Predicting Graph Operator Output over Multiple Graphs

Tasos Bakogiannis[1], Ioannis Giannakopoulos[1], Dimitrios Tsoumakos[2], and Nectarios Koziris[1]

[1] CSLab, School of ECE, National Technical University of Athens, Greece
{abk,ggian,nkoziris}@cslab.ece.ntua.gr

[2] Department of Informatics, Ionian University, Greece
dtsouma@ionio.gr

Abstract. A growing list of domains, in the forefront of which are Web data and applications, are modeled by graph representations. In content-driven graph analytics, knowledge must be extracted from large numbers of available data graphs. As the number of datasets (a different type of volume) can reach immense sizes, a thorough evaluation of each input is prohibitively expensive. To date, there exists no efficient method to quantify the impact of numerous available datasets over different graph analytics tasks. To address this challenge, we propose an efficient graph operator modeling methodology. Our novel, operator-agnostic approach focuses on the inputs themselves, utilizing graph similarity to infer knowledge about them. An operator is executed for a small subset of the available inputs and its behavior is modeled for the rest of the graphs utilizing machine learning. We propose a family of similarity measures based on the degree distribution that prove capable of producing high quality models for many popular graph tasks, even compared to modern, state of the art similarity functions. Our evaluation over both real-world and synthetic graph datasets indicates that our method achieves extremely accurate modeling of many commonly encountered operators, managing massive speedups over a brute-force alternative.

Keywords: Graph Analytics · Operator Modeling · Graph Similarity

1 Introduction

A huge amount of data originating from the Web can be naturally expressed as graphs, e.g., product co-purchasing graphs [25], community graphs [28], etc. Graph analytics is a common tool used to effectively tackle complex tasks such as social community analysis, recommendations, fraud detection, etc. Many diverse graph operators are available [12], with functionality including the computation of centrality measures, clustering metrics or network statistics [8], all regularly utilized in tasks such as classification, community detection and link prediction.

Yet, as Big Data technologies mature and evolve, emphasis is placed on areas not solely related to data (i.e., graph) size. A different type of challenge steadily shifts attention to the actual content. In content-based analytics [14], data is processed for sense-making. Similarly, in *content-sensitive* applications the quality of insights derived is mainly attributed to the input content. The plethora of available sources for content-sensitive analytics tasks now creates an issue: Data scientists have to decide which of the available datasets will be used for a given workflow, in order to maximize its impact. Yet, as modern analytics tasks have evolved into increasingly long and complex series of diverse operators, evaluating the utility of immense numbers of inputs is prohibitively expensive. This is notably true for graph operators, whose computational cost has led to extensive research on approximation algorithms (e.g., [30, 11]).

As a motivating example, let us consider a dataset consisting of a very large number of citation graphs. We wish to identify those graphs that have the most well-connected citations and contain highly-cited papers. As a result, the clustering coefficient [8], a good measure of neighborhood connectivity, would have to be computed for all the graphs in the dataset in order to allow the identification of the top-k such graphs. To quantify the importance of each paper, we consider a centrality measure such as betweenness centrality [8]. Consequently, we would have to compute the maximum betweenness centrality score for each citation graph and combine the results with those obtained from the analysis based on the clustering coefficient. Yet, this could be a daunting task due to the operators' complexity and the size of the dataset.

The challenge this work tackles is thus the following: Given a graph analytics operator and a large number of input graphs, can we reliably predict operator output at low cost? In this work, we introduce a novel, *operator-agnostic* dataset profiling mechanism. Rather than executing the operator over each input graph, our work leverages the relationship between the dataset's graphs, expressed through a similarity measure, and infers knowledge about them. In our example, instead of exhaustively computing the clustering coefficient, we calculate a similarity matrix for our dataset, compute the clustering coefficient for a small subset of graphs and utilize the similarity matrix to estimate its value for the remaining graphs. We may then compute the maximum betweenness centrality for also a small subset of citation graphs and *reuse* the already calculated similarity matrix to estimate the scores for the rest of the graphs.

Our method is based on the intuition that, for a given graph operator, similar graphs produce similar outputs. This intuition is solidly supported by the existence of strong correlations between different graph operators ([22, 7, 20]). Hence, by assuming a similarity measure that correlates to a set of operators, we can use machine learning techniques to approximate their outcomes. Given a graph dataset and an operator to model, our method utilizes a similarity measure to compute the similarity matrix of the dataset, i.e., all-pairs similarity scores between the graphs of the dataset. The given operator is then run for a small subset of the dataset; using the similarity matrix and the available operator outputs, we are able to approximate the operator for the remaining graphs. To the best of our knowledge, this is the first effort to predict graph operator output over large datasets. In summary, we make the following contributions:

- We propose a novel, similarity-based method to estimate graph operator output for large graph datasets. This method shifts the complexity of numerous

graph computations to less expensive graph similarities. This choice offers two major advantages: First, our scheme is *operator-agnostic* since the computed similarity matrix can be reused. As a result, the similarity matrix computation is amortized and the cost of our method is ultimately dominated by the computation of that operator for a small subset of the dataset. Second, the method is agnostic to the similarity measure that is used. This property gives us the ability to utilize or arbitrarily combine different similarity measures.
- We introduce a family of similarity measures based on degree distribution with a gradual trade-off between detail and computational complexity. Despite their simplicity, they prove capable of producing highly accurate models, comparable or even surpassing other more costly, state-of-the-art similarity measures ([32, 35]).
- We offer an open-source implementation[3] of our method and perform an extensive experimental evaluation using both synthetic and real datasets. Our results indicate that we can accurately model a variety of popular graph operators, with errors that can be $< 1\%$, sampling a mere 5% of the graphs for execution. Amortizing the similarity cost over six operators, the process can produce up to $18\times$ speed-up. Our proposed similarity measures produce comparable or more accurate results to state-of-the-art similarity measures but run more than 5 orders of magnitude faster.

2 Methodology

In this section, we formulate the problem and describe the methodology along with different aspects of the proposed solution. We start off with some basic notation followed throughout the paper and a formal description of our method and its complexity.

Let a graph G be an ordered pair $G = (V, E)$ with V being the set of vertices and E the set of edges of G, respectively. The degree of a vertex $u \in V$, denoted by $d_G(u)$, is the number of edges of G incident to u. The degree distribution of a graph G, denoted by $P_G(k)$, expresses the probability that a randomly selected vertex of G has degree k. A dataset D is a set of N simple, undirected graphs $D = \{G_1, G_2, ..., G_N\}$. We define a graph operator to be a function $g\colon D \to \mathbb{R}$, mapping an element of D to a real number. In order to quantify the similarity between two graphs $G_a, G_b \in D$ we use a graph similarity function $s\colon D \times D \to \mathbb{R}$ with range within $[0, 1]$. For two graphs $G_a, G_b \in D$, a similarity of 1 implies that they are identical while a similarity of 0 the opposite.

Consequently, the problem we are addressing can be formally stated as follows: Given a dataset of graphs D and a graph operator g, without knowledge of the range of g given D, we wish to infer a function $\hat{g}\colon D \to \mathbb{R}$ that approximates g. Additionally, we wish our approximation to be both accurate (i.e., $|g - \hat{g}| < \epsilon$, for some small ϵ) and efficient (i.e., $O(\hat{g}) < O(g)$). In this formulation, our goal is to provide an accurate approximation of g, while avoiding its exhaustive execution over the entire D. To achieve this goal, we utilize the similarity matrix R,

[3] https://github.com/giagiannis/data-profiler

an $N \times N$ matrix with $R[i,j] = s(G_i, G_j)$, where s is a given similarity measure. As a result, R contains all-pairs similarity scores between the graphs of D. R is symmetric, its elements are in $[0,1]$ and the entries of its main diagonal equal 1.

Our method takes as input a dataset D and an operator g to model. It forms a pipeline that begins with the computation of the similarity matrix R based on s; calculates the actual values of g for a ratio $p \in (0,1)$ of randomly selected graphs of D and, finally, estimates g for the remaining graphs of D by running a weighted version of the k-Nearest-Neighbors (kNN) algorithm [19]. The inferred function $\hat{g}$ is then given by the following equation:

$$\hat{g}(G_x) = \frac{\sum_{i \in \Gamma_k(x)} w_{xi} g(G_i)}{\sum_{i \in \Gamma_k(x)} w_{xi}} \tag{1}$$

Where $w_{xi} = R[x,i]$ is the similarity score for graphs G_x, G_i, i.e., $w_{xi} = s(G_x, G_i)$, $\Gamma_k(x)$ is the set of the k most similar graphs to G_x for which we have already calculated g and $g(G_i)$ the value of the operator for G_i. Our approach is formally described in Algorithm 1. The complexity of Algorithm 1 can be broken down

Algorithm 1 Graph Operator Modeling

```
 1: procedure APPROXIMATE($[G_1, G_2, ..., G_N]$, $g$, $s$, $p$, $k$)
 2:     $R \leftarrow [\,], T \leftarrow \{\,\}, A \leftarrow \{\,\}$
 3:     for $(i,j) \leftarrow [1,N] \times [1,N]$ do
 4:         $R[i,j] \leftarrow s(G_i, G_j)$
 5:     for $i \leftarrow 1, p \cdot N$ do
 6:         $r \leftarrow randint(1,N)$
 7:         $T[G_r] \leftarrow g(G_r)$
 8:     for $x \leftarrow [G_1, G_2, ..., G_N], x \notin keys(T)$ do
 9:         $t \leftarrow findNeighbors(R,T,k,x)$
10:         $A[x] \leftarrow calcApproximation(R,t)$
11:     return $A$
```

to its three main components: 1) The calculation of the similarity matrix R in lines $3-4$, for a given similarity measure s with complexity S. 2) The component which computes the operator g for pN graphs (lines $5-7$), assuming that g has complexity M. And 3) the approximation of the operator for the remaining graphs (lines $8-10$) using kNN. Thus, the overall complexity of our method is:

$$O(N^2S + pNM + (N(1-p))((pN)log(pN) + k)) \tag{2}$$

From Eq. 2, we deduce that the complexity of our method is dominated by its first two components. Consequently, the lower the computational cost of s, the more efficient our approach will be. Additionally, we expect our training set to be much smaller than our original dataset (i.e., $p \ll 1$).

It is important to note here that the $O(N^2S)$ component corresponds to a calculation performed only *once*, whether modeling a single or multiple operators. Thus, given that the similarity matrix calculation happens once per dataset, its cost gets amortized over multiple graph operators, making the $O(pNM)$ factor the dominant one for our pipeline.

2.1 Similarity Measures

The similarity matrix is an essential tool in our effort to model graph operators under the hypothesis that similar graphs produce similar operator outputs. Relative to graph analytics operators, we propose a family of similarity measures based on graph degree distribution. Reinforced by the proven correlations between many diverse graph operators ([22, 7, 20]), we intend the proposed similarity measures to express graph similarity in a way that enables modeling of various operators.

▷ **Degree Distribution:** In order to quantify the similarity between two graphs we rely on comparing their degree distributions. We choose the Bhattacharyya coefficient BC [5] to perform this task. BC is considered a highly advantageous method for comparing distributions [1]. BC divides the output space of a function into m partitions and uses the cardinality of each partition to create an m-dimensional vector representing that space. As a measure of divergence between two distributions, the square of the angle between the two vectors is considered. In our case, the points of the output space are the degrees of the nodes of each graph. By dividing that space into partitions and considering the cardinalities of each partition, we effectively compare the degree distributions of those graphs. In our implementation, we use a k-d tree [4], a data structure used for space partitioning, to compute BC. We build a k-d tree once, based on a predefined percentage of vertex degrees from all the graphs in D. We then use the created space partitioning to compute degree distributions for each graph.

▷ **Degree Distribution + Levels:** As an extension of the degree distribution-based similarity measure, we consider a class of measures with increasing level of information. Intuitively, the degree of a vertex is a measure of its connectivity based on its immediate neighbors. Adding a level of indirection, we can consider the degree of a vertex at *level 1* as the degree of a super-node containing the vertex and all its immediate neighbors. Generalizing this idea to more than one levels gives us a measure of the indirect connectivity of a vertex. By combining the degrees of a vertex for multiple levels we obtain information up to *level* hops away. As an illustrative example, in Figure 1 vertex u_0 has degree 4, when considering its direct neighbors, 1 when its neighborhood is expanded to *level* 1, and for *level* 2 it becomes 3. As a result, we quantify the similarity between graphs by calculating the degrees up to a certain level and use BC to compare the resulting degree distributions. A good property of this class of measures is that they provide us with a nice trade-off between accuracy and computational cost. Increasing the number of degree distribution levels involves additional computations but also incorporates more graph topological insights to it. In order to calculate the degrees for a given level, for each vertex we perform a depth-limited Depth First Search up to *level* hops away in order to mark the internal edges of the super-node. We then count the edges of the border vertices (vertices *level* hops away from the source) that do not connect to any internal vertices.

▷ **Degree Distribution + Vertex Count** A second extension to our degree distribution-based similarity measure is based on the ability of our method to combine similarity matrices. Graph size (vertex count) is another graph attribute

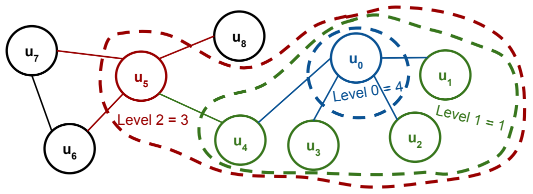


Fig. 1: Example of Degree Distribution + Levels

to measure similarity on. We formulate similarity in terms of vertex count as: $s(G_i, G_j) = \frac{min(|V_{G_i}|,|V_{G_j}|)}{max(|V_{G_i}|,|V_{G_j}|)}$. Intuitively, s approaches 1 when $|V_{G_i}| - |V_{G_j}|$ approaches 0, i.e., when G_i, G_j have similar vertex counts. To incorporate vertex count into the graph comparison, we can combine the similarity matrices computed with degree distributions and vertex counts using an arbitrary formula (e.g., linear composition).

2.2 Discussion

In this section, we consider a series of issues that relate to the configuration and performance of our method as well as to the relation between modeled operators, similarity measure and input datasets.

▷ **Graph Operators:** This work focuses on graph analytics operators, namely centralities, clustering metrics, network statistics, etc. Research on this area has resulted in a large collection of operators, also referred to as *topology metrics* (e.g., [12, 8, 7, 20]). Topology metrics can be loosely classified in three categories ([22, 7, 20]), those related to *distance*, *connectivity* and *spectrum*. In the first class, we find metrics like *diameter*, *average distance* or *betweenness centrality*. In the second, *average degree*, *degree distribution*, etc. Finally, the third class comes from the spectral analysis of a graph and contains the computation of *eigenvalues*, *eigenvectors* or other spectral-related metrics.

▷ **Combining Similarity Measures:** We can think of use cases where we want to quantify the similarity of graphs based on parameters unrelated to each other. For example, we might want to compare two graphs based on their degree distributions but also take under account their vertex count. This composition can be naturally implemented in our system by computing independent similarity matrices and "fuse" those matrices into one using a formula. This technique is presented in our evaluation and proves effective for a number of operators.

▷ **Regression Analysis:** Although there exist several approaches to statistical learning [19], we have opted for the kNN method. We choose kNN for its simplicity and because we do not have to calculate distances between points of our dataset (we already have that information from the similarity matrix). The kNN algorithm is also suitable for our use case since it is sensitive to localized data and insensitive to outliers. A desired property, since we expect similar graphs to have similar operator scores and should therefore be of influence in our estimations.

▷ **Scaling Similarity Computations:** Having to compute all-pairs similarity scores for a large collection of graphs can be prohibitively expensive. To this end, we introduce a preprocessing step which we argue that improves on the existing computational cost, reducing the number of similarity calculations performed. As, in order to approximate a graph operator, we employ kNN, we observe that, for each graph, we only require the similarity scores to its k most similar graphs for which we have the value of g, i.e., the weights in Equ. 1. Therefore we propose to run a clustering algorithm which will produce clusters of graphs with high similarity. Then for each cluster compute all-pairs similarity scores between its members, setting inter-cluster similarities to zero. By creating clusters of size much larger than k, we expect minimal loss in accuracy while avoiding a considerable number of similarity computations. As a clustering algorithm we use a simplified version of k-medoids in combination with k-means++, for the initial seed selection ([23, 2]). For an extensive experimental evaluation of this technique we refer the reader to the extended version of our work in [34] which we have not included here due to space constraints.

3 Experimental Evaluation

▹ **Datasets:** For our experimental evaluation, we consider both real and synthetic datasets. The real datasets comprise a set of ego graphs from Twitter (TW) which consists of 973 user "circles" as well as a dataset containing 733 snapshots of the graph that is formed by considering the Autonomous Systems (AS) that comprise the Internet as nodes and adding links between those systems that communicate to each other. Both datasets are taken from the Stanford Large Network Dataset Collection [26].

We also experiment with a dataset of synthetic graphs (referred to as the BA dataset) generated using the SNAP library [27]. We use the `GenPrefAttach` generator to create random scale-free graphs with power-law degree distributions using the Barabasi-Albert model [3]. We keep the vertex count of the graphs constant to 4K. We introduce randomness to this dataset by having the *initial outdegree* of each vertex be a uniformly random number in the range $[1, 32]$. The Barabasi-Albert model constructs a graph by adding one vertex at a time. The *initial outdegree* of a vertex is the maximum number of vertices it connects to, the moment it is added to the graph. The graphs of the dataset are simple and undirected. Further details about the datasets can be found in Table 1.

▹ **Similarity Measures:** We evaluate all the similarity measures proposed in Section 2.1, namely *degree distribution + levels*, for levels $0, 1, 2$ and *degree distribution + vertex count*. When combining vertex count with degree, we use the following simple formula: $R = w_1 R_d + w_2 R_n$, with R_d, R_n the degree distribution and vertex count similarity matrices respectively. In our evaluation, $w_1 = w_2 = 0.5$. To investigate their strengths and limitations, we compare them against two measures functioning as our baselines. The first is a sophisticated similarity measure not based on degree but rather on distance distributions (from which the degree distribution can be deduced). *D-measure* [32] is based on the concept of network node dispersion (NND) which is a measure of the heterogeneity of a graph in terms of connectivity distances. It is a state-of-the-art graph similarity measure with very good experimental results for both real and syn-

Table 1: Datasets overview

Name	**Size (N)**	$\overline{\lvert V\rvert}$	$\overline{\lvert E\rvert}$	**Range** $\lvert V\rvert$	**Range** $\lvert E\rvert$
TW	973	132	1,841	min: 6 max: 248	min: 9 max: 12,387
AS	733	4,183	8,540	min: 103 max: 6,474	min: 248 max: 13,895
BA	1,000	4,000	66,865	4,000	min: 3,999 max: 127,472

thetic graphs. Our second baseline comes from the extensively researched area of graph kernels. For the purposes of our evaluation, we opted for the geometric *Random Walk Kernel* (*rw-kernel*) [16] as a widely used representative of this class of similarity measures. In order to avoid the *halting* phenomenon due to the kernel's decay factor (λ^k) we set $\lambda = 0.1$ and the number of steps $k \leq 4$, values that are considered to be reasonable for the general case [33].

▷ **Graph Operators:** In our evaluation, we model operators from all the categories mentioned in Section 2.2. As representatives of the distance class, we choose betweenness (**bc**), edge betweenness (**ebc**) and closeness centralities (**cc**) ([29, 8]), three metrics that express how central a vertex or edge is in a graph. From the spectrum class, we choose spectral radius (**sr**) and eigenvector centrality (**ec**). The first is associated with the robustness of a network against the spreading of a virus [21], while the second also expresses vertex centrality [6]. Finally, as a connectivity related metric we consider PageRank (**pr**), a centrality measure used for ranking web pages based on popularity [9].

All measures, except spectral radius, are centrality measures expressed at vertex level (edge level in the case of edge betweenness). Since we wish all our measures to be expressed at graph level, we will be using a method attributed to Freeman [13] to make that generalization. This is a general approach that can be applied to any centrality [8], and measures the average difference in centrality between the most central point and all others. All the graph operators are implemented in R. We use the R package of the `igraph` library [10] which contains implementations of all the algorithms mentioned.

▷ **kNN:** The only parameter we will have to specify for kNN is k. After extensive experimentation (omitted due to space constraints), we have observed that small values of k tend to perform better. As a result, all our experiments are performed with $k = 3$.

▷ **Error Metrics:** The modeling accuracy of our method is quantified using two widely used measures from the literature, the *Median Absolute Percentage Error* and the *Normalized Root Mean Squared Error*.

▷ **Setup:** All experiments are conducted on an Openstack VM with 16 Intel Xeon E312 processors at 2GHz, 32GB main memory running Ubuntu Server 16.04.3 LTS with Linux kernel 4.4.0. We implemented our prototype in Go language (v.1.7.6).

3.1 Experiments

▷ **Modeling Accuracy:** To evaluate the accuracy of our approximations, we calculate *MdAPE* and *nRMSE* for a randomized 20% of our dataset. We vary

Table 2: Modeling Errors and Execution Speedup

		MdAPE (%)			**nRMSE**			**Speedup ×**			**A. Speedup ×**		
		5%	**10%**	**20%**	**5%**	**10%**	**20%**	**5%**	**10%**	**20%**	**5%**	**10%**	**20%**
AS	**sr**	1.3	1.1	0.9	0.05	0.03	0.02	6.4	3.8	3.3	18.0	9.5	4.9
	ec	0.1	0.1	0.0	0.01	0.00	0.00	5.7	4.5	3.1			
	bc	1.4	1.2	1.1	0.04	0.03	0.03	15.7	8.8	4.7			
	ebc	3.1	2.7	2.4	0.04	0.04	0.04	17.3	9.3	4.8			
	cc	0.4	0.4	0.3	0.01	0.01	0.01	14.0	8.2	4.5			
	pr	0.9	0.8	0.7	0.05	0.04	0.03	5.7	4.4	3.1			
TW	**sr**	16.3	15.3	14.7	0.10	0.10	0.10	13.3	8.0	4.4	14.8	8.5	4.6
	ec	8.0	7.7	7.7	0.14	0.14	0.13	13.1	7.9	4.4			
	bc	17.8	17.5	16.8	0.16	0.15	0.14	13.0	7.8	4.4			
	ebc	29.5	29.8	28.6	0.12	0.12	0.12	13.5	8.0	4.4			
	cc	3.3	3.0	2.9	0.10	0.10	0.09	13.0	7.9	4.4			
	pr	9.2	7.7	7.2	0.07	0.06	0.05	13.2	7.9	4.4			
BA	**sr**	3.3	1.8	0.9	0.04	0.03	0.03	5.6	4.4	3.0	16.3	9.0	4.7
	ec	0.4	0.3	0.3	0.01	0.01	0.01	3.7	3.1	2.4			
	bc	10.3	10.1	9.6	0.10	0.05	0.02	12.6	7.7	4.4			
	ebc	10.9	9.3	8.5	0.10	0.09	0.01	13.6	8.1	4.5			
	cc	2.4	2.2	2.1	0.04	0.04	0.03	9.9	6.6	4.0			
	pr	6.7	6.1	5.9	0.06	0.05	0.05	3.6	3.0	2.3			

the sampling ratio p, i.e., the number of graphs for which we actually execute the operator, divided by the total number of graphs in the dataset. The results are displayed in Table 2. Each row represents a combination of a dataset and a graph operator with the corresponding error values for different values of p between 5% and 20%.

The results in Table 2 showcase that our method is capable of modeling different classes of graph operators with very good accuracy. Although our approach employs a degree distribution-based similarity measure, we observe that the generated similarity matrix is expressive enough to allow the accurate modeling of distance- and spectrum-related metrics as well, achieving errors well below 10% for most cases. In AS graphs, the $MdAPE$ error is less than 3.2% for all the considered operators when only a mere 5% of the available graphs is examined. Operators such as closeness or eigenvector centralities display low $MdAPE$ errors in the range of $< 8\%$ for all datasets. Through the use of more expressive or combined similarity measures, our method can improve on these results, as we show later in this Section. We also note that the approximation accuracy increases with the sampling ratio. This is expressed by the decrease of both $MdAPE$ and $nRMSE$ when we increase the size of our training set. These results verify that modeling such graph operators is not only possible, but it can also produce highly accurate models with marginal errors.

Specifically, in the case of the AS dataset, we observe that all the operators are modeled more accurately than in any other real or synthetic dataset. This can be attributed to the topology of the AS graphs. These graphs display a linear relationship between vertex and edge counts. Their clustering coefficient displays very little variance, suggesting that as the graphs grow in size they keep the same topological structure. This gradual, uniform evolution of the AS

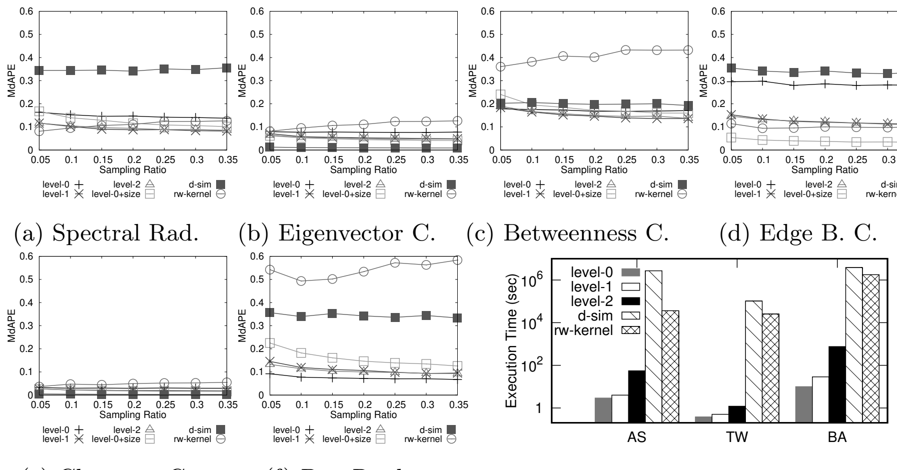
graphs leads to easier modeling of the values of a given graph topology measure.

On the other hand, our approach has better accuracy for degree- than distance-related metrics in the cases of the TW and BA datasets. The similarity measure we use is based on the degree distribution that is only indirectly related to vertex distances. This can be seen, for example, in the case of BA if we compare the modeling error for the betweenness centrality (bc) and PageRank (pr) measures. Overall, we see that eigenvector and closeness centralities are the two most accurately approximated metrics across all datasets. Next up, we find PageRank, spectral radius, betweenness and edge betweenness centralities. Willing to further examine the connection between modeling accuracy and similarity measures, we have included *D-measure* and *rw-kernel* in our evaluation as well as the degree-level similarity measures and the similarity matrice combination technique.

▷ **Execution Speedup:** Next, we evaluate the gains our method can provide in execution time. The similarity matrix computation is a time-consuming step, yet an advantage of our scheme is that the matrix can be reused for different graph operators and thus its cost can be amortized. In order to provide a better insight, we calculate two types of speedups: One that considers the similarity matrix construction from scratch for each operator separately (provided in the *Speedup* column of Table 2) and one that expresses the average speedup for all six measures for each dataset, where the similarity matrix has been constructed once (provided in the *A. Speedup* column of Table 2).

The observed results highlight that our method is not only capable of providing models of high quality, but also does so in a time-efficient manner. A closer examination of the Speedup columns shows that our method is particularly efficient for complex metrics that require more computation time (as in the *ebc* and *cc* cases for all datasets). The upper bound of the theoretically anticipated speedup equals $\frac{1}{p}$, p being the sampling ratio. Interestingly, the *Amortized Speedup* column indicates that when the procedure of constructing the similarity matrix is amortized to the six operators under consideration, the achieved speedup is very close to the theoretical one. This is indeed the case for the AS and BA datasets that comprise the largest graphs, in terms of number of vertices: For all p values, the amortized speedup closely approximates $\frac{1}{p}$. In the case of the TW dataset which consists of much smaller graphs and, hence, the time dedicated to the similarity matrix estimation is relatively larger than the previous cases, we observe that the achieved speedup is also sizable. In any case, the capability of reusing the similarity matrix, which is calculated on a per-dataset rather than on a per-operator basis, enables our approach to scale and be more efficient as the number and complexity of graph operators increases.

▷ **Comparing Similarity Measures:** The results of the similarity measure comparisons, in the case of the TW dataset, are displayed in Figure 2, where $MdAPE$ is used to express the modeling error. We compare six similarity measures: The *degree distribution + levels* measure (for levels equal from 0 to 2), a combination of *level-0* degree distribution with vertex count (denoted by *level-0 + size*), *D-measure* and the *Random Walk Kernel* based similarity measure (de-



(a) Spectral Rad. (b) Eigenvector C. (c) Betweenness C. (d) Edge B. C.

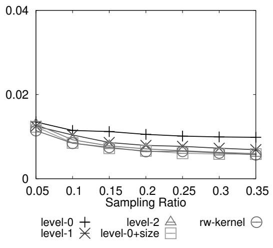
(e) Closeness C. (f) PageRank (g) Execution Time (sec)

Fig. 2: Similarity Metrics Comparison for *TW* Dataset

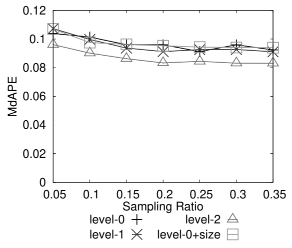
noted by rw-kernel). The results indicate the impact that the choice of similarity measure has on modeling accuracy. A more suitable to the modeled operator and detailed similarity measure is more sensitive to topology differences and can lead to better operator modeling.

In all Figures, with the exception of PageRank, we observe that the *degree distribution + levels* similarity measure, for a number of levels, can model an operator more accurately than the simple degree distribution-based, effectively reducing the errors reported in Table 2. Indeed, the addition of more levels to the degree distribution incorporates more information about the connectivity of each vertex. This additional topological insights contribute positively to better estimate the similarity of two graphs. Examining the modeling quality, we observe that it increases but only up to a certain point, in relation to the topology of the graphs in the dataset. For example, since *TW* comprises of ego graphs, all the degrees of level > 2 are zero, since there exist no vertices with distance greater than 2; therefore, employing more levels does not contribute any additional information about the topology of the graphs when computing their similarity. Finally, we observe that, in specific cases, such as PageRank (Figure 2f), enhancing the degree distribution with degrees of more levels introduces information that is interpreted as noise during modeling. PageRank is better modeled with the simple degree distribution as a similarity measure. As such, we argue that for a given dataset and graph operator, experimentation is required to find the number of levels that give the best trade-off between accuracy and execution time.

We next concentrate on the effect of the combination of degree distribution with vertex count in the modeling accuracy. We note that the vertex count contributes positively in the modeling of distance-related metrics while having a neutral or negative impact on degree- and spectrum-related metrics. This is attributed to the existence of, at least, a mild correlation, between vertex count



(a) *AS* dataset



(b) *BA* dataset

Fig. 3: Similarity Metric Comparison for Betweenness C.

and *bc*, *ebc* and *cc* [22]. For our least accurately approximated task, edge betweenness centrality, employing the combination of measures results in a more than 6× decrease in error.

For *D-measure*, our experiments show that, for distance-related metrics it performs at least as good as the *degree distribution + levels* similarity measures for a given level, with the notable exception of the PageRank case. On the other hand, the degree distribution can be sufficiently accurate for degree- or spectrum-related metrics. As *D-measure* is based on distance distributions between vertices, having good accuracy for distance-related measures is something to be expected. A good example of the effectiveness of *D-measure* is shown in the case of closeness centrality that involves all-pairs node distance information directly incorporated in *D-measure* as we have seen in Section 3. In Figure 2e we observe that by adding levels we get better results, vertex count contributes into even better modeling but *D-measure* gives better approximations. Yet, our methods' errors are already very small (less than 3%) in this case. Considering the *rw-kernel* similarity measure, we observe that it performs poorly for most of the operators. Although its modeling accuracy is comparable to *degree distribution + levels* for some operators, we find that for a certain level or in combination with vertex count a degree distribution-based measure has better accuracy. Notably, *rw-kernel* has low accuracy for degree and distance related operators while performing comparably in the case of spectrum operators.

Identifying betweenness centrality as one of the hardest operators to model accurately, we present $MdAPE$ approximation errors for AS and BA in Figures 3a, 3b. These Figures do not include *D-measure*, since it was not possible to compute it because of its running time. We note that the approximation error is below 12% and that the *degree distribution + levels* measures further improve on it for both datasets. Compared to TW (Fig. 2), we observe that the *level-2* similarity measure provides better results for AS and BA but not TW, attributed to the ego graph structure with *level-2* degrees being zero. Finally, it is expected that *level-0 + size* for BA to be no different than plain *level-0*, since all the graphs in BA have the same vertex count by construction.

The aforementioned similarity measures have striking differences in their execution time. A comparison in computation time for different levels of the *degree distribution + levels* similarity measure is presented in Figure 2g. In the case of *D-measure*, the actual execution time is presented for the TW dataset, since it was prohibitively slow to compute it for the other two datasets. For the remaining two datasets, we have computed *D-measure* on a random number of pairs of graphs and then projected the mean computation time to the number of comparisons performed by our method for each dataset.

Our results show that the overhead from *level-0* to *level-1* is comparable for all the datasets. However, that is not the case for *level-2*. The higher the level, the more influential the degree of the vertices becomes in the execution time. Specifically, while we find *level-0* to be 3.2× faster than *level-2* for TW, we observe that in the case of AS and BA it is 19× and 76× faster. The computation of the *D-measure* and the *rw-kernel*, on the other hand, are orders of magnitude slower. Given the difference in modeling quality between the presented similarity functions, we observe a clear trade-off between quality of results and execution time in the context of our method.

4 Related Work

Our work relates to the actively researched areas of graph similarity, graph analytics and machine learning. The available techniques for quantifying graph similarity can be classified into three main categories ([24, 36]):
▷ **Graph Isomorphism - Edit Distance:** Two graphs are considered similar if they are isomorphic. A generalization of the graph isomorphism problem is expressed through the *Edit Distance*, i.e., the number of operations that have to be performed in order to transform one graph to the other [31]. The drawback of approaches in this category is that graph isomorphism is hard to compute.
▷ **Iterative Methods:** This category of graph similarity algorithms is based on the idea that two vertices are similar if their neighborhoods are similar. Applying this idea iteratively over the entire graph can produce a global similarity score. Such algorithms compare graphs based on their topology, we choose to map graphs to feature vectors and compare those vectors instead.
▷ **Feature Vectors:** These approaches are based on the idea that similar graphs share common properties such as degree distribution, diameter, etc and therefore represent graphs as feature vectors. To assess the degree of similarity between graphs, statistical tools are used to compare their feature vectors instead. Such methods are not computationally demanding. Drawing from this category of measures, we base our graph similarity computations on comparing degree distributions.
▷ **Graph Kernels:** A different approach to graph similarity comes from the area of machine learning where kernel functions can be used to infer knowledge about samples. Graph kernels are kernel functions constructed on graphs or graph nodes for comparing graphs or nodes respectively. Extensive research on this area (e.g., [17, 15]) has resulted in many kernels based on walks, paths, etc. While computationally more expensive they provide a good baseline for our modeling accuracy evaluation.
▷ **Graph Analytics and Machine Learning** Although graph analytics is a very thoroughly researched area, there exist few cases where machine learning techniques are used. On the subject of graph summarization, a new approach

is based on *node representations* that are learned automatically from the neighborhood of a vertex [18]. *Node representations* are also applicable in computing node or graph similarities as seen in [18]. However, we do not find works employing machine learning techniques in the field of graph mining through graph topology metric computations.

5 Conclusion

In this work we present an operator-agnostic modeling methodology which leverages similarity between graphs. This knowledge is used by a kNN classifier to model a given operator allowing scientists to predict operator output for any graph without having to actually execute the operator. We propose an intuitive, yet powerful class of similarity measures that efficiently capture graph relations. Our thorough evaluation indicates that modeling a variety of graph operators is not only possible, but it can also provide results of high quality at considerable speedups. Finally, our approach appears to present similar results to state-of-the-art similarity measures, such as *D-measure*, in terms of quality, but requires orders of magnitude less execution time.

References

1. Aherne, F.J., et al.: The bhattacharyya metric as an absolute similarity measure for frequency coded data. Kybernetika **34**(4), 363–368 (1998)
2. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007. pp. 1027–1035 (2007)
3. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
4. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM **18**(9), 509–517 (1975)
5. Bhattacharyya, A.: On a measure of divergence between two statistical populations defined by their probability distributions. Bulletin of Calcutta Mathematical Society **35**(1), 99–109 (1943)
6. Bonacich, P.: Power and centrality: A family of measures. American Journal of Sociology **92**(5), 1170–1182 (1987)
7. Bounova, G., de Weck, O.: Overview of metrics and their correlation patterns for multiple-metric topology analysis on heterogeneous graph ensembles. Phys. Rev. E **85**, 016117 (2012)
8. Brandes, U., Erlebach, T.: Network Analysis: Methodological Foundations (Lecture Notes in Computer Science). Springer-Verlag New York, Inc. (2005)
9. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks **30**(1-7), 107–117 (1998)
10. Csardi, G., Nepusz, T.: The igraph software package for complex network research. InterJournal **Complex Systems**, 1695 (2006)
11. Eppstein, D., Wang, J.: Fast approximation of centrality. J. Graph Algorithms Appl. **8**, 39–45 (2004)
12. da F. Costa, L., et al.: Characterization of complex networks: A survey of measurements. Advances in Physics **56**(1), 167–242 (2007)

13. Freeman, L.C.: A set of measures of centrality based on betweenness. Sociometry **40**(1), 35–41 (1977)
14. Gandomi, et al.: Beyond the hype. Int. J. Inf. Manag. **35**(2), 137–144 (2015)
15. Gärtner, T.: A survey of kernels for structured data. SIGKDD **5**(1), 49–58 (2003)
16. Gärtner, T., et al.: On graph kernels: Hardness results and efficient alternatives. In: 16th Annual Conference on Computational Learning Theory. pp. 129–143 (2003)
17. Ghosh, S., Das, N., Gonçalves, T., Quaresma, P., Kundu, M.: The journey of graph kernels through two decades. Computer Science Review **27**, 88–111 (2018)
18. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: SIGKDD. pp. 855–864. ACM (2016)
19. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. Springer New York (2009)
20. Hernández, J.M., Mieghem, P.V.: Classification of graph metrics. pp. 1–20 (2011)
21. Jamakovic, A., et al.: Robustness of networks against viruses: the role of the spectral radius. In: Symposium on Communications and Vehicular Technology. pp. 35–38 (2006)
22. Jamakovic, A., Uhlig, S.: On the relationships between topological measures in real-world networks. NHM **3**(2), 345–359 (2008)
23. Kaufmann, L., Rousseeuw, P.: Clustering by means of medoids pp. 405–416 (01 1987)
24. Koutra, D., Parikh, A., Ramdas, A., Xiang, J.: Algorithms for graph similarity and subgraph matching. Technical Report Carnegie-Mellon-University (2011), https://people.eecs.berkeley.edu/ aramdas/reports/DBreport.pdf
25. Leskovec, J., Adamic, L.A., Huberman, B.A.: The dynamics of viral marketing. TWEB **1**(1), 5 (2007)
26. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data (Jun 2014)
27. Leskovec, J., Sosič, R.: Snap: A general-purpose network analysis and graph-mining library. ACM Transactions on Intelligent Systems and Technology **8**(1), 1 (2016)
28. McAuley, J.J., Leskovec, J.: Learning to discover social circles in ego networks. In: Bartlett, P.L., Pereira, F.C.N., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States. pp. 548–556 (2012)
29. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E **69**(2), 026113 (2004)
30. Riondato, M., Kornaropoulos, E.M.: Fast approximation of betweenness centrality through sampling. Data Min. Knowl. Discov. **30**(2), 438–475 (2016)
31. Sanfeliu, A., Fu, K.: A distance measure between attributed relational graphs for pattern recognition. IEEE Trans. Systems, Man, and Cybernetics **13**(3) (1983)
32. Schieber, T.A., et al.: Quantification of network structural dissimilarities. Nature communications **8**, 13928 (2017)
33. Sugiyama, M., Borgwardt, K.M.: Halting in random walk kernels. In: Annual Conference on Neural Information Processing Systems. pp. 1639–1647 (2015)
34. Tasos Bakogiannis, Ioannis Giannakopoulos, D.T., Koziris, N.: Graph operator modeling over large graph datasets. CoRR **abs/1802.05536** (2018), http://arxiv.org/abs/1802.05536
35. Vishwanathan, S.V.N., Schraudolph, N.N., Kondor, R., Borgwardt, K.M.: Graph kernels. Journal of Machine Learning Research **11**, 1201–1242 (2010)
36. Zager, L.A., Verghese, G.C.: Graph similarity scoring and matching. Appl. Math. Lett. **21**(1), 86–94 (2008)