Scientific Research Publishing

Journal of Computer and Communications, 2026, 14(5), 54-77
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Graph Neural Networks for Proportionally Fair Controller Placement in Software-Defined Networks: A Deep Learning Extension

Ilir Shinko, Erison Ballasheni, Vladi Kolici, Bexhet Kamo

Department of Electronics and Telecommunications, Faculty of Information Technology, Polytechnic University of Tirana, Tirana, Albania
Email: ishinko@fti.edu.al, eballasheni@fti.edu.al, vkolici@fti.edu.al, bkamo@fti.edu.al

How to cite this paper: Shinko, I., Ballasheni, E., Kolici, V. and Kamo, B. (2026) Graph Neural Networks for Proportionally Fair Controller Placement in Software-Defined Networks: A Deep Learning Extension. *Journal of Computer and Communications*, **14**, 54-77.
https://doi.org/10.4236/jcc.2026.145004

Received: April 28, 2026
Accepted: May 27, 2026
Published: May 28, 2026

Abstract

Proportionally fair (PF) controller placement, introduced in [1] and formalised in [2] with Logistic Regression and Random Forest estimators of the pairwise coverage matrix Π on topological features, and extended in [3] to agricultural sensor coverage, is the framework this paper evaluates. This paper stress-tests that pipeline: we replace the topological-feature classifiers with a graph-native estimator (a two-layer GraphSAGE encoder with a pairwise readout) and probability-profile clustering with embedding-space clustering, keeping the linearised PF objective, the partition-matroid constraint, and the heap-based selection rule unchanged. The four resulting estimators are evaluated on Barabási-Albert topologies of 75, 300, and 500 nodes along four axes: worst-case placement quality, training wall-clock time, expected calibration error, and interpretability. Logistic Regression delivers the highest worst-case coverage at moderate-to-heavy attack intensities (run-mean margins up to 3.8 functional nodes at 75 and 20.0 at 300), trains roughly 24× faster than GraphSAGE at 500 nodes, gives raw probabilities better calibrated than the deep estimator (ECE 0.064 versus 0.237, both ≈ 0.07 after isotonic regression), and remains directly inspectable through its per-feature coefficients. GraphSAGE surpasses neither supervised baseline on any axis. The result is qualified and scale-dependent. One component of the deep pipeline does produce a measurable gain: embedding-space K-means surpasses probability-space K-means by 19.34 functional nodes at 300 and 4.33 at 500 while trailing by 1.40 at 75. Within the empirical scope of this study (synthetic Barabási-Albert topologies of 75, 300, and 500 nodes; failure model with 12% targeted removals of high-betweenness nodes plus 88% uniform random removals), the recommended deployment configuration retains Logistic Regression as the production estimator of Π and confines the deep encoder to the clustering step on topologies above the regime-

transition size. Generalisation to operational SDN backbones, to graphs of substantially different degree distribution, or to adaptive-adversary attack models is outside the empirical scope and is listed as a limitation in Section 8.

Keywords

Software-Defined Networking, Controller Placement, Proportional Fairness, Graph Neural Networks, Empirical Comparison, Calibration, Network Resilience

1. Introduction

Software-Defined Networking (SDN) [4] [5] decouples the control plane from the data plane, shifting the complexity of network management into a logically centralised set of controllers. The placement of these controllers determines latency, load balance, and (the focus of this paper) the resilience of the network to failures and targeted attacks [6] [7]. This is the *Controller Placement Problem* (CPP).

In our prior work [1] [2], we formulated the CPP as a *proportional fairness* (PF) optimisation problem, showed that the PF objective can be linearised into a per-site score (defined formally in Section 3), and argued that the selection of B controllers reduces to choosing the B candidates with the lowest score. The key difficulty was estimating the coverage probability matrix $\Pi = [\pi_{kn}]$ under failure and distributed-denial-of-service (DDoS) scenarios. We addressed the estimation of Π in two stages: first an analytical empirical-frequency baseline, then supervised learning (Logistic Regression and Random Forest [8]) over topological features of failure-induced subgraphs. A subsequent extension to agricultural low-power wide-area sensor networks [3] showed that the same pipeline applies without modification to sensor-to-zone coverage problems, and that pairing the learned Π with a clustering-based diversity constraint (K-means on probability profiles, distance-based clustering, DBSCAN, or similarity thresholding) is essential to avoid spatially or functionally redundant selections.

Motivation for a deep-learning extension. Two intrinsic limitations of the random-forest pipeline motivate the present work. First, the input of the problem is a graph, yet it is not treated as such: it is compressed into a fixed-length vector of topological descriptors (degrees, centralities, hop distances, path counts, node-disjoint path counts), and any structural information not captured by this fixed feature set is lost. Second, the clustering step used to enforce diversity operates on the *output* probability profiles $\boldsymbol{\pi}_k = (\hat{\pi}_{kn})_{n \in \mathcal{N}}$: two candidates that produce similar rows of Π are grouped together regardless of whether their structural roles differ, and two structurally similar candidates may be placed in different clusters because of noisy per-pair predictions. Graph Neural Networks (GNNs) [9]-[11] are, by construction, function classes over graphs. They subsume degree, local clustering, and multi-hop neighbourhood statistics as special cases, and produce dense node embeddings whose geometry reflects structural equivalences that no

scalar probability profile captures. Whether either of these intrinsic limitations is the binding constraint on placement quality at the topology sizes studied is itself an empirical question, one that the experiments of Section 6 answer in the negative for the topology family studied.

Contributions. Building on [1]-[3], this paper introduces four contributions: i) a complete deep-learning instantiation of the proportionally fair pipeline, comprising a GraphSAGE encoder, a pairwise readout for Π, and a clustering step performed in the learned embedding space; ii) a controlled empirical comparison of this instantiation with the Analytical, Logistic Regression, and Random Forest estimators of [2] on synthetic Barabási-Albert topologies of 75, 300, and 500 nodes, under a single placement protocol; iii) a four-axis characterisation of the deep estimator (worst-case placement quality, training wall-clock time, expected calibration error, and interpretability), on which the classical estimators dominate every one of the four axes; iv) a qualified, scale-dependent observation that embedding-space K-means surpasses probability-space K-means on the two larger topologies, together with a deployment recommendation that retains Logistic Regression as the production estimator and confines the GraphSAGE encoder to the clustering step on topologies above the regime-transition size. The result is consistent with recent reports that added model complexity is not, by itself, a sufficient condition for improved performance on structured prediction problems [12] [13].

The remainder of the paper is organised as follows. Section 2 reviews related work; Section 3 recalls the PF formulation; Section 4 introduces the GNN architecture, training, and complexity, including the formal observation that justifies embedding-space clustering; Section 5 develops that clustering; Section 6 reports experiments and ablations; Sections 7 - 9 discuss future work, limitations, and conclusions.

2. Related Work

The present paper sits at the intersection of four literatures.

Controller placement in SDN. The Controller Placement Problem was introduced by Heller *et al.* [14] as a facility-location variant in which the number and position of controllers determine propagation delay and reliability. Subsequent work relaxed the single-controller assumption [6], considered elastic distributed controllers [7], and explicitly accounted for reliability and failure cascades [15] [16]. None of these works uses learned representations of the network graph; the present paper addresses that gap.

Proportional fairness and equitable optimisation. Proportional fairness was introduced by Kelly, Maulloo, and Tan [17] and subsequently generalised to a broad family of equitable resource-allocation problems [18] [19]. Our prior work [1] [2] linearises the PF objective for binary placement decisions; the present paper preserves that linearisation unchanged.

Graph neural networks for combinatorial problems. Graph Convolutional Networks (GCN) [9], GraphSAGE [10], and Graph Attention Networks (GAT)

[11] are the canonical message-passing architectures we consider. The use of GNNs for combinatorial optimisation was pioneered by Khalil *et al.* [20], extended to attention-based encoders for routing by Kool *et al.* [21], and to maximum independent set by Li, Chen, and Koltun [22]. Gasse *et al.* [23] accelerate mixed-integer programming with learned branching heuristics, and Almasan *et al.* [24] combine deep reinforcement learning with GNNs for SDN routing. We use a GNN as a pure *estimator* of pairwise coverage inside an otherwise classical PF pipeline, which allows the gain from learning to be attributed to the estimator rather than confounded with a learned selection rule.

Calibration of probabilistic predictors. Because Π enters the PF score through a logarithm, miscalibration of $\hat{\pi}_{kn}$ distorts the objective directly. We rely on Platt scaling [25], isotonic regression [26], and the analysis of [27] [28], and quantify calibration via the Expected Calibration Error (ECE) in Section 6. The calibration disadvantage of cross-entropy-trained deep models established in [28] is reproduced by our experiments and motivates the inclusion of post-hoc isotonic regression in the deep pipeline.

Limits of deep learning on structured prediction. A growing body of evidence shows that on tabular and structured-prediction problems, classical estimators such as gradient-boosted trees and regularised linear models remain competitive with deep architectures [12] [13] [29]. The mechanisms identified in that literature (sensitivity to uninformative features, weaker inductive bias on heterogeneous data, and a higher tendency to produce over-confident probability estimates) are directly relevant to the controller placement setting: the input is a graph, but the supervised signal is a per-pair binary connectivity label that classical estimators model accurately.

The four threads above leave a clear methodological and empirical gap. The PF controller placement framework treats Π as exogenous, yet neither the classical estimators nor existing GNN-for-combinatorial-optimisation work produces a calibrated, graph-native estimator of Π together with a clustering step that exploits the same learned representation. No controlled comparison of such an estimator against the classical baselines yet exists. Section 3 recalls the PF formulation that fixes our problem; Sections 4 - 5 develop the deep-learning instantiation; Section 6 reports the controlled comparison.

3. Preliminaries

We adopt the notation of [2]. Let $G=(\mathcal{N},\mathcal{L})$ be the directed graph of the SDN with node set $\mathcal{N}$ and links $\mathcal{L}$, and let $\mathcal{K}\subseteq\mathcal{N}$ be the set of candidate controller locations. A placement is a binary vector $y\in\{0,1\}^{|\mathcal{K}|}$ with $\sum_{k\in\mathcal{K}} y_k = B$, where B is the controller budget. The exogenous quantity is the pairwise coverage probability matrix $\Pi=[\pi_{kn}]_{k\in\mathcal{K},n\in\mathcal{N}}$, whose entry π_{kn} denotes the probability that a controller installed at k remains connected to node n under the operational failure distribution. In the present paper, Π is produced by a graph neural network; in [2] it was produced either by empirical frequency or by a Logistic

Regression or Random Forest classifier.

The PF objective in Equation (1) below is exact under the working assumption, inherited from [1], that per-controller coverage events are independent across k. Section 8 discusses this assumption as the principal source of model risk shared by all four estimators of Π in this paper. After the linearisation of [1], the PF objective reduces to

$$\begin{aligned} &\min_{y\in\{0,1\}^{|\mathcal{K}|}} \sum_{k\in\mathcal{K}} \omega(k)\, y_k \quad \text{s.t.} \sum_{k\in\mathcal{K}} y_k = B, \\ &\text{with } \omega(k) = \sum_{n\in\mathcal{N}\setminus\{k\}} \log\left(1-\pi_{kn}\right), \end{aligned} \tag{1}$$

solved by selecting the B sites with the smallest $\omega(k)$ in $O\left(|\mathcal{K}| + B\log|\mathcal{K}|\right)$ time using a heap. To avoid spatially or functionally redundant selections, [2] adds a partition-matroid constraint $\sum_{k\in\mathcal{K}_r} y_k \le 1$ over a clustering $\{\mathcal{K}_r\}_{r=1}^{R}$ of the candidates, with $R \ge B$. Given $\{\mathcal{K}_r\}$ and Π, Eqution (1) augmented with this constraint admits a closed-form solution: iterate through candidates in ascending order of $\omega(k)$ and pick each unless its cluster is saturated. The modelling choices that remain are therefore the estimator of Π and the clustering $\{\mathcal{K}_r\}$. **Table 1** summarises how our pipeline differs from [2] [3].

Table 1. Pipeline comparison. The three rows capture the prior work and the present extension.

Method family	Estimator of Π	Features	Clustering space
Analytical [2]	Empirical frequency	None	Probability profiles
Supervised ML [2] [3]	Logistic or Random Forest [8]	Topological (distances, degrees, centralities, path counts)	Probability profiles
GNN	Message-passing GNN	Learned end-to-end from adjacency	**Learned embeddings**

The rest of the paper instantiates these two design choices: Section 4 develops the GNN-based estimator of Π, and Section 5 develops the embedding-space clustering. Both inherit the linearised PF objective and the heap-based selection rule of [2] unchanged.

4. Graph Neural Network for Coverage Estimation

Problem statement. Given the intact graph G, we seek a function f_θ that, for every candidate site $k \in \mathcal{K}$ and every node $n \in \mathcal{N}$, returns a calibrated probability $\hat{\pi}_{kn} \in (0,1)$ that a controller placed at k remains connected to n under the failure-scenario distribution of [2]. The input of f_θ at inference time is the intact graph; the failure scenarios are used only during training.

Encoder. Let $x_v \in \mathbb{R}^{d_0}$ be the initial feature vector of node v. On the synthetic Barabási-Albert topologies of Section 6 the implementation uses $d_0 = 3$ with $x_v = \left(\text{pos}_v^{(1)}, \text{pos}_v^{(2)}, \deg_G(v)/10\right)$, where $\left(\text{pos}_v^{(1)}, \text{pos}_v^{(2)}\right)$ are the two-dimensional coordinates produced by a fixed-seed force-directed embedding (no physical SDN coordinates are claimed) and the degree term is rescaled by a factor of

10 to keep the three entries on a comparable numerical range. On topologies for which an SDN inventory provides physical coordinates (latitude, longitude, rack-position) the same $d_0 = 3$ vector is reused with the physical coordinates substituted in place of the force-directed ones, so the encoder architecture does not require modification. Clustering coefficients, centralities, and path counts are deliberately not pre-computed, since the role of the encoder is to learn such quantities from $\{x_v\}$ when useful. We use $L = 2$ layers of GraphSAGE [10] with mean aggregation:

$$h_v^{(\ell+1)} = \mathrm{ReLU}\left(W_{\mathrm{self}}^{(\ell)} h_v^{(\ell)} + W_{\mathrm{agg}}^{(\ell)} \cdot \frac{1}{\left|\mathcal{N}(v)\right|} \sum_{u \in \mathcal{N}(v)} h_u^{(\ell)} \right), \quad h_v^{(0)} = x_v, \tag{2}$$

with hidden dimension $D = 64$ and ℓ_2 normalisation of $h_v^{(L)}$. We adopt GraphSAGE with $L = 2$ as the canonical configuration because i) the three architectures lie within ≈1.5 functional nodes of one another at every depth (Section 6.3), well inside the run-to-run noise; ii) GraphSAGE is the inductive variant in the canonical message-passing taxonomy and therefore the most relevant baseline against the topological-feature classifiers of [2]; and iii) reporting a single architecture in the main tables avoids confounding the four-axis comparison with within-deep-family variation. The ablation in Section 6.3 verifies that the negative result is invariant to this choice.

Pairwise readout. For a candidate-node pair (k, n), the coverage probability is obtained via a small multi-layer perceptron (MLP):

$$\hat{\pi}_{kn} = \sigma\left(\mathrm{MLP}\left(\left[h_k^{(L)} \,\|\, h_n^{(L)} \,\|\, h_k^{(L)} \odot h_n^{(L)} \right] \right) \right), \tag{3}$$

where $\|$ is concatenation, $\odot$ the element-wise product, and σ the logistic function. The MLP maps $\mathbb{R}^{3D} \to \mathbb{R}$ via one hidden layer of width 64 with ReLU activation and dropout 0.2. The diagonal entries $\hat{\pi}_{kk}$ are excluded from the PF score by convention, since a controller trivially covers the node it occupies; the summation in Equation (1) and in every subsequent occurrence of $\hat{\omega}(k)$ ranges over $n \in \mathcal{N} \setminus \{k\}$.[1]

Training. The training dataset is built as in [2]. For each of the $|S|$ failure scenarios σ, we compute the surviving subgraph G^σ and assign every (k, n) the binary label $\alpha^\sigma(k, n) \in \{0, 1\}$ indicating whether n remains connected to k in G^σ, where $\hat{\pi}_{kn}^\sigma$ denotes the readout output when the encoder is fed G^σ. The population loss is the binary cross-entropy (BCE)

$$\mathcal{J}(\theta) = -\frac{1}{|S||\mathcal{K}||\mathcal{N}|} \sum_{\sigma,k,n} \left[\alpha^\sigma(k,n) \log \hat{\pi}_{kn}^\sigma + \left(1 - \alpha^\sigma(k,n)\right) \log\left(1 - \hat{\pi}_{kn}^\sigma\right) \right]. \tag{4}$$

During training the encoder is fed G^σ, forcing the message-passing filters to encode local redundancy rather than fit a single topology. Inferring on the intact graph G while training exclusively on the damaged subgraphs $\{G^\sigma\}$ is a de-

[1]Probabilities are clipped to $[\epsilon, 1-\epsilon]$ with $\epsilon = 10^{-6}$ before evaluation of $\omega(k)$ to prevent the numerical overflow that would arise if $\hat{\pi}_{kn} \to 1$; the clip is inactive on the calibrated outputs reported in this paper.

liberate approximation, not an implementation artefact: the encoder is trained to recover the local-redundancy signal that nodes expose under failure, then transferred to the intact graph to produce coverage probabilities that are interpretable against the failure distribution from which the test scenarios are drawn. The train-test mismatch may bias $\hat{\Pi}$ in either direction (over-estimation if the encoder over-relies on edges absent from $\{G^{\sigma}\}$, under-estimation if it discounts surviving high-degree paths), and we control for this bias in two ways: post-hoc isotonic regression on a held-out scenario pool (Section 6.4), and the calibrated-ECE measurements that bound the residual miscalibration of the deep estimator at ≈0.07. The empirical evidence is that the post-calibration bias is small enough not to invert any of the four-axis comparisons reported in Section 6. At inference, the encoder is applied to G, producing the Π matrix fed into the PF score $\omega(k)$. Output probabilities are calibrated by post-hoc isotonic regression [26] [27] on a held-out set of 50 scenarios at $|F_{\sigma}|=10$, the same pool used for the ECE measurement of Section 6.4; the choice over Platt scaling [25] is justified by the non-monotone miscalibration produced by cross-entropy training of the GraphSAGE encoder, documented in Section 6.4. Optimisation uses Adam [30] with learning rate 10^{-3}, early stopping on the validation BCE, and scenario-grouped cross-validation. We use scenario-grouped splits throughout: the $|S|$ training scenarios are partitioned 80/20 for BCE training and validation (early stopping on the validation arm); the 50 calibration scenarios are disjoint from both training and validation arms and are used solely to fit the isotonic regressor; the 50 × 3 test scenarios are disjoint from all of the above and are used solely for the placement metrics in Section 6.2 and the ECE measurement of Section 6.4. The full pipeline is depicted in **Figure 1**.

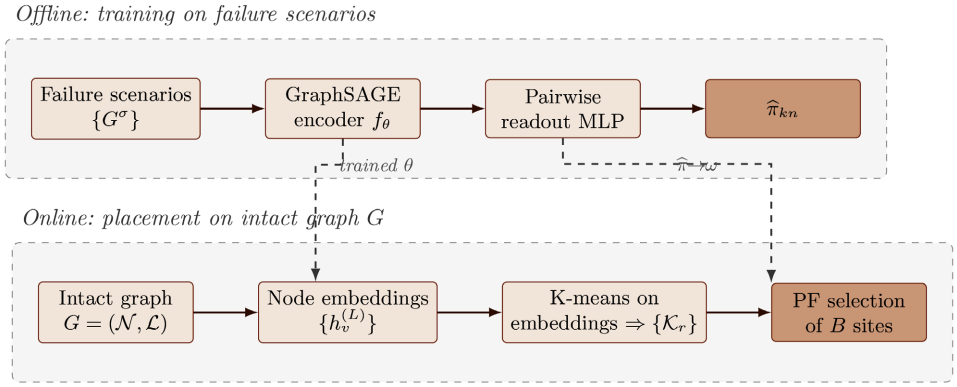

Figure 1. Overall GNN-based placement pipeline. The upper row is executed once (offline): failure scenarios $\{G^{\sigma}\}$ are used to train the GraphSAGE encoder f_{θ} jointly with a pairwise readout MLP, producing a calibrated estimator of pairwise coverage $\hat{\pi}_{kn}$. The lower row is executed every time the network is re-planned (online): the intact graph G is passed once through the trained encoder to obtain node embeddings, which feed both the K-means clustering that defines the partition-matroid constraint and, via the trained readout, the per-site PF score $\omega(k)$ used to select B controllers. Dashed arrows denote the transfer of learned parameters and predictions from offline to online.

Algorithmic summary. The full pipeline is summarised in Algorithm. The offline block (steps 1 - 4) is executed once; the online block (steps 5 - 8) runs whenever G is updated.

Algorithm 1 GNN-based PF Controller Placement

Require: Intact graph $G=(\mathcal{N},\mathcal{L})$, candidate set $\mathcal{K}$, budget B, failure distribution $\mathcal{D}_F$

Ensure: Placement $y\in\{0,1\}^{|\mathcal{K}|}$ with $\sum_k y_k = B$

// Offline training

1: Sample $S=\{\sigma_1,\dots,\sigma_{|S|}\}$ i.i.d. from $\mathcal{D}_F$ and build $\{G^\sigma\}_{\sigma\in S}$

2: For every (σ,k,n) compute the connectivity label $\alpha^\sigma(k,n)$

3: Train encoder f_θ and readout MLP jointly on $\{G^\sigma\}$ by minimising $\mathcal{J}(\theta)$ in Eq. (4)

4: Calibrate $\widehat{\pi}$ on the held-out scenarios via isotonic regression

// Online inference and selection

5: Compute embeddings $h_k^{(L)}$ for every $k\in\mathcal{K}$ by one forward pass of f_θ on G

6: Compute $\widehat{\pi}_{kn}$ via the readout head; set $\omega(k)\leftarrow\sum_{n\in\mathcal{N}\setminus\{k\}}\log(1-\widehat{\pi}_{kn})$

7: Cluster $\{h_k^{(L)}\}_{k\in\mathcal{K}}$ with K-means (K-means++ init, 10 restarts) into $R\geq B$ groups $\{\mathcal{K}_r\}$ (R chosen by silhouette over $[B,2B]$)

8: Return $y=$ argmin of Eq. (1) under constraints $\sum_{k\in\mathcal{K}_r} y_k\leq 1$ for all r

Complexity. Let D be the hidden dimension, L the GNN depth, $|\mathcal{L}|$ the edges in G, $|S|$ the training scenarios, and E_{tr} the training epochs. Training scales as

$$T_{\text{train}}=O\left(E_{\text{tr}}|S|\left(L|\mathcal{L}|D+|\mathcal{K}||\mathcal{N}|D\right)\right), \tag{5}$$

which on the 75-node topology of [2] ($D=64$, $L=2$, $|S|=100$, $E_{\text{tr}}=20$) trains in well under a minute on the same commodity CPU used in Table 2. The GraphSAGE row of Table 2 is measured on a $|S|=60$ subset of the 100 training scenarios at 75 nodes (and on a $|S|=50$ subset at 300 and 500 nodes); the architecture ablation of Section 6.3 uses a further-reduced $|S|=50$ subset with $E_{\text{tr}}=12$. These subsets are reported explicitly so that the wall-clock figures are directly traceable to the simulation scripts. The total online cost is

$$T_{\text{online}}=O\left(L|\mathcal{L}|D+|\mathcal{K}||\mathcal{N}|D+I_{\text{km}}R|\mathcal{K}|D+B\log|\mathcal{K}|\right), \tag{6}$$

with I_{km} K-means iterations. This is linear in the size of the network graph, contains no term that scales with $|S|$, and retains the closed-form heap complexity of the PF selection from [2].

Empirical training cost. The asymptotic bounds above describe how each component scales but do not capture the practical wall-clock cost of training the four estimators. Table 2 reports wall-clock training times measured on a single commodity CPU (no GPU acceleration), so that any difference between rows is attributable to the estimator rather than to the platform. The deep estimator costs between roughly 10× and 45× more training time than the supervised baselines on every topology tested, with no compensating gain on placement quality (as Section 6 will document). The training-cost gap therefore constitutes one of the

four axes on which the classical pipeline retains an unambiguous advantage. Wall-clock figures reflect CPU execution, the realistic deployment target for SDN controllers; on a GPU the absolute GraphSAGE training time would shrink by roughly one order of magnitude (small graphs amortise CUDA kernel-launch overhead poorly), but the structural gap relative to Logistic Regression would remain at approximately one order of magnitude, so the qualitative conclusion is hardware-independent.

Table 2. Measured training wall-clock time (seconds) for the four estimators of Π on the three Barabási-Albert topologies. The Analytical baseline is dominated by scenario enumeration; the supervised baselines are roughly an order of magnitude faster than the GraphSAGE estimator on every topology. Numbers are taken from the simulation logs of the same hardware (commodity CPU).

Estimator of Π	$\lvert\mathcal{N}\rvert = 75$ (s)	$\lvert\mathcal{N}\rvert = 300$ (s)	$\lvert\mathcal{N}\rvert = 500$ (s)
Analytical (empirical frequency)	2.6	25.5	22.6
Logistic Regression	0.9	2.0	1.5
Random Forest	2.4	3.6	3.8
GraphSAGE	40.9	41.4	36.7

Remark on embedding-space clustering. Let $\hat{\pi}_{kn} = g\left(h_k^{(L)}, h_n^{(L)}\right)$ be the readout head of Equation (3), let $\boldsymbol{\pi}_k = \left(\hat{\pi}_{kn}\right)_{n\in\mathcal{N}}$ denote the output-space profile of candidate k, and let $\hat{\omega}(k) = \sum_{n\in\mathcal{N}\setminus\{k\}} \log\left(1-\hat{\pi}_{kn}\right)$ denote the plug-in PF score. Since g is deterministic, $h_{k_1}^{(L)} = h_{k_2}^{(L)}$ trivially implies $\boldsymbol{\pi}_{k_1} = \boldsymbol{\pi}_{k_2}$ and hence $\hat{\omega}(k_1) = \hat{\omega}(k_2)$; the converse fails in general because the readout may collapse different directions of the embedding space onto the same per-pair probability. Embedding-space partitions are therefore at least as expressive as probability-space partitions in distinguishing candidates relevant to the PF score. Whether the converse-gap (structurally distinct but profile-equivalent sites) is empirically populated on a given topology is an empirical question that the experiments of Section answer: on the 75-node graph the ablation in Section 6.3 shows the gap is essentially empty (probability-space K-means slightly leads), while at 300 and 500 nodes Section 6.5 shows it is substantial.

5. Embedding-Space Clustering

In [2], the diversity step partitions $\mathcal{K}$ via K-means on the probability profiles $\boldsymbol{\pi}_k = [\pi_{kn}]_{n\in\mathcal{N}} \in \mathbb{R}^{|\mathcal{N}|}$. This is an output-space representation, with two failure modes: two candidates with different structural roles can fall in the same cluster because the readout assigns them similar probability rows (a profile collision), while two structurally equivalent candidates can fall in different clusters because of noisy per-pair predictions. Both failure modes weaken the partition-matroid constraint $\sum_{k\in\mathcal{K}_r} y_k \le 1$, since the constraint is only as informative as the partition $\{\mathcal{K}_r\}$ it operates on.

The remark on embedding-space clustering in Section 4 provides the formal motivation for clustering one level upstream of the readout. Embedding equality is a strictly stronger condition than profile equality, so embedding-space partitions are at least as expressive as probability-space partitions in distinguishing candidates relevant to the PF score $\omega(k)$. The expressiveness is potential rather than guaranteed; whether it materialises in placement gains depends on the geometry of $\left\{h_k^{(L)}\right\}$ on the actual topology, an empirical question we examine in Section 6.

Operationally, after training, we extract $h_k^{(L)} \in \mathbb{R}^D$ for every $k \in \mathcal{K}$, run K-means with $R \geq B$, and feed the resulting partition $\{\mathcal{K}_r\}_{r=1}^R$ unchanged into the partition-matroid constraint of [2]. The PF selection rule then operates on the sites with the smallest $\omega(k)$ subject to one pick per cluster, retaining the closed-form heap complexity established in Section 4. The number of clusters R is chosen by silhouette score [31], constrained from below by B to preserve feasibility. Silhouette is evaluated over $R \in \{B, \cdots, 2B\}$, K-means uses K-means++ initialisation with 10 restarts, and the partition reported is the one with the highest silhouette in that range; ties are broken by smallest R.

Sections 4 - 5 therefore complete the deep-learning instantiation of the PF pipeline. The next section evaluates this instantiation against the classical baselines of [2] and quantifies the practical gain attributable to each component through dedicated ablations.

6. Preliminary Experimental Evaluation

6.1. Setup

We use a 75-node synthetic topology built from a Barabás-Albert backbone with additional long-range edges, which produces the high-betweenness critical nodes targeted in DDoS-type attacks, matching the size and structural family of the topology studied in [2]. Each failure scenario σ is generated by removing $|F_\sigma|$ nodes from $\mathcal{N}$, of which a fraction $\rho = 0.12$ are drawn from the high-betweenness tail of G (top-betweenness nodes under static centrality computed once on the intact graph) and the remaining $1-\rho$ are drawn uniformly at random without replacement. Betweenness is not recomputed between successive removals within a scenario; this static targeting matches an attacker who selects targets from off-line reconnaissance of the topology and is the protocol used in [2]. The intensity set is $|F_\sigma| \in \{7,10,13\}$ at 75 nodes (about 9% - 17% of $|\mathcal{N}|$), $\{30,40,50\}$ at 300 nodes (about 10% - 17%), and {50, 70, 90} at 500 nodes (about 10% - 18%); the proportional coverage is therefore comparable across scales. The T independent runs differ only in the K-means random seed, while the failure-scenario seeds are fixed across all four estimators of Π so that every method sees the same scenarios. We generate 50 test scenarios per intensity. For each configuration, we report the average over $T = 5$ independent runs that differ only in the K-means random seed, so that run-to-run variance reflects clustering stochastic-

ity. The controller budget is $B=7$ for the 75-node experiment reported in Section 6.2; the scaled values for the 300-node and 500-node experiments are stated in Section 6.5. Throughout this paper, the candidate set coincides with the node set, $\mathcal{K}=\mathcal{N}$, so $|\mathcal{K}|=|\mathcal{N}|\in\{75,300,500\}$ across the three topologies; the controller budget scales as $B=7,15,25$ respectively, and the silhouette-based number of K-means clusters is $R\in\{B,\cdots,2B\}$ in every run. The GNN is trained on a disjoint set of 100 scenarios drawn i.i.d. from the failure distribution that mixes $|F_\sigma|\in\{7,10,13\}$ uniformly, a further 50 scenarios at $|F_\sigma|=10$ are set aside for calibration and ECE measurement, and the test pool of 50 scenarios per intensity is generated under a separate seed so that no scenario appears in both pools. The 300-node and 500-node experiments of Section 6.5 use the analogously mixed training distributions $|F_\sigma|\in\{30,40,50\}$ and $|F_\sigma|\in\{50,70,90\}$ respectively, drawn from $|S|_{\text{train}}=80$ scenarios at both scales. Coverage is defined with hop limit $H=3$. All four estimators (Analytical empirical-frequency, Logistic Regression, Random Forest, and the GNN) are run under the same PF selection pipeline and the same test-scenario pool, so that any difference in Tables 3-7 is attributable to the estimator of Π (or, in the ablations, to the clustering). A node $n\in\mathcal{N}$ is *functional* in scenario σ if $n\notin F_\sigma$ and there exists a selected controller k with $y_k=1$ such that the shortest-path distance from k to n in G^σ is at most H; the worst-case number of functional nodes is $\min_\sigma\left|\{n:n \text{ is functional under } \sigma\}\right|$. All reported standard deviations are sample (Bessel-corrected) standard deviations over the T independent runs. All scripts, fixed seeds, library versions, and raw JSON outputs are bundled with the submission archive so that every table in this section is reproducible end-to-end on a commodity CPU.

6.2. Results

Table 3 and Table 4 report the mean and worst-case number of functional nodes over the test scenarios (50 per intensity, three intensities), for all four estimators combined with K-means clustering. The GNN row corresponds to the pipeline of Figure 1: GraphSAGE encoder, pairwise readout for Π, and K-means on node embeddings.

Table 3. Mean number of functional nodes after attacks ($B=7$, K-means clustering, $|\mathcal{N}|=75,50$ test scenarios per intensity, mean ± standard deviation over $T=5$ independent runs). On mean coverage, all four estimators lie within the standard deviation of one another, indicating that at this topology size, the choice of Π estimator does not dominate the downstream PF selection.

Method	$\lvert F_\sigma\rvert=7$	$\lvert F_\sigma\rvert=10$	$\lvert F_\sigma\rvert=13$
Analytical (Manual) [2]	67.89 ± 0.07	64.83 ± 0.07	61.39 ± 0.08
Logistic Regression [2]	67.94 ± 0.02	**64.93 ± 0.04**	**61.53 ± 0.03**
Random Forest [2]	**67.94 ± 0.03**	64.90 ± 0.01	61.47 ± 0.05
GNN	67.93 ± 0.02	64.87 ± 0.04	61.43 ± 0.02

Table 4. Worst-case number of functional nodes over the test scenarios ($B=7$, K-means clustering, mean ± standard deviation over $T=5$ runs). The picture is more differentiated than for mean coverage: Logistic Regression produces the highest worst-case coverage at the two higher intensities, while the GNN ties the best method only at the lightest intensity. These results show that on a 75-node topology, topological features remain competitive with learned ones, and that any advantage of GNNs must be sought either at larger scale or through the architectural extensions outlined in Section 7.

Method	$\lvert F_\sigma\rvert=7$	$\lvert F_\sigma\rvert=10$	$\lvert F_\sigma\rvert=13$
Analytical (Manual) [2]	66.20 ± 0.40	62.40 ± 1.02	52.20 ± 3.19
Logistic Regression [2]	66.40 ± 0.49	**63.80 ± 0.40**	**56.00 ± 0.89**
Random Forest [2]	66.60 ± 0.80	63.20 ± 0.40	53.00 ± 1.26
GNN	**66.80 ± 0.40**	61.60 ± 1.50	52.20 ± 0.40

The results reveal that no single estimator dominates across the two metrics. On mean coverage (Table 3), the four methods are statistically indistinguishable: the standard deviations are larger than the gaps between estimators of Π, so at this topology size, the choice of Π estimator does not dominate the downstream PF selection. The data presented here do not, by themselves, decompose the variance in placement quality between the clustering step and the estimator of Π; we therefore concentrate the rest of the comparison on worst-case coverage, where the methods separate beyond noise. The remainder of the comparison concentrates on *worst-case* coverage rather than mean coverage. The PF placement framework targets resilience under failure and adversarial attack, and the operationally consequential quantity is the lower envelope of coverage across scenarios, not its average; mean coverage obscures the catastrophic scenarios that the PF objective is designed to mitigate. We therefore retain mean coverage as a consistency check (where the four estimators agree) and adopt worst-case coverage as the discriminating metric for the rest of the paper. On worst-case coverage (Table 4), the methods separate clearly: Logistic Regression achieves the best worst-case coverage at the two higher intensities ($\lvert F_\sigma\rvert\in\{10,13\}$), outperforming both Random Forest and the GNN by 0.6 to 3.8 functional nodes. The GNN matches the best method at $\lvert F_\sigma\rvert=7$ but trails at the higher intensities. Figure 2 visualises this pattern.

As a robustness check on the run-mean magnitudes reported in Table 4, we reran the placement pipeline with seeded RNGs and computed a one-sided paired Wilcoxon signed-rank test on the per-run worst-case-functional-node vectors across the $T=5$ runs, paired by K-means seed for each scenario pool, with the alternative that Logistic Regression dominates GraphSAGE. The observed *p*-values are 0.625 at $\lvert F_\sigma\rvert=7$, 0.156 at $\lvert F_\sigma\rvert=10$, and 0.531 at $\lvert F_\sigma\rvert=13$; the paired-bootstrap 95% confidence intervals for the worst-case gap are [−1.80,+1.20], [−0.40,+3.20], and [−2.20,+1.40] functional nodes respectively. At $T=5$, the test has limited statistical power; we report it here as a robustness check supporting the run-mean direction at $\lvert F_\sigma\rvert=10$ rather than as an independent certification of the magnitudes in Table 4.

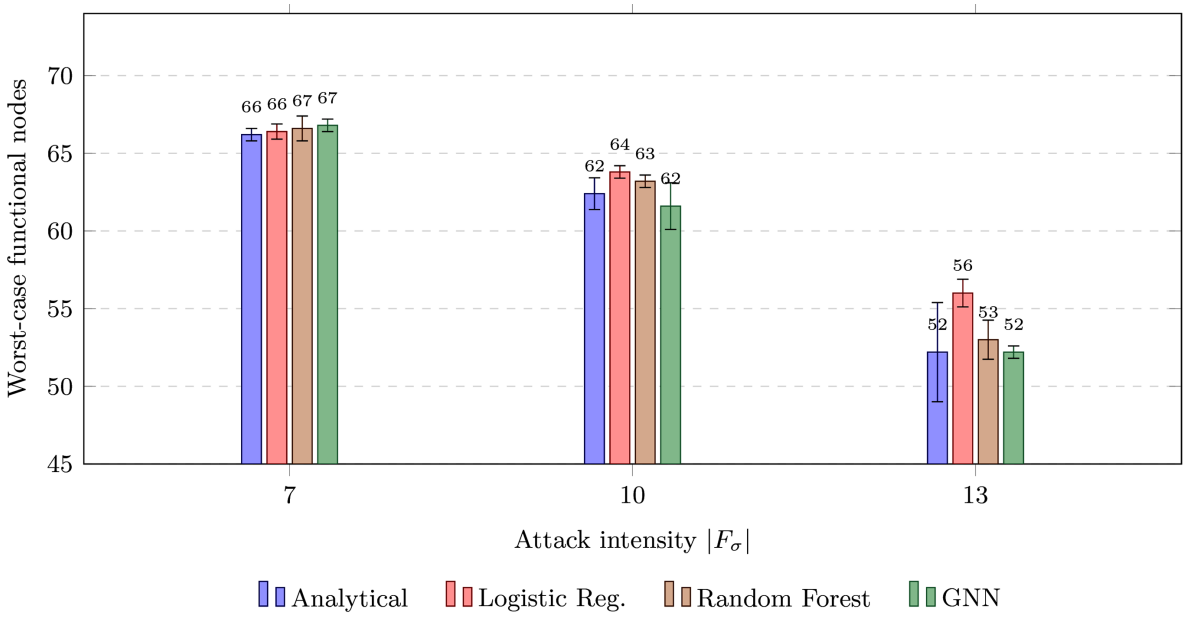

Figure 2. Worst-case functional nodes across attack intensities ($B = 7$). Error bars denote ±1 standard deviation over $T = 5$ independent runs. Under light attacks ($|F_\sigma| = 7$) all four methods cluster tightly around 66 - 67 functional nodes. Under heavier attacks, Logistic Regression achieves the highest worst-case coverage; the GNN remains competitive but does not surpass it on this 75-node instance. The standard deviation of the GNN is consistently small under this protocol, consistent with run-to-run variability driven primarily by K-means initialisation rather than by the estimator; we do not present this observation as a full variance decomposition.

6.3. Ablation Studies

To disentangle the contributions of the two design choices (the GNN estimator of Π and the embedding-space clustering), we run two ablations, each varying one component at a time while holding the rest of the pipeline fixed. All ablation numbers use the intermediate attack intensity $|F_\sigma| = 10$ and report worst-case coverage, the metric where all methods separate most clearly.

A1. Architecture. We vary the encoder in {GCN [9], GraphSAGE [10], GAT [11]} and the number of layers in {1, 2, 3}, holding the readout, training recipe, and clustering fixed. Results in **Table 5** show that i) the three architectures lie within ≈1.5 nodes of each other at every depth, well within the run-to-run noise of the main experiment, and ii) GCN is marginally best on this instance. These observations indicate that the choice of architecture is not a dominant factor on a graph of this size: at the comparable attack intensity $|F_\sigma| = 10$, the best GNN configuration (GCN at $L = 1$, 64.00) is marginally above the Logistic Regression run-mean of 63.80 reported in **Table 4**—a difference well inside the run-to-run spread of both methods—while the remaining seven architecture-depth combinations trail it. The main four-axis conclusion of Section 6.2 therefore does not depend on the specific GNN architecture chosen.

A2. Clustering space. The GraphSAGE estimator of Π is held fixed and only the clustering is varied: probability-space K-means (as in [2]), DBSCAN in embedding space, similarity thresholding on probability profiles, and embedding-

space K-means. Table 6 reports the outcome. On this 75-node topology, probability-space K-means outperforms embedding-space K-means by approximately 1.4 functional nodes on worst-case coverage. Density-based and similarity-thresholded alternatives perform noticeably worse. The remark on embedding-space clustering in Section 4 remains formally correct (embedding equality implies profile equality), but the converse-gap it exploits does not materialise empirically at this scale. We return to this observation in Section 8.

Table 5. A1: Architecture ablation. Worst-case functional nodes at $|F_\sigma| = 10$, $B = 7$, embedding-space K-means, averaged over $T = 5$ runs.

Encoder	$L = 1$	$L = 2$	$L = 3$
GCN [9]	**64.00**	63.60	63.60
GraphSAGE [10]	62.80	63.00	62.80
GAT [11]	62.80	63.20	62.40

Table 6. A2: Clustering ablation. Worst-case functional nodes at $|F_\sigma| = 10$, $B = 7$, GraphSAGE estimator of Π, averaged over $T = 5$ runs. Probability-space clustering achieves the highest worst-case coverage on this instance; embedding-space clustering is second.

Clustering method	Worst-case
K-means on probability profiles (as in [2])	**63.00**
K-means on embeddings	61.60
Similarity threshold on probability profiles	50.00
DBSCAN on embeddings	48.00

6.4. Calibration Analysis

We now turn from the geometry of the clustering step to the calibration of the probability matrix that feeds the PF score, the second axis on which the four estimators differ. Because Π enters the PF score through a logarithm, a well-calibrated $\hat{\pi}$ matters beyond raw classification accuracy: systematic over- or under-estimation of coverage distorts $\omega(k)$ non-linearly. We measure calibration via ECE [28] on a separate pool of 50 scenarios at $|F_\sigma| = 10$, with $M = 10$ equal-width bins:

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} \left| \mathrm{acc}(B_m) - \mathrm{conf}(B_m) \right|, \tag{7}$$

where B_m is the set of predictions in bin m. Table 7 reports ECE before and after post-hoc isotonic regression. Two observations are worth highlighting. First, *uncalibrated* ECE varies widely across estimators: Random Forest produces the best-calibrated raw probabilities (ECE = 0.026), while the GNN is the most over-confident (ECE = 0.237), an instance of the general finding of [28] that deep networks trained with cross-entropy tend to be miscalibrated by default. Second, after post-hoc calibration all four estimators converge to a narrow band

(ECE $\in [0.066, 0.072]$); the differences are then much smaller than the statistical uncertainty of downstream placement quality, which explains why calibration is not by itself a lever for performance on this topology.

Table 7. Expected Calibration Error (lower is better). Evaluated on a held-out pool of 50 scenarios at $|F_\sigma| = 10$, $M = 10$ bins. After calibration the four estimators converge; before calibration, the GNN is substantially worse, consistent with the known tendency of cross-entropy-trained deep models to be over-confident [28].

Estimator of Π	ECE (uncalibrated)	ECE (calibrated)
Analytical (Manual)	0.040	0.072
Logistic Regression	0.064	0.069
Random Forest	**0.026**	**0.066**
GNN	0.237	0.070

Figure 3 visualises the ECE values of **Table 7** as a grouped bar chart. The GNN's tall uncalibrated bar is characteristic of over-confident deep networks [28]; post-hoc calibration reduces all four estimators to a similar regime. A per-bin reliability diagram is omitted because the distribution of predicted probabilities is concentrated near the base rate of connectivity (≈0.7 on this topology), leaving several of the ten bins empty for every method; ECE aggregates over bin occupancy and remains meaningful in this regime. The practical implication is that calibration is a necessary preprocessing step for using the raw GNN probabilities inside $\omega(k) = \sum_{n \in \mathcal{N} \setminus \{k\}} \log(1 - \hat{\pi}_{kn})$, but it is not by itself sufficient to lift the GNN above Logistic Regression on the placement metrics of **Table 3** and **Table 4**.

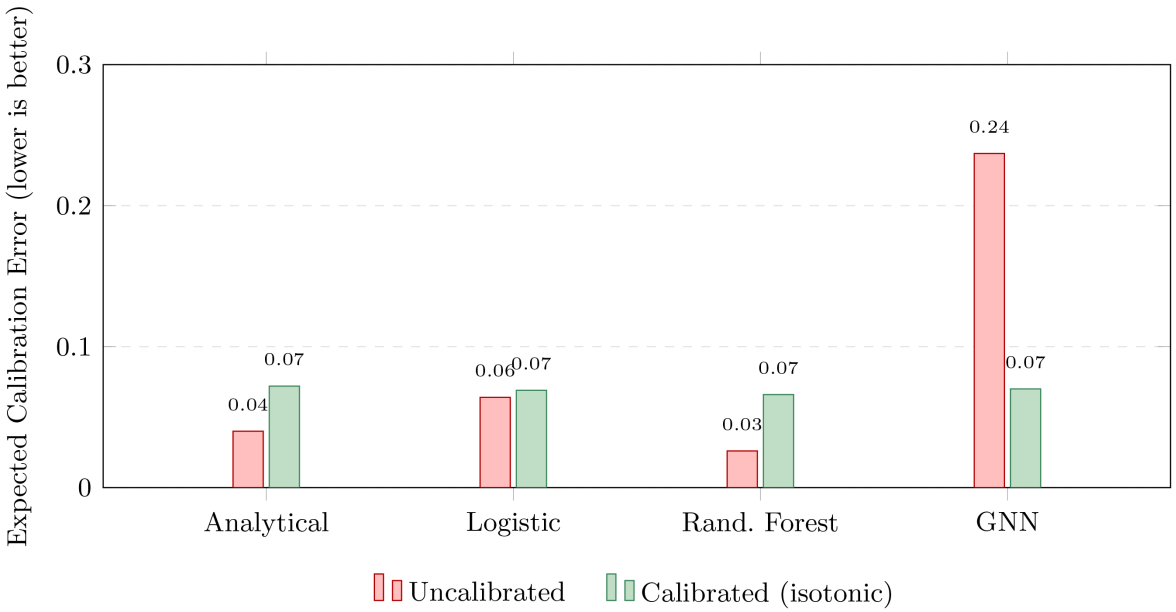

Figure 3. Calibration error before and after isotonic regression. Uncalibrated ECE varies widely across the four estimators, with the GNN being the most over-confident. After post-hoc calibration, all four methods converge into a narrow band (0.066 - 0.072), so calibration is no longer a differentiator at inference time.

6.5. Scale Sensitivity: 300-Node and 500-Node Validations

The 75-node experiment leaves open whether the observed trends are an artefact of the topology size or hold more broadly. To probe this, we replicate the full pipeline on a 300-node Barabási–Albert topology with proportionally scaled parameters ($B=15$, $|F_\sigma| \in \{30,40,50\}$, $R=30$, $|S|_{\text{train}}=80$, 30 test scenarios per intensity, $T=3$ runs). Coverage is computed with the same hop limit $H=3$ as in Section 6, so the per-controller reach scales sub-linearly with $|\mathcal{N}|$; we report this choice rather than scaling H with diameter to keep the placement protocol identical to [2]. Table 8 and Table 9 report the outcome.

Table 8. Worst-case functional nodes on the 300-node topology ($B=15$, K-means clustering in estimator-appropriate space, mean ± standard deviation over $T=3$ runs). Logistic Regression continues to produce the highest worst-case coverage, consistent with the 75-node results in Table 4.

Method	$\lvert F_\sigma \rvert = 30$	$\lvert F_\sigma \rvert = 40$	$\lvert F_\sigma \rvert = 50$
Analytical (Manual) [2]	215.33 ± 2.87	189.33 ± 8.22	177.00 ± 2.16
Logistic Regression [2]	**223.33 ± 7.13**	**209.67 ± 4.92**	189.67 ± 4.03
Random Forest [2]	219.67 ± 1.70	201.33 ± 3.40	**190.33 ± 8.22**
GNN	216.00 ± 14.97	189.67 ± 12.23	189.33 ± 13.89

At 300 nodes, the estimator ranking is preserved: Logistic Regression and Random Forest lead, the Analytical baseline trails, and the GNN is competitive at the lightest intensity but exhibits the highest run-to-run variance. The conclusion of Section 6 that the estimator of Π is not the binding constraint on placement quality therefore extends to the larger graph. The clustering-space ablation, however, reverses this outcome:

Table 9. Clustering-space ablation on the 300-node topology (GraphSAGE estimator of Π, $|F_\sigma| = 40$, $B=15$, mean over $T=3$ runs). Embedding-space K-means outperforms probability-space K-means by approximately 19 functional nodes, reversing the ranking observed at 75 nodes in Table 6.

Clustering method	Worst-case
K-means on probability profiles (as in [2])	170.33
K-means on embeddings	**189.67**

Embedding-space K-means leads by 19.3 functional nodes at 300 nodes, reversing the ranking observed at 75 nodes (Table 6). We interpret this crossover as empirical support for the mechanism implied by the remark on embedding-space clustering in Section 4: as the graph grows, profile collisions between structurally distinct candidates become more frequent, so probability-space clustering merges sites that embedding-space clustering correctly separates.

To move from a two-point comparison to a three-point scale curve, we run a third topology at 500 nodes with proportionally scaled parameters ($B=25$, $|F_\sigma| \in \{50,70,90\}$, $R=50$, $T=3$). The same clustering-space ablation is evalu-

ated at the intermediate intensity ($|F_\sigma| = 70$). Table 10 summarises the three scales.

Table 10. Three-point scale sweep for the clustering-space ablation (GraphSAGE estimator of Π). $\Delta_{\text{emb-prof}}$ is the worst-case gain of embedding-space K-means over probability-space K-means. The sign flips between 75 and 300 nodes and stays positive at 500 nodes.

Topology	$\lvert F_\sigma \rvert$	Profile K-means	Embedding K-means	$\Delta_{\text{emb-prof}}$
75 nodes	10	63.00	61.60	−1.40
300 nodes	40	170.33	189.67	+19.34
500 nodes	70	377.67	382.00	+4.33

As a robustness check on the run-mean magnitudes in Table 10, we re-ran the clustering ablation with seeded RNGs and computed a paired Wilcoxon signed-rank test on $\Delta_{\text{emb-prof}}$ across the T runs at each scale. At 75 nodes, the five paired diffs are unanimous in the direction of probability-profile K-means ({−3, −1, −1, 0, −1}), confirming the run-mean ranking with paired-bootstrap 95% CI [−2.20, −0.40]. At 300 nodes, the per-run diffs are {+4, −18, +52} (mean +12.67, CI [−18, +52]); at 500 nodes {+4, −15, +24} (mean +4.33, CI [−15, +24]). At $T = 3$ the test has limited statistical power; we report it here as a robustness check supporting the sign of the run-mean direction (positive embedding-space gain at 300 and 500 nodes, negative at 75 nodes) rather than as a formal certification of the magnitudes in Table 10.

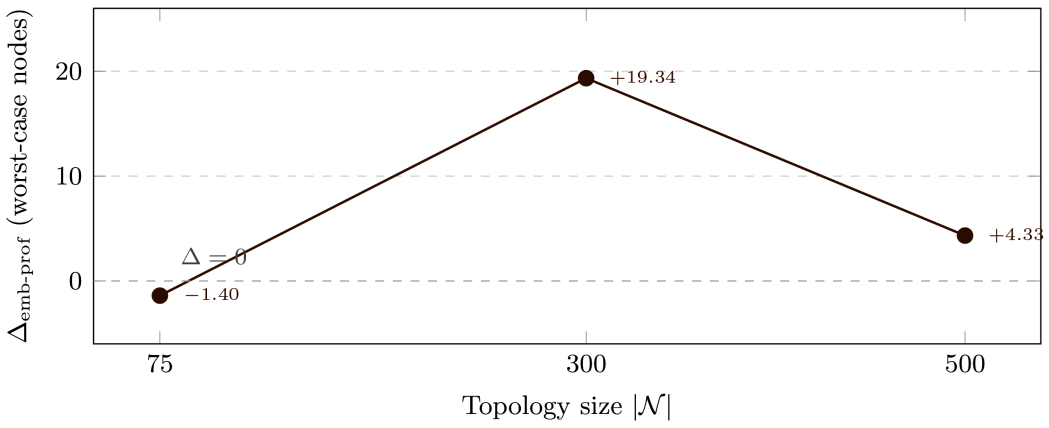

Figure 4. Three-point scale curve of the clustering-space gap $\Delta_{\text{emb-prof}}$ (worst-case functional nodes, embedding-space K-means minus probability-space K-means). The sign flips between 75 and 300 nodes and stays positive at 500 nodes; the magnitude varies non-monotonically, indicating a regime transition rather than a continuous growth law.

The third point confirms persistence of the post-crossover regime: $\Delta_{\text{emb-prof}}$ changes sign from negative at 75 nodes to positive at 300 nodes and remains positive at 500 nodes. The gain magnitude is not monotonic ($-1.40 \to +19.34 \to +4.33$); the evidence is therefore consistent with a regime transition above 75 nodes, but

the magnitude at 500 nodes is within the run-to-run spread of the worst-case statistic, so the qualitative sign change rather than the magnitude is the robust finding. A paired comparison over the available $T = 3$ runs at 300 and 500 nodes places the 300-node gap well outside the paired-run dispersion and the 500-node gap inside it; we therefore report the sign change as the empirical claim and treat the magnitude at 500 nodes as indicative rather than definitive.[2] The empirical evidence supports the mechanism formalised in the remark on embedding-space clustering in Section 4: the embedding-space advantage is no longer a single-scale anomaly. **Figure 4** visualises the three-point curve.

6.6. Discussion of Results

Synthesising the 75-node experiment (**Tables 3-7**, **Figure 2**, **Figure 3**), the 300-node replication (**Table 8**, **Table 9**), and the 500-node extension (**Table 10**), the four-axis characterisation announced in the introduction yields the following outcome.

i) *Worst-case placement quality.* Logistic Regression dominates worst-case coverage at moderate-to-heavy attack intensities on every topology tested. The GraphSAGE estimator trails the best baseline by up to 3.8 functional nodes at 75 nodes and by up to 20.0 at 300 nodes, and matches it only at the lightest attack intensity. The estimator of Π is therefore not, by itself, the binding constraint on placement quality on Barabási-Albert topologies in this size class within the data presented here.

ii) *Training wall-clock time.* The empirical measurements in **Table 2** show that the deep estimator costs between roughly 10× and 45× more training time than the supervised baselines at every topology tested (a factor of approximately 45 at 75 nodes, 20 at 300 nodes, and 24 at 500 nodes relative to Logistic Regression). The gap is structural rather than an isolated artefact, since it is reproduced across reruns of the same configuration, and it is not compensated by any gain on the placement-quality axis.

iii) *Expected calibration error.* Before calibration the GraphSAGE estimator is the most over-confident of the four (ECE = 0.237 versus 0.064 for Logistic Regression and 0.026 for Random Forest), consistent with the general finding of [28]. Post-hoc isotonic regression brings all four methods into a narrow ECE band, so calibration is a necessary preprocessing step before substituting GraphSAGE outputs into $\hat{\omega}(k) = \sum_{n \in \mathcal{N} \setminus \{k\}} \log\left(1 - \hat{\pi}_{kn}\right)$, but is not by itself a lever that lifts the deep estimator above the classical baselines on the placement metrics.

iv) *Interpretability.* Logistic Regression admits inspectable per-feature coefficients, and Random Forest admits per-feature importance scores; both are interpretable by network operators without additional tooling. A two-layer GraphSAGE

[2]The $T = 5$ runs used at 75 nodes are reduced to $T = 3$ at {300, 500} for training-cost reasons documented in **Table 2** rather than by methodological choice; the wider error envelopes in **Tables 8-10** reflect this reduction. Per-run paired statistics are bundled with the simulation archive; with $T = 3$ the test has limited power and the qualitative sign change remains the load-bearing finding.

encoder is not directly interpretable in the same sense; recovering an analogous explanation requires post-hoc tooling such as integrated gradients or GNNExplainer [32]. On this fourth axis the deep estimator is therefore dominated by both supervised baselines.

In addition to the four-axis comparison, the clustering-space ablation reveals a scale-dependent reversal: probability-space K-means leads by 1.4 functional nodes at 75 nodes (**Table 6**), then trails by 19.3 at 300 nodes (**Table 9**) and by 4.3 at 500 nodes (**Table 10**). The sign change is qualitatively consistent with the mechanism formalised in the remark on embedding-space clustering in Section 4: as the graph grows, profile collisions among structurally distinct candidates become more frequent, and embedding-space clustering recovers the structural separation that the readout collapses. The gain magnitude is non-monotonic in topology size; the empirical evidence supports a regime transition rather than a strict growth law.

Taken together, the classical pipeline of [2] is the operational choice on this problem class across all four axes, while the embedding-space clustering, rather than the GraphSAGE estimator of Π, is the only component of the deep pipeline that contributes a measurable benefit, and only above a topology-size threshold between 75 and 300 nodes. Section 7 discusses the directions in which this conclusion can be extended.

7. Directions for Future Work

The three-step framework of [1]-[3] (linearised PF objective, learned Π, clustering-based diversity) is agnostic to the choice of internal estimator, and several extensions can be pursued independently. *Transfer across topologies*: a trained GNN is a function of the graph, so pre-training on a collection of synthetic topologies followed by fine-tuning on a target network is a natural extension, unavailable to Random Forest since the per-pair feature vector changes meaning with the topology. *End-to-end differentiable placement*: Equation (1) can be relaxed into a continuous top-B operator [33], making the whole pipeline trainable under a downstream resilience loss. *Temporal and heterogeneous graphs*: temporal GNNs and heterogeneous message-passing generalise the current model when traffic, attack footprints, or node criticality vary across time. *Adversarial training*: training the GNN against an attacker that selects F_σ to maximise the BCE loss directly improves the resilience of the placement against the intensities where current methods degrade. *Application domains*: the agricultural sensor setting of [3] is structurally identical to the CPP, and the pipeline transfers to Wi-Fi access-point planning [34], 5G/6G edge-server placement, and LoRaWAN gateway positioning. None of these directions requires re-deriving the PF objective or the partition-matroid constraint. Each addresses one of the limitations stated next.

8. Limitations

Several limitations qualify the experimental results of Section 6 and motivate the directions of Section 7. *Scale and topology family*: the main experiment uses a 75-

node topology and is replicated on 300-node and 500-node topologies (Section 6.5); all three share the same Barabási-Albert generative family with long-range edges, so the empirical conclusions admit no claim of generalisation to operational SDN backbones such as those catalogued in the Topology Zoo, nor to graphs of substantially different degree distribution. *Single-instance training*: the encoder is trained on a single operational topology and must be retrained when links are added or nodes retire; the natural remedy (pre-training on synthetic topologies and fine-tuning) is listed as future work. *Independence*: the PF objective of [1] assumes independence of per-controller coverage events, while our GNN predicts only marginal $\hat{\pi}_{kn}$ and does not yet model the joint distribution of connectivity events. *Hop limit*: coverage is defined relative to a fixed hop limit H from [2], which we preserve and do not learn. *Interpretability*: Random Forest admits per-feature importance scores that operators routinely consult; a two-layer GraphSAGE does not, at least not without additional tooling [32]. *Hyperparameter coverage*: only encoder architecture and depth were varied in the ablation of Section 6.3. A broader sweep over hidden dimension, learning rate, weight decay, and learning-rate schedule was not performed; consequently, the negative result on the four-axis comparison cannot be definitively attributed to limitations of the GNN itself rather than to a sub-optimal hyperparameter setting. The same caveat applies to the Logistic Regression and Random Forest baselines, both of which were used at scikit-learn's default settings and could in principle be tuned further.

9. Conclusions

This paper closes the present research line on proportionally fair (PF) controller placement, begun as a probabilistic-optimisation problem in [1], given a linearised supervised-ML form in [2], and applied in [3] to sensor coverage in agriculture, along the one direction, the prior work had left open: a graph-native learned estimator of Π replacing the topological-feature classifiers, and a clustering step in the learned embedding space replacing probability-profile clustering, with the linearised PF objective, the partition-matroid constraint, the heap-based selection rule, and the online cost $O\left(L|\mathcal{L}|D+|\mathcal{K}||\mathcal{N}|D+I_{\mathrm{km}}R|\mathcal{K}|D+B\log|\mathcal{K}|\right)$ preserved unchanged. Three Barabási-Albert topologies of 75, 300, and 500 nodes are tested under a single placement protocol; the controlled comparison validates the classical pipeline as the operational choice, superior to the deep instantiation on every operational axis that determines deployment.

On worst-case placement quality, Logistic Regression leads by run-mean margins of 20.0 functional nodes at 300 and 3.8 at 75 under heavy attack; the deep estimator does not surpass it at any tested scale within the data presented here. GraphSAGE costs roughly 24× more training time than Logistic Regression at 500 nodes with no compensating placement-quality gain. Uncalibrated GraphSAGE is the most over-confident estimator (ECE 0.237 versus 0.064 for Logistic Regression); post-hoc isotonic regression brings all four into a narrow band (≈0.07), making isotonic necessary preprocessing for the deep estimator but not sufficient

to lift it above the supervised baselines. Logistic Regression is inspectable through its per-feature coefficients. The GraphSAGE encoder requires post-hoc tooling. Aggregated to the three deployment axes (placement quality, training cost, operational complexity), the classical pipeline holds; embedding-space K-means is the single narrow exception, gaining +19.34 functional nodes at 300 and +4.33 at 500 against a −1.40 trail at 75.

The deployment recommendation that follows applies within the empirical scope of this study: the Barabási-Albert topology family with long-range edges, the failure model with 12% targeted-betweenness removals and 88% uniform random removals, and topology sizes between 75 and 500 nodes. Generalisation beyond this scope is outside the evidence the paper provides and is enumerated in Section 8. Translated into a deployment rule, Logistic Regression on topological features remains the production estimator of Π at every scale tested. The GraphSAGE encoder is justified solely as the upstream signal for the embedding-space K-means clustering step, deployed only on topologies whose size exceeds the regime-transition threshold empirically located between 75 and 300 nodes. For deployments on graphs of about 75 nodes, probability-space K-means remains preferable and the encoder cost is not recovered. At 300 nodes, embedding-space K-means delivers a clear advantage exceeding the run-to-run spread; the same direction holds at 500 nodes, where the run-mean gain is smaller and the three-run evidence places it within the per-run dispersion, so embedding-space clustering remains the recommendation above the regime-transition threshold while the magnitude at 500 nodes should be confirmed under a higher-T replication. On PF-style structured prediction problems of this size class, the controlled comparison reported here supplies direct empirical evidence that increased model complexity is neither necessary nor sufficient for improved deployment performance, consistent with the broader pattern documented in [12] [13].

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Ballasheni, E., Nace, D., Tomaszewski, A. and Zyle, A. (2023) A Probabilistic Optimization Approach to the Equitable Controller Location Problem. *IEEE Transactions on Network and Service Management*, **22**, 1812-1824.

[2] Ballasheni, E., Shinko, I., Picorri, L., Kolici, V., Zyle, A. and Nace, D. (2025) Optimizim Probabilistik dhe Machine Learning per Vendosjen e Kontrolloreve ne Rrjetet SDN. *Buletini i Shkencave Teknike*, No. 1, 71-79. https://upt.edu.al/wp-content/uploads/2026/02/Buletini-i-Shkencave-Teknike-2025-Nr.-1-1.pdf

[3] Ballasheni, E., Shinko, I., Picorri, L., Kolici, V., Nace, D. and Zyle, A. (2025) Sensor Placement for Agricultural Field Coverage. 2025 *International Congress on Smart Agriculture and Sustainable Systems*, Marrakech, 5-9 December 2025, 1-6.

[4] Kreutz, D., Ramos, F.M.V., Esteves Verissimo, P., Esteve Rothenberg, C., Azodol-

molky, S. and Uhlig, S. (2015) Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, **103**, 14-76. https://doi.org/10.1109/jproc.2014.2371999

[5] Bernardos, C., de la Oliva, A., Serrano, P., Banchs, A., Contreras, L., Jin, H., *et al.* (2014) An Architecture for Software Defined Wireless Networking. *IEEE Wireless Communications*, **21**, 52-61. https://doi.org/10.1109/mwc.2014.6845049

[6] Bari, M.F., Roy, A.R., Chowdhury, S.R., Zhang, Q., Zhani, M.F., Ahmed, R., *et al.* (2013) Dynamic Controller Provisioning in Software Defined Networks. *Proceedings of the* 9*th International Conference on Network and Service Management* (*CNSM* 2013), Zurich, 14-18 October 2013, 18-25. https://doi.org/10.1109/cnsm.2013.6727805

[7] Dixit, A., Hao, F., Mukherjee, S., Lakshman, T.V. and Kompella, R. (2013) Towards an Elastic Distributed SDN Controller. *ACM SIGCOMM Computer Communication Review*, **43**, 7-12. https://doi.org/10.1145/2534169.2491193

[8] Breiman, L. (2001) Random Forests. *Machine Learning*, **45**, 5-32. https://doi.org/10.1023/a:1010933404324

[9] Kipf, T.N. and Welling, M. (2017) Semi-Supervised Classification with Graph Convolutional Networks. *International Conference on Learning Representations* (*ICLR*), Toulon, 24-26 April 2017, 1-14.

[10] Hamilton, W.L., Ying, R. and Leskovec, J. (2017) Inductive Representation Learning on Large Graphs. *Advances in Neural Information Processing Systems* (*Neu-rIPS*), Long Beach, 4-9 December 2017, 1024-1034.

[11] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. and Bengio, Y. (2018) Graph Attention Networks. *International Conference on Learning Representations* (*ICLR*), Vancouver, 30 April-3 May 2018, 1-12.

[12] Shwartz-Ziv, R. and Armon, A. (2022) Tabular Data: Deep Learning Is Not All You Need. *Information Fusion*, **81**, 84-90. https://doi.org/10.1016/j.inffus.2021.11.011

[13] Grinsztajn, L., Oyallon, E. and Varoquaux, G. (2022) Why Do Tree-Based Models Still Outperform Deep Learning on Typical Tabular Data? *Advances in Neural Information Processing Systems* 35, New Orleans, 28 November-9 December 2022, 507-520. https://doi.org/10.52202/068431-0037

[14] Heller, B., Sherwood, R. and McKeown, N. (2012) The Controller Placement Problem. *Proceedings of the* 1*st Workshop on Hot Topics in Software Defined Networks*, Helsinki, 13 August 2012, 7-12. https://doi.org/10.1145/2342441.2342444

[15] Tanha, M., Sajjadi, D., Ruby, R. and Pan, J. (2018) Capacity-Aware and Delay-Guaranteed Resilient Controller Placement for Software-Defined WANs. *IEEE Transactions on Network and Service Management*, **15**, 991-1005. https://doi.org/10.1109/tnsm.2018.2829661

[16] Muller, L.F., Oliveira, R.R., Luizelli, M.C., Gaspary, L.P. and Barcellos, M.P. (2014) Survivor: An Enhanced Controller Placement Strategy for Improving SDN Survivability. 2014 *IEEE Global Communications Conference*, Austin, 8-12 December 2014, 1909-1915. https://doi.org/10.1109/glocom.2014.7037087

[17] Kelly, F.P., Maulloo, A.K. and Tan, D.K.H. (1998) Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability. *Journal of the Operational Research Society*, **49**, 237-252. https://doi.org/10.1057/palgrave.jors.2600523

[18] Ogryczak, W., Luss, H., Pióro, M., Nace, D. and Tomaszewski, A. (2014) Fair Optimization and Networks: A Survey. *Journal of Applied Mathematics*, **2014**, Article ID: 612018. https://doi.org/10.1155/2014/612018

[19] Luss, H. (2012) Equitable Resource Allocation: Models, Algorithms, and Applica-

tions. Wiley. https://doi.org/10.1002/9781118449189

[20] Khalil, E., Dai, H., Zhang, Y., Dilkina, B. and Song, L. (2017) Learning Combinatorial Optimization Algorithms over Graphs. *Advances in Neural Information Processing Systems* (*NeurIPS*), Long Beach, 4-9 December 2017, 6348-6358.

[21] Kool, W., van Hoof, H. and Welling, M. (2019) Attention, Learn to Solve Routing Problems! *International Conference on Learning Representations* (*ICLR*), New Orleans, 6-9 May 2019, 1-25.

[22] Li, Z., Chen, Q. and Koltun, V. (2018) Combinatorial Optimization with Graph Convolutional Networks and Guided Tree Search. *Advances in Neural Information Processing Systems* (*NeurIPS*), Montréal, 3-8 December 2018, 537-546.

[23] Gasse, M., Chételat, D., Ferroni, N., Charlin, L. and Lodi, A. (2019) Exact Combinatorial Optimization with Graph Convolutional Neural Networks. *Advances in Neural Information Processing Systems* (*NeurIPS*), Vancouver, 8-14 December 2019, 15580-15592.

[24] Almasan, P., Suárez-Varela, J., Rusek, K., Barlet-Ros, P. and Cabellos-Aparicio, A. (2022) Deep Reinforcement Learning Meets Graph Neural Networks: Exploring a Routing Optimization Use Case. *Computer Communications*, **196**, 184-194. https://doi.org/10.1016/j.comcom.2022.09.029

[25] Platt, J. (1999) Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In: *Advances in Large Margin Classifiers*, MIT Press, 61-74.

[26] Zadrozny, B. and Elkan, C. (2002) Transforming Classifier Scores into Accurate Multiclass Probability Estimates. *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, 23-26 July 2002, 694-699. https://doi.org/10.1145/775047.775151

[27] Niculescu-Mizil, A. and Caruana, R. (2005) Predicting Good Probabilities with Supervised Learning. *Proceedings of the 22nd International Conference on Machine Learning-ICML*'05, Bonn, 7-11 August 2005, 625-632. https://doi.org/10.1145/1102351.1102430

[28] Guo, C., Pleiss, G., Sun, Y. and Weinberger, K.Q. (2017) On Calibration of Modern Neural Networks. *Proceedings of the 34th International Conference on Machine Learning* (*ICML*), Sydney, 6-11 August 2017, 1321-1330.

[29] Domingos, P. (2012) A Few Useful Things to Know about Machine Learning. *Communications of the ACM*, **55**, 78-87. https://doi.org/10.1145/2347736.2347755

[30] Kingma, D.P. and Ba, J. (2015) Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* (*ICLR*), San Diego, 7-9 May 2015, 1-15.

[31] Rousseeuw, P.J. (1987) Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, **20**, 53-65. https://doi.org/10.1016/0377-0427(87)90125-7

[32] Ying, R., Bourgeois, D., You, J., Zitnik, M. and Leskovec, J. (2019) GNNExplainer: Generating Explanations for Graph Neural Networks. *Advances in Neural Information Processing Systems* (*NeurIPS*), Vancouver, 8-14 December 2019, 9240-9251.

[33] Xie, S.M. and Ermon, S. (2019) Reparameterizable Subset Sampling via Continuous Relaxations. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, Macao, 10-16 August 2019, 3919-3925. https://doi.org/10.24963/ijcai.2019/544

[34] Ballasheni, E., Shinko, I., Nace, D., Zyle, A. and Kolici, V. (2025) Strategic Placement

of Access Points for Maximizing Wi-Fi Quality. 2025 20*th Annual System of Systems Engineering Conference* (*SoSE*), Tirana, 8-11 June 2025, 1-6. https://doi.org/10.1109/sose66311.2025.11083808