COARSERL: A GRAPH REINFORCEMENT LEARNING METHOD FOR ALGEBRAIC MULTIGRID COARSENING

Soha Yusuf * **Zechen Zhang** † **Kowshik Thopalli** ‡ **Rui Peng Li** ‡

ABSTRACT

Solving large sparse linear systems $\mathbf{Ax} = \mathbf{b}$ is central to many scientific and engineering applications. Algebraic Multigrid (AMG) achieves optimal linear complexity for suitable problems, but its performance critically depends on coarsening strategies that are largely heuristic-driven and sensitive to anisotropy, heterogeneity, and problem geometry. We introduce COARSERL, a graph-based reinforcement learning (RL) framework that learns coarse–fine (CF) splitting policies directly from the sparse matrix $\mathbf{A}$. CF splitting is formulated as a sequential decision process on the matrix graph, in which an agent selects coarse variables to optimize cumulative reward signals derived from classical AMG principles such as diagonal dominance. We present a systematic empirical study evaluating combinations of two RL algorithms, two GNN architectures, multiple reward formulations, and a range of diffusion and anisotropic diffusion problems on both structured and unstructured meshes. Our experiments show that COARSERL can achieve coarsening quality comparable to, and in many cases exceeding, that of classical greedy heuristics. These findings provide practical insights and guidelines for applying RL to AMG coarsening and demonstrate a reproducible pathway toward data-driven, robust coarsening algorithms for large-scale PDE simulations. [1]

1 INTRODUCTION

Large sparse linear systems of the form

$$\mathbf{Ax} = \mathbf{b}, \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{N \times N}$ is a symmetric positive-definite sparse matrix, arise ubiquitously in the numerical solution of partial differential equations (PDEs) for applications in fluid dynamics, solid mechanics, subsurface flow, and electromagnetics. For elliptic and parabolic problems, Algebraic Multigrid (AMG) methods are among the most effective solvers, offering (near-)optimal linear complexity while relying solely on the algebraic structure of the system matrix $\mathbf{A}$ (Brandt, 1986; Ruge & Stüben; Briggs et al., 2000). The efficiency of AMG hinges on a hierarchy of progressively coarser representations of the original problem, constructed through a process known as *coarsening*, which involves partitioning unknowns into coarse and fine sets and designing operators that transfer information between levels. Despite decades of development by Brandt (1977); Brandt et al. (1985); Vaněk et al. (1996); Brezina et al. (2006); Henson & Yang (2002), effective AMG coarsening algorithms remain a central challenge, particularly in the context of fully automated and robust solvers. Classical approaches rely on problem-dependent algorithmic heuristics including strength-of-connection measures, the coloring scheme, greedy maximal independent sets, and compatible relaxation. While highly successful for many classes of problems, these strategies can degrade or require substantial tuning for operators with strong anisotropy, highly heterogeneous coefficients, or unstructured discretizations (Stüben, 1983; MacLachlan & Saad, 2007; Brannick & Falgout, 2010a).

These limitations motivate the search for *data-driven* approaches that can learn coarsening strategies directly from the matrices. Unlike supervised learning, which requires ground-truth labels, AMG coarsening often lacks a unique "correct" solution: multiple valid partitions may yield similar solver efficiency and the quality of a coarsening is only revealed through downstream solver performance.

*Rensselaer Polytechnic Institute, New York, United States
†University of Minnesota, Minneapolis, United States
‡Lawrence Livermore National Laboratory, California, United States

[1] Our codes and dataset can be found at `https://github.com/SohaYusuf/AMG-COARSERL`

This makes the problem particularly well-suited to *unsupervised* and *reinforcement learning* (RL) paradigms, where an agent learns through interaction with the environment rather than from labeled data. RL is especially appealing because it naturally accommodates the sequential and combinatorial nature of coarsening decisions, where each node assignment affects subsequent choices and enables optimization of global objectives, such as AMG convergence, through delayed reward signals.

Recent work has begun to explore such data-driven approaches for AMG. Luz et al. (2020) proposed learning prolongation operators using graph neural networks (GNNs), demonstrating that neural networks can capture problem structure relevant to multigrid efficiency. More directly related to our focus, Taghibakhshi et al. (2021) introduced an RL framework for learning coarse–fine (CF) splitting policies directly from the matrix graph. Their work showed that an RL agent can learn coarsening strategies that generalize across problem sizes, establishing the viability of this approach. However, this pioneering study explored only a single point in what is actually a large design space: one RL algorithm (Dueling DQN), one GNN architecture (TAGConv), one problem class (diffusion equation), and a specific reward formulation. It remains unclear how sensitive the results are to these choices, and whether alternative configurations might yield better or more robust performance.

In this work, we address this gap by presenting a systematic empirical study of RL paradigms for AMG coarsening across four key design axes:
(a) **RL algorithm**: Dueling DQN (Wang et al., 2016) versus Proximal Policy Optimization (PPO) (Schulman et al., 2017);
(b) **GNN architecture**: TAGConv (Du et al., 2017) versus standard Graph Convolutional Networks (GCN) (Kipf & Welling, 2017);
(c) **Problem and mesh diversity**: diffusion and anisotropic diffusion benchmarks from Brannick & Falgout (2010a), on both structured and unstructured meshes; and
(d) **Reward design**: local step penalties combined with different terminal reward functions.

We clarify that our goal is not to propose a single new method, but rather to provide the community with actionable guidance on how these design choices affect learned coarsening quality.

Our Contributions. (i) We present a systematic empirical study of RL-based AMG coarsening, evaluating combinations of two RL algorithms, two GNN architectures, two problem classes, two grid types, and multiple reward formulations; (ii) We show that PPO matches or exceeds Dueling DQN while offering more stable training, and that GCN performs comparably to TAGConv despite its simpler architecture; (iii) We demonstrate that learned coarsening policies generalize from isotropic to anisotropic diffusion and from structured to unstructured meshes, achieving coarsening quality comparable to or better than classical greedy (MacLachlan & Saad, 2007) algorithms; and (iv)We provide practical guidelines for practitioners to apply RL to AMG coarsening, including recommendations on algorithm selection, architecture architecture, and reward design.

2 PRELIMINARIES

This section summarizes the AMG method, the PPO and Deep Q-Learning (DQN) RL algorithms, and the GNN architectures GCN and TAGConv employed in this work.

2.1 PROBLEM STATEMENT

We consider a linear system in equation 1. If we split the degrees of freedom into two disjoint sets: fine F and coarse C ($N = n_F + n_C$, where n_F and n_C are number of fine and coarse nodes respectively), and reorder rows and columns accordingly, $\mathbf{A}$ can be written in 2×2 block form:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FC} \\ \mathbf{A}_{CF} & \mathbf{A}_{CC} \end{pmatrix} \tag{2}$$

where $\mathbf{A}_{FF} \in \mathbb{R}^{n_F \times n_F}$ couples the fine nodes, $\mathbf{A}_{CC} \in \mathbb{R}^{n_C \times n_C}$ couples the coarse nodes, and $\mathbf{A}_{FC}$ and $\mathbf{A}_{CF}$ couple the two sets. Since $\mathbf{A}$ is symmetric we have $\mathbf{A}_{FC} = \mathbf{A}_{FC}^{\top}$. This block partition plays a central role in multilevel methods. In the algebraic recursive multilevel solver (ARMS) (Saad & Suchomel, 2002), the Schur complement on C is formed by $\mathbf{S} = \mathbf{A}_{CC} - \mathbf{A}_{CF}\mathbf{A}_{FF}^{-1}\mathbf{A}_{FC}$, with small fill-in entries dropped to preserve sparsity. When $\mathbf{A}_{FF}$ is well-conditioned, this procedure yields a numerically stable and inexpensive sparse approximation of $\mathbf{S}$. In the AMG context, a well-conditioned $\mathbf{A}_{FF}$ similarly indicates the existence of a sparse and

local approximation to the "ideal" interpolation operator $\mathbf{P}^\star = \begin{pmatrix}\mathbf{A}_{CF}\mathbf{A}_{FF}^{-1} & \mathbf{I}\end{pmatrix}^\top$, which yields a (tight) upper bound on the ideal two-grid convergence factor, $\mu(\mathbf{P}^\star) = 1 - \lambda_{\min}(\mathbf{M}_{FF}^{-1}\mathbf{A}_{FF})$ (Falgout & Vassilevski, 2004). Here, $\mathbf{M}$ denotes the smoother and $\lambda_{\min}(\cdot)$ the smallest eigenvalue. This is precisely the quantity measured by compatible relaxation (CR) (Brannick & Falgout, 2010b). This provides a strong theoretical justification for using the conditioning of $\mathbf{A}_{FF}$ as an optimization criterion in designing coarsening algorithms that guarantee robust two-grid convergence.

In this work, we enforce this condition by constructing the C/F partition to guarantee a prescribed diagonal dominance (DD) threshold θ_d for $\mathbf{A}_{FF}$. We define DD in the standard row-wise sense. For a matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$, row i is said to be DD with respect to θ_d (abbreviated as θ_d-DD) if

$$|a_{ii}| \;\geq\; \theta_d \sum_{j=1}^{n} |a_{ij}|, \tag{3}$$

where $\theta_d \in (0, 1]$ is a threshold parameter that controls the strictness of the condition. The goal of our coarsening algorithm is to maximize n_F while ensuring $\mathbf{A}_{FF}$ satisfies the θ_d-DD property. Maximizing the number of F nodes keeps the coarse grid small, and thus reduces both memory usage and computational cost in the multilevel hierarchy. This balance between the size and quality of the coarse grid is essential for achieving efficient and scalable AMG performance.

2.2 Algebraic Multigrid

AMG achieves linear computational complexity by constructing a hierarchy of progressively coarser linear systems using only the information of the matrix $\mathbf{A}$ without requiring explicit geometric data. At each level of the AMG hierarchy, the degrees of freedom are divided into a coarse set C and a fine set F. The quality of CF splitting is critical to AMG performance as it directly affects the construction of interpolation operators and the spectral properties of the coarse-level operators.

AMG relies on the complementary roles of relaxation (smoothing) and coarse-grid correction. Standard relaxation schemes effectively damp high-frequency error components, while coarse-grid correction targets the remaining low-frequency error modes. To enable this separation of scales, it must be ensured that error components that are not efficiently reduced by relaxation on the fine grid can be accurately represented on the coarse grid. This requirement is commonly formalized through principles such as CR and weak approximation properties (Briggs et al., 2000). Given a CF splitting, an interpolation operator $\mathbf{P} \in \mathbb{R}^{N\times n_C}$ is constructed to interpolate corrections of the coarse-grid to the fine grid. The coarse-grid operator is defined via the Galerkin projection $\mathbf{A}_c = \mathbf{P}^\top\mathbf{A}\mathbf{P}$. The multigrid hierarchy is obtained recursively by applying the same procedure to $\mathbf{A}_c$.

While many AMG components admit robust algebraic constructions, the design of effective coarsening strategies remains a central challenge. Classical approaches are typically greedy and heuristic-driven, relying on measures such as strength of connection, diagonal dominance, or relaxation-based criteria. These methods can be sensitive to anisotropy, coefficient heterogeneity, and problem structure, motivating data-driven approaches that learn CF splitting policies directly from the matrix $\mathbf{A}$.

2.3 Reinforcement learning

Reinforcement learning (RL) is a natural fit for the CF-splitting task (see Algorithm 1) because coarsening is inherently sequential, combinatorial, and governed by long-range, nonlocal objectives: each choice of a coarse node changes the remaining action set and the future spectral properties of the fine-level operator $\mathbf{A}_{FF}$. An RL agent can optimize a global objective through trial-and-error, exploit structure in the matrix graph via GNN-based policies, and learn to trade off immediate and delayed effects of node selections without requiring hand-tuned heuristics.

We model the RL environment as an episodic Markov Decision Process (see Section 3.1 for detail). In this work, we compare two representative RL paradigms: value-based methods (DQN) and policy-gradient methods (PPO). Each offers distinct trade-offs for the CF-splitting task.

Deep Q-Networks (DQN). DQN (Mnih et al., 2015) learns a state-action value function $Q(s, a; \Theta)$, where Θ denotes the neural network parameters, that estimates the expected return. At time step t, with state s_t, the policy selects an action $a_t = \arg\max_{a\in\mathcal{A}(s_t)} Q(s_t, a; \Theta)$, where

Algorithm 1 Reinforcement-learning Coarsening Environment

Require: Dataset of system matrices $\mathbf{A}$; violation criteria based on θ_d
Ensure: Environment API: `reset()`, `step(`a_t`)`

procedure INITIALIZE
  Set terminal reward function $f \in \{f_0, f_1, f_2, f_3, f_4\}$
  Set coarse set $C \leftarrow \varnothing$, fine set $F \leftarrow \{1, \dots, N\}$.
  Define action space: select node index $i \in F$.
  Define state $s_t = G_t = (V_t, E_t)$.
end procedure
function RESET
  Sample new system matrix $\mathbf{A}$ from dataset.
  Initialize $C = \varnothing,\ F = \{1, \dots, N\}$.
  Construct initial state $s_0 = G_0 = (V_0, E_0)$ and evaluate violation indicators on F w.r.t. θ_d
  return s_0.
end function
function STEP(action $a_t = i$)
  Coarsen node i: $F \leftarrow F \setminus \{i\},\ C \leftarrow C \cup \{i\}$.
  Construct next state $s_{t+1} = G_{t+1} = (V_{t+1}, E_{t+1})$ and update $\mathbf{A}_{FF}$ violations (w.r.t. θ_d).
  Set step penalty $r_t \leftarrow -1$; done $\leftarrow$ no violations in F (or no valid actions)
  Compute episode return: $R_t \leftarrow r_t + (\text{done})\, f(R_F)$
  return $(s_{t+1}, R_t, \text{done}, .$
end function

$\mathcal{A}(s_t)$ is the set of valid actions. Training minimizes the temporal-difference error using experience replay and a target network for stability. DQN is off-policy, allowing reuse of past transitions, but requires computing Q-values for all valid actions at each step.

Proximal Policy Optimization (PPO). PPO (Schulman et al., 2017) directly learns a stochastic policy $\pi_\theta(a \mid s)$ parameterized by θ. PPO updates θ by maximizing the clipped surrogate objective

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\big(\rho_t(\theta)\,\hat{A}_t,\ \text{clip}(\rho_t(\theta), 1-\varepsilon, 1+\varepsilon)\,\hat{A}_t\big)\Big], \qquad \rho_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)},$$

where $\pi_{\theta_{\text{old}}}$ is the data-collecting policy, $\varepsilon > 0$ is the clipping (trust-region) parameter, and $\hat{A}_t$ is an estimator of the advantage at time step t (e.g., $\hat{A}_t \approx R_t - V_\phi(s_t)$). Here R_t denotes the empirical return and V_ϕ is a value function with parameters ϕ (updated by minimizing the squared error between $V_\phi(s_t)$ and R_t). PPO is on-policy (requires fresh trajectories for each update) and naturally supports action masking for the shrinking action set in CF-splitting.

Key differences. The two algorithms differ in several ways relevant to CF-splitting: (i) **Variable, shrinking action set:** actions correspond to indices $i \in F$ and F changes after each selection. PPO samples from a masked distribution over F, while DQN must compute Q-values for all nodes in F and take the maximum; and (ii) **On-policy vs. off-policy**: DQN reuses past experience via replay, which may become stale as the policy improves. PPO uses only fresh on-policy data, avoiding distribution mismatch but requiring more environment interactions.

3 EXPERIMENTAL SETUP

This section describes the experimental setup for our benchmark. We formulate AMG coarsening as a sequential decision problem on graphs (Section 3.1), describe the two GNN architectures under comparison (Section 3.2), and detail the reward formulations we evaluate (Section 3.3).

3.1 GRAPH REINFORCEMENT LEARNING

The RL environment for AMG coarsening presented in Algorithm 1 is defined as an episodic Markov decision process whose state, actions and rewards are tailored to the coarsening task. At step t the observation (state) s_t is a graph representation of the system matrix $\mathbf{A}$, i.e.,

$$s_t = G_t = \big(V_t, E_t\big),$$

where V_t is the set of nodes and E_t is the set of edges. Each node $i \in V_t$ carries a node attribute vector $\mathbf{v}_i \in \mathbb{R}^{n_v}, n_v = 2$, and each edge $(i,j) \in E$ is associated with an edge attribute $\mathbf{e}_{ij} \in \mathbb{R}^{n_e}, n_e = 1$. In our framework the edge attributes are simply the nonzero matrix entries,

$$e_{ij} = a_{ij}, \qquad (i,j) \in E = \{(p,q) \ : \ a_{pq} \neq 0\}.$$

Each node feature encodes the fine/coarse label and a violation indicator:

$$\mathbf{v}_i \ = \ \begin{bmatrix} f_i \\ \nu_i \end{bmatrix}, \qquad f_i \in \{0,1\},\ \nu_i \in \{0,1\},$$

where $f_i = 1$ denotes that node i is currently labeled as a fine node (and $f_i = 0$ denotes coarse), and $\nu_i = 1$ indicates that node i violates diagonal dominance criteria θ_d-DD (see equation 3). Thus $n_v = 2$ in our implementation. These node features are updated after each coarsening action so that the state s_t always reflects the current CF splitting and violation status. Together, the node features $\{\mathbf{v}_i\}_{i \in V_t}$ and edge attributes $\{\mathbf{e}_{ij}\}_{(i,j) \in E_t}$ provide the agent with local structural and status information—which nodes are fine/coarse, which nodes violate θ_d-DD, and the strength of connections between nodes—enabling informed selection of actions during training.

The action space is discrete over node indices, where an action at step t is defined as $a_t = i$, that is selecting node $i \in V_t$ to be promoted from the fine set to the coarse set. Applying a_t updates the graph state according to $G_{t+1} = \mathcal{T}(G_t, a_t)$, where $\mathcal{T}$ denotes the transition operator that modifies the CF splitting and updates the associated node and edge attributes. Specifically, the fine and coarse sets are updated as

$$F_{t+1} = F_t \setminus \{i\}, \qquad C_{t+1} = C_t \cup \{i\}.$$

An episode terminates when no violation nodes remain in the fine–fine submatrix $\mathbf{A}_{FF}$. Each episode corresponds to the complete coarsening of a single linear system $\mathbf{Ax} = \mathbf{b}$, beginning with initialization $C = \varnothing$, $F = \{1, \ldots, N\}$, and initial state s_0. See section 3.3 for reward design.

3.2 Graph Neural Network

We parameterize the policy (for PPO) or Q-function (for DQN) by a GNN that maps the graph state G_t to per-node scores. We compare two architectures that differ in how they aggregate neighborhood information.

Graph Convolutional Network (GCN). GCN (Kipf & Welling, 2017) uses a simple first-order approximation of spectral graph convolutions. Each layer updates node representations as:

$$\mathbf{h}_i^{(\ell+1)} = \sigma\Big(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{d_i d_j}} \mathbf{W}^{(\ell)} \mathbf{h}_j^{(\ell)} \Big),$$

where $\mathcal{N}(i)$ denotes the neighbors of node i, d_i is the degree of node i, $\mathbf{W}^{(\ell)}$ are learnable weights, and σ is a nonlinearity. GCN is computationally efficient and widely used, but each layer only aggregates information from immediate neighbors.

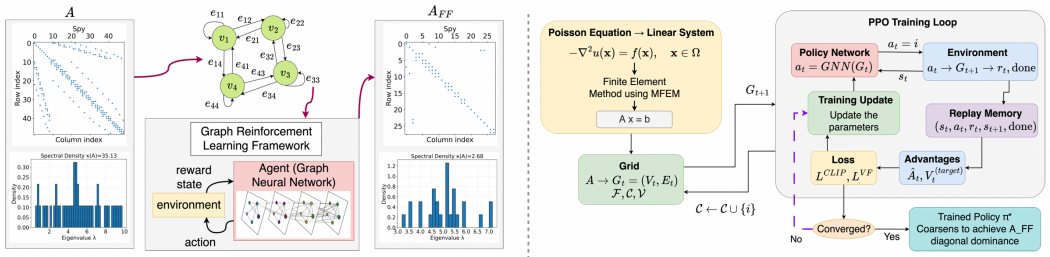
Topology Adaptive Graph Convolutional Network (TAGConv). TAGConv (Du et al., 2017) generalizes GCN by using a fixed-size set of graph filters based on powers of the adjacency matrix:

$$\mathbf{h}_i^{(\ell+1)} = \sigma\Big(\sum_{k=0}^{K} \mathbf{W}_k^{(\ell)} \big(\mathbf{A}^k \mathbf{H}^{(\ell)}\big)_i \Big),$$

where K is the filter order and $\mathbf{A}^k$ captures k-hop neighborhood information. This allows TAGConv to aggregate information from multi-hop neighbors within a single layer, potentially capturing longer-range dependencies relevant to coarsening quality Taghibakhshi et al. (2021) used TAGConv in their RL coarsening framework.

Both architectures output per-node scores that are passed through a softmax (for PPO) or used directly as Q-values (for DQN) to select coarsening actions. We use the same number of layers and hidden dimensions for both architectures to ensure a fair comparison. More detailed discussion about optimization with Deep Q-learning and Proximal Policy Optimization is mention in the Appendix.

We illustrate the overall RL-based coarsening framework with GNNs and the PPO training loop as well as the reward structure in Figure 1.



(a) High-level RL-based AMG coarsening framework.

(b) Detailed Visualization of PPO based training.

Figure 1: Overview of our RL formulation for AMG coarsening. (a) A sparse SPD matrix $\mathbf{A}$ obtained from finite-element discretization is represented as a graph $G = (V, E)$ and provided to a GNN-based policy. At each step, the agent selects a node to coarsen, progressively updating the fine set until a convergence criterion on $\mathbf{A}_{FF}$ is satisfied. (b) Visualizes the training through the PPO algorithm, where each coarsening action incurs a step penalty and the episode terminates once diagonal dominance or compatible-relaxation thresholds are met.

3.3 Reward Design

The reward signal is a key design choice in RL-based coarsening. We compare several formulations that trade off simplicity against incorporation of domain knowledge.

Step penalty. The simplest reward assigns a constant penalty $r_t = -1$ for each coarsening action. This encourages the agent to minimize the number of nodes selected into the coarse set while still producing a valid DD configuration. The cumulative return is

$$\sum_t r_t = -|C| = |F| - N,$$

so maximizing the return is equivalent to maximizing the F-fraction, i.e., $R_F = |F|/N$.

Terminal bonus. To give a stronger problem-dependent signal, we add a terminal bonus $f(x)$, where x denotes the F-fraction R_F at the end of each episode. The terminal bonus directly couples the reward to the quality of the final coarsening outcome, encouraging policies that optimize the global coarsening structure rather than local decisions.

Our first terminal reward function is a piecewise linear function f_0 defined as

$$f_0(x) = \{a_i x - 1,\ x_i < x \le x_{i+1}\}, \quad i = -1, \dots m,$$

where x denotes the final R_F, $x_{-1} = 0$, $x_{m+1} = 1$, and x_0 serves as a reference value of R_F, for example obtained from a classical DD based coarsening algorithm. For $x \le x_0$, we let $a_{-1} = 0$, so that $f_0(x)$ assigns a constant negative reward of -1, corresponding to a uniform penalty when the RL-generated coarsening is inferior to the reference baseline. The slopes $\{a_i\}$ and breakpoints $\{x_i\}$ were chosen manually based on empirical experiences, with progressively steeper rewards assigned to higher R_F in order to emphasize improvements beyond typical heuristic baselines.

While this terminal reward function can strongly promote high-quality coarsenings, the discontinuities at the breakpoints can introduce noise into value learning and encourage threshold-seeking behavior. Next, we consider several options for smooth terminal bonus functions. All of these functions are designed to take the value 0 at $x = t$ and exhibit a rapid decay to -1 when $x < t$ and $x \to 0$, thereby penalizing coarsenings that underperform the reference baseline, while increasing monotonically for $x \in (0, 1)$. The first option is a logit–tanh function defined by

$$f_1(x) = \tanh\big(k[\text{logit}(x) - \text{logit}(t)]\big), \qquad \text{logit}(y) = \ln \frac{y}{1-y}.$$

This construction yields a bonus bounded in $(-1, 1)$ centered at x_0. The logit transform enhances the sensitivity to the change of R_F at both endpoints. The parameter $k > 0$ controls the sharpness of the transition around $x = t$, with larger values producing behavior closer to a hard threshold and smaller values yielding a smoother gradient. Another option is a fractional power softmax function,

$$f_2(x) = \left(\frac{\zeta(y)}{\ln 2}\right)^p - 1, \quad y = k(x - t), \quad p \le 1$$

where ζ denotes the softplus function, $\zeta(u) = \ln(1 + e^u)$, and the normalization by $\ln 2$ ensures $f_2(t) = 0$. This formulation provides a smooth approximation to a shifted hinge function. The exponent $p \in (0, 1]$ controls the rate of increase for $x > t$, with $p < 1$ producing sublinear growth that moderates large rewards. The last option is

$$f_3(x) = \frac{\ln\big(1 + \zeta(y)\big)}{\ln\big(1 + \ln 2\big)} - 1, \quad y = k(x - t)$$

where ζ again denotes the softplus function. This construction applies an additional logarithmic compression to the softplus-based reward, resulting in a smoothly increasing function. The parameters and plots of these terminal bonus functions are provided in the appendix for visual comparison.

Finally, the terminal reward is added at the end of each episode $R_{\text{terminal}} = \lambda\, f_i(x)$, where $\lambda > 0$ is a scalar that places the terminal bonus on the same numerical scale as the cumulative step penalties. In all experiments we choose $\lambda = 40.0$. The total episode return is the sum of the per-step penalties and the terminal reward R_{terminal}.

4 EXPERIMENTS

We evaluate COARSERL on a controlled, reproducible benchmark organized into four complementary parts. Section 4.1 describes how we generate problem instances and meshes used throughout the study. Section 4.2 isolates the effect of terminal reward design on coarsening quality. Section 4.3 compares GNN backbones (GCN vs. TAGConv) and their impact on policy reliability and sample efficiency. For each experiment we report baselines and evaluation metrics; implementation details and full hyperparameter tables are provided in the Appendix.

4.1 DATA GENERATION

We generate a controlled, reproducible dataset of sparse system matrices by discretizing four diffusion problem classes on the unit square $\Omega = [0, 1]^2$. Three classes follow the standard 2D anisotropic benchmarks from Brannick & Falgout (2010a), and the fourth is a variable-coefficient scalar diffusion problem used to exercise coefficient variability and mesh-dependent behavior.

Following Brannick & Falgout (2010a), we consider three 2-D diffusion test problems defined by

$$-\nabla\cdot\big(K\nabla u\big) + d\,u = f, \qquad K = \begin{pmatrix} a & c \\ c & b \end{pmatrix}, \qquad \begin{aligned} a &= \cos^2\phi + \varepsilon\sin^2\phi, \\ b &= \varepsilon\cos^2\phi + \sin^2\phi, \\ c &= (1-\varepsilon)\cos\phi\sin\phi, \end{aligned} \qquad 0 < \varepsilon \le 1,$$

where ε controls anisotropy strength and ϕ its orientation. The first problem is the isotropic Laplacian (2D-Lap), obtained by setting $\varepsilon = 1$ and $d = 0$, in which case ϕ is irrelevant. The second is the axis-aligned anisotropic Laplacian (2D-ALap), defined by $\varepsilon = 0.01$, $\phi = 0$, and $d = 0$. The third is the rotated anisotropic Laplacian (2D-RLap), defined by $\varepsilon = 0.01$, $\phi = \pi/3$, and $d = 0$.

In addition, we consider a scalar diffusion class (Anderson et al., 2021) of the form

$$-\nabla\cdot\big(k(\mathbf{x})\nabla u\big) = f$$

with a scalar coefficient $k(\mathbf{x})$ parameterized by c_x, c_y, c_z and a frequency parameter ($\kappa = 2\pi \cdot \text{freq}$). In our dataset, we use 2D meshes so that all produced matrices are directly comparable.

All coarsening methods (including the standard greedy heuristic from MacLachlan & Saad (2007)) are applied to the same matrix instances and evaluated with a common pipeline. Our principal coarsening-quality metric is the fine-node fraction $R_F = \frac{|F|}{N}$. Smaller R_F denotes more aggressive coarsening. We also record the condition number of $\mathbf{A}_F F$ after coarsening, and learning dynamics (mean reward versus training iterations). Training dataset distribution, training histories of mean rewards and optimization with PPO and DQN are provided in the Appendix.

4.2 EFFECT OF TERMINAL REWARD FUNCTION ON COARSENING QUALITY

The variants COARSERL–f_0, COARSERL–f_1, COARSERL–f_2, and COARSERL–f_3 correspond to our framework with the terminal reward functions defined in Section 3.3. In addition, COARSERL–f_4 denotes the case of zero terminal bonus ($f_4 = 0$), i.e., no terminal reward.

Table 1: Comparison of different terminal rewards for the same matrix ($N = 63$) at different θ_d. For each method we report the number of fine and coarse nodes (n_F, n_C), the condition number of the relevant block, and F-fraction R_F. The original fine–fine block has $\kappa(\mathbf{A}_{FF}) = 28.643$.

θ_d	Method	$\kappa(\mathbf{A}_{FF})$	(n_F, n_C, R_F)
0.6	Scott	3.202	(39, 24, 0.619)
	ddSym	3.181	(39, 24, 0.619)
	RL–f_0	3.710	(41, 22, 0.651)
	RL–f_1	3.612	(41, 22, 0.651)
	RL–f_2	3.550	(39, 24, 0.619)
	RL–f_3	4.078	(42, 21, 0.667)
	RL–f_4	4.562	**(44, 19, 0.698)**
0.7	Scott	2.630	**(34, 29, 0.540)**
	ddSym	2.639	(33, 30, 0.524)
	RL–f_0	2.628	(25, 38, 0.397)
	RL–f_1	2.936	(28, 35, 0.444)
	RL–f_2	3.212	(30, 33, 0.476)
	RL–f_3	3.223	(32, 31, 0.508)
	RL–f_4	3.220	(30, 33, 0.476)
0.9	Scott	1.000	(14, 49, 0.222)
	ddSym	2.000	**(15, 48, 0.238)**
	RL–f_0	2.000	(9, 54, 0.143)
	RL–f_1	2.000	(9, 54, 0.143)
	RL–f_2	2.000	(11, 52, 0.175)
	RL–f_3	2.000	(8, 55, 0.127)
	RL–f_4	2.000	(10, 53, 0.159)

Table 2: For each coarsening method, the maximum R_F is shown below for matrix sizes $N \in \{7, 13, 63\}$ and $\theta_d \in \{0.6, 0.7, 0.9\}$. Entries are shown as triples ($N = 7$, 13, 63) corresponding to the three matrix sizes.

θ_d	Method	Max R_F
0.6	Scott	(0.857, 0.538, 0.508)
	ddSym	(0.857, 0.538, 0.508)
	RL–f_0	**(0.857, 0.769, 0.540)**
	RL–f_1	(0.857, 0.769, 0.492)
	RL–f_2	(0.857, 0.769, 0.508)
	RL–f_3	(0.857, 0.769, 0.492)
	RL–f_4	(0.857, 0.615, 0.540)
0.7	Scott	(0.857, 0.462, 0.508)
	ddSym	**(0.857, 0.538, 0.508)**
	RL–f_0	(0.857, 0.538, 0.381)
	RL–f_1	(0.857, 0.538, 0.413)
	RL–f_2	(0.857, 0.538, 0.365)
	RL–f_3	(0.857, 0.538, 0.413)
	RL–f_4	(0.714, 0.462, 0.413)
0.9	Scott	(0.429, 0.231, 0.508)
	ddSym	**(0.429, 0.231, 0.508)**
	RL–f_0	(0.429, 0.231, 0.222)
	RL–f_1	(0.286, 0.308, 0.254)
	RL–f_2	(0.429, 0.308, 0.270)
	RL–f_3	(0.429, 0.231, 0.270)
	RL–f_4	(0.286, 0.308, 0.222)

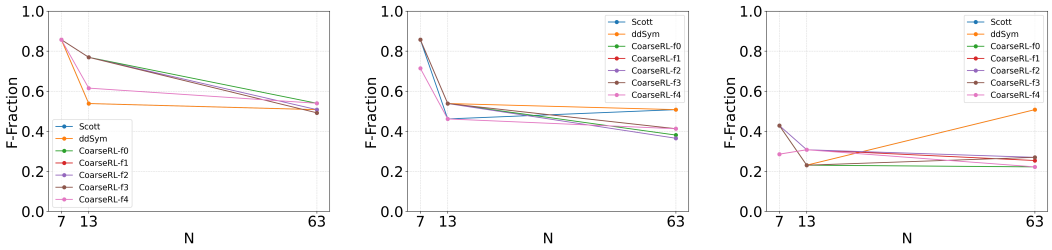
Figure 6 presents the evolution of the mean training reward for $\theta_d \in \{0.6, 0.7, 0.8, 0.9\}$. A consistent observation across all θ_d values is that a nonzero terminal bonus is essential for effective learning. When $f_4 = 0$, the reward signal is weak and the policy improves slowly, confirming that terminal feedback plays a critical role in guiding CF-splitting decisions. Among the nonzero bonuses, the smooth monotone reward f_2 yields the most stable training dynamics, with a steady increase in mean reward and comparatively low variance. The bounded logit–tanh reward f_1 shows similarly stable behavior. In contrast, the piecewise linear reward f_0 exhibits larger oscillations due to its breakpoint structure, which introduces abrupt changes in incentives and encourages threshold-seeking behavior.

Figure 2, Table 1 and Table 2 quantify the impact of these reward designs on coarsening quality. At $\theta = 0.6$, the less restrictive diagonal-dominance requirement allows more aggressive coarsening, and CoarseRL–f_0 attains the largest R_F at larger N, indicating that its strong, piecewise incentives are effective when the constraint is mild. However, as θ_d increases to 0.7 and 0.9, the problem becomes more restrictive and feasible high-R_F splittings are harder to find. In this regime, CoarseRL–f_2 consistently performs better among the learned variants, achieving higher or more competitive R_F values with more stable condition numbers. The smoother growth of f_2 appears to provide better gradient information when the feasible region shrinks.

Overall, performance degrades for all reward functions as θ_d increases, reflecting the intrinsic difficulty of satisfying stricter diagonal-dominance constraints. Under these harder settings, the RL framework struggles to match classical heuristics in maximum R_F, but smooth terminal rewards, particularly f_2, provide the most robust behavior across thresholds.

4.3 GNN Architecture Comparison: GCN vs. TAGConv

We compare GCN and TAGConv using PPO on diffusion and anisotropic diffusion problem. Both architectures use the same number of layers (3) and hidden dimensions (64). Table 3 reports the



(a) $\theta_d = 0.6$ (b) $\theta_d = 0.7$ (c) $\theta_d = 0.9$

Figure 2: Maximum R_F versus matrix size $N \in \{7, 13, 63\}$ for each coarsening method. Panels (a)–(c) show results for diagonal-dominance thresholds $\theta_d = 0.6, 0.7, 0.9$, respectively; points are taken from Table 2. Overall, looser diagonal-dominance requirements (smaller θ_d) permit larger maximum R_F values, and most methods show a net decline in R_F as N increases. At $\theta_d = 0.6$ all methods drop from $R_F \approx 0.86$ at $N = 7$ to roughly 0.50–0.54 at $N = 63$ (CoarseRL–f_0 and CoarseRL–f_4 retain the largest fractions at $N = 63$). For $\theta_d = 0.7$ the decrease with N is stronger for several CoarseRL variants (final values $\approx$ 0.36–0.41), while Scott/ddSym recover to ≈ 0.51 at $N = 63$. At $\theta_d = 0.9$ maximum R_F is generally lower and method-dependent: Scott and ddSym reach the highest values at $N = 63$ (≈ 0.51), whereas most CoarseRL variants remain below ≈ 0.28.

Table 3: Comparison of R_F for GCN and TAGConv for one matrix instance of size $N = 63$ and $\kappa(\mathbf{A}) = 28.643$.

Method	$\kappa(\mathbf{A}_{FF})$	(n_F, n_C, R_F)
GCN	3.863	(42, 21, 0.667)
TAGConv	4.059	(43, 20, 0.683)

R_F and related metrics for one representative matrix instance of size $N = 63$ and $\kappa(\mathbf{A}) = 28.643$. Although TAGConv achieves a slightly higher F-fraction ($R_F = 0.683$ vs. 0.667 for GCN), the difference is minor, and the convergence behavior ($\kappa(\mathbf{A}_{FF})$) is comparable. This indicates that while the simpler GCN architecture performs comparably on smaller matrices at lower computational cost, TAGConv achieves slightly better F-fraction and convergence for the matrix sizes considered here. Therefore, we use TAGConv in all our experiments, noting that for smaller problems, GCN would provide similar performance more efficiently.

5 Conclusions and Future Work

We presented an empirical study of reinforcement learning for AMG coarsening, exploring different design choices. Our experiments show that RL can learn effective coarsening policies for diffusion problems, with performance sensitive to design. PPO provides stable, high-quality policies; TAGConv yields slight gains, while GCN performs comparably on smaller matrices at lower computational cost. Smooth rewards with a terminal F-fraction bonus further improve stability and coarsening quality.

While our study focuses on moderate-size diffusion problems, results may not generalize to highly anisotropic PDEs, unstructured meshes, or very large-scale systems. Future work includes scaling RL to larger and more diverse PDEs, developing sample-efficient methods, designing multi-objective rewards, exploring automated GNN architecture search, integrating coarsening with full AMG pipelines, enabling distributed or accelerated training, and extending benchmarks with interpretability tools.

ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract No. DE-AC52-07NA27344, Lawrence Livermore National Security, LLC and was supported by the LLNL-LDRD Program under Project No. 24ERD033.

REFERENCES

Robert Anderson, Julian Andrej, Andrew Barker, Jamie Bramwell, Jean-Sylvain Camier, Jakub Cerveny, Veselin Dobrev, Yohann Dudouit, Aaron Fisher, Tzanio Kolev, Will Pazner, Mark Stowell, Vladimir Tomov, Ido Akkerman, Johann Dahm, David Medina, and Stefano Zampini. Mfem: A modular finite element methods library. *Computers & Mathematics with Applications*, 81:42–74, January 2021. ISSN 0898-1221. doi: 10.1016/j.camwa.2020.06.009. URL http://dx.doi.org/10.1016/j.camwa.2020.06.009.

A. Brandt, S. McCormick, and J. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In David J. Evans (ed.), *Sparsity and its Applications*, pp. 257–284. Cambridge University Press, Cambridge, 1985. ISBN 0521262720.

Achi Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 31(138):333–390, 1977. doi: 10.1090/S0025-5718-1977-0431719-X.

Achi Brandt. Algebraic multigrid theory: The symmetric case. *Applied Mathematics and Computation*, 19(1):23–56, 1986. ISSN 0096-3003. doi: https://doi.org/10.1016/0096-3003(86)90095-0. URL https://www.sciencedirect.com/science/article/pii/0096300386900950.

James J. Brannick and Robert D. Falgout. Compatible relaxation and coarsening in algebraic multigrid. *SIAM Journal on Scientific Computing*, 32(3):1393–1416, 2010a. doi: 10.1137/090772216. URL https://doi.org/10.1137/090772216.

James J. Brannick and Robert D. Falgout. Compatible relaxation and coarsening in algebraic multigrid. *SIAM Journal on Scientific Computing*, 32(3):1393–1416, 2010b. doi: 10.1137/090772216. URL https://doi.org/10.1137/090772216.

Marian Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, Steve McCormick, and John Ruge. Adaptive algebraic multigrid. *SIAM Journal on Scientific Computing*, 27:1261–1286, 01 2006. doi: 10.1137/040614402.

William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A Multigrid Tutorial, Second Edition*. Society for Industrial and Applied Mathematics, second edition, 2000. doi: 10.1137/1.9780898719505. URL https://epubs.siam.org/doi/abs/10.1137/1.9780898719505.

Jian Du, Shanghang Zhang, Guanhang Wu, José M. F. Moura, and Soummya Kar. Topology adaptive graph convolutional networks. *arXiv preprint arXiv:1710.10370*, 2017.

Robert D. Falgout and Panayot S. Vassilevski. On generalizing the algebraic multigrid framework. *SIAM Journal on Numerical Analysis*, 42(4):1669–1693, 2004. doi: 10.1137/S0036142903429742. URL https://doi.org/10.1137/S0036142903429742.

Van Henson and Ulrike Yang. Boomeramg: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41:155–177, 04 2002. doi: 10.1016/S0168-9274(01)00115-5.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Ilay Luz, Meirav Galun, Haggai Maron, Ronen Basri, and Irad Yavneh. Learning algebraic multigrid using graph neural networks. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6489–6499. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/luz20a.html.

S. MacLachlan and Yousef Saad. A greedy strategy for coarse-grid selection. *SIAM Journal on Scientific Computing*, 29(5):1825–1853, 2007. doi: 10.1137/060654062. URL https://doi.org/10.1137/060654062.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. doi: 10.1038/nature14236. URL https://doi.org/10.1038/nature14236.

J. W. Ruge and K. Stüben. *4. Algebraic Multigrid*, pp. 73–130. doi: 10.1137/1.9781611971057.ch4. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611971057.ch4.

Y. Saad and B. Suchomel. Arms: an algebraic recursive multilevel solver for general sparse linear systems. *Numerical Linear Algebra with Applications*, 9(5):359–378, 2002. doi: https://doi.org/10.1002/nla.279. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nla.279.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Klaus Stüben. Algebraic multigrid (amg): experiences and comparisons. *Applied Mathematics and Computation*, 13(3):419–451, 1983. ISSN 0096-3003. doi: https://doi.org/10.1016/0096-3003(83)90023-1. URL https://www.sciencedirect.com/science/article/pii/0096300383900231.

Ali Taghibakhshi, Scott MacLachlan, Luke Olson, and Matthew West. Optimization-based algebraic multigrid coarsening using reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 13823–13835. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/6531b32f8d02fece98ff36a64a7c8260-Abstract.html.

Petr Vaněk, Jan Mandel, and Miroslav Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56(3):179–196, 1996. doi: 10.1007/BF02238511.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1995–2003. PMLR, 2016.

A Appendix

A.1 Training data distribution

We summarize the statistical properties of the training set in Figure 3. The three histograms (average node degree, matrix condition number, and mesh size) show that the dataset spans a range of connectivity patterns, numerical conditioning, and problem sizes rather than concentrating on a single regime. In particular, average node degree varies with mesh topology (quadrilateral versus triangular) and refinement level, condition numbers exhibit a spread that includes moderately ill-conditioned matrices, and mesh sizes cover the small-to-medium scale problems used in our experiments. The scatter plot (mesh size versus average degree) indicates that average degree is not tightly coupled to size alone — mesh topology and refinement control connectivity — so the dataset provides varied structural examples for the learning algorithm.

Figure 4 illustrates representative sparsity patterns (top row) alongside their corresponding two-dimensional meshes (bottom row). The first two columns show quadrilateral meshes and their spy plots, which display more regular, banded sparsity reflecting the grid-like connectivity; the last two columns show triangular meshes and their spy plots, which are less regular and typically exhibit

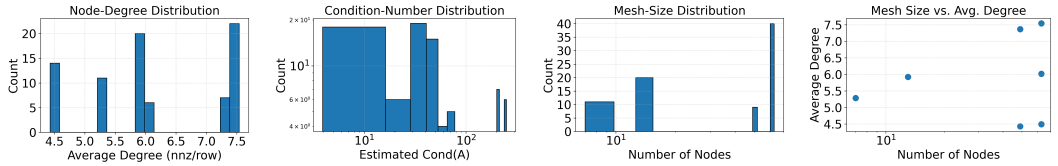


Figure 3: Distributional statistics of the training dataset. From left to right: histogram of average node degree, histogram of matrix condition numbers, histogram of mesh sizes (degrees of freedom), and a scatter plot relating mesh size to average node degree.

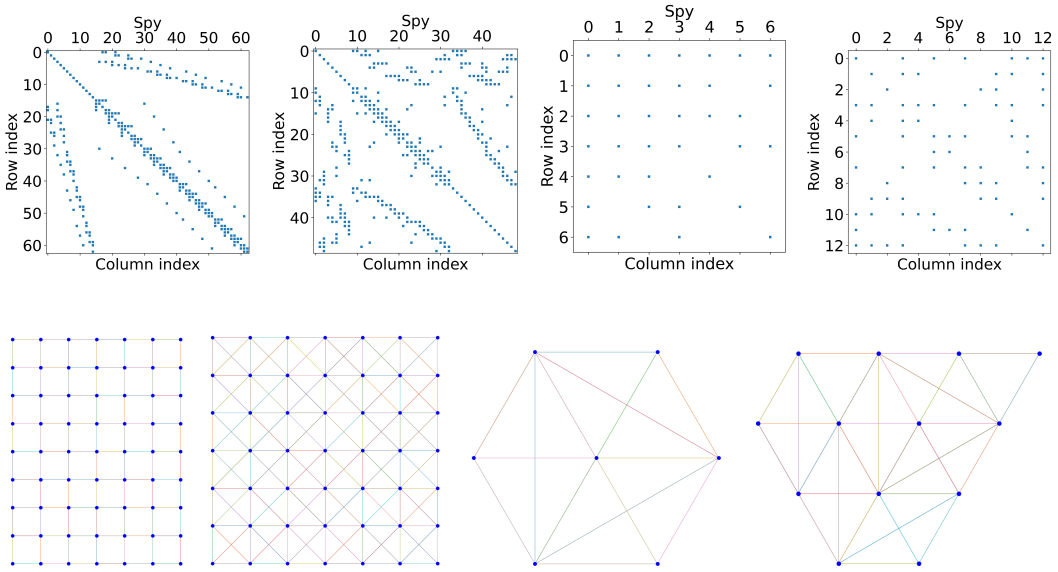


Figure 4: Sparsity patterns of the system matrices $\mathbf{A}$ (top row) and the corresponding two-dimensional meshes (bottom row) for four representative examples. The first two columns correspond to quadrilateral meshes, while the last two columns correspond to triangular meshes.

wider bandwidth and more irregular fill. These paired images make explicit the relationship between mesh geometry and the structure of the system matrix $\mathbf{A}$; together, the distributional statistics and the example pairs demonstrate that the training data contain diverse connectivity and conditioning scenarios that challenge and thus help to robustly train the coarsening policies.

A.2 Optimization with Deep Q-learning

We optimize a state-action value function $Q(s, a; \Theta)$ using the DQN algorithm, where s and a denote the environment state and a discrete action (node index) as defined above, and Θ are the network parameters. The Q-network is trained to satisfy the Bellman optimality relation by minimizing the temporal-difference (TD) error. Given a transition $(s_t, a_t, r_t, s_{t+1}, \text{done})$ sampled from experience, the DQN target is

$$y_t \;=\; r_t + \gamma\,(1 - \text{done}) \max_{a'} Q(s_{t+1}, a'; \Theta^-),$$

where $\gamma \in [0, 1)$ is the discount factor and Θ^- are the (periodically updated) parameters of a target network. The training loss for a minibatch of transitions is the mean squared TD error,

$$\mathcal{L}(\Theta) \;=\; \mathbb{E}_{(s,a,r,s',\text{done})}\Big[\big(y - Q(s, a; \Theta)\big)^2\Big],$$

and Θ is updated by stochastic gradient descent on $\mathcal{L}$.

In our coarsening environment the action space is the discrete set of node indices, so the Q-network produces value estimates for candidate node selections given the graph-structured state s_t. To stabilize learning we use two standard DQN ingredients: (i) an experience replay buffer from which transitions are sampled i.i.d. for minibatch updates, and (ii) a target network with parameters Θ^- that are held fixed for several gradient steps before being synchronized with Θ. Action selection during interaction uses an ε-greedy policy (select $\arg\max_a Q(s_t, a; \Theta)$ with probability $1 - \varepsilon$, otherwise a random valid action). Episodes terminate according to the environment criteria described above; terminal transitions are handled by the $(1 - \text{done})$ factor in the target. This off-policy formulation is well suited to the discrete, graph-structured coarsening task and allows a GNN parameterization of Q to learn value estimates that generalize across problem instances.

A.3 Optimization with Proximal Policy Optimization

We learn a stochastic policy $\pi_\phi(a \,|\, s)$ parameterized by ϕ and a value function $V_\psi(s)$ parameterized by ψ. Interaction with the environment yields on-policy trajectories $\{(s_t, a_t, r_t, s_{t+1})\}$ collected under the behavior policy $\pi_{\phi_{\text{old}}}$. Here r_t denotes the scalar environment reward at step t, $\gamma \in [0, 1)$ the discount factor, and T the trajectory length used for advantage estimation. To stabilize policy updates we form the probability ratio

$$\rho_t(\phi) \;=\; \frac{\pi_\phi(a_t \mid s_t)}{\pi_{\phi_{\text{old}}}(a_t \mid s_t)},$$

and optimize the clipped surrogate objective

$$\mathcal{L}^{\text{CLIP}}(\phi) = \mathbb{E}_t\Big[\min\Big(\rho_t(\phi)\,\hat{A}_t,\; \text{clip}\left(\rho_t(\phi), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_t\Big)\Big],$$

where $\varepsilon > 0$ is the clipping parameter and $\hat{A}_t$ denotes an estimate of the advantage at time t. The complete training loss combines the negative surrogate (we minimize) with a value-function regression term and an entropy bonus:

$$\mathcal{L}(\phi, \psi) = -\mathcal{L}^{\text{CLIP}}(\phi) \;+\; c_v\,\mathbb{E}_t\big[\big(V_\psi(s_t) - \hat{R}_t\big)^2\big] \;-\; c_e\,\mathbb{E}_t\big[\mathcal{H}\big(\pi_\phi(\cdot \mid s_t)\big)\big],$$

where $\hat{R}_t$ is the return target (discounted sum of rewards), $\mathcal{H}$ denotes policy entropy, and $c_v, c_e \geq 0$ are weighting coefficients. Advantage estimates $\hat{A}_t$ are computed with generalized advantage estimation (GAE). Define the one-step TD residual

$$\delta_t \;=\; r_t + \gamma V_\psi(s_{t+1}) - V_\psi(s_t),$$

and set

$$\hat{A}_t \;=\; \sum_{l=0}^{T-t-1} (\gamma\lambda)^l\,\delta_{t+l},$$

with GAE parameter $\lambda \in [0, 1]$. In practice we normalize $\hat{A}_t$ across the batch to stabilize learning. Training proceeds in epochs: collect trajectories under $\pi_{\phi_{\text{old}}}$, compute $\hat{A}_t$ and $\hat{R}_t$, then perform multiple minibatch gradient steps on $\mathcal{L}(\phi, \psi)$ before updating $\phi_{\text{old}} \leftarrow \phi$. The policy and value networks are instantiated as GNNs that output masked action probabilities over valid node indices (so that already-coarsened or invalid nodes receive zero probability). This on-policy, clipped objective yields stable updates well suited to the discrete, graph-structured coarsening task and directly optimizes policy.

A.4 Terminal bonus functions

The bonus functions f_0–f_3 discussed in Sec. 3.3 are illustrated in Fig. 5, where $x_0 = 0.5$, $x_1 = 0.6$, $x_2 = 0.7$, $x_3 = 0.8$, $a_0 = 2$, $a_1 = 4$, $a_2 = 5$, and $a_3 = 6$ are used in f_0, $p = 0.65$ is used in f_2, and $k = 24$ is used in both f_2 and f_3.

A.5 Training rewards evolution

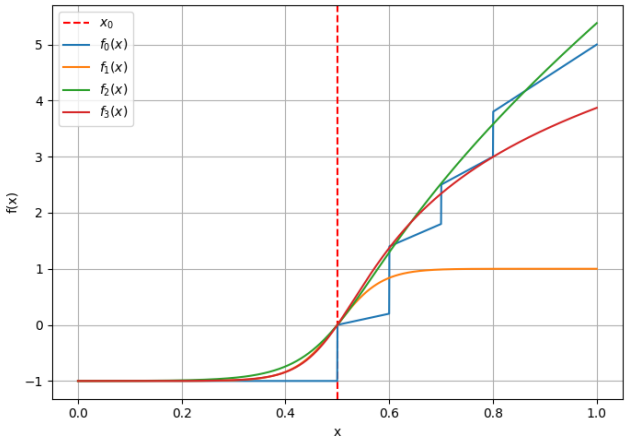


Figure 5: Terminal bonus functions

Figure 6: Training reward evolution for CoarseRL for different terminal reward functions. Mean reward versus training iteration for different diagonal-dominance thresholds $\theta_d \in \{0.6, 0.7, 0.8, 0.9\}$ (rows) and terminal reward functions f_0–f_4 (columns). All experiments use learning rate $= 0.001$ and $\lambda = 40.0$.