Graph Sampling Quality Prediction for Algorithm Recommendation

S. Haleh S. Dizaji*, Reza Farahani*, Jože M. Rožanec†, Dragi Kimovski*, Ahmet Soylu‡, Radu Prodan*
*Institute of Information Technology, University of Klagenfurt, Austria
†Jožef Stefan Institute, Ljubljana, Slovenia
‡Kristiania University College, Oslo, Norway

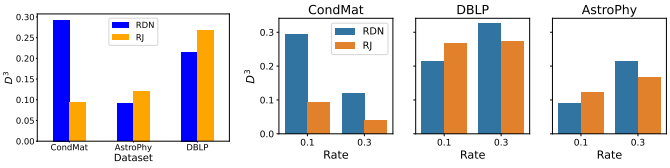
***Abstract*—The increasing size of graph structures in real-world applications, such as distributed computing networks, social media, or bioinformatics, requires appropriate sampling algorithms that simplify them while preserving key properties. Unfortunately, predicting the outcome of *graph sampling algorithms* is challenging due to their irregular complexity and randomized properties. Therefore, it is essential to identify appropriate graph features and apply suitable models capable of estimating their sampling outcomes. In this paper, we compare three machine learning (ML) models for predicting the divergence of five metrics produced by twelve node, edge, and traversal-based graph sampling algorithms: degree distribution (D^3), clustering coefficient distribution ($\mathrm{C}^2\mathrm{D}^2$), hop-plots distribution (HPD^2) (including the largest connected component (HPD^2_C)), and execution time. We use these prediction models to recommend suitable sampling algorithms for each metric and conduct mutual information analysis to extract relevant graph features. Experiments on six large real-world graphs demonstrate a prediction error under $20\,\%$ in $\mathrm{C}^2\mathrm{D}^2$ and HPD^2 prediction for most algorithms despite their relatively high dissimilarity with the training data. Sampling algorithm recommendations on ten real-world graphs show higher hits@3 for D^3 and $\mathrm{C}^2\mathrm{D}^2$ and comparable results for HPD^2 and HPD^2_C compared to the K-best baseline method. Finally, ML models show superior runtime recommendations compared to baseline methods, with hits@3 over $86\,\%$ for synthetic and real graphs and hits@1 over $60\,\%$ for small graphs. These findings are promising for algorithm recommendation systems, particularly when balancing quality and runtime preferences.**

***Keywords*—Graph sampling; machine learning; quality prediction; algorithm recommendation.**

I. Introduction

With the growing size of graph data structures in real-world applications, reducing their complexities becomes increasingly critical to speed up computations. Moreover, the unavailability of entire graphs for privacy or scalability reasons requires gathering relevant samples for analyzing and estimating their global properties. For example, the increasing use of machine learning (ML) methods on graphs, such as graph neural networks (GNNs) or in-memory analytics on small edge devices, requires the selection of realistic subgraphs for scalable training, achievable only through qualitative sampling strategies.

Unfortunately, selecting a suitable *sampling algorithm* across various categories (e.g., scale-free, power-law, binomial) can be burdensome, as no solution performs equally well across different problems [1]. Sampling algorithms are of three categories (i.e., node, edge, and traversal-based), producing outcomes dependent on graph and method properties, such as sampling metric and rate. To illustrate the variability in quality, Figure 1 compares the *degree distribution divergence (D^3)*, representing the dissimilarity of the sample and original graph, using two sampling algorithms, *random jump (RJ)* and *random degree node (RDN)* [2] for three graphs representing research co-authorship and citation networks (`Astro physics`, `CondMat` and `DBLP`). RDN performs better than RJ for the `Astro physics` and `DBLP` graphs. However, RJ significantly outperforms for `CondMat`. A possible interpretation is the size of `CondMat`, containing a larger maximum connected component preferred by RJ to traverse the graph. Additionally, the higher degree variance of the `Astro physics` graph might result in better sampling for the degree-based RDN algorithm. The sampling rate further aggravates the problem, RJ underperforming RDN for a rate of 0.1 and outperforming it for a rate of 0.3 for the `Astro physics` and `DBLP` datasets (see Figure 1(b)) by better collecting structural information.



(a) Dataset impact. (b) Sampling rate impact.

Fig. 1: D^3 of RJ and RDN algorithms on three example data sets.

To address this challenge, various analytical methods [3], [4] formulate the sampling quality of algorithms for metrics such as degree and clustering coefficient (CC) distributions. Although computing the bounds for graph sampling qualities is feasible [2], [5], providing an exact formalism to estimate their performance for particular cases is challenging, especially for covering recent advancements. On the other side, empirical analysis methods [6], [7], [8], [9] evaluate graphs algorithms without thoroughly considering graph properties, providing non-generalizable results.

Algorithm outcome prediction guides its selection depend-

This manuscript is the accepted version of a paper published in the Proceedings of the IEEE 31st International Conference on High Performance Computing, Data, and Analytics (HiPC), 2024. The final published version is available at https://ieeexplore.ieee.org/document/10884475.

ing on the desired metric, input graph data, or algorithm properties [10], [11]. Numerous predictive models forecast processing tasks' performance, such as the runtime for iterative graph algorithms (e.g., PageRank or top-k ranking) [12] or query execution times over graph databases [13]. Nevertheless, research on ML methods to forecast graph sampling performance in the literature is scarce.

We bridge this gap by introducing a predictive tool for graph sampling algorithm selection using three ML models: *k-nearest neighbor (kNN)*, *multi-layer perception (MLP)*, and *random forest (RF)*. We consider five quality metrics: degree distribution divergence (D^3), clustering coefficient distribution divergence ($\mathrm{C}^2\mathrm{D}^2$), hop-plots distribution divergence (HPD^2), including the largest connected component (HPD^2_C), and execution time. Our experiments involve 664 synthetically generated and 16 real-world graphs providing evaluations at a scale larger than the limited training set. Evaluation results indicate superior prediction accuracy of MLP and kNN on synthetic graphs for D^3, $\mathrm{C}^2\mathrm{D}^2$, and HPD^2, and of RF on real graphs for $\mathrm{C}^2\mathrm{D}^2$ and HPD^2, RF showing better generalizability for unseen features. We provide quality and runtime sampling algorithm recommendations based on our predictions with superior ranking confidence across most metrics compared with two random and K-best baseline methods. We delve into the impact of different features on predictions, enhancing the explainability of the ML methods and emphasizing the importance of sampling characteristics and highlighted graph features. We conclude by discussing the current limitations of our method on scalability, dissimilarity, and feature extraction, which are subject to future research.

The paper has eight sections. Section II summarizes the related work in graph algorithm analysis and prediction methods. Section III describes the proposed methodology, including sampling algorithm selection, feature extraction, ML models, and learning goals. Section IV details the experimental design, followed by the results and discussion in Section V. Section VI analyzes performance optimizations to the ML models and the runtime prediction of the sampling algorithms. Section VII discusses limitations, and Section VIII concludes the paper.

II. Related Work

This section analyzes and classifies graph analysis-related works into two categories: empirical evaluations of sampling algorithms and predictions.

A. Empirical graph sampling

Leskovec and Faloutsos [6] studied three types of sampling algorithms affecting the graph structure for static and dynamic graphs considering nine metrics, including degree, CC, connected component size, singular vectors, adjacency matrix, and hop-plots distributions. They concluded the best overall performance of traversal methods for static networks.

Yoon et al. [7] analyzed the random walk (RW) algorithm considering degree distribution, CC, and degree-degree correlation, preserving most topological features of synthetic and real-world scale-free graphs with high powers.

Zhang et al. [8] evaluated multiple sampling algorithms on static synthetic and real-world graphs under different sampling rates considering different distribution divergence metrics (e.g., Kolmogorov–Smirnov (KS)) and runtime. Visualization results confirmed the outcomes dependant not only on the algorithm but also on the graph properties.

Yousuf et al. [9] studied five traversal sampling methods considering six graph properties (i.e., degree, local and global CC, path length, assortativity, modularity) and two metrics (i.e., root mean square error (RMSE) and Jensen-Shannon divergence). Experiments on synthetic and real-world graphs confirmed no better algorithm for all metrics.

Limitations: Existing works revealed that the sampling quality depends on algorithms and graph properties but lacks a thorough, generalizable study. Therefore, estimating their quality depending on the input features is necessary.

B. Graph sampling prediction

We summarize the research on graph sampling algorithm analysis and prediction.

1) Analytical models: Stumpf and Wiu [3] and Lee et al. [4] formulated the sampled graph degree as a function of the original graph degree distribution and sampling rate.

Illenberger and Flötteröd [14] studied the snowball algorithm, derived an expression for node selection probability, and developed estimators for mean degree, degree correlation, and CC. They found a decreasing estimator performance with increasing distribution variance in the original graph.

Ribeiro and Towsley [15] predicted the original degree distribution using the unbiased Horvitz-Thompson estimator considering the sample distribution and degree and reported bounds for the estimated errors.

Everitt and Hutter [16] estimated runtimes for breadth-first search and depth-first search algorithms over trees as a function of the tree depth.

Limitations: Existing methods estimate particular sampling outcomes based on a minimal set of graph characteristics (and vice-versa). However, they cannot learn associations between complex structural properties and metrics of interest.

2) Predictive models: Popescu et al. [12] proposed a prediction model for iterative algorithms over large graphs, such as PageRank, semi-clustering, and top-k ranking. They modeled the iteration runtime as a multivariate linear function of input graph features and acknowledged its appropriateness for algorithms on homogeneous graph structures.

Hutter et al. [10] developed ML models (e.g., RF, neural networks) to predict the runtimes for traveling salesperson algorithms using features such as problem size, minimum spanning tree (MST), and cluster distance.

Ahmad et al. [17] developed a runtime prediction framework for graph analytics using decision trees to select dynamic parameters within and across multiple accelerators. It uses graph features, like node and edge counts and diameter, and graph analytics features, such as node processing and scheduling, memory access, data movement, compute type, and synchronization information.

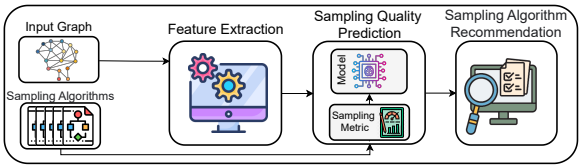


Fig. 2: Graph sampling prediction architecture.

Park et al. [11] proposed a graph meta-learning evaluation-free link prediction model selection relying on the prior performance metrics and various statistics of structural graph features, such as degrees, length-two paths, triangle counts, PageRank, eccentricity, and k-core numbers.

Chu et al. [18] developed an LSTM model to predict query execution times over graph databases, converting query plan graphs into a sequence of operations and selecting five system features to learn sequential patterns.

Limitations: Related works forecast graph processing outcomes and performance and provide insights into relevant features and models for algorithm prediction. However, they disregard graph sampling algorithms.

III. Methodology

This section outlines the methodology for sampling quality prediction based on five phases displayed in Figure 2: 1) sampling algorithm selection and characterization, 2) sampling goals and quality metrics formulation, 3) feature extraction, processing, and selection, 4) ML model design for sampling prediction, and 5) sampling algorithm recommendation.

A. Sampling algorithm selection

We selected twelve popular sampling algorithms covering three classes (see Table I): node, edge, and traversal-based.

1) Node-based sampling:

Random node (RN): samples graph nodes with uniform probability, failing to capture complex structural properties, such as the power-law degree distribution [3], [19].

RDN: samples graph nodes with a probability proportional to their degrees [6], biased toward high-degree nodes.

2) Edge-based sampling:

Random edge (RE): samples with a uniform probability, biased toward high-degree nodes and failing to preserve the graph structure. Furthermore, it generates graphs with higher average path lengths for larger samples and smaller CC [4].

Random node edge (RNE): performs one RN and RE step in each iteration. While it decreases the bias towards high-degree nodes [6], it could result in sparse connections [19].

Induced random edge (IRE): extends RE with an induction step of including the remaining edges between nodes in the sample, which contains more topological information [20].

3) Traversal-based sampling: captures more structural properties than the other categories but is usually biased towards high-degree nodes.

TABLE I: Graph sampling algorithm selection.

Algorithm	*RN*	*RDN*	*RNE*	*RE*	*IRE*	*RJ*	*SB*	*FF*	*FS*	*RD*	*MHRW*	*XS*
Node	✓	✓	✓									
Edge			✓	✓	✓							
Traversal						✓	✓	✓	✓	✓	✓	✓

RJ: addresses the RW's problem of getting stuck in a region and traversing a connected area by jumping to an arbitrary node. However, it results in low neighborhood traversing and convergence of the sample [21].

Metropolis-Hastings random walk (MHRW): selects neighbors of a node proportional to their degree ratios to preserve the degree distribution [2]. However, it poorly estimates the degree distribution [15].

Snowball (SB): uses the breadth-first traversal to replace nodes with a fixed number of neighbors [2]. It can better preserve CC but may cause boundary bias [4].

Forest fire (FF): is a probabilistic algorithm derived from SB based on a graph evolution model [22] that burns node neighbors at each step with a geometric distribution to prevent local sampling. However, it exhibits a bias towards high-degree nodes and a tendency to remain isolated in cluster regions [19].

Frontier sampling (FS): starts from a random node set, probabilistically selects nodes proportional to their degree, and replaces them with random neighbors. An infinite increase of seeds yields uniform node and edge distributions [2].

Rank degree (RD): [23] ranks node neighborhoods by degrees and preserves community structures [9]. Given a seed set, it selects a random node and its top-k-ranked neighbors as sampled edges and replaces the current set with them.

Expansion sampling (XS): maintains community structure by initiating from a random seed. A greedy neighborhood sampling [24] selects nodes that maximize the expansion factor, indicating the number of outgoing edges in the sampled graph. However, its computational cost is considerable.

B. Sampling quality metrics

We frame the learning of a sampling algorithm quality metric as a regression problem, considering an input data graph and its sampling characterization features. We consider five learning goals, studied in [6]:

- D^3 using the KS distance between the normalized degree distributions of the original and sampled graphs;
- C^2D^2 using the KS distance between the normalized CC distributions of the original and sampled graphs;
- HPD^2 using the KS distance between the hop-plots (distance) distributions of the original and sample graphs;
- HPD^2_C using HPD^2 in the largest connected component;
- *Execution time* describing the overall sampling algorithm runtime performance.

C. Feature extraction and selection

We first extracted vectors of 82 dimensions containing statistics of 15 structural and size features and calculation times from 664 graphs and evaluated the quality of the sampling algorithms according to the metrics specified in

TABLE II: Selected input graph and sampling features.

	Feature	*Notation*	*Normal.*	*Learning Metrics*
Graph features	Node number	$\|N\|$	M	C^2D^2, HPD^2, HPD^2_C
	Edge number	$\|E\|$	M	D^3
	Density	D	–	D^3, HPD^2
	Node-edge division	$\|N\|/\|E\|$	EL	D^3
	Edge-node division	$\|E\|/\|N\|$	EL	D^3
	Degree	Deg	EL, M	D^3
	Clustering coefficient	CC	EL, M	D^3, C^2D^2, HPD^2_C
	Node betweenness centrality	NBC	EL, M	D^3, C^2D^2, HPD^2
	Shortest path length	SPL	EL, M	D^3, C^2D^2, HPD^2, HPD^2_C
	Eigenvector centrality	EIC	EL, M	D^3, C^2D^2
	Pagerank centrality	PRC	EL, –	D^3, C^2D^2, HPD^2, HPD^2_C
	Degree assortativity	–	–	D^3, C^2D^2
	Minimum spanning tree degree	MSTD	M	D^3, C^2D^2, HPD^2, HPD^2_C
	Number of connected components	$\|CComp\|$	M	C^2D^2, HPD^2, HPD^2_C
	Connected component size	CCS	M	C^2D^2, HPD^2, HPD^2_C
Sampling features	Algorithm type	T-(Node, Edge, Trav)	–	D^3, C^2D^2
	Algorithm	Alg.	–	D^3, C^2D^2
	Rate (0.1, 0.3)	–	–	D^3, C^2D^2

Section III-B. We conducted a mutual information analysis of these features and the sampling outcomes, evaluated non-linear dependencies and extracted those with values higher than 0.99 for any algorithm resulting in 50 dimensional features. Then, we selected the important features of the graphs alongside sampling features for training and testing the models grouped in two categories, as represented in Table II.

a) Graph features: include size and structural features and their statistics represented as maximum ($f_{\max}$), minimum ($f_{\min}$), average ($\overline{f}$), median (f_{med}), and variance (f_{var}) of feature f over the graph (see Table II).

b) Sampling features: containing 16 dimensions describe the algorithm (twelve-dimensional one-hot vector), type (three-dimensional), and sampling rate (one-dimensional).

We used two normalization strategies to scale the features in the $[0, 1]$ interval, as required by specific ML algorithms:

- *M*, denoting the value normalized by the maximum feature value in the graph $\frac{f_i}{f_{\max}}$, applied to the minimum, average and medium values.
- *EL*, denoting exponential-logarithmic mapping $e^{-\log(f_i)}$, applied for maximum and variance values, calculation times, and raw features.

D. Samping quality prediction

Since a lightweight linear regression cannot handle the complexity and non-deterministic outcome of graph sampling, we selected three ML regression methods to predict the sampling algorithms' C^2D^2, D^3, HPD^2, and HPD^2_C:

- *RF* is resilient to probabilistic sampling noise and versatile to rather high dimensional features (66) [10] and outperformed linear, linear-SVM, and Gaussian-process in our preliminary experiments;
- *MLP* is a flexible regression appropriate for high dimensional graph data used in the literature for runtime prediction [10], and appropriate for synthetic graph predictions with replicating properties of train-test data;
- *kNN* has fast training, low hyperparameter size, suitable for resembling data properties.

E. Sampling algorithm recommendation

This final phase recommends a selection of k "best" sampling algorithms based on three quality metrics:

a) RMSE: as the average difference between the predicted and actual metrics, sensitive to outliers when high errors are undesirable;

b) Hits@k: as the average percentage of the top-k best sampling algorithms correctly predicted (without ordering);

c) Local interpretable model-agnostic explanations: for obtaining the most relevant features [25] for any input graph by perturbing them to obtain vicinity points and generate simple (e.g., linear) regressor that locally approximates the black-box model. The feature weights of the simple model represent their importance factor for any input graph.

IV. Experimental Design

This section explains our datasets, ML model configuration, and evaluation metrics, released together with the code.

A. Experimental graphs

We evaluated our method using a combination of synthetic and real graphs. We performed the model training based on a carefully generated synthetic dataset that enables flexible generative parameterization for various graphs displaying certain characteristics. We tested our method using a combination of synthetic and real graphs to evaluate its generalizability.

1) Synthetic graphs: We generated graphs of different types and sizes to train and evaluate our prediction modules using the `NetworkX` and Stanford Network Analysis Project for Python libraries. We considered six types of synthetic graphs that yield different properties of real-world networks:

a) Albert-Barabasi (AB): are scale-free graphs that mimic biological, social, and communication networks [26].

b) Watts-Strogatz (WS): has small-world and high clustering properties, observed in social networks [26].

c) Erdos-Renyi (ER): produces random graphs considering binomial distribution of any particular vertex [26].

d) Stochastic block model (SBM): possesses a community structure evident in many real-world graphs.

e) Power-law cluster (PLC): generates graphs with the desired CC as an important real-world characteristic.

f) Forest fire model (FFM): provides a superlinear dependency between edge and node numbers and shrinking diameters property over time.

We generated 664 training graphs of all types with $10\,000$ – $100\,000$ nodes. We tested the models on 36 larger graphs of AB, ER, and WS types for $150\,000$ – $450\,000$ nodes (see Table III). We analyzed various real-world graph properties, such as density and CC, to generate graphs with realistic properties. For AB, WS, ER, SBM, and FFM, we adjusted the graph parameters to produce density values in the range of real-world graphs. For PLC, we set parameters to generate graphs with CCs close to real-world ones. We avoid duplicates in the training and testing datasets to prevent leakage. The appendix provides more details on the graph generation process.

TABLE III: Synthetic graph characteristics.

Dataset	*Type*	$\|N\|$	D	$\overline{CC}$	*Graphs*
Train	AB, ER, WS, SBM	10 000 – 100 000	0.000 01 – 0.001	0 – 0.16	568
	PLC	10 000 – 100 000	0.000 01 – 0.001	0.1 – 0.6	60
	FFM	10 000 – 100 000	0.000 04 – 0.001	0.01 – 0.4	36
Test	AB, ER, WS	150 000 – 450 000	0.000 008 – 0.0001	0 – 0.001	36

TABLE IV: Real-world graph characteristics.

Category	*Dataset*	$\|N\|$	$\|E\|$	D	$\overline{CC}$
Citation	DBLP-small	12 590	607 610	0.0006	0.12
	HepTh	27 770	352 285	0.0009	0.31
	HepPh	34 546	420 877	0.0007	0.28
	Cora	23 166	89 157	0.0003	0.27
Co-authorship	Arxiv	18 771	198 050	0.001	0.63
	Astro physics	18 771	121 251	0.0007	0.57
	CondMat-2003	30 460	120 029	0.0003	0.65
	CondMat-2005	39 577	175 691	0.0002	0.65
Collaboration	Actor	382 219	15 038 083	0.000 21	0.78
	DBLP	317 080	1 049 866	0.000 02	0.63
Technology	Caida	190 914	607 610	0.000 03	0.16
	Gnutella	62 586	147 892	0.000 08	0.005
	Topology	34 761	107 720	0.000 18	0.29
Social	Gowalla	196 591	950 327	0.000 05	0.24
	Digg	770 799	5 907 132	0.000 02	0.09
	Hyves	1 402 673	2 777 419	0.000 003	0.04

2) Real-world graphs: We tested the prediction 16 public real-world graphs [27] corresponding to citation, co-authorship, social, collaboration, and technology domains (see Table IV), categorized into two size ranges:

- SMALL consisting of graphs with around 12 000 – 62 000 nodes [28], [29];
- LARGE consisting of graphs with around 190 000 – 1 400 000 nodes [28].

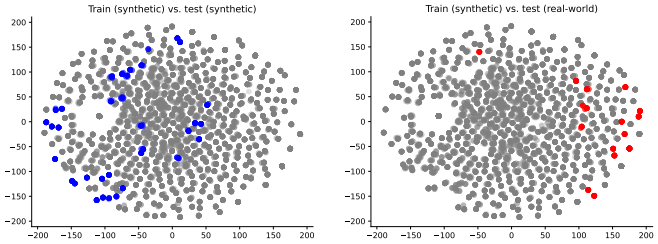
3) Approximation methods: We approximated features for LARGE graphs to speed up their calculations or determine their infeasibility, as follows:

- *Hop-plots approximation* [30] implemented in `Networkit` library for shortest paths on a neighborhood using a maximum of 1024 parallel executions;
- *NBC approximation* [31] based on the random samplings of shortest paths with a quality factor of 0.8 and 0.9.

4) Similarity analysis: We compare the similarities of the synthetic and real-world training and testing graphs by performing a dimensionality reduction over the normalized features using the t-distributed stochastic neighbor embedding (t-SNE) displayed in Figure 3.

a) Synthetic testing data: Figure 3(a) reveals a better coverage of the synthetic training dataset due to the similar graph types, sizes, densities, and CC features adjusted during generation. We also generated synthetic test graphs with dissimilar features to evaluate the models' generalizability.

b) Real-world testing data: The distribution of the real-world testing data in Figure 3(b) shows a higher divergence from training data failing to capture all graph properties, such as NBC and SPL, due to the limited adjustable generation model parameters. However, this divergence allows us to test the generalizability of the models to unseen data features, which is important for practical applications.



(a) Synthetic test graphs. (b) Real-world test graphs.

Fig. 3: Similarity of the training (gray) and testing data sets.

TABLE V: ML model hyper-parameter configuration for all metrics.

Model	*Parameter*	*Metric*			
		D^3	C^2D^2	HPD^2	HPD^2_C
kNN	algorithm	auto	ball_tree	ball_tree	ball_tree
	n_neighbors	4	4	15	10
	weights	distance	distance	distance	distance
RF	bootstrap	true	true	true	true
	max_depth	none	none	none	none
	max_features	3	3	3	3
	min_samples_leaf	2	2	2	2
	min_samples_split	8	8	8	8
	n_estimators	100	400	400	400
MLP	activation	tanh	tanh	relu	relu
	alpha	0.0001	0.0001	0.0001	0.0001
	early_stopping	true	true	true	true
	hidden_layer_sizes	100,100	30,30	100,100	50,50
	learning_rate	constant	constant	constant	constant
	shuffle	true	true	true	true
	solver	adam	adam	adam	adam

B. ML model configuration

We trained and optimized the models using the `scikit-learn` Python library and employed a five-fold cross-validation to explore and configure the model hyperparameters, illustrated in Table V:

- RF configurations assuming a maximum tree depth of at least 90 and a minimum number of 100 decision trees;
- MLP consisting of one and two layers of maximum size 100 with early stopping criteria;
- kNN with 4 – 15 neighbors and uniform and distance-based weights.

V. Evaluation Results

This section first analyzes the prediction accuracy for synthetic graphs and provides sampling recommendations at scale. We then challenge the analysis with real graphs with higher similarity divergence, expected in real-world scenarios.

A. Synthetic graphs

We tested the scalability of the sampling prediction on synthetic graphs up to 450 000 nodes (see Table III). We excluded the XS algorithm due to its high time complexity on large graphs of $\mathcal{O}\left(p \cdot |N|^2\right)$, where p is the number of sampled nodes and $|N|$ the number of nodes. We evaluated sampling algorithm recommendations based on the Hits@1 and Hits@3 metrics and compared them to two baseline methods:

- *K-best* selection of the algorithms with the highest average rankings in the training set portfolio;
- *Random* sampling algorithm selection for every graph.

TABLE VI: Sampling Hits@k ranking on synthetic graphs.

Method	*Hits@1*				*Hits@3*			
	D^3	C^2D^2	HPD^2	HPD^2_C	D^3	C^2D^2	HPD^2	HPD^2_C
Random	0.08	0.07	0.07	0.06	0.32	0.31	0.27	0.29
K-best	0.66	0.54	**0.54**	**0.49**	0.66	0.54	0.54	0.49
MLP	0.75	0.54	0.41	0.35	**0.89**	0.60	0.59	**0.76**
RF	0.75	**0.62**	0.42	0.35	0.83	0.61	**0.63**	0.69
kNN	**0.87**	0.58	0.37	0.35	0.77	**0.65**	0.55	0.75

TABLE VII: Sampling algorithms Hits@3 on synthetic graphs.

Method	D^3				C^2D^2				HPD^2				HPD^2_C			
	AB	*ER*	*WS*	*All*	*AB*	*ER*	*WS*	*All*	*AB*	*ER*	*WS*	*All*	*AB*	*ER*	*WS*	*All*
RF	0.79	**0.82**	**0.95**	0.83	0.65	0.33	**0.82**	0.61	0.5	**0.78**	**0.72**	**0.63**	0.61	0.82	0.72	0.69
MLP	**0.94**	0.78	0.93	**0.89**	**0.89**	0.20	0.47	0.60	0.47	0.72	0.68	0.59	**0.74**	0.72	**0.85**	**0.76**
kNN	0.75	0.72	0.86	0.77	0.86	0.30	0.63	**0.65**	0.46	0.60	0.70	0.55	0.68	**0.83**	0.79	0.75

We used MLP for algorithm recommendation regarding its overall better selection performance for the top three results across all metrics according to Table VI and detailed algorithm recommendations specific to each graph type in Table VIII.

1) HPD^2_C *and* HPD^2 *forecast:* We found no significant differences in accuracy for HPD^2_C and HPD^2 prediction on synthetic graphs (see Figures 4(a) and 4(b)). However, MLP and RF result in a relatively lower RMSE for HPD^2_C, while MLP and kNN achieve lower RMSE for HPD^2 of maximum $20\,\%$ (see Figures 5(a) and 5(b)). Overall, the selection of the best model depends on the sampling algorithm. RNE and RJ are among the most predictable algorithms, expected from their best, respectively worst HPD^2 preservation [32].

Sampling recommendation: Table VI indicates the highest Hits@1 of $54\,\%$ for HPD^2 and $49\,\%$ for HPD^2_C by single best (FF) selection and the highest Hits@3 of $63\,\%$ and $76\,\%$ by RF and MLP. Predictably, the random selection exhibits the lowest performance. Further analysis in Table VII demonstrates the overall higher Hits@3 of RF for HPD^2 on ER and WS graphs. MLP yields better results considering HPD^2_C recommendations for WS and AB, as it effectively learns from connected subgraphs containing discernible patterns (WS and AB). Table VIII presents the three sampling algorithms recommended by MLP, along with the predicted HPD^2 and HPD^2_C divergence across different graph types. Notably, RJ, FF, and SB received the most recommendations, where FF is consistent with the findings in [6]. In conclusion, MLP recommends IRE for AB graphs and FF for WS and ER graphs for both HPD^2_C and HPD^2.

2) C^2D^2 *forecast:* MLP is the best model for C^2D^2 prediction on synthetic graphs. Figure 4(d) shows fewer outliers for MLP, and Figure 5(d) depicts its maximum C^2D^2 prediction error of $12\,\%$. Figure 4(d) shows the concentration of RF predictions near the identity line for lower C^2D^2 (below 0.3). However, it highly diverges from the line for higher scores. Well-performing sampling algorithms, such as IRE, RDN, RE, RJ, RN, and RNE, are more predictable.

Sampling recommendation: Table VI shows an expected higher Hits@1 of $62\,\%$ for RF due to its low prediction error for high-quality algorithms, and a Hits@3 of $65\,\%$ for kNN. A detailed evaluation in Table VII revealed a higher Hits@3 on small-world and scale-free graphs, such as RF on WS and MLP on AB. No model yields Hits@3 higher than $33\,\%$ for ER graphs. The three best algorithms recommended by MLP, shown in Table VIII, vary across graph types and yield a maximum high-quality C^2D^2 RMSE of $13\,\%$. Based on these results, we recommend RN for most graph types considering C^2D^2, corroborating prior studies [32].

TABLE VIII: MLP sampling recommendation for synthetic graphs.

Topology	D^3		C^2D^2		HPD^2		HPD^2_C	
	Alg.	*Divergence*	*Alg.*	*Divergence*	*Alg.*	*Divergence*	*Alg.*	*Divergence*
AB	FF	0.34	RN	0.00	IRE	0.2	IRE	0.15
	RJ	0.39	RNE	0.09	RJ	0.2	FF	0.14
	IRE	0.44	MHRW	0.13	RDN	0.21	RD	0.2
WS	FF	0.67	RDN	0.00	FF	0.76	FF	0.69
	SB	0.85	RN	0.01	MHRW	0.87	SB	0.83
	MHRW	0.86	RNE	0.01	SB	0.86	RJ	0.87
ER	FF	0.56	IRE	0.00	FF	0.69	FF	0.68
	RJ	0.66	RD	0.11	SB	0.73	SB	0.74
	IRE	0.71	FS	0.01	RJ	0.83	RJ	0.82

3) D^3 *forecast:* We observed D^3 prediction errors below $20\,\%$ on synthetic graphs for all models. Figure 5(c) shows the dominance of MLP, with a maximum RMSE of $8\,\%$. Figure 4(c) shows the concentration of its predictions around the identity line, learning similar patterns in the testing and training graphs. Figure 4(c) represents more outliers for RF. We found FF the most challenging algorithm for D^3 prediction.

Sampling recommendation: kNN and MLP achieved a better D^3 sampling recommendation, with Hits@1 and Hits@3 over $87\,\%$, surpassing the K-best (FF, RJ, and SB) selections. Table VII represents the highest RF Hits@3 of $95\,\%$ on WS and $82\,\%$ on ER graphs. MLP yields higher Hits@3 of $94\,\%$ on AB graphs. FF, RJ, and IRE are the top three algorithms recommended by MLP for D^3 in Table VIII, consistent across AB and ER graphs. SB and MHRW emerge as other top-performing algorithms for WS graphs. MLP recommends FF as the sampling algorithm with the highest quality for all graph types with a minimum D^3 of $34\,\%$.

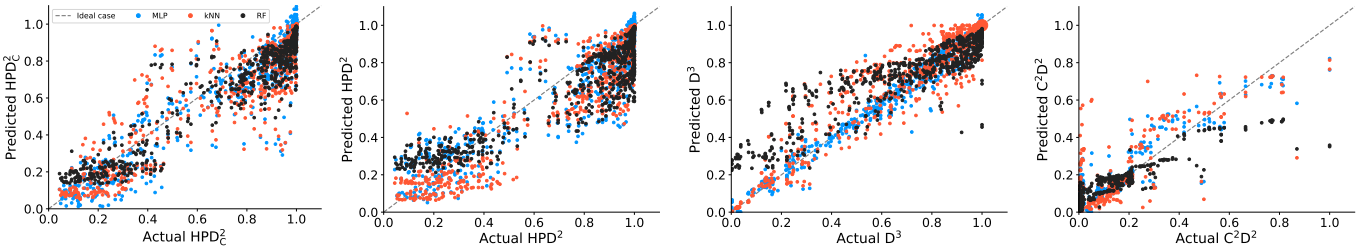
B. Real-world graphs

In most cases, real graphs show higher prediction errors and variety, expected considering their divergence from the training data (see Figure 3). To compensate, our recommendation includes features with an average maximum ranking of ten.

1) HPD^2_C *and* HPD^2 *forecast:* We observe lower errors for all models with increasing graph size, which might arise from most algorithms' lower HPD^2_C.

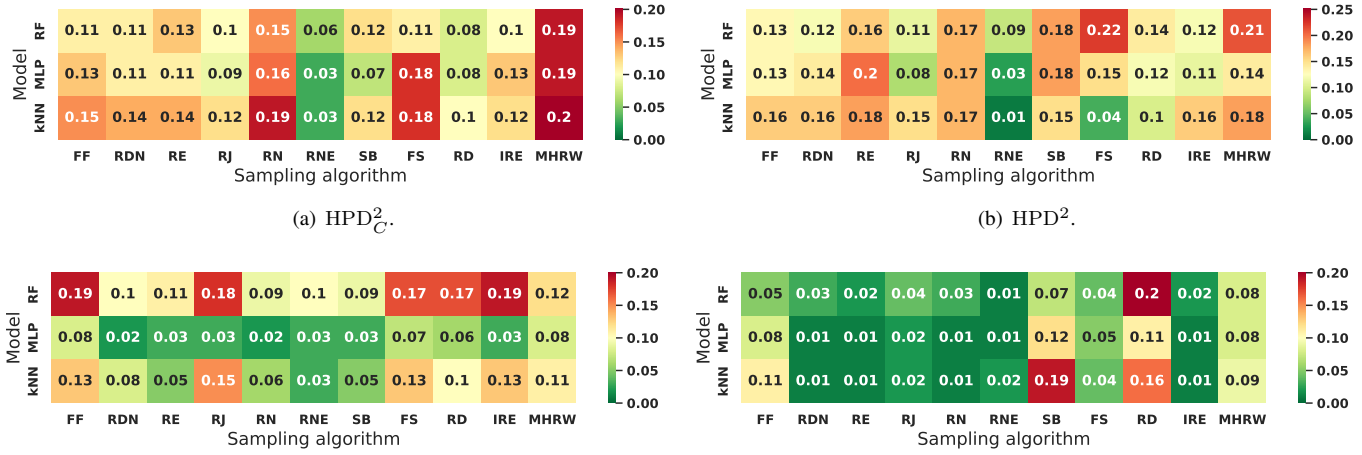
a) SMALL graphs: Prediction RMSEs in Figures 6(a) and 6(b) represent RF as the dominant ML method irrespective of the algorithm with prediction errors under $30\,\%$ for HPD^2_C and $35\,\%$ for HPD^2. MLP and kNN weakly predict with errors higher than $20\,\%$ for most algorithms.

b) LARGE graphs: RF is still the best choice for larger graphs on all algorithms with a maximum prediction error of $37\,\%$. Exceptionally, kNN and MLP outperform RF with a maximum HPD^2 error of $13\,\%$ for RNE and MHRW (see Figures 6(a) and 6(b)). RF can provide better predictions for more accurate algorithms such as FF, XS, IRE, and RJ but fails to predict RN, FS, and SB with the overall lower HPD^2 quality [32]. The variability in HPD^2_C arises from the random outcome of RN and the SB's bias over a region affecting RF's



(a) HPD^2_C. (b) HPD^2. (c) D^3. (d) $\mathrm{C}^2\mathrm{D}^2$.

Fig. 4: Actual versus predicted synthetic graph sampling metrics for all algorithms.



(a) HPD^2_C. (b) HPD^2. (c) D^3. (d) $\mathrm{C}^2\mathrm{D}^2$.

Fig. 5: Predicted RMSE for different sampling algorithms and quality metrics on synthetic graphs.

prediction accuracy. All models predict RNE well as a weak performing algorithm with high HPD^2_C.

c) Predictability and recommendation: FF, RNE, MHRW, and XS demonstrate the highest predictability, with hop-plot prediction RMSEs below 16 %. FF and XS yield higher-quality samples, while RNE produces lower-quality ones, contributing to their predictability. Conversely, RN, SB, FS, and RDN pose challenges for hop-plot preservation prediction. RN, SB, and FS are among the poorest algorithms [32] generating more diverse results. We identified RF as the dominant hop-plot prediction model regarding RMSE, particularly on large graphs with over 100 thousand nodes. Table IX represents a lower Hits@1 results of HPD^2 for all methods with relatively higher (15 %) for RF and kNN. MLP yields the highest Hits@1 of 30 % for HPD^2_C as it better learns from connected components. Hits@3 of kNN and K-best selection are comparable by about 53 %.

d) Feature importance: Sampling features ranked among the highest for predicting hop plots, while $\overline{\mathrm{PRC}}$, CCS, $|\mathrm{CComp}|$ and Deg are more relevant graph features (Figure 7). LCCs have a higher variance and influence on HPD^2_C, which is sensitive to their size. Furthermore, $\mathrm{MSTD}_{\max}$ is also critical for RF's HPD^2_C prediction, while MST depends on the structure of LCC.

TABLE IX: Sampling Hits@k quality ranking on real graphs.

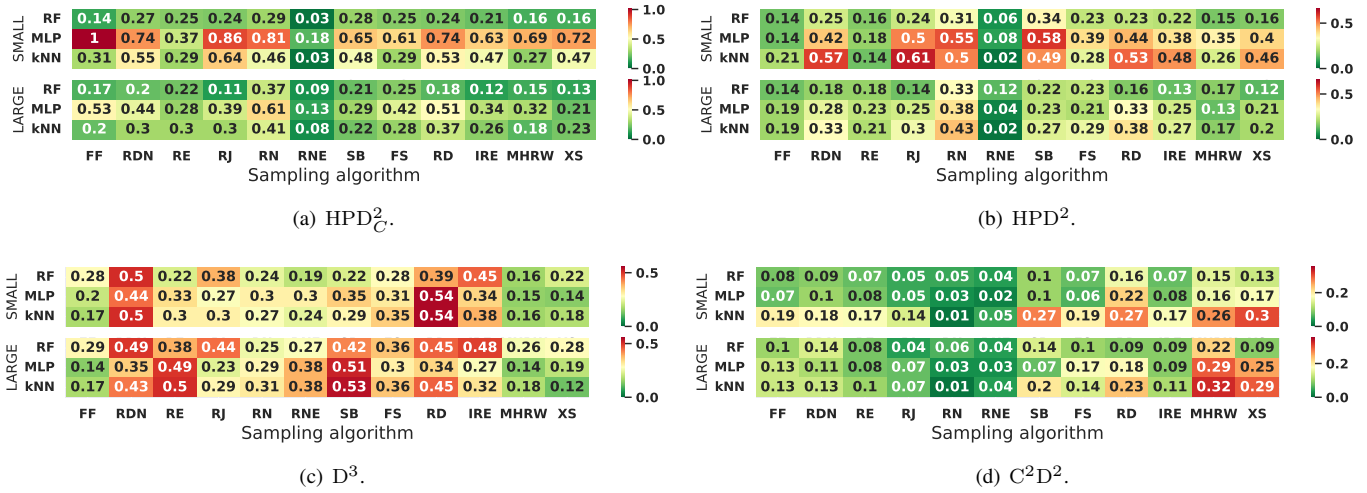
Method	*Hits@1*				*Hits@3*			
	D^3	$\mathrm{C}^2\mathrm{D}^2$	HPD^2	HPD^2_C	D^3	$\mathrm{C}^2\mathrm{D}^2$	HPD^2	HPD^2_C
Random	0.00	0.10	0.05	0.20	0.25	0.30	0.17	0.27
K-best	0.10	**1.00**	0.05	0.05	0.40	0.63	**0.53**	**0.53**
MLP	**0.25**	0.95	0.05	**0.30**	**0.43**	**0.72**	0.48	0.52
RF	0.10	**1.00**	**0.15**	0.05	**0.43**	0.68	0.47	0.47
kNN	0.15	0.20	**0.15**	0.15	**0.43**	0.37	0.52	**0.53**

2) $\mathrm{C}^2\mathrm{D}^2$ forecast: Overall, $\mathrm{C}^2\mathrm{D}^2$ is a more predictable metric due to its lower values for most instances in the training and real testing experiments, decreasing the prediction diversity. RF is the most generalizable model in $\mathrm{C}^2\mathrm{D}^2$ prediction, showing lower errors.

a) SMALL graphs: Figure 6(d) shows better predictions by RF and MLP for SMALL graphs, with lower errors (maximum 16 %) for RF, possibly due to the higher sensitivity to the training data diversity with less diversity in $\mathrm{C}^2\mathrm{D}^2$. kNN fails to predict on these graphs except for RN and RNE.

b) LARGE graphs: Figure 6(d) illustrates better RF predictions with a maximum RMSE of 22 % on LARGE graphs. However, the majority of models have good accuracy with errors below 15 % for most algorithms (see Figure 6(d)).

c) Predictability and recommendation: We find the majority of the well-performing sampling algorithms in CC



(a) HPD^2_C.

(b) HPD^2.

(c) D^3.

(d) $\mathrm{C}^2\mathrm{D}^2$.

Fig. 6: Predicted sampling RMSE for different algorithms and metrics on real-world graphs.

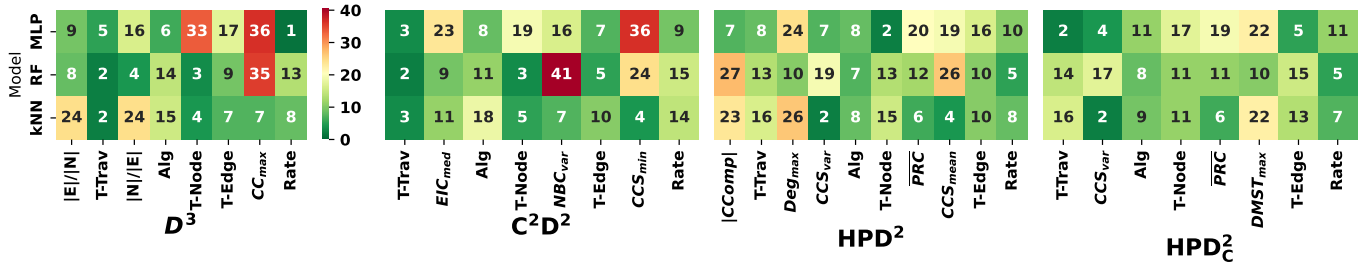


Fig. 7: Average feature rankings for all ML models.

preservation as the most predictable with $\mathrm{C}^2\mathrm{D}^2$ RMSEs under $10\,\%$ by the best ML model. The higher $\mathrm{C}^2\mathrm{D}^2$s of RD, MHRW, and XS (and slightly SB) [32] results in higher prediction errors due to their diversity from other instances. We find all models suitable for $\mathrm{C}^2\mathrm{D}^2$ prediction (according to RMSE), especially RF, due to the lower diversity of training data. MHRW is the most difficult algorithm, for which no model delivered a prediction error lower than $15\,\%$. Table IX demonstrates accurate top algorithm selection by K-best and RF. MLP results in the highest Hits@3 of $72\,\%$ for $\mathrm{C}^2\mathrm{D}^2$. Overall, MLP and RF are the dominant models.

d) Feature importance: Sampling algorithm features, particularly traversal type, are important for $\mathrm{C}^2\mathrm{D}^2$, and the specific algorithm and the sample rate are more influential for MLP (see Figure 7). Among the graph features, $\overline{\mathrm{EIC}}$, CCS and NBC have the highest rank.

3) D^3 forecast: We find D^3 the most difficult metric to predict. Due to the power-law characteristic of real graphs, node degrees have no lower bound, and thus, preserving their distribution by samplings is rather difficult.

a) SMALL graphs: Figure 6(c) shows higher D^3 RMSEs for all models compared to the other metrics. It shows RMSEs higher than $25\,\%$ for RDN, RJ, FS, RD, and IRE on SMALL graphs. Overall, RF yielded slightly better predictions driven by its generalizability to unseen features. FF, MHRW, and XS are predictable by MLP and kNN with a $20\,\%$ RMSE, while RNE, RE, SB, and RN by RF with a maximum of $25\,\%$ RMSE.

b) LARGE graphs: We observed higher model accuracy on FF, MHRW, XS, and RN algorithms for LARGE graphs. MLP and kNN are good choices for most traversal algorithms (i.e., FF, RJ, MHRW, XS, and IRE), whereas RF fails to predict D^3 because of its sensitivity to its diversity. RF predicts RNE and RN algorithms better.

c) Predictability and recommendation: FF, RJ, MHRW, and XS are more predictable on large graphs. Conversely, SB, RE, RD, and RDN are the most unpredictable algorithms, exhibiting the highest RMSE. MLP and kNN are dominant models for D^3 prediction on most traversal algorithms, while RF is appropriate for RNE and RN. Table IX represents the algorithm selection with an expected lower Hits@1 due to ML models' higher D^3 RMSE on real graphs. MLP yields slightly better Hits@1 of $25\,\%$. However, ML models better retrieve the three best algorithms with $43\,\%$ of Hits@3.

d) Feature importance: Figure 7 indicates that the sampling algorithm, type, and rate are the most influential factors for D^3 prediction. Traversal algorithms generally preserve degree distribution, notably affecting the predictions. RF and kNN predictions heavily rely on sampling type information, whereas MLP requires specific details about the sampling algorithm and rate for accurate estimates. Regarding graph

TABLE X: Actual and predicted top algorithms with the least HPD^2 for five large graphs.

Graph	Caida		Gowalla		Digg		Hyves		Actor	
Rate	0.1	0.3	0.1	0.3	0.1	0.3	0.1	0.3	0.1	0.3
Actual	RJ	RN	RJ	RN	RN	RJ	RJ	FF	RJ	RJ
RF-predicted	RJ	RJ	RDN	RJ	RJ	RJ	RJ	RJ	RJ	RJ

TABLE XI: Recommended sampling algorithm prediction model for real graphs (S: SMALL, L: LARGE).

	D^3			$\mathrm{C}^2\mathrm{D}^2$			HPD^2_C			HPD^2		
	RF	*MLP*	*kNN*	RF	*MLP*	*kNN*	RF	*MLP*	*kNN*	*RF*	*MLP*	*kNN*
FF		✓	✓	✓	✓	L	✓			✓		
RDN				✓	✓	L	✓			✓		
RE				✓	✓	L	✓			✓		
RJ		L	L	✓	✓	L	✓			✓		
RN	✓			✓	✓	✓						
RNE	✓			✓	✓	✓	✓		✓	✓	✓	✓
SB				✓	✓		✓		L	✓		
FS				✓			✓					
RD				✓			✓			✓		
IRE				✓	✓	L	✓			✓		
MHRW	S	✓	✓				✓			S	L	
XS		✓	✓	L			✓			✓		

features, node-edge ratios are more pertinent for MLP and RF, while $\mathrm{CC}_{\max}$ is crucial for kNN.

C. State-of-the-art comparison

The evaluations by [6] indicate better performance of the FF algorithm compared to RN, RDN, RE, RNE, and RJ on HPD^2 for real graphs analyzed (considering only the algorithms within the scope of our research). However, our evaluations on five large graphs indicate that the RJ algorithm outperforms the others in most experiments, as represented in Table X, correctly identified by RF in $50\,\%$ of the cases. Only one instance for the `Hyves` graph with a rate of 0.3 aligns with the expectation of [6], where FF is better. The limited generalizability of the findings in [6] to large graphs is due to the restricted experiments to four graph types with maximum $75\,000$ nodes and $510\,000$ edges. Nevertheless, it identified RJ as a high-quality algorithm with HPD^2 of 0.26.

D. Summary

According to the varying accuracy of the ML models for different sampling algorithms and metrics, we provide model recommendations producing lower RMSE for different sampling algorithms in Table XI.

VI. Runtime Performance

We analyze performance optimizations to our ML models and the runtime prediction of the sampling algorithms.

A. ML optimisations

We applied three optimization methods during data preprocessing and ML model training:

a) Feature extraction parallelism: achieved a 20-fold speedup for ten major feature computations over ten graphs of a minimum of $100\,000$ nodes using 100 cores;

b) Sampling preparation parallelism: achieved approximately 25-fold speedup in sampling from 30 graphs with over $100\,000$ nodes including sample evaluation.

TABLE XII: Graph sampling runtime recommendations.

(a) Hits@k for synthetic graphs.

Model	*Hits@1*	*Hits@3*
Random	0.07	0.26
K-best	0.13	0.68
kNN	0.15	**0.87**
RF	0.08	0.32
MLP	**0.76**	**0.86**

(b) MLP recommendation.

Topology	*Algorithm*	*Runtime*
AB	RJ	0.07
	RN	0.94
	FF	3.08
WS	RJ	0.42
	FF	1.05
	RN	3.51
ER	RJ	-0.12
	FF	1.22
	RN	2.26

(c) Average Hits@k for synthetic graph types.

Model	*Hits@1*				*Hits@3*			
	AB	*ER*	*WS*	*All*	*AB*	*ER*	*WS*	*All*
RF	0.02	0.05	0.15	0.08	0.29	0.3	0.36	0.32
MLP	**0.85**	**0.60**	**0.72**	**0.76**	**0.86**	0.83	**0.93**	0.86
kNN	0.05	0.10	0.31	0.15	0.83	**0.93**	0.90	**0.87**

(d) Average Hits@k for real-world graphs.

Dataset	*Model*	*Hits@1*	*Hits@3*
SMALL	Random	0.15	0.27
	K-best	0.55	0.85
	kNN	**0.60**	**0.87**
	RF	0.40	0.40
	MLP	0.15	0.62
LARGE	Random	0.00	0.28
	K-best	0.17	0.86
	kNN	0.20	**0.93**
	RF	0.08	0.36
	MLP	**0.42**	0.67

c) Approximation-based optimization: includes `Networkit-Riondato` NBC approximation, achieved a speedup of up to 200 having 0.01 error with minimum $90\,\%$ probability for large graphs and `Networkit-ANF` approximating hop-plots, achieved around 70-fold speedup.

B. Runtime prediction

Sampling runtime is crucial for large graphs, aiding in algorithm selection based on desired quality trade-offs. However, predicting runtime proved challenging in our experiments, with high error rates, due to the diverse runtimes spanning from ms to $27\,\mathrm{h}$. Despite the large prediction errors, the hits@k algorithm rankings surprisingly surpassed baseline algorithm selections and are a good basis for recommendations. However, RF struggled to learn effectively and make predictions due to the high variability in the training runtimes.

1) Synthetic graphs: MLP achieved a higher runtime Hits@1 and Hits@3 of $76\,\%$ and $86\,\%$ compared to random and K-best baselines that fail in best algorithm selection (see Table XII(a)). kNN effectively retrieves the top three algorithms by $87\,\%$. A detailed analysis of algorithm recommendations across graph types in Table XII(c) demonstrates the general superiority of MLP for all graph types in algorithm selection thanks to the similarity between training and test graphs. The sole exception was the ER graphs with intricate patterns that hindered the MLP's ability to capture the properties accurately, for which kNN attained the highest hits@3 of $93\,\%$. MLP yields especially accurate predictions for fast algorithms on large graphs with execution times below $10\,\mathrm{s}$. In

TABLE XIII: Sampling algorithms' runtime complexity tradeoffs.

(a) DBLP dataset predictions.

Algorithm	HPD^2	*Runtime [s]*
RJ	0.13	2.20
RN	0.82	0.68
MHRW	0.68	1.11

(b) Best (Ω) and worst ($\mathcal{O}$) case complexity.

Algorithm	*Complexity*
RE, RNE	$\mathcal{O}(\lvert E\rvert)$
RN, IRE	$\mathcal{O}(k \cdot \lvert E\rvert)$
RDN	$\mathcal{O}(N + k \cdot \log N + k \cdot \lvert E\rvert)$
RJ, SB, FF, FS	$\Omega(k)$
MHRW, RD	$\Omega(k \cdot N)$
XS	$\mathcal{O}(k \cdot N^2)$

particular, MLP recommended RJ, RN, and FF as the fastest sampling algorithms predicted below $4\,\mathrm{s}$ (see Table XII(b)).

2) Real-world graphs: The dissimilarity of real graphs lowers the hits@1 recommendations, as detailed in Table XII(d). Nonetheless, Hits@3 demonstrates enhanced accuracy.

a) SMALL: graphs achieved a Hits@1 of $60\,\%$ and a Hits@3 of $87\,\%$ using kNN that could find better neighborhood graphs thanks to the similar training data, comparable to the k-best selection.

b) LARGE: graphs exhibited significantly improved ML accuracy over baseline methods. MLP excels with a Hits@1 of $42\,\%$, while kNN remains the most accurate in the Hits@3 algorithm selection, achieving a remarkable $93\,\%$. The different fastest algorithms among SMALL and LARGE graphs led to the failure of K-best selection.

VII. Discussion and Limitations

This section discusses challenges in sampling algorithm runtime quality tradeoffs and feature extraction for handling dissimilar real-world graphs.

1) Model selection: In preliminary experiments, a message-passing GNN with naive initialization and without sampling produced lower accuracy than manual feature extraction by capturing limited neighbourhood graph properties, insufficient for higher-order features such as distance requiring deep traversals. Combining GNNs and manual feature extraction for initializing embeddings may extend our ML portfolio.

2) Runtime and quality tradeoff: Since fast algorithms can degrade sampling quality, it is valuable to consider a tradeoff between runtime and quality to guide the algorithm selection. For example, Table XIII(a) shows the predicted runtime and HPD^2 tradeoffs for the DBLP graph. kNN predicts RN, MHRW, and RJ as the fastest algorithms, while RF predicts RJ with the lowest HPD^2. Therefore, RJ is a tradeoff choice since RN and MHRW have low-quality HPD^2 samples.

3) Sampling complexity: Although the computational complexity of algorithms can help estimate their runtimes, it is highly dependent on the accuracy of the approximation. Since we could not find a thorough and accurate complexity analysis for the sampling algorithms in the literature, we derived non-tight upper or lower bounds (see Table XIII(b)) assuming a constant $\mathcal{O}(1)$ random node and edge selection. The lower complexity for RJ, SB, FF, and FS is in the best case, and the high complexity for XS is in the worst case, which is partially consistent with our measurements.

4) Graph scalability: The lower prediction errors (especially MLP) for large synthetic graphs exhibiting higher similarity to the training data (see Figure 3) are promising for the generalizability and scalability of the ML models.

5) Graph dissimilarity: Real graphs exhibit diverse properties that are difficult to replicate with synthetic data, impacting each evaluation criterion differently. Specifically, the power-law degree distribution of real graphs can lead to more varied D^3s, negatively affecting model training. In contrast, the diversity in HPD^2_C and HPD^2 has less impact on predictions (having lower RMSEs) due to the shrinking diameter of growing graphs, resulting in shorter paths [33]. Consequently, sampling quality largely depends on the size of the components and the properties of the sampling method. Effective path or connectivity-preserving algorithms, particularly traversal-based methods, can yield high-quality samples, enhancing prediction accuracy for unseen graph features. Additionally, prediction accuracy relies more on algorithmic than graph features, suggesting transfer learning across different graph datasets as particularly beneficial for metrics such as D^3.

6) Sampling predictability: Our study indicates that ML models are more effective at predicting sampling algorithms that yield extreme outcomes, allowing us to identify well-performing and exclude suboptimal ones. For other algorithms, placing higher emphasis on graph features is crucial to enhance prediction accuracy. We can improve their predictability by identifying each algorithm's most influential graph features, such as those derived from the MI process. Specifying the graph characteristics under which these algorithms are less predictable will further refine our predictive models. For instance, analyzing D^3 prediction results of the most unpredictable algorithms, SB and RE, on large real graphs with a sampling rate of 0.1 reveals that RF usually fails to predict them for graphs with low $\overline{CC}$. Additionally, graphs with low $|N|$ and $|CComp|$, result in RE's low predictability by RF.

7) Feature extraction: Large graph processing is highly complex, especially for BC and SPL extraction over hundreds of thousands of nodes. For example, approximating BC for a graph with 450 thousand nodes and 4.9 million edges takes around three hours on a server with 64 cores and $754\,\mathrm{GB}$ of memory. The high complexity hinders feature extraction and challenges real-time usability. Moreover, calculations are inefficient for dense graphs with a high edge count. We plan several strategies to improve scalability: 1) Explore more approximation techniques or use feature learning methods (e.g., GNN); 2) Replace time-intensive features like NBC and SPL with simpler ones such as k-length distances or linear complexity ones (according to $|E|$), like k-cores, along with additional statistics for accuracy; 3) Implement streaming graph probing and feature updates instead of recalculation.

8) Feature importance: As a remedy for the high complexity of feature calculation, the feature ranking results in Figure 7 suggest eliminating or weighting the following features with low importance and complex extraction time (beyond square-polynomial in number of nodes) for different metrics: SPL, assortativity, and betweenness for D^3, SPL, assortativity and

CC for C^2D^2 and SPL, CC and betweenness for HPD^2 and HPD^2_C. The feature reduction will improve model efficiency without high degradation in accuracy.

VIII. Conclusions

We proposed three ML models for predicting the quality of sampling algorithms and formulating recommendations regarding four metrics: D^3, C^2D^2, HPD^2, and HPD^2_C. The experimental findings indicate improved MLP and kNN prediction accuracy on synthetic graphs, while RF yields lower prediction errors on real graphs, particularly for C^2D^2 and HPD^2. Algorithm recommendation results indicate improved accuracy over most metrics' baseline algorithm selection models. Algorithm ranking evaluations on synthetic graphs indicate Hits@3 of $89\,\%$ by MLP for D^3, $65\,\%$ by kNN for C^2D^2, $63\,\%$ by RF for HPD^2 and $76\,\%$ by MLP for HPD^2_C. Algorithm recommendations on real graphs yield hits@3 of $43\,\%$ by all ML models for D^3, $72\,\%$ by MLP for C^2D^2 and $53\,\%$ by kNN for HPD^2_C. Runtime algorithm recommendation shows the hits@3 over $86\,\%$ on synthetic and real graphs and hits@1 over $60\,\%$ on small graphs, surpassing the random and K-best baseline methods. Finally, we discuss the predictability of the algorithms concerning quality metrics, along with ML explainability analysis validating the models by identifying influential sampling and graph features.

In the future, we plan to address the following aspects: 1) more efficient feature extraction methods like GNNs; 2) mutual information analysis on real graphs to extract relevant features; 3) near-realistic generative graph models for improved training; 4) intelligent graph sampling selection using reinforcement learning for static and streaming graphs.

Acknowledgment

This work received funding from: 1) *Horizon Europe* research and innovation program, grant agreements 101093202 (Graph-Massivizer); 2) *Austrian Research Promotion Agency (FFG)*, grant agreement 888098 (Kärntner Fog); 3) *Romanian Ministry of Research, Innovation, and Digitalization*, grant agreement CN760046/3.05.2024 (JobKG).

References

[1] D. H. Wolpert and W. G. Macready, "No Free Lunch Theorems for Search," Santa Fe Institute, Tech. Rep. 12, 1995.

[2] P. Hu and W. C. Lau, "A Survey and Taxonomy of Graph Sampling," *arXiv preprint arXiv:1308.5865*, 2013.

[3] M. P. Stumpf *et al.*, "Subnets of Scale-free Networks are not Scale-free: Sampling Properties of Networks," *Proceedings of the National Academy of Sciences*, vol. 102, no. 12, pp. 4221–4224, 2005.

[4] S. H. Lee *et al.*, "Statistical Properties of Sampled Networks," *Physical Review E*, vol. 73, no. 1, p. 016102, 2006.

[5] N. K. Ahmed *et al.*, "Graph Sample and Hold: A Framework for Big-graph Analytics," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14. New York, NY, USA: ACM, 2014, p. 1446–1455.

[6] J. Leskovec and C. Faloutsos, "Sampling from Large Graphs," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '06. Philadelphia, PA, USA: ACM, 2006, p. 631–636.

[7] S. Yoon *et al.*, "Statistical Properties of Sampled Networks by Random Walks," *Physical Review E*, vol. 75, p. 046114, 2007.

[8] F. Zhang *et al.*, "A Visual and Statistical Benchmark for Graph Sampling Methods," in *Exploring Graphs at Scale Workshop*, vol. 3, 2015.

[9] M. I. Yousuf *et al.*, "Empirical Characterization of Graph Sampling Algorithms," *Social Network Analysis and Mining*, vol. 13, no. 1, p. 66, 2023.

[10] F. Hutter *et al.*, "Algorithm Runtime Prediction: Methods & Evaluation," *Artificial Intelligence*, vol. 206, pp. 79–111, 2014.

[11] N. Park *et al.*, "Metagl: Evaluation-free Selection of Graph Learning Models via Meta-Learning," in *The Eleventh International Conference on Learning Representations*, 2022.

[12] A. D. Popescu *et al.*, "Predict: Towards Predicting the Runtime of Large Scale Iterative Analytics," *Proceedings of the VLDB Endowment*, vol. 6, no. 14, p. 1678–1689, Sep. 2013.

[13] H. P. Samoaa *et al.*, "TEP-GNN: Accurate Execution Time Prediction of Functional Tests using Graph Neural Networks," in *International Conference on Product-Focused Software Process Improvement*. Springer, 2022, pp. 464–479.

[14] J. Illenberger and G. Flötteröd, "Estimating Network Properties from Snowball Sampled Data," *Social Networks*, vol. 34, no. 4, pp. 701–711, 2012.

[15] B. Ribeiro and D. Towsley, "On the Estimation Accuracy of Degree Distributions from Graph Sampling," in *51st IEEE Conference on Decision and Control (CDC)*. Maui, HI, USA: IEEE, 2012, pp. 5240–5247.

[16] T. Everitt and M. Hutter, "Analytical Results on the BFS vs. DFS Algorithm Selection Problem. Part I: Tree Search," in *AI 2015: Advances in Artificial Intelligence: 28th Australasian Joint Conference, November 30–December 4, 2015, Proceedings 28*. Springer, 2015, pp. 157–165.

[17] M. Ahmad *et al.*, "Heteromap: A Runtime Performance Predictor for Efficient Processing of Graph Analytics on Heterogeneous Multi-Accelerators," in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2019, pp. 268–281.

[18] Z. Chu *et al.*, "A Novel Deep Learning Method for Query Task Execution Time Prediction in Graph Database," *Future Generation Computer Systems*, vol. 112, pp. 534–548, 2020.

[19] A. Myakushina, "Exploring Sampling Techniques in Large Graphs and Networks," Ph.D. dissertation, University of Rochester, Spring 2023.

[20] N. K. Ahmed *et al.*, "Network Sampling: From Static to Streaming Graphs," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 8, no. 2, pp. 1–56, 2013.

[21] X. Qi, "A Review: Random Walk in Graph Sampling," *arXiv preprint arXiv:2209.13103*, 2022.

[22] J. Leskovec *et al.*, "Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations," in *11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. Chicago Illinois USA: ACM, 2005, pp. 177–187.

[23] E. Voudigari *et al.*, "Rank Degree: An Efficient Algorithm for Graph Sampling," in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. San Francisco, CA, USA: IEEE, 2016, pp. 120–129.

[24] A. S. Maiya and T. Y. Berger-Wolf, "Sampling Community Structure," in *19th International Conference on World Wide Web*. Raleigh North Carolina USA: ACM, 2010, pp. 701–710.

[25] M. T. Ribeiro *et al.*, ""Why Should I Trust You?" Explaining the Predictions of any Classifier," in *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco California USA: ACM, 2016, pp. 1135–1144.

[26] A.-L. Barabasi *et al.*, "Network Science Book Project," *Network Science Book Project*, 2012.

[27] J. Kunegis, "Konect: The Koblenz Network Collection," in *Proceedings of the 22nd International Conference on World Wide Web*. Rio de Janeiro Brazil: ACM, 2013, pp. 1343–1350.

[28] R. Rossi and N. Ahmed, "The Network Data Repository with Interactive Graph Analytics and Visualization," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, vol. 29, no. 1. Austin Texas: AAA Press, 2015.

[29] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford Large Network Dataset Collection," http://snap.stanford.edu/data, jun 2014.

[30] C. R. Palmer *et al.*, "ANF: A Fast and Scalable Tool for Data Mining in Massive Graphs," in *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Edmonton Alberta Canada: ACM, 2002, pp. 81–90.

TABLE XIV: Models space complexity.

Model	*Space complexity*	*Size*
kNN	$\mathcal{O}(\lvert F\rvert \cdot \lvert TS\rvert + \lvert H\rvert)$	3 MB – 5 MB
MLP	$\mathcal{O}(\lvert F\rvert \cdot \lvert L_1\rvert + \lvert L_1\rvert \cdot \lvert L_2\rvert + \lvert L_2\rvert)$	100 kB – 400 kB
RF	$\mathcal{O}(\lvert Tr\rvert \cdot \lvert TS\rvert)$	10 MB – 80 MB

TABLE XV: Models training time complexity.

Model	*Training time complexity*	*Time*
kNN	$\mathcal{O}(\lvert F\rvert \cdot \lvert TS\rvert \cdot \lvert Ne\rvert)$	~ 10 min
MLP	$\mathcal{O}\left(\lvert E\rvert \cdot \lvert TS\rvert \cdot (\lvert F\rvert \cdot \lvert L_1\rvert + \lvert L_1\rvert \cdot \lvert L_2\rvert + \lvert L2\rvert)\right)$	~ 5 h
RF	$\mathcal{O}\left(\lvert T\rvert \cdot F_{\max} \cdot \lvert TS\rvert \cdot \log(\lvert TS\rvert)\right)$	~ 5 h

[31] M. Riondato and E. M. Kornaropoulos, "Fast Approximation of Betweenness Centrality through Sampling," *Data Mining and Knowledge Discovery*, vol. 30, pp. 438–475, 2016.

[32] S. H. S. Dizaji *et al.*, "An Extensive Characterization of Graph Sampling Algorithms," in *Companion of the 15th ACM/SPEC International Conference on Performance Engineering*. London, United Kingdom: ACM, 2024, p. 135–140.

[33] J. Leskovec *et al.*, "Graph Evolution: Densification and Shrinking Diameters," *ACM transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, pp. 2–es, 2007.

[34] J. Pineau *et al.*, "Improving Reproducibility in Machine Learning Research (a Report from the Neurips 2019 Reproducibility Program)," *Journal of machine learning research*, vol. 22, no. 164, pp. 1–20, 2021.

APPENDIX

Reproducibility artifact

We followed the ML checklist proposed in the *NeurIPS-2019 Reproducibility Program* [34] and released the code and dataset information in the `GitHub` repository summarized in this section (https://github.com/halehdizaji/Graph-Sampling-Quality-Prediction-for-Algorithm-Recommendation).

1) Sample complexity: multiplies the number of training and testing graphs (700), graph and sampling features (66), totaling 14 700 examples and approximately 25 MB per metric.

2) Model space complexity: in Table XIV depends on the sizes of the input feature set F, training data set TS, hyperparameter set H, the two MLP layers L_1 and L_2, and the number of trees $|Tr|$, where $|.|$ is the set modulo operator.

3) Time complexity: with respect to training (Table XV) and inference (Table XVI) depends on the number of neighbors Ne for kNN, the number of epochs E and model size for MLP, and the number of estimators T and maximum features at a split point $F_{\max}$ for RF trees.

4) Synthetic data: parameters for training and testing graphs displayed in Table XVII are the number of nodes $|N|$, new edges per node in AB E_{new}, degree in WS graphs Deg_{ws}, edge probability in ER graphs P_{er}, triangle probability in PLC P_{tri}, forward/backward probabilities in FFM graphs $P_{fwd/bwd}$, number of clusters C and inter-cluster to intra-cluster edge probability (P_{inter} and P_{intra}) ratio in SBM α_{sbm}. We calculated the required parameters for generating AB, WS, and SBM graphs, given the predefined graph features (N, D, C and α_{sbm}), according to the following equations (with equal size clusters in SBM graphs):

$$
\begin{aligned}
E_{new} &= \left\lfloor |N| \cdot \frac{D}{2} \right\rfloor; \\
Deg_{ws} &= \lfloor D \cdot (|N| - 1) \rfloor; \\
P_{intra} &= \frac{D \cdot (|N| - 1) \cdot C}{|N| - C + \alpha_{sbm} \cdot \frac{|N|}{C-2}}; \\
P_{inter} &= P_{intra} \cdot \alpha_{sbm}.
\end{aligned}
$$

TABLE XVI: Models inference time complexity.

Model	*Inference time complexity*
kNN	$\mathcal{O}(\lvert F\rvert \cdot \lvert TS\rvert \cdot \lvert Ne\rvert)$
MLP	$\mathcal{O}\left(\lvert F\rvert \cdot \lvert L_1\rvert + \lvert L_1\rvert \cdot \lvert L_2\rvert\right)$
RF	$\mathcal{O}(\lvert T\rvert \cdot \log(\lvert TS\rvert))$

TABLE XVII: Synthetic graph generation parameters.

Dataset	*Type*	*Parameters*		*Graphs*	*Library*
		Name	*Range*		
Training	*AB*	$\lvert N\rvert$	10.000 – 100.000	120	`Networkx`
		$\lvert E_{new}\rvert$	1 – 42		
	WS	$\lvert N\rvert$	10.000 – 100.000	119	`Networkx`
		Deg_{ws}	2 – 98		
	ER	$\lvert N\rvert$	10.000 – 100.000	174	`Networkx`
		P_{er}	0.000 01 – 0.001		
	PLC	$\lvert N\rvert$	10 000 – 45.000	60	`Networkx`
		$\lvert E_{new}\rvert$	2 – 20		
		P_{tri}	0.2 – 1		
	FFM	$\lvert N\rvert$	5000 – 50.000	36	`SnapStanford`
		$P_{fwd/bwd}$	0.01 – 0.3		
	SBM	$\lvert N\rvert$	5000 – 50 000	155	`Networkx`
		C	10 – 25		
		α_{sbm}	0.01 – 0.1		
Testing	AB	$\lvert N\rvert$	150 000 – 450 000	17	`Networkx`
		$\lvert E_{new}\rvert$	1 – 42		
	WS	$\lvert N\rvert$	150 000 – 450.000	9	`Networkx`
		Deg_{ws}	2 – 44		
	ER	$\lvert N\rvert$	150 000 – 450 000	10	`Networkx`
		P_{er}	0.000 008 – 00.0001		

We used `Networkx` and `SanpStanford` libraries for generating graphs.

5) Code: comprises five important modules.

a) Software dependencies: involve `Python` and `bash` scripting, with the following requirements:

- `Python-3.8` and Jupyter Notebook for model development and evaluation;
- `Networkx-2.8`, `Networkit-11.0`, and `SnapStanford-6.0` for graph processing;
- `SLURM-19.05` for job scheduling on servers.

b) Preprocessing: includes graph feature extraction, sampling with quality and performance evaluation (means and variances of metrics over five iterations), mutual information analysis for feature selection, and normalization (Table II).

c) Data analysis: includes duplicate elimination and visualization using `sklearn-tSNE`.

d) Training and evaluation: use `sklearn` for ML and `LIME` explainability.

e) `README` file: includes data preparation, model training, and testing instructions.

6) Experimental results:

a) Hyperparameter: tuning of ML models utilizes `GridSearchCV` with five-fold cross-validation (see Section IV-B and the `GitHub` repository for the detailed search space), summarized in Table V.

b) Training: runs 500 MLP epochs and full-batch for RF and kNN.

c) Evaluation: runs one inference iteration.

d) Evaluation metrics: see Section III-E.

e) Result description: see Section V.

f) Computing infrastructure: involve two high-performance servers for running the experiments:

- Two servers with Intel `Xeon-Gold 5218` processors at 2.30 GHz with 64 cores and 754 GB of memory, respectively 128 cores and 384 GB of memory;
- One Nvidia `Quadro RTX8000` GPU with 48 GB of memory.