Algebraic multiscale grid coarsening using unsupervised machine learning for subsurface flow simulation

Ramesh Kumar, Kishan; Tene, Matei

DOI
10.1016/j.jcp.2023.112570

Publication date
2023

Document Version
Final published version

Published in
Journal of Computational Physics

Citation (APA)
Ramesh Kumar, K., & Tene, M. (2023). Algebraic multiscale grid coarsening using unsupervised machine learning for subsurface flow simulation. *Journal of Computational Physics*, *496*, Article 112570. https://doi.org/10.1016/j.jcp.2023.112570

J. Comput. Phys. 496 (2024) 112570

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Short note

Algebraic multiscale grid coarsening using unsupervised machine learning for subsurface flow simulation

Kishan Ramesh Kumar [a,*], Matei Tene [b]

[a] Faculty of Civil Engineering and Geosciences, Department of Geoscience and Engineering, Delft University of Technology, Stevinweg 1, 2628CV, Delft, the Netherlands

[b] SLB Norway Technology Center Fornebuveien 3, 1366 Lysaker, Norway

ARTICLE INFO

Keywords:
Reservoir simulation
Unsupervised learning
Graph-based partitioning
Computational performance
Algebraic multiscale methods

ABSTRACT

Subsurface flow simulation is vital for many geoscience applications, including geoenergy extraction and gas (energy) storage. Reservoirs are often highly heterogeneous and naturally fractured. Therefore, scalable simulation strategies are crucial to enable efficient and reliable operational strategies. One of these scalable methods, which has also been recently deployed in commercial reservoir simulators, is algebraic multiscale (AMS) solvers. AMS, like all multilevel schemes, is found to be highly sensitive to the types (geometries and size) of coarse grids and local basis functions. Commercial simulators benefit from a graph-based partitioner; e.g., METIS to generate the multiscale coarse grids. METIS minimizes the amount of interfaces between coarse partitions, while keeping them of similar size which may not be the requirement to create a coarse grid. In this work, we employ a novel approach to generate the multiscale coarse grids, using unsupervised learning methods which is based on optimizing different parameter. We specifically use the Louvain algorithm and Multi-level Markov clustering. The Louvain algorithm optimizes modularity, a measure of the strength of network division while Markov clustering simulates random walks between the cells to find clusters. It is found that the AMS performance is improved when compared with the existing METIS-based partitioner on several field-scale test cases. This development has the potential to enable reservoir engineers to run ensembles of thousands of detailed models at a much faster rate.

1. Introduction

One of the main challenges in computational geosciences for flow and transport simulations is that the reservoir is highly heterogeneous and naturally fractured. This will become computationally very expensive to solve at the fine (high-resolution) scale, because of the large number of cells and nonlinearities to be resolved. In addition, because of the uncertainties involved in geoscience applications, several forward simulations have to be run on many possible realizations [1]. As such, advanced simulation methods are crucial to enable safe, efficient and reliable operational strategies at field-scales. To address this, algebraic multi-scale methods [2–8] were developed and evolved in the past decade to accelerate both academic and commercial reservoir simulators. Their performance, however, is highly dependent on the choice of local "basis functions", used to map between the fine- and coarser-scale systems [9]. These basis functions form partitions of unity, and are enriched once fractures are present [10].

* Corresponding author.
E-mail addresses: k.rameshkumar-2@tudelft.nl (K. Ramesh Kumar), matei.tene@slb.com (M. Tene).

https://doi.org/10.1016/j.jcp.2023.112570
Received 30 March 2023; Received in revised form 27 August 2023; Accepted 13 October 2023

Available online 24 October 2023

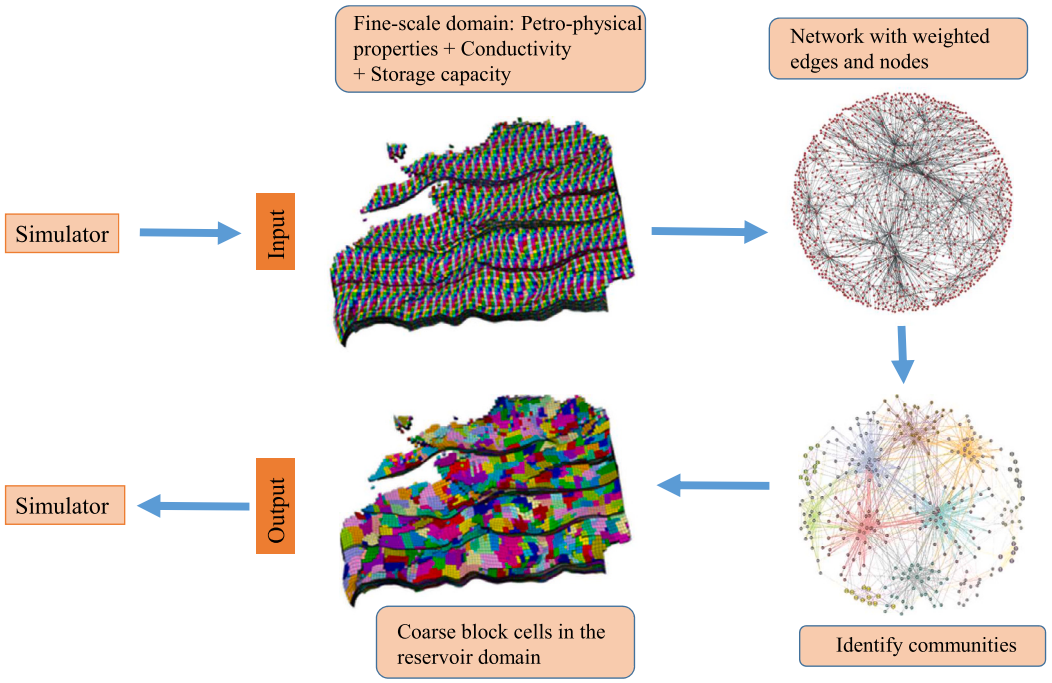


Fig. 1. Schematic of graph clustering in reservoir simulation using algebraic multiscale methods.

In the recent era, machine-learning methods are being widely used in the reservoir flow modeling applications like using smart finite elements [11] or employing a physics-informed neural network to model the Buckley-Leverett problem to understand the drainage of gas into a water-filled porous medium [12]. To reduce the computational time and complexity of the system, several multiscale methods with different definitions are prevalently, such as upscaling material properties [13], algebraic multiscale methods [4] or FE2 methods for composites [14]. Few authors in recent years have used machine learning in the area of algebraic multiscale or multigrid methods [15–20]. A deep neural network was used in the geometric multigrid method, by considering every operation as a network layer [19]. Recently, a graph-based neural network (GNN) was employed to train algebraic multi-grid prolongation operator for linear systems using a single GNN [21], reinforcement learning [22] and encoder-decoder-based idea of pooling to coarsen the mesh graph [23]. A surrogate model was also developed to generate basis functions in the multiscale solver [24].

Based on the literature, most of the work is focused on employing machine learning to improve prolongation operators or optimize the AMG solvers. However, enhancing the solver performance using machine learning to find the location of the coarse grid cells in the reservoir domain was never attempted. Traditionally, commercial simulators, use the graph cut methods such as METIS package [25] to partition the fine-scaled domain into a coarse-scale system based on the connectivity of the cells. The improvement in the performance of the partitioning can be indirectly reflected in reducing the total number of Newton iterations in each timestep. The METIS package minimizes the number of edges between partitions (coarse block cells) while keeping the partitions of similar size. This is an important consideration for parallel computing; however, it might not be important for a multiscale sequentially fully implicit (SFI) solver. To tackle this, in this work, unsupervised graph-based learning algorithms are employed to cluster the fine-scale domain into a coarse-scale system. The schematic of the approach is shown in Fig. 1.

The data, which are stored in the reservoir, are translated to a graph which captures the connectivity between the cells into the edge weights of the graph (transmissibility) and the node weights (pore volume). Using graph clustering, communities are detected and further translated to the coarse grid. Graph clustering has been extensively used in the past to find communities based on connectivity between the nodes or edge weights and also node weights for different applications such as image segmentation [26], load balancing in parallel computing, very large scale integration systems (VLSI) [25], genetic mapping [27,28] and social networks [29].

Popular graph-based clustering algorithms include spectral clustering [30–32], hierarchical clustering [33], modularity-based methods [34,35] and neural-network-based methods [36]. The spectral algorithm involves the computation of eigenvalues and vectors with approximate methods such as the Nystrom and Lanczos algorithms. The obtained eigenvectors can be sensitive to the approximation error that contributes to the poor quality of the results and does not scale well to large test cases. Neural-network-based algorithms need extensive training and testing data, which is not appropriate in this application mainly because the performance is case dependent due to high heterogeneity and insufficient data available to train the network.

In this work, the two chosen advanced algorithms to compare with METIS are the Louvain algorithm [35] and Multi-level Regularized Markov clustering (MLR-MCL) [37]. The Louvain algorithm is based on modularity optimization while the Markov clustering algorithm is iteratively employed until its stochastic matrix is converged. The Louvain algorithm, a recursive method that builds on previous work [38–42] and variants of Markov clustering algorithms based on random walk [43–45] have been previously

implemented in different applications extensively. This work presents the first attempt to employ unsupervised clustering algorithms within algebraic multiscale methods to accelerate reservoir simulation. This is different from traditional surrogate models or neural-network-based models that generally require a lot of training and testing data to obtain a generic model suitable to all simulation cases.

The paper is organized in the subsequent manner. Initially, a brief explanation is given about the mass conservation system used to calculate pressure, as well as the algebraic multiscale solver utilized in this study. Following that, the graph-based clustering methods are introduced, providing detailed explanations of two employed algorithms: Louvain and Markov clustering. Various numerical experiments are conducted, gradually escalating in complexity, where these two algorithms are employed. The performance of the solver is then compared to the established METIS package. Lastly, the paper concludes with final remarks.

2. Governing equations

In this work, the conservation of mass equation for single-phase incompressible flow is solved to compute fluid pressure across the reservoir. It reads as

$$-\nabla(\lambda \cdot \nabla p) = q + \nabla(\rho g \lambda \cdot \nabla h). \tag{1}$$

Here p is the pore pressure, λ is the positive definite mobility tensor, q is the source term, g is the gravitational acceleration, ∇h is the depth and ρ is the density of the fluid. The conservation of mass equation is solved to compute the pressure in the reservoir domain as dictated by the location of the wells.

Applying finite volume (FV) discretization scheme and integrating the mass conservation equation over a control volume, the fine-scale FV system can be used to solve for incremental pressure (Δp). In the residual format, it can be written as

$$\boldsymbol{J}\Delta p = \boldsymbol{r} \tag{2}$$

To improve the computational efficiency and solve for larger testcases, an algebraic multiscale system is employed as presented in the next section.

3. Multiscale solver

In this section, the AMS method which was extensively researched in the past, is briefly elaborated. Recently, the AMS method was also released in the commercial simulation space [46,47,7]. The approximate incremental fine-scale pressure p_f can be written in terms of incremental coarse-scale pressure p_c using the prolongation operator $\boldsymbol{P}$ as

$$\Delta p_f = \boldsymbol{P}\Delta p_c. \tag{3}$$

The prolongation operator is $\boldsymbol{P} = [N_1, N_2 ... N_H]$ where the number H is the number of coarse blocks in the domain (user defined) and p_f is the approximate fine-scale pressure solution obtained from AMS. This number and locations of the coarse grids are obtained from unsupervised learning methods in this work. The pressure increment is computed as

$$(\boldsymbol{R}\boldsymbol{J}\boldsymbol{P})\Delta p_c = \boldsymbol{R}r. \tag{4}$$

Here $\boldsymbol{R}$ is the restriction operator. The prolongation operator is computed from the finite volume Restriction matrix $\boldsymbol{R}_{FV}$ that is given by

$$\boldsymbol{R}_{FV} = \begin{cases} 1, & \text{if } x_i \in \Omega_j^b \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Here Ω_j^b is the jth coarse grid. The prolongation operator ($\boldsymbol{P}$) is computed iteratively by employing steady-state formulation which is computed from the Jacobians. Initially the prolongation operator is set as $\boldsymbol{P}^0 = \boldsymbol{R}^T$, which is further modified iteratively as

$$\boldsymbol{P}^{n+1} = \boldsymbol{P}^n - \mathcal{M}(\omega \boldsymbol{D}_b^{-1} A_b \boldsymbol{P}^n). \tag{6}$$

Here A_b is computed by regularizing the fluxes over the interface which is given by

$$\boldsymbol{A_b} = \frac{1}{2}\left\{ \boldsymbol{J} + \boldsymbol{J}^T - \text{diag}[(\boldsymbol{J} + \boldsymbol{J}^T)\boldsymbol{1}]. \right\} \tag{7}$$

Here $\boldsymbol{D}_b$ is the diagonal of A_b, ω is the relaxation factor, $\mathcal{M}$ is the function which modifies the incremental pressure ensuring that the basis function is limited to the support region. Once the prolongation operator is computed, the restriction operator is defined as $\boldsymbol{R} = \boldsymbol{P}^{-1}$ similar to finite element multiscale method. The detailed procedure can be found in [48,7,49].

For all the testcases, clusters were employed in the multiscale solver. Each cluster of cells forms a coarse block and will possess a corresponding basis function centered within it. The support of the basis function is the coarse block and includes a “halo” consisting of three layers of neighboring cells surrounding it. This halo overlaps with the support areas of adjacent basis functions, resulting in off-diagonal elements within the coarse-scale system. The thicker the halo, the bigger the support and overlap, consequently increasing the density of the coarse-scale stencil. While a dense stencil might offer a more accurate portrayal of the pressure problem’s

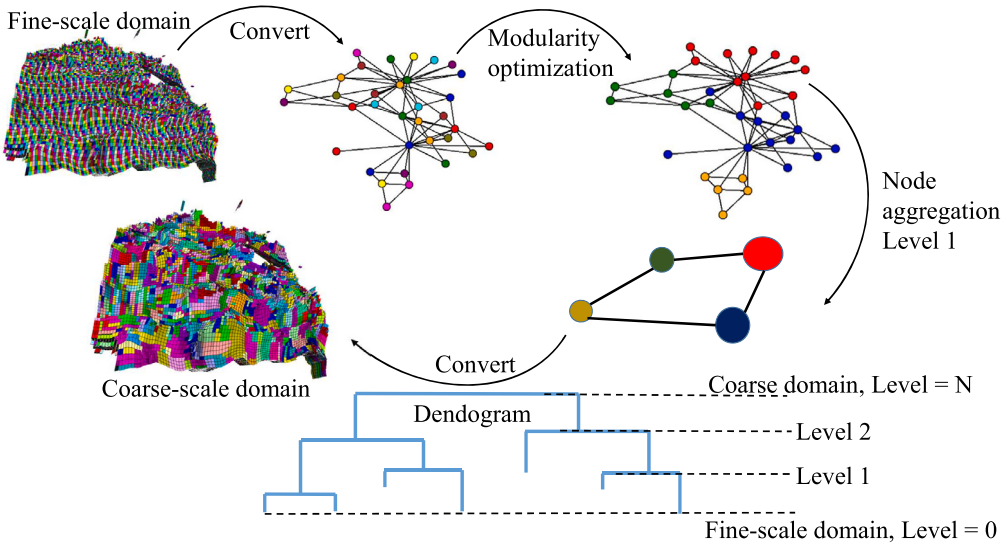


Fig. 2. Illustration of Louvain algorithm.

global nature, it can also lead to solving a more difficult rigid coarse linear system. The computation of the basis functions follows the iterative MsRSB procedure [48].

In the next section, graph clustering algorithms employed to generate the multiscale grid are elaborated.

4. Graph clustering algorithms

Graph clustering is a technique in data analysis and machine learning that involves grouping similar nodes in a graph into clusters. A graph is a mathematical representation of a network, consisting of a set of nodes and edges connecting them. Clustering allows us to identify groups of nodes that are densely connected to each other within the graph, while being sparsely connected to nodes outside the group. In this work the Louvain algorithm and MLR-MCL have been employed and are elaborated below.

4.1. Louvain algorithm

The Louvain algorithm is based on optimizing a parameter called modularity. It is suitable for directed and weighted graphs. It is already a proven scalable algorithm, which can cluster millions of nodes within seconds. Fig. 2 shows the schematic of the Louvain algorithm applied in multiscale methods.

Initially, the fine-scale domain is converted to a weighted and directed graph. Then, based on modularity optimization, the initial set of similar nodes are identified. Modularity is expressed as [50]

$$M = \frac{1}{2m}\sum_{i,j}[A_{ij} - \frac{t_i t_j}{2m}]\delta(v_i, v_j) \tag{8}$$

v_i is the community to which vertex i is assigned, $\delta(v_i, v_j) = 1$ when $v_i = v_j$ else $\delta(v_i, v_j) = 0$. Here t_i, t_j are the sum of weights of edges attached to the nodes of i and j, A_{ij} represents the weight of the edge between i and j nodes, $m = \frac{1}{2}\sum_{ij} A_{ij}$. The range of modularity is between 0 and 1. After modularity optimization, the nodes are aggregated to form a new cluster that results in a network with aggregated nodes. The pseudo-code of this algorithm can be seen in [35].

The Louvain algorithm outputs the number of clusters at each level of the dendogram. There are several parameters, such as the optimization tolerance, which is the minimum increase to enter a new modularity optimization, the number of aggregations, which is the level of the dendogram at which the user is requesting. These parameters affect the number and shape of the clusters. Depending on these parameters for each level of dendogram, different numbers of clusters are output. Accordingly, the user chooses the dendogram level to obtain a certain number of clusters. Once the number of clusters are output, the cluster number and the number of cells in the respective cluster will be employed to solve the algebraic multiscale system. In this work, the Louvain algorithm package deployed in the scikit network was employed [35,51,52]. The default values of parameters in these packages were reasonable to get a good estimate of the number of clusters. The edge and node weights employed for this algorithm are transmissibility and porosity, respectively. To accommodate the effect of transmissibility and porosity equally, all the weights are normalized.

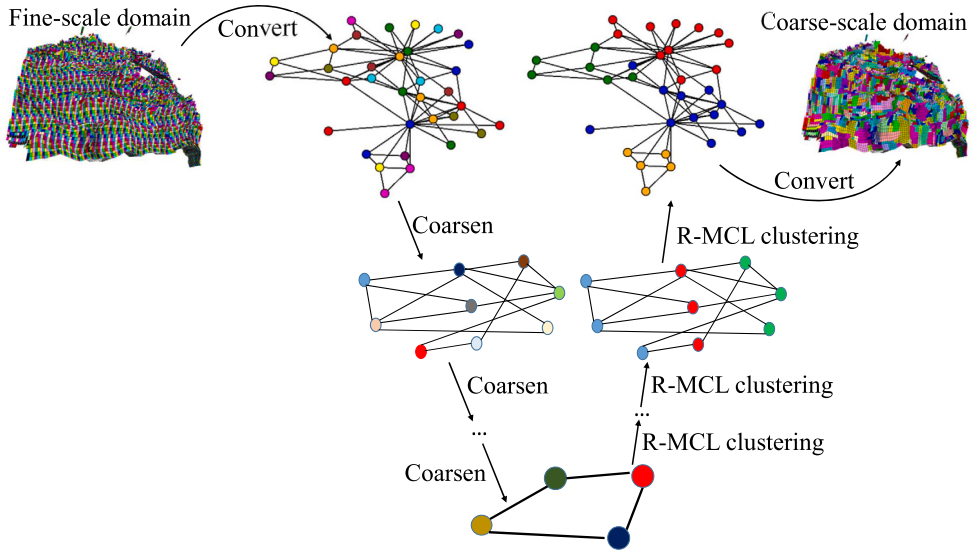


Fig. 3. Schematic of MLR-MCL algorithm.

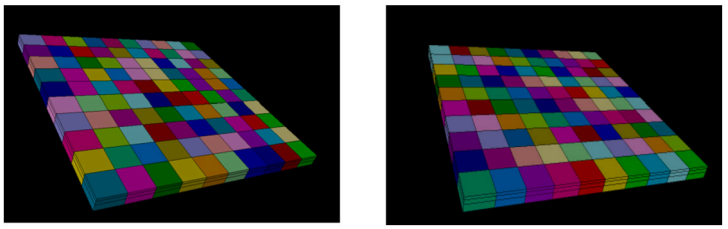


Fig. 4. Coarse grid obtained from METIS (left) and MLR-MCL (right) simulation for this test case. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

4.2. MLR-MCL algorithm

MLR-MCL is the other method that was employed in this study, which uses expansion and inflation operators. The method performs deterministic random walks through the network graph using the two operators by converting one set to another set to identify the nodes that belong in the same cluster. In this work, an extension of Markov clustering which is Regularized multi-level Markov clustering as employed by [37] is used. The schematic can be seen in Fig. 3.

The graph based representation of the fine-scale domain is coarsened using edge-matching techniques, then the coarsest graph is obtained. By employing Markov clustering on the coarse graphs, aggregated clustered graphs are identified. Further, depending on the user conditions, the number of clusters are obtained. This is further translated to the coarse-scale domain. This algorithm is applicable to undirected and unweighted graphs. This algorithm is highly scalable which benefits the commercial solvers usability.

5. Numerical experiments

In this section, METIS, MLR-MCL and Louvain algorithm are compared based on the performance of the simulator. Pressure iterations of the linear solver and the time taken to solve for the pressure solution (t_p) are the two parameters chosen for comparison. In the increasing level of complexity, the following test cases are ordered.

5.1. Test case A

This is a small homogeneous test case that has 300 fine grid cells and 100 coarse blocks from the METIS algorithm. Louvain clustering and Markov clustering also output 100 coarse blocks with similar coarse grid structure as seen in Fig. 4. The performance of the solver for different algorithms is shown in Table 1. Here the coarse blocks are grouped vertically (z direction) because of more dominant connections compared to the horizontal ones. In the vertical (z) direction, there are 3 cells, while in the horizontal (x and y) directions, there are 10x10 cells each. The various colors within the grid indicate distinct coarse cells that have been clustered in the z direction. This implies that each coarse block cell consists of 3 finer cells. Upon visual inspection, it's evident that the quantity of coarse block cells is consistent across all three algorithms, and they are also arranged in the same manner, aligning with the z-direction. The computational performance and the time shown by all the algorithms are similar.

Table 1
Comparison of solver performance for different algorithms with number of fine cells = 300 cells. N_c is the number of coarse cells and t_p is the time taken to solve for pressure solution.

Algorithm	Linear pressure iterations	t_p [s]	N_c
METIS	854	0.7	100
Louvain	867	0.6	100
MLR-MCL	867	0.6	100

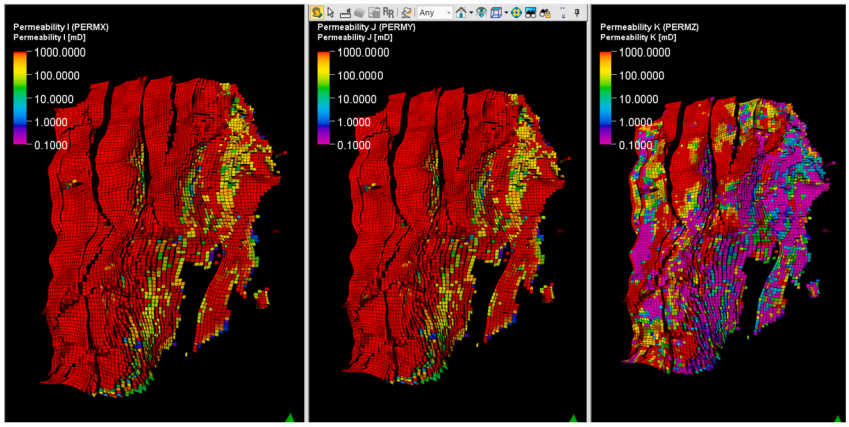


Fig. 5. Permeability fields in x, y and z directions from left to right respectively for this testcase.

Table 2
Comparison of solver performance for different algorithms with number of fine cells = 79423. (EW: Edge weight, NW: Node weight). Here M is modularity. Lowest linear pressure iterations and least pressure solution time is bolded.

Algorithm	Linear pressure iterations	t_p [s]	N_c
METIS	**507**	35.1	4019
METIS (+EW)	531	38.7	4700
METIS (+EW + NW)	541	35.4	4200
Louvain [$M = 0.67$]	654	**32.8**	3028
Louvain (+EW) [$M = 0.965$]	786	38	3516
Louvain (+EW + NW) [$M = 0.978$]	693	34.2	3206
MLR-MCL	580	37.8	4415

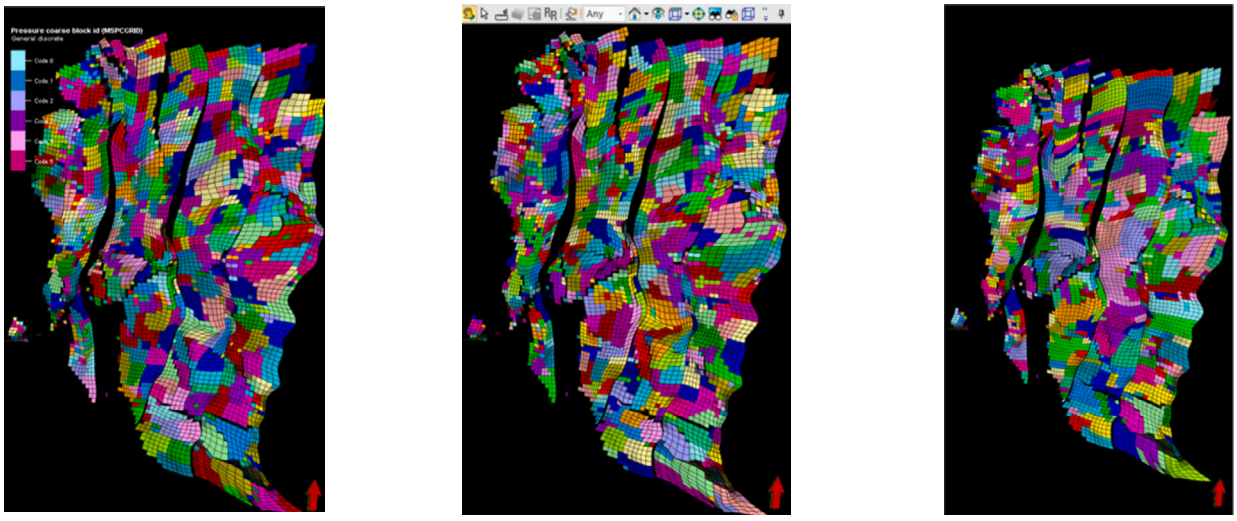
The effect of adding weights to METIS and Louvain algorithms did not show any improvement in the performance of the solver because of a highly simplified testcase. This simple testcase builds confidence in the employed algorithms because of similar computational performance compared to traditional algorithm METIS.

5.2. Test case B

The reservoir is medium sized, geometrically complex and heterogeneous in nature in this testcase. The number of fine-cells is 79423 and the number of coarse block cells are slightly different for different algorithms. The permeability fields in x, y and z direction are shown in Fig. 5. The range of permeability is between 0.1 mD to 1000 mD with relatively low permeability along the z direction compared to the other two directions. The permeability range is similar for x and y direction. The coarse blocks obtained from the three algorithms are shown in Fig. 6. The performance of the solver and the number of coarse blocks for different algorithms are shown in Table 2.

The coarse blocks as shown in Fig. 6 depict the grids for METIS, Markov clustering and Louvain clustering. The coarse grids look similar for METIS and MLR-MCL; however, the coarse grids for Louvain algorithm show nonuniform-sized coarse blocks with different numbers of fine cells. The METIS algorithm was found to have the fewest linear pressure iterations, but the Louvain algorithm was found to have the least pressure solution time. This means that the pressure solver performance was improved. The number of coarse blocks for these algorithms are slightly different, which could be causing the change in the performance of the pressure solver.

To gain a deeper insight into the distinctions within the coarse grid arrangement of these algorithms, a statistical assessment is conducted across these datasets. This comparison is visualized through violin plots, depicted in Fig. 7. These visualizations illustrate how the distribution of fine-scale cells varies for METIS when considering weight inclusion, for Louvain with weight consideration, and for the MLR-MCL algorithm. Within these plots, the median is denoted by a white dot, the mean is represented by a blue line,



(a) METIS (b) Markov clustering (c) Louvain clustering

Fig. 6. Coarse grids obtained from the clustering algorithms for this testcase.

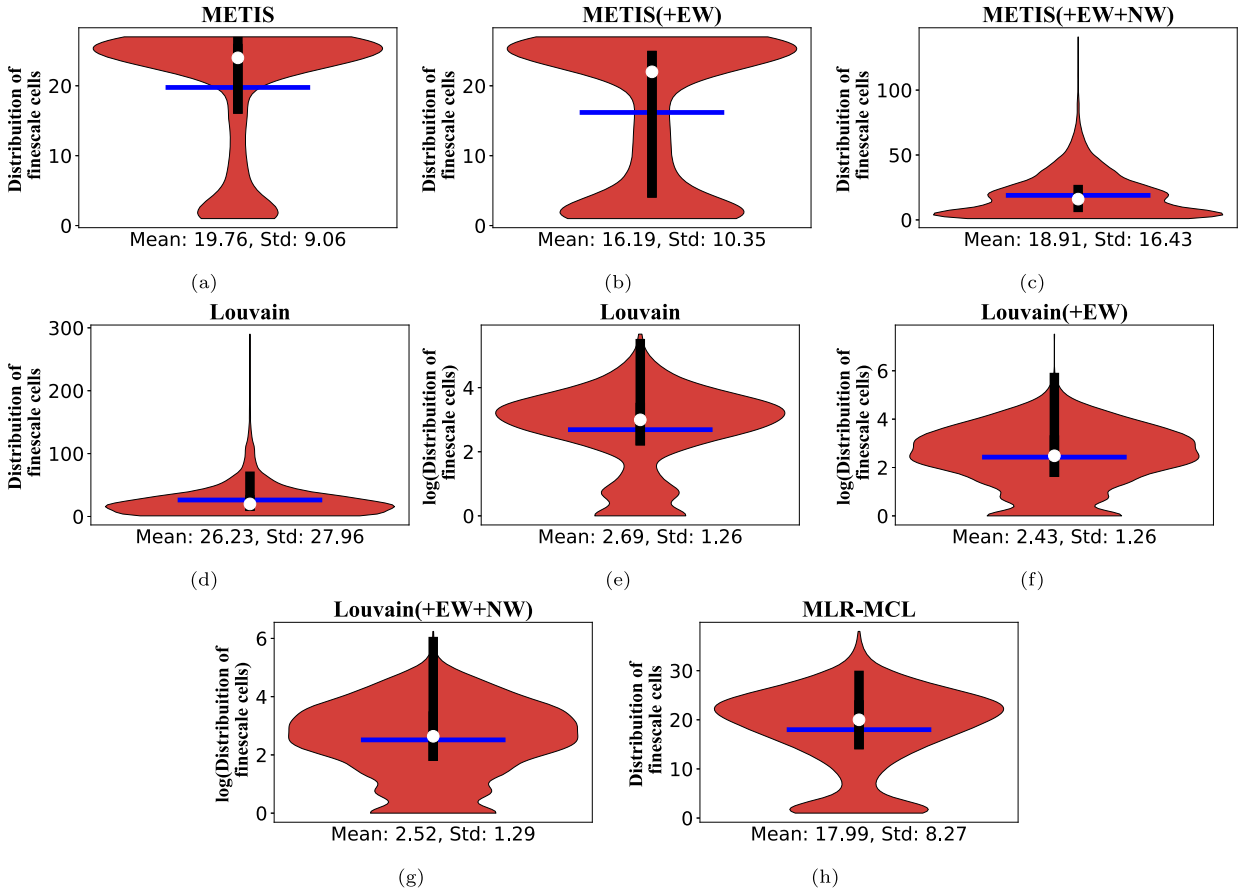
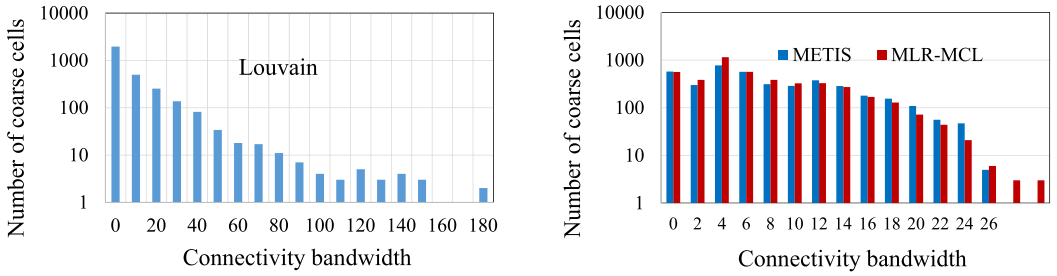


Fig. 7. Violin plots of test case B showing the distribution of finescale cells for each of the algorithm employed. The white dot and the blue horizontal line represents the median and mean respectively. The black box represents 25 to 75 percentile of the data. Figs. 7e, 7f and 7g shows the distribution in log scale for Louvain algorithm.



(a) Louvain algorithm (b) METIS and MLR-MCL algorithm

Fig. 8. Bandwidth of the matrix obtained from the three algorithms.

Table 3
Comparison of solver performance for different algorithms with number of fine cells = 584220. The highlighted numbers show the lowest pressure iterations and least pressure solution time.

Algorithm	Linear pressure iterations	t_p [s]	N_c
METIS	10957	5835.6	11.7 k
Louvain	**8765**	5009.8	12.02 k
MLR-MCL	9317	**4828.8**	12.04 k

and the interquartile range is shown by a black box. Moreover, the plots also display the mean and standard deviation values for each case. As seen in Figs. 7a, 7b, and 7c, these plots display the distribution of fine-scale cells for the METIS algorithm without any weights, with edge weights (EW), and with both edge and node weights respectively.

It's evident that in the cases of METIS and METIS (+EW), the data exhibits a higher-tailed distribution on both sides, with sparser distribution in the central region. Disregarding the outliers, the median (represented by a white dot) indicates that there are roughly 20+ cells in the coarse cell region. When node weights (porosity) are included, the distribution becomes more heavily tailed, with certain outliers showing a high number of fine-scale cells in coarse cells.

Turning to the Louvain algorithm (Fig. 7d), it shows a centered distribution with heavier tails, wherein numerous coarse cells have large number of fine-scale cells compared to other algorithms. To enhance the clarity of this data, a logarithmic distribution of cells is presented, wherein the distribution shows to approximate a normal distribution. This trend remains consistent across all variations of the Louvain algorithm, even when edge and node weights are included. This adherence to log-normal distribution signifies that the algorithm tends to favor large number of fine-scale cells within coarse grids. This is in line with existing literature [53] where modularity-based algorithms favor larger communities. In the case of the MLR-MCL algorithm (Fig. 7h), the distribution appears to be more uniformly distributed. The mean values for METIS and MLR-MCL algorithms are similar with a small standard deviation. However for Louvain algorithm, the mean value and standard deviation values are similar showing the skewness of the data. Another interesting observation is that the standard deviation is least for MLR-MCL algorithm compared to other algorithms which means that the distribution is less spread out and uniform spaced. The data also shows a larger quartile region (data from 25 percentile to 75 percentile) for Louvain algorithm compared to METIS and the least for MLR-MCL algorithm showing the heterogeneous coarsening ratio distribution.

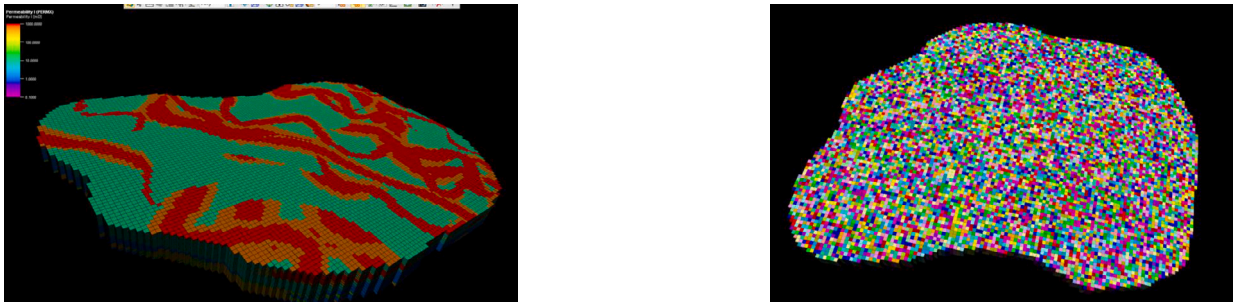
To understand how strong the coupled system is, connectivity bandwidth of the matrix is studied. Connectivity bandwidth means the average number of off-diagonals among the rows. The higher the bandwidth means the stronger is the coupled system. Fig. 8 shows the variation of numbers of coarse cells with the connectivity bandwidth of the algorithms. Fig. 8a shows higher bandwidth compared to the Fig. 8b because the output of the Louvain algorithm shows many large coarse blocks adjacent to the much smaller ones.

The Louvain algorithm optimizes the number of coarse blocks based on the modularity parameter. So based on adding weights (edge and node), the ease at which the clusters could be identified is reflected in parameter modularity. When edge and node weights are accounted for, the modularity of the Louvain algorithm increases as seen in Table 2, which suggests that it is easier to coarsen the computational grid. However, this trend was not observed in the computational solver performance. This could be case dependent or, because the solver is highly complex, the inclusion of weights was not a dominant factor.

5.3. Test case C

This is a medium-sized reservoir with 584220 fine cells and with heterogeneous permeability fields as shown in Fig. 9a. The resulting coarse block from METIS is shown in Fig. 9b. Visually it is not possible to see the difference in the coarse grids obtained from different algorithms. The performance of the solvers is shown in Table 3.

Similar to the previous testcase, the number of linear pressure iterations are least for the Louvain algorithm. The pressure solution is also low for the MLR-MCL and Louvain algorithms compared to the traditional METIS algorithm. The number of coarse blocks are roughly the same obtained from the algorithms. The difference in the number of coarse blocks is less than 3%.



(a) Permeability (b) Coarse block

Fig. 9. Permeability (x direction) and coarse grid obtained from METIS.

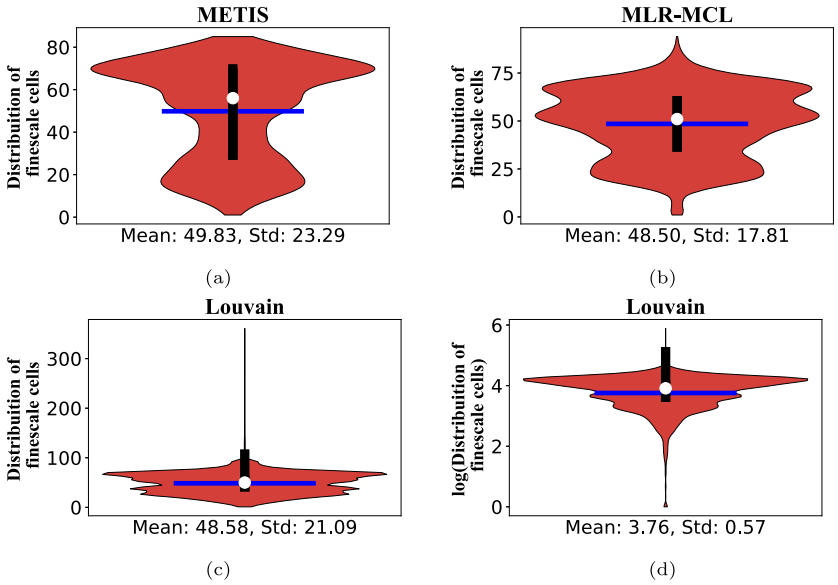


Fig. 10. Violin plots of test case C showing the distribution of finescale cells for each of the algorithm employed. The white dot and the blue horizontal line represents the median and mean respectively. The black box represents 25 to 75 percentile of the data. Fig. 10d shows the distribution in log scale for the Louvain algorithm.

The dissimilarities in the coarse grid structures of each of these algorithms are illustrated through violin plots, as depicted in Fig. 10. In this particular test case, the representation showcases the distribution of fine-scale cells within the coarse grid for METIS without weights, MLR-MCL, the Louvain algorithm, and its logarithmic distribution (no weights), as displayed in Figs. 10a, 10b, 10c, and 10d, respectively.

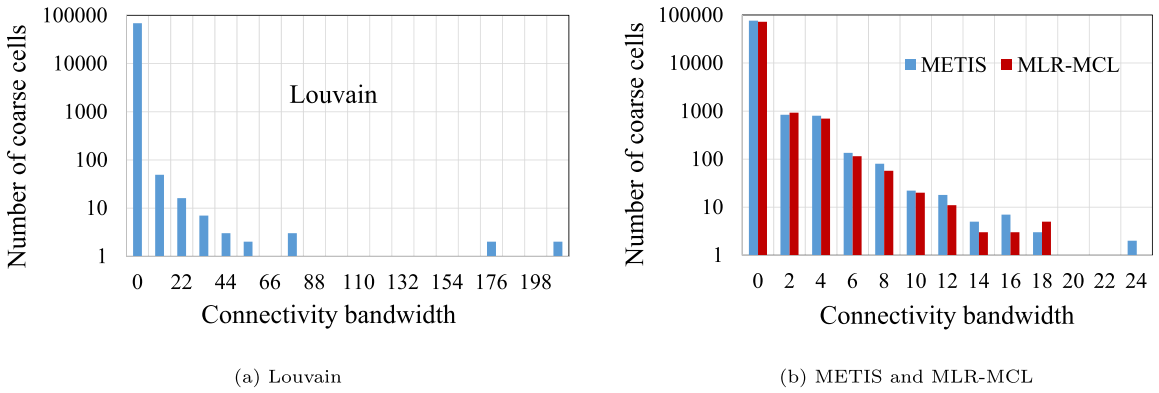
Similar to the prior test case, the Louvain algorithm demonstrates a distribution with heavy tails, revealing a subset of coarse-scale cells containing a considerable number of fine-scale cells. Conversely, the MLR-MCL test case shows a multimodal distribution in contrast to the METIS algorithm, which displays two prominent peaks. However, within this particular test scenario, the mean values were found to be similar. Nonetheless, akin to the preceding test case, the MLR-MCL algorithm exhibits the lowest standard deviation, indicative of a distribution with uniformly spaced characteristics.

Fig. 11 shows the bandwidth obtained from these algorithms. Though the connectivity bandwidth is higher for the Louvain algorithm there are few outliers which are mainly due to few large sized coarse blocks next to small sized coarse blocks.

5.4. Test case D

This is another testcase with 2.36 million cells that was solved for using 32 cores in parallel using openMPI. This is the refined version of the previous reservoir testcase. The performance of the solver is shown in Table 4.

Similar to the previous testcases, though the number of linear pressure iterations are lower for METIS algorithm, lower pressure solution time was observed with Louvain and MLR-MCL algorithms. The lower cost per iteration suggests that the clustering algorithms produce higher quality coarse grids than METIS. The distinctions among these algorithms' coarse grid structures are visualized through violin plots, illustrated in Fig. 12. The distribution is presented by selecting data from a randomly chosen CPU process. In this instance, the distribution of fine-scale cells obtained from the 6th, 15th, and 26th processes is represented by the first, second,



(a) Louvain (b) METIS and MLR-MCL

Fig. 11. Bandwidth of the matrix obtained from the three algorithms.

Table 4
Comparison of solver performance for different algorithms with number of fine cells = 2.36 million.

Algorithm	Linear pressure iterations	t_p [s]	Nc	Elapsed time per iteration [s]
METIS	**10775**	1036.3	77.6 k	0.096
Louvain	12633	**964.65**	69 k	0.0763
MLR-MCL	11808	974.15	74.5 k	0.0824

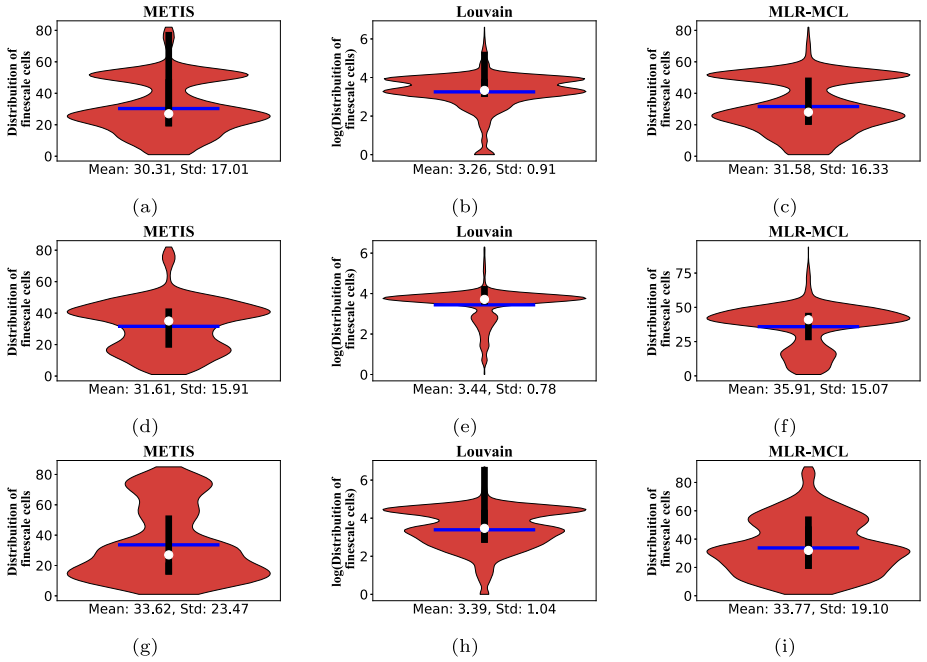


Fig. 12. Violin plots of test case D showing the distribution of finescale cells for each of the algorithm employed. The white dot and the blue horizontal line represents the median and mean respectively. The black box represents 25 to 75 percentile of the data. Figs. 12b, 12e and 12h shows the distribution in log scale for Louvain algorithm.

and third rows respectively, corresponding to the METIS, Louvain, and MLR-MCL algorithms. Similar to previous test cases, the distribution for the Louvain algorithm is displayed in a logarithmic format due to its pronounced one-sided tail. Notably, the standard deviation consistently remains the lowest for the MLR-MCL algorithm similar to previous testcases.

Upon comparing the performance characteristics across all the aforementioned test cases, it becomes evident that, for simpler and smaller test cases (Testcase A and B), the performance of the Louvain or MLR-MCL algorithms is comparable to that of the traditionally employed METIS algorithm. Nevertheless, as the scale of the test cases increases (Testcase C and D), the performance of the MLR-MCL and Louvain algorithms appears to hold a competitive advantage over the METIS algorithm, exhibiting a performance gain of over 8%. This upper hand observed in larger test cases is a consequence of the relatively more pronounced impact of

employing a coarse scale grid. Although this percentage gain might seem modest, it holds significance in the context of commercial software aimed at solving complex problems for extensive test cases. Any enhancement in the performance of solving the coarse system, regardless of its size, is a valuable asset.

It is worth acknowledging that the scalability features and the automatic determination of the coarsening ratio by these algorithms satisfy the requirements of the task at hand. Considering that this marks the inaugural effort to improve the coarse grid structure within an algebraic multiscale algorithm applied to unstructured grids using clustering algorithms, it paves the way for potential avenues of research aimed at optimizing the upscaled grid using machine learning techniques.

6. Conclusion

This work revolves around the novelty of implementing unsupervised learning graph based methods to generate coarse block grids in the algebraic multiscale methods for the first time. This method could also be extended to other upscaling methods such as multigrid methods. Advanced algorithms such as the Louvain and Multi-level Regularized Markov clustering algorithms were implemented into a commercial reservoir simulator to run several test cases. A statistical analysis was also conduced to compare the coarse grid structure for different algorithms. These algorithms were found to be run conveniently on large test cases without any penalty on the computational performance. Few conclusions which are drawn based on this work are

- For small and simpler field test cases, conventional graph cut based algorithm such as METIS showed similar performance as Louvain and MLR-MCL algorithms. However, the unsupervised learning algorithms had an upper edge in terms of computational time for larger and heterogeneous test cases compared to traditional graph cut based METIS algorithm. Based on the violin plots, it could be seen that the Louvain algorithm showed more skewed data implying that the algorithm favors bigger coarsening ratio coarse grid cells and MLR-MCl algorithm showed a more uniform distribution.
- The advantage of these algorithms is that they are highly scalable and automatically determine the number of coarse blocks, eliminating the need for users to input this information as required in the METIS algorithm based commercial solvers. This is particularly beneficial for users who may not be familiar with multiscale methods.
- The impact of adding edge and node weights to the graph was evident in clustering metrics such as modularity and edge cut, showing higher modularity (easier clustering) when weights were included. However, despite this positive correlation, there was no noticeable enhancement in solver performance, possibly due to the heterogeneity of the testcases.
- In general, when dealing with large and heterogeneous test cases, these unsupervised graph-based methods demonstrate superior performance in generating a coarse grid structure compared to the conventional method known as METIS. This is evident in terms of improved computational efficiency.

The forthcoming research will focus on solving for saturation and mechanics in addition to pressure, with the aim of assessing their combined impact on the overall computational performance across various types of test cases.

CRediT authorship contribution statement

Kishan Ramesh Kumar: Conceptualization, Methodology, Software, Writing – original draft. **Matei Tene:** Conceptualization, Software, Supervision, Writing – review & editing.

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.

Data availability

No data was used for the research described in the article.

Acknowledgement

The authors would like to thank all the team members of HPC team, Multiscale team and Machine learning team of SLB for interesting discussions. The authors would like to thank Dr. Hadi Hajibeygi and all the members of the ADMIRE project members at TU Delft, for the fruitful feedback.

References

[1] R.J. de Moraes, J.R. Rodrigues, H. Hajibeygi, J.D. Jansen, Multiscale gradient computation for flow in heterogeneous porous media, J. Comput. Phys. 336 (2017) 644–663.
[2] T. Hou, Y. Efendiev, Multiscale Finite Element Methods, Springer, New York, 2009.
[3] H. Hajibeygi, G. Bonfigli, M.A. Hesse, P. Jenny, Iterative multiscale finite-volume method, J. Comput. Phys. 227 (2008) 8604–8621.
[4] Y. Wang, H. Hajibeygi, H.A. Tchelepi, Algebraic multiscale solver for flow in heterogeneous porous media, J. Comput. Phys. 259 (2014) 284–303.

[5] M. Ţene, M.S.A. Kobaisi, H. Hajibeygi, Algebraic multiscale method for flow in heterogeneous porous media with embedded discrete fractures (f-ams), J. Comput. Phys. 321 (2016) 819–845.
[6] T.Y. Hou, X.-H. Wu, A multiscale finite element method for elliptic problems in composite materials and porous media, J. Comput. Phys. 134 (1997) 169–189.
[7] O. Møyner, K.-A. Lie, A multiscale restriction-smoothed basis method for compressible black-oil models, SPE J. 21 (2016) 2079–2096.
[8] P. Jenny, S.H. Lee, H.A. Tchelepi, Multi-scale finite-volume method for elliptic problems in subsurface flow simulation, J. Comput. Phys. 187 (2003) 47–67.
[9] H. Hajibeygi, Iterative multiscale finite volume method for multiphase flow in porous media with complex physics, Ph.D. thesis, ETH, Zurich, Switzerland, 2011.
[10] M. HosseiniMehr, C. Vuik, H. Hajibeygi, Adaptive dynamic multilevel simulation of fractured geothermal reservoirs, J. Comput. Phys. X 7 (2020) 100061.
[11] G. Capuano, J.J. Rimoli, Smart finite elements: a novel machine learning application, Comput. Methods Appl. Mech. Eng. 345 (2019) 363–381, https://doi.org/10.1016/j.cma.2018.10.046.
[12] M.M. Almajid, M.O. Abu-Al-Saud, Prediction of porous media fluid flow using physics informed neural networks, J. Pet. Sci. Eng. 208 (2022) 109205.
[13] M. Vassaux, K. Gopalakrishnan, R.C. Sinclair, R.A. Richardson, P.V. Coveney, Accelerating heterogeneous multiscale simulations of advanced materials properties with graph-based clustering, Adv. Theory Simul. 4 (2021) 2000234.
[14] F. Feyel, J.-L. Chaboche, Fe2 multiscale approach for modelling the elastoviscoplastic behaviour of long fibre sic/ti composite materials, Comput. Methods Appl. Mech. Eng. 183 (2000) 309–330.
[15] S. Nikolopoulos, I. Kalogeris, V. Papadopoulos, G. Stavroulakis, AI-enhanced iterative solvers for accelerating the solution of large-scale parametrized systems, arXiv:2207.02543, 2022.
[16] T. Louw, S. McIntosh-Smith, Applying recent machine learning approaches to accelerate the algebraic multigrid method for fluid simulations, in: Communications in Computer and Information Science, CCIS, vol. 1512, 2022, pp. 40–57.
[17] V. Fanaskov, Neural multigrid architectures, in: Proceedings of the International Joint Conference on Neural Networks, 2021, July, 2021.
[18] D. Greenfeld, M. Galun, R. Kimmel, I. Yavneh, R. Basri, Learning to optimize multigrid PDE solvers, in: 36th International Conference on Machine Learning, ICML 2019, 2019, June, 2019, pp. 4305–4316.
[19] A. Katrutsa, T. Daulbaev, I. Oseledets, Deep Multigrid: learning prolongation and restriction matrices, arXiv:1711.03825, 2017.
[20] J. Brown, Y. He, S. Maclachlan, M. Menickelly, S.M. Wild, Tuning multigrid methods with robust optimization and local Fourier analysis, SIAM J. Sci. Comput. 43 (2021) 109–138.
[21] I. Luz, M. Galun, H. Maron, R. Basri, I. Yavneh, Learning algebraic multigrid using graph neural networks, in: ICML'20: Proceedings of the 37th International Conference on Machine Learning, 2020, pp. 6489–6499.
[22] A. Taghibakhshi, S. MacLachlan, L. Olson, M. West, Optimization-based algebraic multigrid coarsening using reinforcement learning, Adv. Neural Inf. Process. Syst. 34 (2021) 12129–12140.
[23] Z. Yang, Y. Dong, X. Deng, L. Zhang, AMGNET: multi-scale graph neural networks for flow field prediction, Connect. Sci. 34 (2022) 2500–2519.
[24] S. Chan, A.H. Elsheikh, Data-driven acceleration of multiscale methods for uncertainty quantification: application in transient multiphase flow in porous media, GEM Int. J. Geomath. 11 (2019) 3.
[25] G. Karypis, V. Kumar, S. Comput, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM J. Sci. Comput. 20 (1970).
[26] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000).
[27] J. Cheema, J. Dicks, Computational approaches and software tools for genetic linkage map estimation in plants, Brief. Bioinform. 10 (2009) 595–608.
[28] Uzma, F. Al-Obeidat, A. Tubaishat, B. Shah, Z. Halim, Gene encoder: a feature selection technique through unsupervised deep learning-based clustering for large gene expression data, Neural Comput. Appl. 34 (2022) 8309–8331.
[29] R. Liu, S. Feng, R. Shi, W. Guo, Weighted graph clustering for community detection of large social networks, Proc. Comput. Sci. 31 (2014) 85–94.
[30] S. Sun, J. Zhao, J. Zhu, A review of Nyström methods for large-scale machine learning, Inf. Fusion 26 (2015) 36–48.
[31] A.T. Barker, C.S. Lee, P.S. Vassilevski, Spectral upscaling for graph Laplacian problems with application to reservoir simulation, SIAM J. Sci. Comput. 39 (2017) S323–S346.
[32] A. Choromanska, T. Jebara, H. Kim, M. Mohan, C. Monteleoni, Fast spectral clustering via the Nyström method, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) LNAI, vol. 8139, 2013, pp. 367–381.
[33] M.E. Newman, Fast algorithm for detecting community structure in networks, Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Topics 69 (2004) 5.
[34] A. Clauset, M.E.J. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Topics 70 (2004) 6.
[35] V.D. Blondel, J.L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. Theory Exp. (2008) P10008.
[36] W.L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, C.J. Hsieh, Cluster-GCN: an efficient algorithm for training deep and large graph convolutional networks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019.
[37] V. Satuluri, S. Parthasarathy, Scalable graph clustering using stochastic flows: applications to community discovery, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 737–745.
[38] J.M. Pujol, V. Erramilli, P. Rodriguez, Divide and conquer: partitioning online social networks, arXiv:0905.4918, 2009.
[39] C. Makris, D. Pettas, G. Pispirigos, Distributed community prediction for social graphs based on Louvain algorithm, IFIP Adv. Inf. Commun. Technol. 559 (2019) 500–511.
[40] X. Que, F. Checconi, F. Petrini, J.A. Gunnels, Scalable community detection with the Louvain algorithm, in: Proceedings - 2015 IEEE 29th International Parallel and Distributed Processing Symposium, IPDPS 2015, 2015, pp. 28–37.
[41] Y. Lu, L. Deng, L. Wang, K. Li, J. Wu, Improving metagenome sequence clustering application performance using Louvain algorithm, Commun. Comput. Inf. Sci. 1303 (2020) 386–400.
[42] C. Mao, Three-dimensional tree visualization of computer image data based on Louvain algorithm, Lect. Notes Data Eng. Commun. Technol. 123 (2022) 837–845.
[43] A. Bustamam, V.Y. Nurazmi, D. Lestari, Applications of Cuckoo search optimization algorithm for analyzing protein-protein interaction through Markov clustering on HIV, in: applications of Cuckoo search optimization algorithm for analyzing protein-protein interaction through Markov clustering on HIV, https://doi.org/10.1063/1.5064229, 2023.
[44] L. Gu, Y. Han, C. Wang, W. Chen, J. Jiao, X. Yuan, Module overlapping structure detection in PPI using an improved link similarity-based Markov clustering algorithm, Neural Comput. Appl. 31 (2019) 1481–1490.
[45] A. Bustamam, Fast parallel Markov clustering in bioinformatics using massively parallel computing on GPU with CUDA and ELLPACK-R SparseFormat, IEEE/ACM Trans. Comput. Biol. Bioinform. 9 (2012).
[46] S.K. Khataniar, D. de Brito Dias, R. Xu, Aspects of multiscale flow simulation with potential to enhance reservoir engineering practice, SPE J. 27 (2022) 663–681.
[47] A. Kozlova, Z. Li, J.R. Natvig, S. Watanabe, Y. Zhou, K. Bratvedt, S.H. Lee, A real-field multiscale black-oil reservoir simulator, SPE J. 21 (2016) 2049–2061.
[48] O. Møyner, K.A. Lie, A multiscale method based on restriction-smoothed basis functions suitable for general grids in high contrast media, volume Day 2 Tue, February 24, 2015 of SPE Reservoir Simulation Conference, https://doi.org/SPE-173265-MS, _eprint: https://onepetro.org/spersc/proceedings-pdf/15RSS/2-15RSS/D021S008R002/1466006/spe-173265-ms.pdf, 2015.
[49] O. Møyner, K.-A. Lie, A multiscale restriction-smoothed basis method for high contrast porous media represented on unstructured grids, J. Comput. Phys. 304 (2016) 46–71.
[50] M.E.J. Newman, Analysis of weighted networks, Phys. Rev. E 70 (2004) 056131.
[51] M.J. Barber, Modularity and community detection in bipartite networks, Phys. Rev. E 76 (2007).
[52] N. Dugué, A. Perez, Directed Louvain: maximizing modularity in directed networks, Research Report, Université d'Orléans, 2015.
[53] S. Fortunato, M. Barthélemy, Resolution limit in community detection, Proc. Natl. Acad. Sci. 104 (2007) 36–41.