Self-Supervised Computational Graph Coarsening for Accelerating Two-Dimensional Subsurface Flow Simulations

Alexander Ryabov[1] , Sergei Shumilin[1] , Viacheslav Naumov[1] , Nikolay Yavich[1] , Sayan Ranu[3] , N. M. Anoop Krishnan[3], Evgeny Burnaev[1 2], Vladimir Vanovskiy[1]

[1]Artificial Intelligence Center, Skolkovo Institute of Science and Technology, 121205 Moscow, Russia
[2]Autonomous Non-Profit Organization Artificial Intelligence Research Institute (AIRI), 105064 Moscow, Russia
[3]Indian Institute of Technology Delhi, Hauz Khas, New Delhi, India 110016

Keywords: Graph coarsening, reduced order modelling, computational graph, numerical modelling, differentiable solver, grid coarsening, upscaling, differentiable physics

Abstract. Efficient simulations of complex physics systems often require solving partial differential equations (PDEs) on fine, unstructured grids, leading to high computational costs. We propose a learnable framework that automatically coarsens these grids while preserving essential dynamics, thus achieving substantial speed-up. We present a graph neural network (GNN) based approach to perform differentiable coarsening of nodes in a self-supervised manner. We ensure the coarsened mesh yields simulation results close to those on the original fine mesh. We demonstrate the universality of our approach by integrating both explicit and implicit finite volume solvers into the training loop. Results on 2D PDEs describing subsurface flow dynamics show that the coarsened meshes can greatly reduce node count without significant loss of accuracy. Additionally, the learned coarsening by GNN is able to generalize to unseen scenarios and retains consistent fidelity across different grid configurations.

1. Introduction

Simulating large-scale physical systems governed by partial differential equations (PDEs) is computationally intensive, especially when high spatial or temporal resolution is required [1,2]. Traditional techniques such as upscaling and reduced-order modeling have been used to mitigate this cost, but they often rely on simplifying assumptions, do not easily generalize to new problem instances, or introduce large approximation errors [3,4]. Recent advances in machine learning, particularly graph neural networks (GNNs), have shown promise in modeling or accelerating PDE-based simulations [5,6]. However, purely surrogate-based approaches may suffer from drift over long simulations or fail to ensure physical conservation laws. We present following contributions:

1. We present a coarsening approach that uses GNN-based node aggregation on a fine grid to obtain node positions and corresponding features on a coarsened grid. It leverages differentiable Voronoi tessellation and graph representation of finite volume solvers (FVM) [7] to effectively minimize the difference between the simulation results on two grids.
2. By incorporating explicit or implicit FVM solvers directly in the training loop, we enforce that the coarsened grid reproduces the solution of the original grid while maintaining numerical stability. For ensuring the stability of the explicit solver we design a special stability loss.

2. Methodology

2.1. Graph representation of grids

We represent an unstructured 2D computational grid as a graph $G = (V, E, A, X)$, where V is the set of nodes (cell centers), E is the set of edges indicating adjacency between cells, A is the adjacency matrix, and X contains node features such as spatial coordinates (x, y) and material properties (e.g., permeability). Our goal is to reduce $|V|$ while preserving accuracy in the PDE solution.

2.1. Physics-guided simulation

We focus on PDEs commonly encountered in subsurface flow and wave propagation. For instance, consider the diffusion (or pressure) equation and its discretization:

$$\frac{\partial u}{\partial t} - \text{div}\,(K\nabla u) = f, 0 < t < T \tag{1}$$

$$D\frac{u^{k+1}-u^k}{\tau} + Au^k = Df^k, k = 0 \cdots m-1 \tag{2}$$

where *u(x,y,t)* means unknown concentration (pressure for the subsurface flow for cases considered in the current paper), *K(x,y)* represents the diffusion coefficient (permeability for the subsurface flow), *f(x,y,t)* is the source/sink term. In discretization $u^k \in R^N$ and $f^k \in R^N$ are discrete pressure and source terms, respectively, D is a diagonal matrix of areas $|V_i|$ and $A \in R^{N\times N}$ is the finite-volume system matrix. We denote by V_i the respective Voronoi cells with area $|V_i|$, by e_{ij} the edges separating adjacent cells V_i and V_j with length $|e_{ij}|$, by h_{ij} the distance between the centers of these cells. Matrix A is defined as:

$$A_{ij} = -\frac{|e_{ij}|}{h_{ij}}\frac{2K_iK_j}{K_i+K_j} \tag{3}$$

for the non-diagonal entries and also diagonal entries are:

$$A_{ii} = -\sum_{j=1,j\neq i}^{N} A_{ij} \tag{4}$$

We use differentiable Voronoi tessellation inside the FVM using methodology we have previously developed [7]. We use both explicit and implicit updating schemes leveraging full differentiability of Voronoi tessellation parameters. The solver is made differentiable by constructing its computations (matrix products, system solves) in a frameworks PyTorch (for explicit solver) or JAX (for implicit solver) [8].

2.3. Graph coarsening pipeline

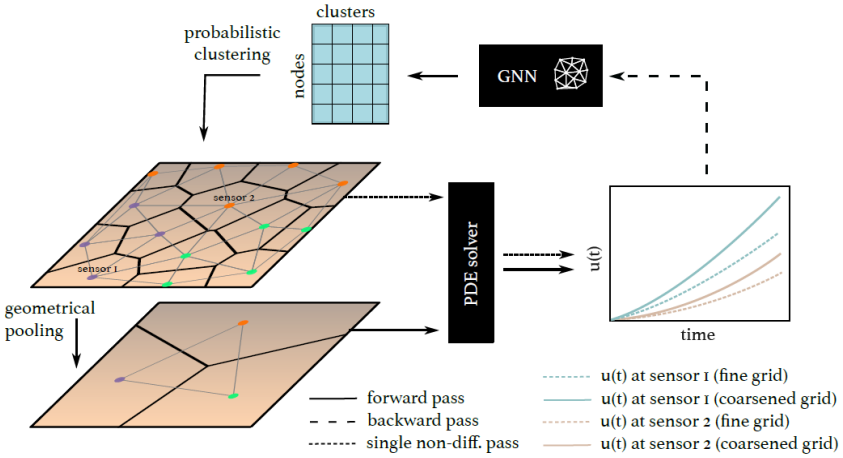

Figure 1. The proposed graph coarsening framework. Graph attention (represented by GNN) predicts the cluster assignments which are used to aggregate the features and obtain a coarsened grid. The GNN is trained by minimizing the error between the field quantity obtained from the simulations using the coarse and the fine (ground truth) grids in a self-supervised way.

Our approach is further development of our previous coarsening approach that optimizes points locations of coarsened grids to make simulation results in critical points of interest (such as wells) close to results on fine grid [9]. We name our previous algorithm as "*Competing algorithm*" because one of our goals is to obtain the results of our new algorithm similar to the previous algorithm in as many scenarios as possible. The unique character of this baseline sets it apart from others. In that work [9],

we tackled the challenging problem of global point optimization, which is conducted on each new grid. In our new method, the GNN directly produces a solution to this optimization problem that is significantly more straightforward and general. That is why we should aim to be close to this baseline [9] (it's more about the best achievable metric we can bet).

In subsurface flow simulations, fluid movement is primarily driven by wells that inject or extract fluids. We model wells as point sources where the intensity of the source depends on the pressure difference between the wellbore and the surrounding area. Below we slightly abuse the standard definitions and use terms "sink point"/"measurement point" interchangeably. Features of source/sink(measurements) nodes (locations of wells) are kept intact during the coarsening process.

Node features x_i (coordinates x and y and permeability K) are aggregated per cluster into coarsened ones using a weighted average where S_{ij} is the soft assignment of node i to cluster j learned by GNN:

$$\mathrm{c}_j = \frac{\sum_{i=1}^{N} S_{ij}\mathrm{x}_i}{\sum_{i=1}^{N} S_{ij}} \tag{5}$$

2.4. Loss function

Our loss has two main components that are summed together with weights (which are hyperparameters):

1. Physics Loss: A mean-squared error comparing the coarsened-grid simulation at measurement points to the ground truth:

$$L_{\text{physics}} = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{M}\sum_{s=1}^{M}\left(p_s(t) - p_s^*(t)\right)^2\right)^{1/2} \tag{6}$$

2. Stability Loss (for Explicit Solver): When using an explicit time integrator, we include a term that penalizes very small cells or high local ratios. This term is inspired by the sufficient condition for stability of the forward Euler scheme and modified for enabling its effective optimization:

$$L_{\text{stability}} = -\frac{4}{C}\frac{nV_{min}}{|V|}. \tag{7}$$

   The actual stability loss is computed as a sigmoid function applied to the scaled stability loss: sigmoid weight × stability loss. This ensures that the stability loss is bounded between 0 and 1, preventing extreme gradients during optimization.

Here $p_s(t)$ and $p_s^*(t)$ are the time series for variable u (pressure) at a sensor point s modeled on initial grid S and coarsened grid S^*. The sensor point notation is the same on both grids because these points remain untouched by the coarsening procedure. Also, M is the number of sensors, T is the number of timesteps, V_{min} is the smallest cell area, $|V|$ is the area of domain, n is the number of points in coarsened grid and C is a dimensionless constant, which depends on the geometry of the grid.

During training, we first warm up with the stability loss alone (during the first 50 epochs) and then jointly optimize both terms. For implicit solver stability loss is not used due to greatly reduced stability restrictions.

3. Experiments and results

We tested the approach on 2D grids for the subsurface flow on the "*loop*" scenario and "*sinusoidal permeability scenario*" and also made a small generalizability test for the wave equation.

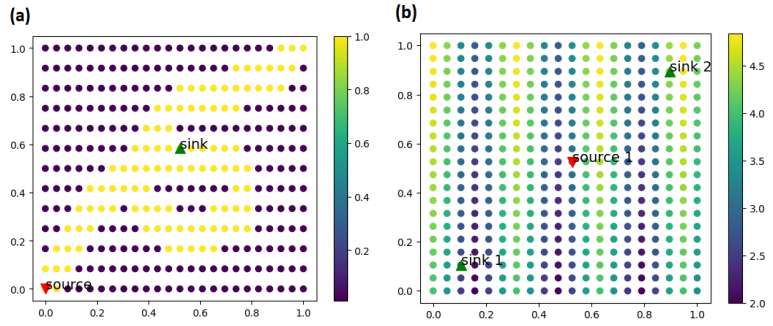

Figure 2. Permeability fields : (a) ”loop” scenario. (b) sinusoidal permeability scenario.

Loop scenario (diffusion equation). As a first scenario, we use the configuration representing the form of “loop”. This scenario represents an evenly spaced point cloud with 312 points where the discrete permeability field K has values 0.1,1.

Sinusoidal permeability scenario (diffusion equation). The 2D domain is set to a unit square with side lengths of $l = 1$. The grid is initialized with a resolution of 20×20 points, resulting in a total of 400 grid points. Each grid point has coordinates (x,y) and the permeability *K(x,y)* is generated using a sinusoidal function that varies spatially across the grid. Specifically, the permeability field *K(x,y)* is defined as $K(x, y) = \cos(20x) + \sin(y) + shift$ where last term represents shift needed for positivity of permeability: $shift = 2.0 + \text{abs}(\min(K(x, y)))$.

Our method is compared with: a naive k-means-based coarsening (features averaged per cluster), a global optimization approach that repositions grid points [9], METIS partitioning [10] and a data-driven surrogate method (DMD) [11] for future prediction.

In a "loop" scenario with piecewise-constant permeability, we reduce the grid size by up to 90-95% while maintaining comparable accuracy to our previous global optimization approach [9]. We calculate RMSE on sink (or measurement) points between dynamics on a coarsened grid and fine grid.

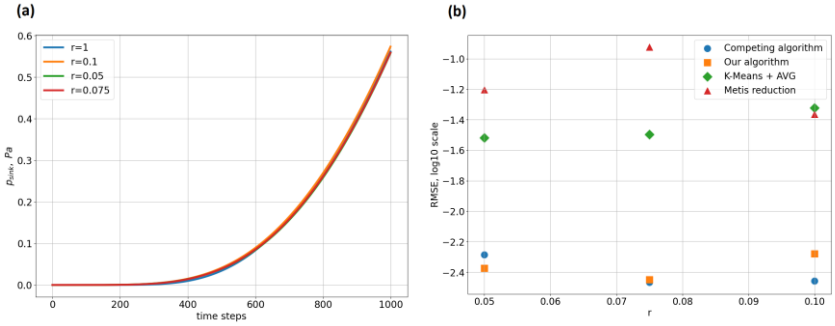

Figure 3. Results for the loop scenario: (a) pressure curves at different reduction levels (b) comparison of RMSE with other baselines.

Similarly, in a sinusoidal permeability scenario, we again observe that the coarsened solutions remain close to the reference solutions, outperforming naive or purely geometric methods:

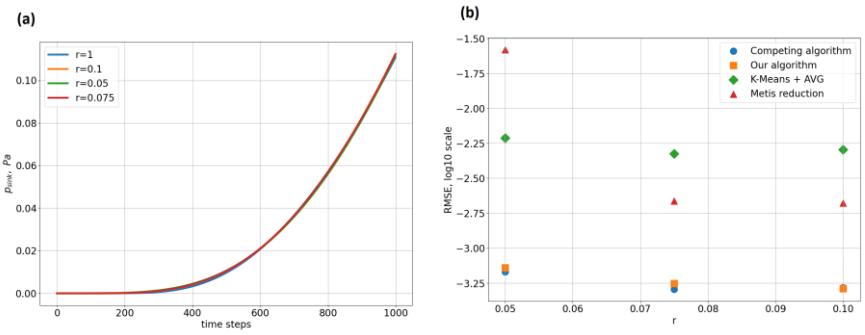

Figure 4. Results for sinusoidal scenario: (a) pressure curves at different reduction levels (b) comparison of RMSE with other baselines.

In a "loop" scenario with piecewise-constant permeability, we reduce the grid size by up to $90-95\%$ while maintaining comparable accuracy to the previous global optimization approach [9].

DMD has a different nature and is aimed at future prediction. We compared our algorithm with DMD in future prediction scenario and have shown that our algorithm demonstrates superior performance:

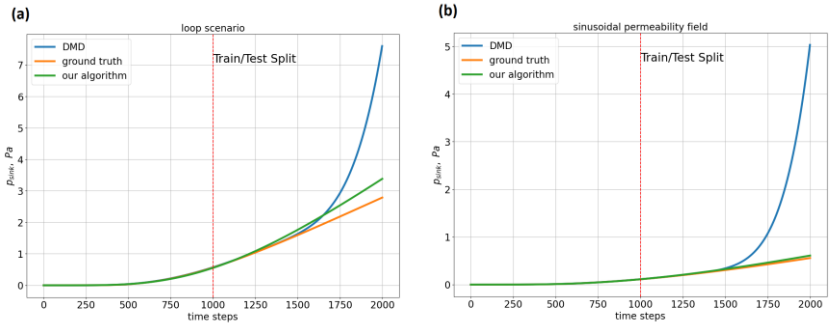

Figure 5. Comparison with DMD: (a) “loop” scenario (b) Sinusoidal scenario.

Additionally, we analyzed our model for coarsening level 7.5% for loop scenario on inference with different sinks positions (different from train sample). In the first variant the new position of sink were (0.48,0.58), in second - (0.39,0.5), and in third - (0.22,0.33) (original position was (0.52,0.58)):

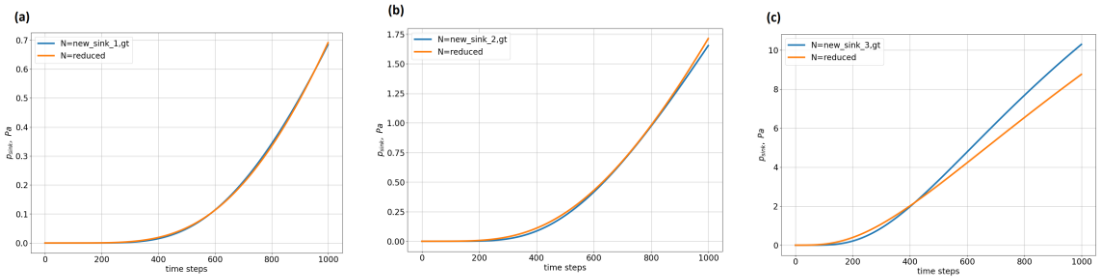

Figure 6. Comparison of pressure curves for different locations of our new sinks: a) result for first sink, b) result for second sink, c) result for third sink.

The results indicate that without retraining our model we achieve accurate pressure results on the coarsened grid (only the last far point leads to noticeable discrepancy). It is important that a competing algorithm needs retraining for every new grid.

Moreover, we demonstrated that our approach also can be used with implicit solver, for diffusion equation [Link to the video] for the reduced simulation time and stable rollouts over many timesteps. Also to study the approach generalizability to other fluid dynamics equations we also tested it on the wave equation (5) [Link to the video].

4. Conclusions

We introduced a learnable GNN-based coarsening framework for physics simulations on unstructured grids. By integrating a differentiable finite volume solver into the training loop, our method ensures consistency with the governing PDEs, reducing node count significantly without sacrificing accuracy. The inclusion of a stability term helps to avoid divergence in explicit schemes, while switching to an implicit solver naturally addresses stability concerns. Simulation results on the subsurface flow problem confirm that the approach consistently produces high-fidelity solutions on coarsened meshes and generalizes to new conditions. Future directions include extending to 3D domains, exploring advanced PDE systems (e.g., multiphase flows or wave equation with nonlinear terms or directly the Navier-Stokes equation), and incorporating more sophisticated graph architectures.

Acknowledgments

The authors are grateful to the support of the Russian Science Foundation (Grant No. 24-41-02035).

References

1. Gerritsen, M. G. and Durlofsky, L. J. Modeling fluid flow in oil reservoirs. Annu. Rev. Fluid Mech., 37(1):211–238, 2005.
2. Huyakorn, P. S. Computational methods in subsurface flow. academic press, 2012.
3. Qi, D. and Hesketh, T. An analysis of upscaling techniques for reservoir simulation. Petroleum Science and Technology, 23(7–8):827–842, July 2005. ISSN 1532-2459. doi: 10.1081/lft-200033132.
4. Guo, M. and Hesthaven, J. S. Data-driven reduced order modeling for time-dependent problems. Computer Methods in Applied Mechanics and Engineering, 345:75–99, March 2019. ISSN 0045-7825. doi: 10.1016/j.cma.2018. 10.029. URL http://dx.doi.org/10.1016/j. cma.2018.10.029.
5. Alet, F., Jeewajee, A. K., Villalonga, M. B., Rodriguez, A., Lozano-Perez, T., and Kaelbling, L. Graph element networks: adaptive, structured computation and memory. In International Conference on Machine Learning, pp. 212–222. PMLR, 2019.
6. Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. arXiv preprint arXiv:2010.03409, 2020.
7. Shumilin, S., Ryabov, A., Barannikov, S., Burnaev, E., & Vanovskii, V. (2023). A Method for Auto-Differentiation of the Voronoi Tessellation. *arXiv preprint arXiv:2312.16192*.
8. Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., et al. Jax: composable transformations of python+ numpy programs. 2018b.
9. Shumilin, S., Ryabov, A., Yavich, N., Burnaev, E., and Vanovskiy, V. Self-supervised coarsening of unstructured grid with automatic differentiation. In Forty-first International Conference on Machine Learning, 2024.
10. Karypis, G. and Kumar, V. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on scientific Computing, 20(1):359–392, 1998.
11. Schmid, P. J. Dynamic mode decomposition of numerical and experimental data. Journal of fluid mechanics, 656: 5–28, 2010.