA) \tag{1}$$

$\tilde{A}_i^\delta = A_i^\delta + h L_i, \quad R_0 : Interpolation operator$

$L : Learned\ interface\ values, \quad \delta : Overlap, D_i^\delta : Subdomains$

$R_i^\delta : Restriction\ to\ D_i^\delta, \quad \tilde{R}_i^\delta : Modified\ restriction\ to\ D_i^\delta$

Loss function

Recap: error propagation operator: $T = (I - CA)(I - MA)$

For $K \in \mathbb{N}$, and m uniformly random vectors x_i:

$$Y_K = \left\{ \left\| (T)^K x_1 \right\|, \left\| (T)^K x_2 \right\|, \ldots, \left\| (T)^K x_m \right\| \right\},$$

To prevent vanishing gradients:

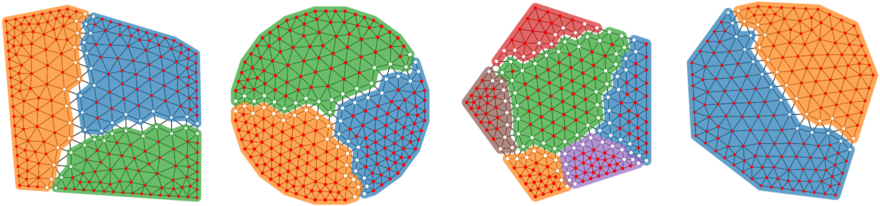
$$Z = \left(\max((Y_1)^1), \max((Y_2)^{\frac{1}{2}}), ..., \max((Y_K)^{\frac{1}{K}}) \right).$$

$$\mathcal{L} = \langle \mathrm{softmax}(Z) \cdot Z \rangle.$$

We use $K = 10$ for our experiments.

Training

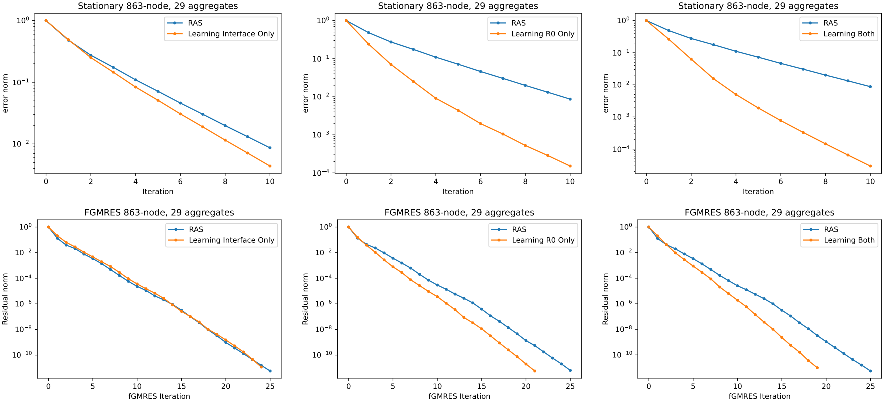
- 100 unstructured grids of size 800-999 (shown below).
- Subdomains obtained by Lloyd (K-means based) clustering.
- Batch size of 10, for 30 epochs, and with learning rate of 10^{-4}.



Learning L_0, R_0, or both

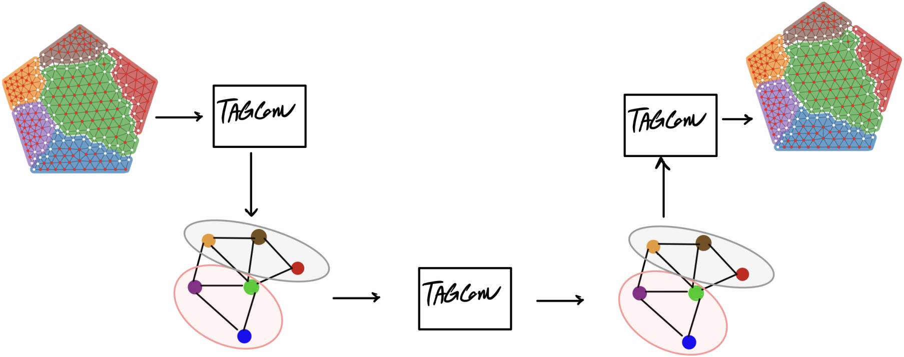
Overtraining the HGNN on a single grid:

L_0 $\quad$ R_0 $\quad$ $R_0 + L_0$



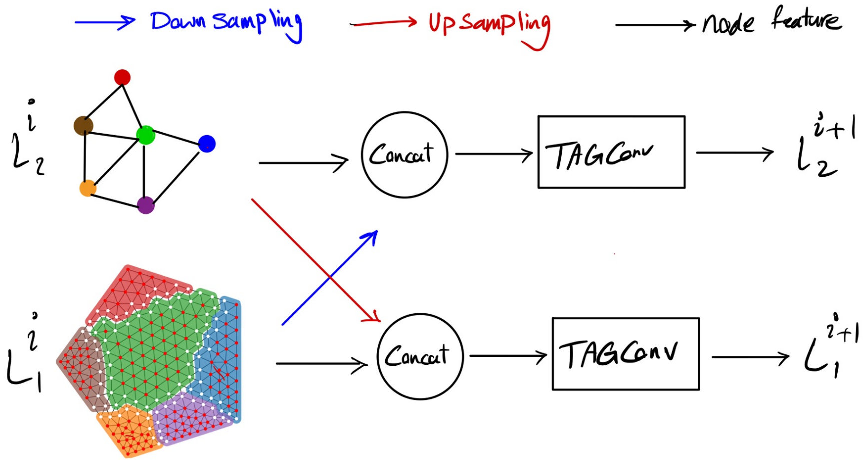
Graph U-net architecture

1 layer of graph U-net:



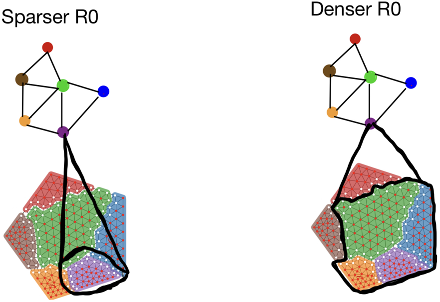
Hierarchical GNN architecture

1 layer of hierarchical GNN:



Results: Loss function variants

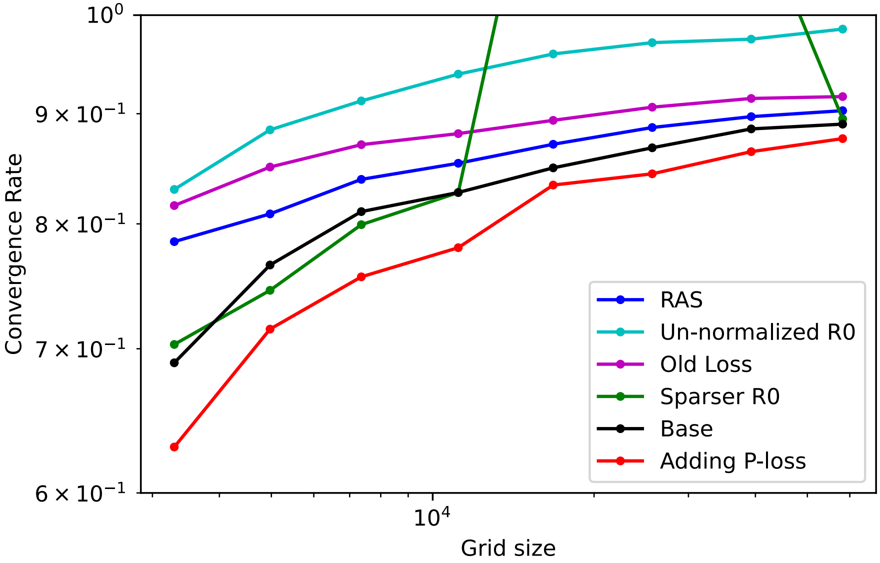
- Un-normalized R_0: No restriction of row sum $= 1$
- Old loss: Loss from [1] ($\mathcal{L} = \max(Y_K)$).
- P-loss: Inspired by [2], adding $\gamma \sum_{i \in \text{columns}} ||R_{0_i}||_A^2$ to the loss.
- Sparser R_0:



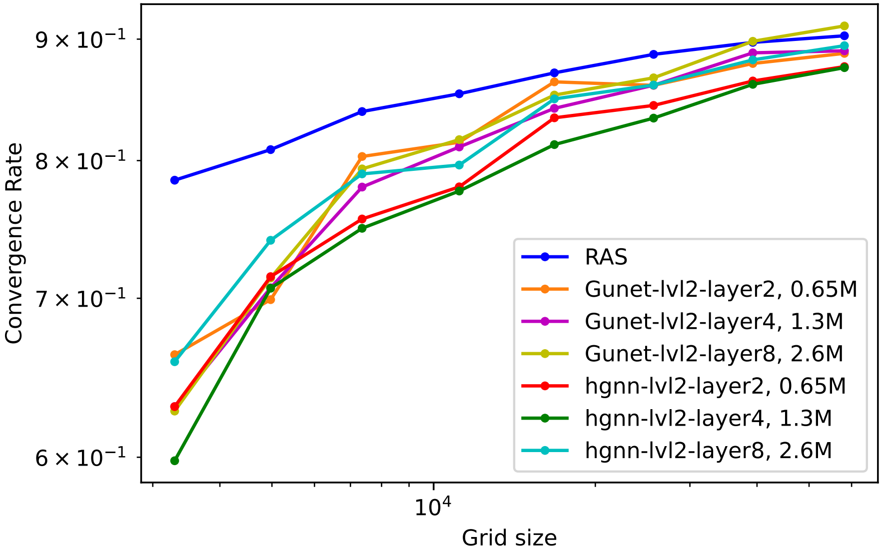
[1] Taghibakhshi *et. al.*, Learning Interface Conditions in Domain Decomposition Solvers, NeurIPS 2022

[2] Olson *et. al.*, A general interpolation strategy for algebraic multigrid using energy minimization, SIAM.

Results: Loss function variants



Results: HGNN vs. Graph U-net



Conclusions

In this project, we studied:

- Learning interpolation operator and interface conditions for 2-level ORAS
- A modified loss function to facilitate leaning 2-level ORAS
- Hierarchical GNN for learning ORAS and comparison with Graph U-net
- An ablation study on number of each architecture's number of layers
- Testing for larger grids than training grid size.

Future directions:

- Neural operator learning for multilevel ORAS.
- Leaning smoothers for AMG using HGNN.
- Trying 3D grids.