# 1 Background

Here, we are attempting to predict both aggregates and interpolation for an AMG solver given some matrix $\boldsymbol{A}$. We will be exploring 4 test problems:

1. 1D Poisson with Dirichlet conditions
2. 1D Poisson with Neumann conditions
3. 2D isotropic Poisson with Dirichlet conditions
4. 2D rotated anisotropic Poisson with Dirichlet conditions. This problem is rotated by $\theta = \frac{\pi}{6}$ and with a scaling of 0.001 in the $y$-direction.

# 2 Aggregation and Interpolation Algorithm

To compute the aggregates and final interpolation operator, two individual networks are trained concurrently to perform each action, which we will label $\Theta_{\text{Agg}}$ and $\Theta_{\text{Interp}}$, both taking some weighted graph and returning a new graph with same connectivity, but different weight values. The algorithm for finding the aggregates and interpolation is roughly sketched below:

1. Let $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ be the system we are trying to solve and $\alpha \in (0, 1]$ be some parameter that determines the ratio of aggregates to vertices, i.e. we will be roughly coarsening the graph by $1/\alpha$. Define $k := \lceil \alpha n \rceil$, the number of aggregates we will be outputting.
2. Convolve the graph of $\boldsymbol{A}$ with $\Theta_{\text{Agg}}$ to obtain a new set of node values and edge values. We will use the node values as a *scoring* and the edge values as path weights. Define the aggregate centers as the indices of the largest $k$ node scores. Then, run Bellman-Ford on the graph with these aggregate centers and new edge weights to obtain a tentative aggregation operator, Agg.
3. Now, again convolve the graph of $\boldsymbol{A}$ but with $\Theta_{\text{Interp}}$ (with aggregate information) to obtain the aggregate smoother $\hat{\boldsymbol{P}}$. Form $\boldsymbol{P} := \hat{\boldsymbol{P}}\text{Agg}$.

# 3 Genetic Training

Training of both networks is done at the same time with PyGAD[1], a genetic algorithm that can be used to train networks without any gradient information. A basic overview of the training algorithm is that the method is seeded with some number of randomly generated networks. A subset of the best performing (most fit) networks are selected and "bred" with one another (crossing weights/traits) and "mutations" inserted (random perturbations to weights) to create another population of networks. This is then repeated for many "generations" until a set of hopefully trained networks is obtained, from which we can pick the best fit as our final network.

# 4 Results

Results for selecting aggregates and interpolation are visualized in figures 1 - 4. Aggregates are visualized as colored blobs, aggregate centers are $\star$s, and edge weights for Bellman Ford are shown mapped from dark coloring (low weight) to vivid colors (high weight).

# 5 Future

Perhaps an obvious future direction to explore is to perform an ablation study, comparing results for the following cases:

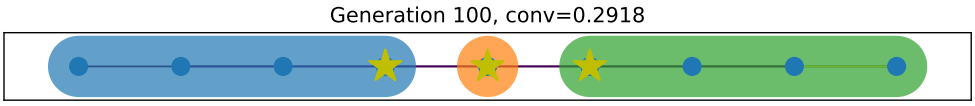
1. Lloyd aggregation and SA interpolation (baseline)

2. ML Aggregation and SA interpolation
3. Lloyd aggregation and ML interpolation (somewhat what was done before)
4. ML Aggregation and interpolation trained together (results from above)
5. ML Aggregation and interpolation trained separately, then combined

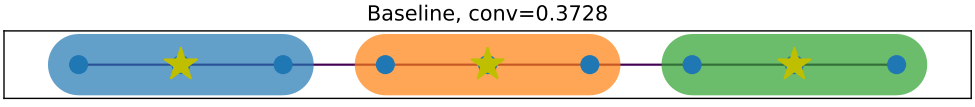
Also, it would likely be worth exploring use of the GA vs. traditional gradient based optimizers. The results above suggest that the GA is able to broadly learn effective networks, and the algorithm was able to effectively learn an interpolation for the 1D Neumann case where previously the gradient optimizer could not. Perhaps a hybrid approach could be employed where the network is first broadly trained with the genetic algorithm then refined with gradient descent.

## References

[1] A. F. Gad, *Pygad: An intuitive genetic algorithm python library*, 2021.

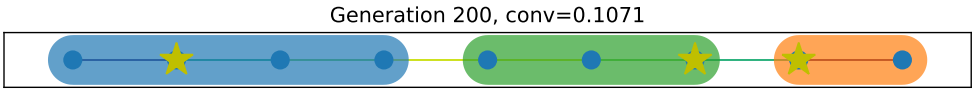



(a) ML Aggregates

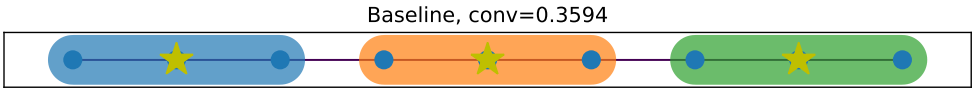



(b) Baseline Aggregates (coarsening by 3)

Figure 1: Aggregates for 1D Dirichlet problem

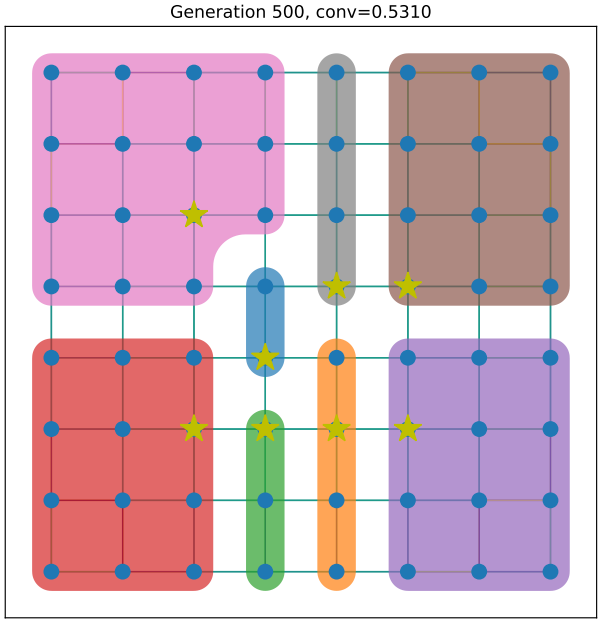



(a) ML Aggregates

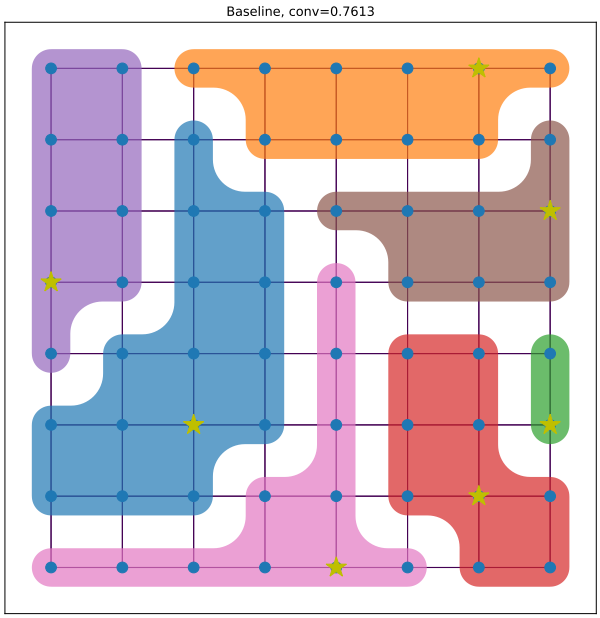



(b) Baseline Aggregates (coarsening by 3)

Figure 2: Aggregates for 1D Neumann problem

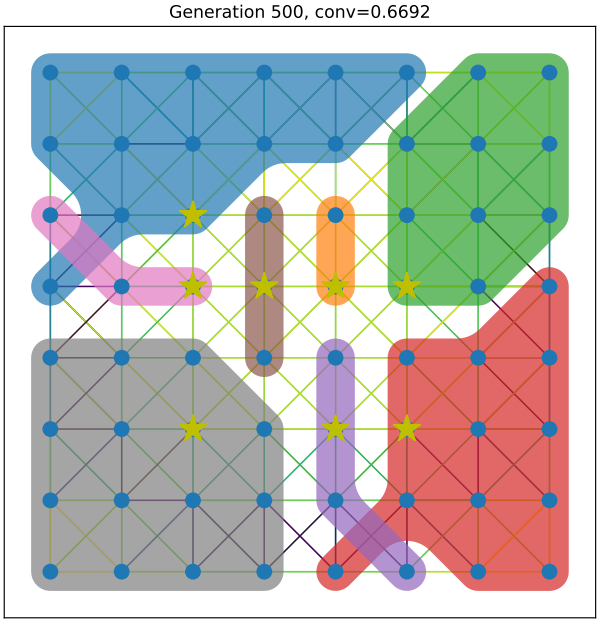



(a) ML Aggregates

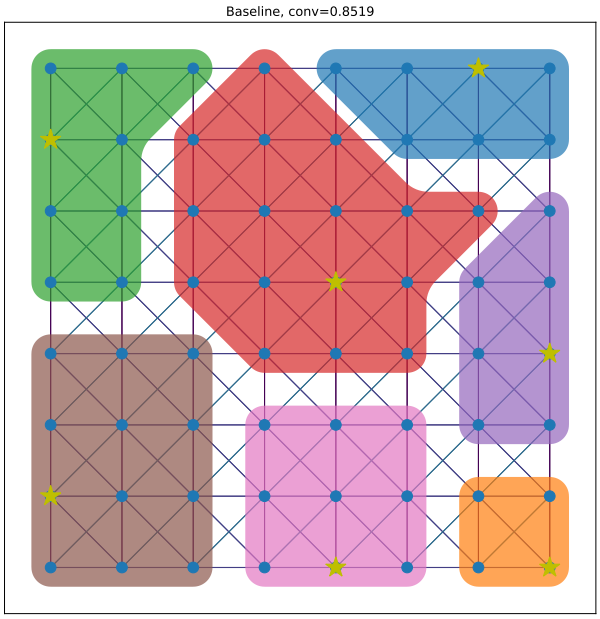



(b) Lloyd Aggregates

Figure 3: Aggregates for isotropic 2D Poisson




(a) ML Aggregates




(b) Lloyd Aggregates

Figure 4: Aggregates for anisotropic 2D Poisson