VerSaChI: Finding Statistically Significant Subgraph Matches using Chebyshev's Inequality

Shubhangi Agarwal
Indian Institute of Technology Kanpur
India
sagarwal@cse.iitk.ac.in

Sourav Dutta
Huawei Research Centre
Dublin, Ireland
sourav.dutta2@huawei.com

Arnab Bhattacharya
Indian Institute of Technology Kanpur
India
arnabb@cse.iitk.ac.in

ABSTRACT

Approximate subgraph matching, an important primitive for many applications like question answering, community detection, and motif discovery, often involves large labeled graphs such as knowledge graphs, social networks, and protein sequences. Effective methods for extracting matching subgraphs, in terms of label and structural similarities to a query, should depict accuracy, computational efficiency, and robustness to noise. In this paper, we propose VerSaChI for finding the top-k most similar subgraphs based on 2-hop label and structural overlap similarity with the query. The similarity is characterized using Chebyshev's inequality to compute the chi-square statistical significance for measuring the degree of matching of the subgraphs. Experiments on real-life graph datasets showcase significant improvements in terms of accuracy compared to state-of-the-art methods, as well as robustness to noise.

CCS CONCEPTS

• **Information systems** → **Information extraction**; *Information systems applications*; *Data mining*.

KEYWORDS

Subgraph Similarity; Approximate Matching; Statistical Significance; Chi-Square; Labeled Graph; Chebyshev's Inequality

ACM Reference Format:
Shubhangi Agarwal, Sourav Dutta, and Arnab Bhattacharya. 2021. VerSaChI: Finding Statistically Significant Subgraph Matches using Chebyshev's Inequality. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3459637.3482217

1 INTRODUCTION

With the growth of Open Linked Data in the form of knowledge graphs, social networks, bioinformatics structures, and road networks, efficient graph mining poses a challenging problem [3, 48]. Such large data sources are represented as labeled graphs, where entities are modeled as vertices, while their relationships are captured by edges, with labels defining the attributes of entities and relations. *Subgraph querying* is used across several domains including frequent pattern search [33], community detection in IR [27], question answering in NLP [40], object recognition in computer vision [7], and route planning [13]. The problem of subgraph match querying entails the extraction of subgraphs from an underlying graph having similar structure and labels to a given query [32, 36].

Traditional approaches for exact structural and label matching based on isomorphism are computationally infeasible. Thus, approaches based on pruning [29, 42, 53], indexing [45, 47], filtering [9, 23], and dynamic programming [25] have been proposed. However, they fail to scale for modern web-scale graph applications, wherein *approximate subgraph matching* was explored [32, 34] to extract similar subgraphs, with exact matches or with slight variations in structural elements and label mismatches. For example, in bioinformatics, approximate subgraph matching enables the detection of candidate regions in genome, that might have undergone abnormal mutations, for studying the associated medical effects [46, 54]. Although approximate subgraph extraction have been well studied [4, 21, 32, 34, 47, 50], efficiently finding matching subgraphs with improved runtime and accuracy remains an important problem.

Problem Statement. Consider $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G}, \mathcal{L}_\mathcal{G})$ to be an input graph, where $\mathcal{V}_\mathcal{G}$ and $\mathcal{E}_\mathcal{G}$ denote the vertex and edge sets respectively, while $\mathcal{L}_\mathcal{G} : \mathcal{V}_\mathcal{G} \rightarrow \Gamma$ maps the vertices in $\mathcal{G}$ to a finite label (or attribute) set Γ. A similar query graph is considered: $Q = (\mathcal{V}_Q, \mathcal{E}_Q, \mathcal{L}_Q)$. Without loss of generality, we assume that the graph G is deterministic, undirected, vertex labeled, and does not contain hyper-edges. The problem of *approximate subgraph matching* aims to find the top-k subgraphs of $\mathcal{G}$ that are *best* matching (maximum similarity) to Q in terms of vertex label and edge overlap. In our context, VerSaChI finds the *top-k statistically significant subgraphs* of $\mathcal{G}$ as the best approximate matches of Q.

State-of-the-art. (Sub-)graph matching has been extensively studied, and the existing body of work can be broadly categorized into two groups – *exact* methods and *approximate* heuristics. Since graph isomorphism is quasi-polynomial [5] and subgraph isomorphism is NP-complete [15], earlier works on exact graph matching such as Swift-Index [45], VF2 [16, 29], PathBlast [31], SAGA [50], IsoRank [47] and GraphGrep [23] to name a few, explored pruning and indexing techniques. To tackle incomplete and noisy data, approximate matching techniques tolerate small amounts of structural and label mismatches. These methods usually rely on identifying candidate vertices, whose neighborhoods are then progressively expanded in a greedy manner – providing compute efficiency, although with possibly sub-optimal results. Initial approaches like TALE [51], C-Tree [62], GString [28] and SAPPER [61] were based on indexing techniques and graph distance measures to compute

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482217

the degree of similarity. The use of pattern matching, semantic-based search, and graph decomposition for finding matching subgraphs were explored in gIndex [57], FG-Index [12], iGraph [26], Grafil [58], GPTree [60], cIndex [8], iGQ [55], and SIM-T [34]. Surveys on the rich literature of graph matching can be found in [14, 22, 37, 56]. Recent techniques like NeMa [32] and GFinder [36] adopt a combination of efficient indexing and graph traversal based cost measure to efficiently identify candidate matching regions, while VELSET [21] and NAGA [21] use statistical measure to mine subgraphs that demonstrate significant deviations from the background distribution when matched to a query. Graph pattern matching using *graph simulation* has been proposed [10, 11, 38]. Approximate graph matching in probabilistic graphs have also been studied [1, 24, 30, 35, 39, 48, 59].

Contributions. In this paper, we propose the *Vertex Neighborhood Aggregation for Statistically Significant Subgraphs via Chebyshev Inequality*—**VERSACHI**—algorithm for efficient top-k subgraph matching. We show how deviations from the underlying graph characteristics, modeled using probabilistic bounds, can efficiently provide label and structural similarity measures for approximate subgraph matching. Experimental results on real and synthetic datasets showcase how our proposed algorithm outperforms existing techniques in accuracy and is robust to noise.[1]

2 VERSACHI ALGORITHM

We now describe the VERSACHI algorithm for extracting the top-k best approximate matching subgraphs from a target graph $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G}, \mathcal{L}_\mathcal{G})$ with respect to a query graph $\mathcal{Q} = (\mathcal{V}_\mathcal{Q}, \mathcal{E}_\mathcal{Q}, \mathcal{L}_\mathcal{Q})$.

Background. VERSACHI uses *statistical significance* to quantify similarity between query and target vertex pairs. We use *Pearson's chi-square statistic* (χ^2) (an estimate of the p-value [41]) to measure the probability of attributing an observed event to chance or randomness. Mathematically, $\chi^2 = \sum_{\forall i} \left[(O_i - E_i)^2 / E_i\right]$, where O_i and E_i are the observed and expected number of occurrences respectively for all outcomes. The *Chebyshev's inequality* [49] models the probability of deviation for a random variable in terms of number of standard deviations from the mean. For a random variable X with finite mean μ and variance δ^2, $\Pr(|X - \mu| \geq t \cdot \delta) \leq 1/t^2$ for any $t > 0$ ($t \in \mathbb{R}$). Intuitively, the degree of matching (i.e., similarity) between a query and its matching subgraph would demonstrate significant deviations (due to high similarity) from the expected characteristics (considering a random subgraph). The Chebyshev's inequality is used to characterize such *deviations* for computing statistical significance of candidate matching subgraphs. Such techniques were studied for sequence mining [18, 19, 44], substring matching [17], subgraph similarity [1, 21], and clique finding [20].

Overview. The working of VERSACHI comprises the following stages. The first 5 steps are *offline* and computes the different characteristics for each vertex – done only once for the target graph. The next 4 steps, the *online* phase upon query arrival, computes characteristics for the query vertices and identifies candidate neighborhood regions matching the query by using *two-hop* label and structural overlap. The deviation of the observed similarity, from the underlying distribution is then characterized by Chebyshev's

[1]The source code is available from https://github.com/shubhangiat/VerSaChI.

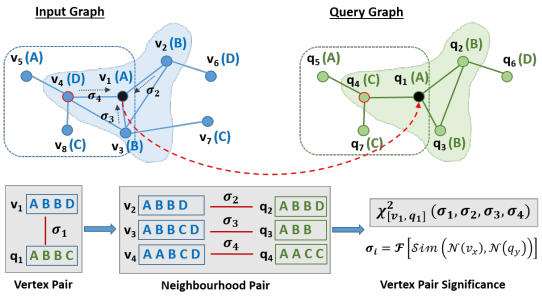


Figure 1: Two-hop neighborhood similarity based computation of χ^2 statistical significance for vertex match in VERSACHI.

inequality and represented as symbols. Based on statistical significance, matching candidate regions are identified and explored in a greedy manner, to obtain the best matching subgraphs. VERSACHI is closely related to and inspired from the works of [20] and [21].

1. Index Creation. Given a target graph $\mathcal{G}$, VERSACHI initially constructs two indexing lists summarizing the labels of the vertices and their neighbors. The first is an *inverted list*, $IL_\mathcal{G}$, that maps vertex labels to the corresponding list of vertices having the label. The second index, the *label neighbor list*, $LNL_\mathcal{G}$, stores the label information of the neighbors for each vertex in $\mathcal{G}$. A *label count vector* index, $LCV_\mathcal{G}(u)$, for each vertex u in $\mathcal{G}$ is also constructed. It stores the count of occurrence of each label (for $|\Gamma|$ labels) in the neighborhood of u. This enables efficient computation of similarity between vertices as described next (step 4 onwards).

2. Similarity Measure. For a vertex pair (u, v) we use a modified *Tversky index* [52] to define the *vertex similarity score*: $\eta_{u,v} = |\mathcal{N}(u) \cap \mathcal{N}(v)| / \left(|\mathcal{N}(u) \cap \mathcal{N}(v)| + |\mathcal{N}(v) \setminus \mathcal{N}(u)|^\gamma\right)$ where $\mathcal{N}(u)$ is the set of labels in the neighborhood of u including the label of u itself. Observe, by setting $\gamma = 1$, we obtain the original Tversky index with $\alpha = 0$ and $\beta = 1$. Intuitively, the similarity of $u \in \mathcal{G}$ is maximized w.r.t. $v \in \mathcal{Q}$ when all neighbor labels of v are present in the neighbors of u (i.e., $\mathcal{N}(v) \subseteq \mathcal{N}(u)$). Since the presence of additional neighbors of $u \in \mathcal{G}$ should not affect the similarity, we set $\alpha = 0$ in the Tversky index. In essence, the equation captures the *neighborhood recall* of $v \in \mathcal{Q}$ provided by $u \in \mathcal{G}$ (thus, $\beta = 1$). The exponential *penalty factor*, γ, penalizes increasing mismatches in the neighborhood label overlap between vertex pairs. It captures fine differences in the neighborhoods by accentuating even smaller mismatches. Empirically, $\gamma = 3$ gave the best results.

3. Initialization. Using $LCV_\mathcal{G}(u)$ and similarity measure $\eta_{u,v}$, VERSACHI computes the vertex similarity scores for every vertex pair in $\mathcal{G}$. This captures the underlying distribution. The expected similarity distribution across random neighborhoods of $\mathcal{G}$ is captured via 3 characteristics computed using $\eta_{u,w} : \forall \langle u \in \mathcal{G}, w \in \mathcal{G} \rangle$

(1) $\psi(\mathcal{G}) = \sum_{u,w \in \mathcal{G}} \eta_{u,w} / |\mathcal{V}_\mathcal{G}|$: *average* vertex similarity score for all vertex pairs in $\mathcal{G}$,

(2) $\delta(\mathcal{G}) = \left(\sum_{u,w \in \mathcal{G}} (\eta_{u,w} - \psi(\mathcal{G}))^2 / (|\mathcal{V}_\mathcal{G}| - 1)\right)^{1/2}$: *standard deviation* of the vertex similarity scores of $\mathcal{G}$, and

(3) $\triangle(\mathcal{G}) = \max_{u,w \in \mathcal{G}} \{(|\eta_{u,w} - \psi(\mathcal{G})| / \delta(\mathcal{G})\}$: *maximum* deviation of vertex similarity score from the average among all the vertex pairs in terms of standard deviations in $\mathcal{G}$.

4. Symbol Categorization. The degree of matching between a target graph vertex and a query vertex is captured by the amount of deviation of the vertex pair similarity score (in terms of the number

of standard deviations) from the underlying expected distribution (computed above). The standard deviations are discretized using the *step size* parameter, κ. It also determines the total number of possible symbols, $\tau = \lceil (\triangle(\mathcal{G}) - 1)/\kappa \rceil$. The set of category symbols, therefore, is $\Sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_\tau\}$. Smaller values of κ is preferred for differentiating between finer-grained structural mismatches.

For a pair of vertices u, w, its similarity is characterized using the symbol σ_i, $1 \leq i \leq \tau$. The first symbol, σ_1, spans the range of standard deviations up to $1+\kappa$, i.e., $\sigma_1 : 0 \leq |\eta_{u,w} - \psi(\mathcal{G})|/\delta(\mathcal{G}) < 1+\kappa$. Subsequent symbols cover step size standard deviations each, $\sigma_i : 1 + (i-1)\cdot\kappa \leq |\eta_{u,w} - \psi(\mathcal{G})|/\delta(\mathcal{G}) < 1 + i\cdot\kappa$ for $2 \leq i \leq \tau$.

5. Expected Probabilities of Symbols. The expected probability of occurrence of the category symbols is next computed using the *Chebyshev's inequality*. Observe, the deviation of vertex pair similarity from the mean can be in negative or positive direction.

Since we are interested in vertices that have higher similarity than the mean (to capture higher degree of matching), we only discretize the similarity (into symbols) when it is greater than the mean. For all similarities that are lesser than the mean, we fold them into symbol σ_1. Thus, assuming symmetric one-sided Chebyshev's inequality, the occurrence probability of symbol σ_i is $\Pr(\sigma_i) = \frac{1}{2}\left[\frac{1}{(1+(i-1)\cdot\kappa)^2} - \frac{1}{(1+i\cdot\kappa)^2}\right]$ for $2 \leq i \leq \tau$, and $\Pr(\sigma_1) = 1 - \sum_{j=2}^{\tau} \Pr(\sigma_j)$.

We also empirically evaluated the variant where the deviation where both the positive and negative side of the mean are considered (i.e., without folding). However, it produced no changes in our results. Since a very low similarity (large negative deviation) can potentially have large chi-square values and, thus, produce false matching results, VerSaChI uses the one-sided version.

The above steps are *offline*, and performed only once for a target graph. Once a query arrives, the following *online* steps take place.

6. Candidate Pair Mapping. Upon arrival of a query graph Q, the *online* processing starts with the construction of indexes IL_Q, LNL_Q, and LCV_Q, analogously to $\mathcal{G}$. For each *label* in Q, VerSaChI creates *candidate pairs* between the vertices of $\mathcal{G}$ and Q having the same label. These candidate pairs form the initial seed vertex for extracting matching subgraphs (to the query) via greedy neighborhood search. Formally, the candidate pairs generated are $CP = \{\langle v \in \mathcal{G}, q \in Q\rangle \mid \mathcal{L}_{\mathcal{G}}(v) = \mathcal{L}_Q(q)\}$.

7. Vertex Symbol Sequence. For a candidate pair $\langle v \in \mathcal{G}, q \in Q\rangle$, VerSaChI computes the vertex pair similarity score, $\eta(v, q)$, and characterizes the similarity score by assigning a category symbol based on the deviation from the expected similarity distribution (as discussed previously). The category symbol $\sigma_{\langle v,q\rangle}$ captures the one-hop neighborhood similarity for vertices v and q.

We next compute "second-order" candidate pairs between the vertex sets adjacent to v and q. A *greedy* best mapping based on the vertex pair similarity score is used to compute the second-order candidate pairs. Similar to $\langle v, q\rangle$, each second-order candidate pair is assigned a category symbol based on the deviation of its similarity score from the expected. The initial category symbol $\sigma_{\langle v,q\rangle}$ is aggregated with the second-order category symbols to form the *vertex symbol sequence*, $O_{\langle v,q\rangle}$, for the candidate pair $\langle v, q\rangle$.

As an example, consider Fig. 1 depicting an initial candidate pair between vertices v_1 and q_1 (both having label A) with category symbol σ_1 assigned to it. The adjacent vertices of v_1 ($\{v_2, v_3, v_4\}$) and the neighbors of q_1 ($\{q_2, q_3, q_4\}$) are then greedily best-matched based on vertex pair similarity to obtain the "second-order" candidate pairs. For instance, v_2 and q_2 provides the best match with the same label and the same neighborhood labels and, thus, forms the next candidate pair (with, say, category symbol σ_2). Subsequently, v_3 and q_3 are matched having the same label and partial neighborhood overlap (consider to be assigned symbol σ_3). Finally, the candidate pair $\langle v_4, q_4\rangle$ is obtained with category symbol σ_4. The corresponding vertex symbol sequence, $O_{\langle v_1,q_1\rangle} = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$, associated to $\langle v_1, q_1\rangle$, captures the two-hop similarity between the candidate pair vertices v_1 and q_1 (Fig. 1).

8. Statistical Significance. The computed symbol sequence $O_{\langle v,q\rangle}$ signifies the degree of matching between the two-hop neighborhoods of v and q. Assuming d to be the *degree* of q, since mapping is performed for the neighbors of q, the length of $O_{\langle v,q\rangle}$ is d. Thus, the *expected occurrence counts* of category symbol σ_i is $E[\sigma_i] = d\cdot\Pr(\sigma_i)$. The *observed occurrence counts* of the category symbols are directly obtained from $O_{\langle v,q\rangle}$. Using the observed and expected counts, VerSaChI computes the *chi-square statistics*, $\chi^2_{\langle v,q\rangle}$, for all the candidate pairs obtained in CP (see step 6).

9. Approximate Matching. The candidate pairs along with their computed chi-square values, $\langle v, q, \chi^2_{\langle v,q\rangle}\rangle$, are inserted into a *primary max-heap* structure. The candidate pair with the largest χ^2 value is extracted (assume $\langle v, q\rangle$) for initializing the top-1 matching subgraph, $Match^{(1)}$, and is considered as the seed vertex pair for greedy expansion to find matching subgraph for the query Q.

Next, candidate pairs between adjacent vertices of the extracted seed pair (v and q) are constructed (as in step 6) and pushed into a *secondary max-heap* structure. As before, vertex symbol sequences of the candidate pairs in the secondary heap are constructed, their statistical significance scores are computed, the pair with the highest χ^2 value is extracted, and added to $Match^{(1)}$. This process is iterated till the secondary heap is empty, or the size of $Match^{(1)}$ equals the number of vertices in Q. The subgraph obtained in $Match^{(1)}$ is reported as the *top-1 best approximate matching subgraph* for Q.

Vertices extracted from the primary and secondary heaps are marked as "done" to prevent duplicate entries in the heap structures, and ensuring that the same region is not repeatedly explored. To retrieve more *top-k* approximate matches for a query, the secondary heap is reset and the process is re-run, starting from picking the currently best candidate pair (with highest statistical significance) from the primary heap. This is repeated until k matches are obtained.

Complexity Analysis. For a target graph having n vertices and $|\Gamma|$ symbols, the offline time for index construction, etc. is $O(n^2)$, while the online querying time for a query having n_Q vertices is $O(n_Q \cdot n/|\Gamma|)$. (Full details are in [2].)

3 EXPERIMENTS

We now discuss the evaluation of VerSaChI with other approaches.

Datasets. We evaluate the algorithms on real datasets from 3 different domains: (i) *Biological Networks:* protein-protein graphs of `Human`, `HPRD` [6] and `Protein` [43]; (ii) *Social Interaction:* user interaction network on the hosting site Flickr [43]; and (iii) *Knowledge Graph:* `IMDb` [43] movie relationship graph. The dataset characteristics are shown in Table 1(a). Using synthetically generated *Barabási-Albert* graphs we study the scalability of VerSaChI.

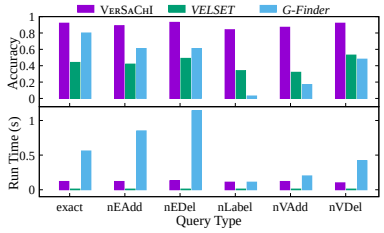
Dataset	# Vertices	# Edges	# Unique Labels
Human	4,674	86,282	44
HPRD	9,460	37,081	307
Protein	43,471	81,044	3
Flickr	80,513	5.9M	195
IMDb	428,440	1.7M	22

(a)

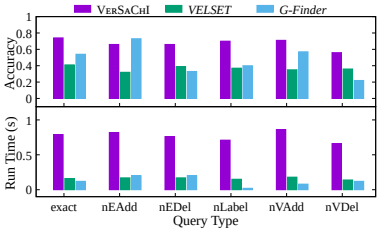
Dataset / Algorithm	Accuracy					Running Time (sec)				
	Human	HPRD	Protein	Flickr	IMDb	Human	HPRD	Protein	Flickr	IMDb
VELSET [21]	0.42	0.65	0.37	0.75	0.53	0.01	0.01	0.16	0.13	1.31
G-Finder [36]	0.45	0.12	0.47	out of memory		0.55	0.01	0.12	out of memory	
VERSACHI	**0.90**	**0.81**	**0.67**	**0.84**	**0.87**	0.12	0.06	0.77	1.98	6.90

(b)

Table 1: (a) Summary of the datasets characteristics. (b) Overall accuracy and runtime performance of the algorithms on the different datasets.



(a) Human



(b) IMDb

Figure 2: Performance for different query types on Human and IMDb datasets.

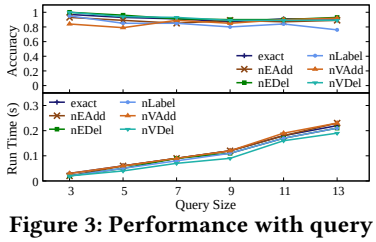
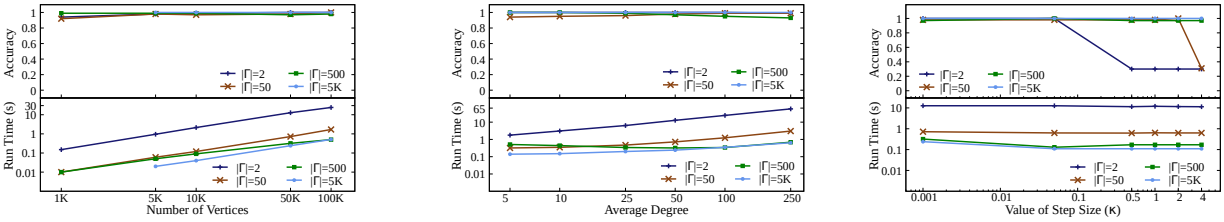


Figure 3: Performance with query size on Human dataset.



(a) Number of Vertices (Avg. Degree = 50, κ = 0.001) **(b)** Average Degree (|V| = 50K, κ = 0.001) **(c)** Step Size (κ) (|V| = 50K, Avg. Degree = 50)

Figure 4: Performance of VERSACHI on Barabási-Albert graphs with varying (a) number of vertices, n, (b) average degree, and (c) step size, κ.

Query. Query graphs (connected) are constructed (from the dataset) by initially selecting a random vertex, and exploring its neighborhood till n_Q vertices are visited. These are referred to as *exact queries*. To study the performance of the algorithms in presence of noise, exact query graphs were perturbed by introducing structural and label noise randomly by (i) modifying vertex labels (*nLabel*), or (ii) inserting or deleting vertices (*nVAdd* and *nVDel* resp.), or (iii) adding or deleting edges (*nEAdd* and *nEDel* resp.). The number of perturbations are limited to 2. Further, for each scenario, we generate queries with sizes varying from 3 to 13 (at intervals of 2), with 20 query graphs extracted for each size. Thus, for each dataset, we consider $(6 \times 6 \times 20) = 720$ queries, and report average results.

Evaluation. The efficiency of the algorithms are measured in terms of edge retrieval accuracy (using the labels of end vertices), that is, the fraction of edges of the query graph Q that are present in the matching subgraph retrieved. Additionally, we report the average runtime required (per query) by the different approaches to extract the approximate matching subgraphs. Since the introduced perturbations do not exist in the original graph, the exact query (for obtaining the noisy query) is considered as the ground truth. For *Barabási-Albert* graphs we use exact queries only.

Baselines. We compare the performance of VERSACHI algorithm against the following: (i) *VELSET* [21], a statistical significance based approach for exploring candidate regions with partial label match; and (ii) *G-Finder* [36], a graph traversal based indexing for dynamic filtering and refinement of matching neighborhoods.

Empirical Results

From Table 1(b), we observe that VERSACHI has a significantly better accuracy than the competing algorithms for finding the best matching subgraphs with more than 20% accuracy improvements (averaged across varying query types and sizes). The run-time of VERSACHI is slightly more than the other approaches due to the two-hop neighborhood similarity computation. In absolute terms, though, it is quite practical. Overall, with a slight increase in compute time, VERSACHI offers a substantial accuracy gain. (*G-Finder* crashes due to out-of-memory issues for Flickr and IMDb datasets.)

Fig. 2 depicts the performance for different query types. (Results on the other datasets are similar and are, thus, omitted due to space constraints). VERSACHI achieves better accuracy across all the different query types, with slight increase in runtime. Fig. 3 shows that with increase in query size, the runtime increases linearly (across query types), while the accuracy remains largely unaffected.

Fig. 4 studies the scalability of VERSACHI using synthetic *Barabási-Albert* graphs. The runtime is seen to increase linearly with number of vertices and average degree, conforming to the analysis of Sec. 2. The accuracy of VERSACHI is unaffected in these scenarios. With increase in the *step size* κ, accuracy decreases, as the number of symbols decrease, limiting the power of VERSACHI to differentiate between finer differences in neighborhood mismatches between the graphs, while the runtime remains mostly constant.

For more experimental results including experimental setup and indexing details, please see [2].

4 CONCLUSIONS

This paper proposed a scalable and highly accurate algorithm, VERSACHI, for approximate labeled graph querying. It shows significantly better accuracy than the competing methods across datasets and noise. Our framework is generic enough to accommodate other similarity measures and application-dependent tail distributions.

REFERENCES

[1] S. Agarwal, S. Dutta, and A. Bhattacharya. 2020. ChiSeL: Graph Similarity Search using Chi-Squared Statistics in Large Probabilistic Graphs. *PVLDB* 13, 10 (2020), 1654–1668.

[2] S. Agarwal, S. Dutta, and A. Bhattacharya. 2021. VerSaChI: Finding Statistically Significant Subgraph Matches using Chebyshev's Inequality. https://arxiv.org/abs/2108.07996.

[3] C. C. Aggarwal and H. Wang. 2010. Graph Data Management and Mining: A Survey of Algorithms and Applications. *Advances in Database Systems* 40 (2010), 13–68.

[4] A. Arora, M. Sachan, and A. Bhattacharya. 2014. Mining Statistically Significant Connected Subgraphs in Vertex Labeled Graphs. In *International Conference on Management of Data (SIGMOD)*. 1003–1014.

[5] L. Babai. 2016. Graph Isomorphism in Quasipolynomial Time. In *STOC*. 684–697.

[6] F. Bi, L. Chang, X. Lin, L. Qin, and W. Zhang. 2016. Efficient Subgraph Matching by Postponing Cartesian Products. In *SIGMOD*. 1199–1214.

[7] A. Bordes, S. Chopra, and J. Weston. 2014. Question Answering with Subgraph Embeddings. In *EMNLP*. 615–620.

[8] C. Chen, X. Yan, P. S. Yu, J. Han, D. Zhang, and X. Gu. 2007. Towards Graph Containment Search and Indexing. In *VLDB*. 926–937.

[9] W. Chen, J. Liu, Z. Chen, X. Tang, and K. Li. 2017. PBSM: An Efficient Top-K Subgraph Matching Algorithm. *International Journal of Pattern Recognition and Artificial Intelligence* 32, 6 (2017).

[10] X. Chen. 2020. Simulation-based Approximate Graph Pattern Matching. In *International Conference on Management of Data (SIGMOD)*. 2825–2827.

[11] X. Chen, L. Lai, L. Qin, X. Lin, and B. Liu. 2021. A Framework to Quantify Approximate Simulation on Graph Data. In *International Conference on Data Engineering (ICDE)*. 1308–1319.

[12] J. Cheng, Y. Ke, W. Ng, and A. Lu. 2007. FG-Index: Towards Verification-free Query Processing on Graph Databases. In *SIGMOD*. 857–872.

[13] T. Y. Cheung. 1983. State of the Art of Graph-based Data Mining. *Transactions on Software Engineering* 5, 1 (1983), 59–68.

[14] D. Conte, P. Foggia, C. Sansone, and M. Vento. 2004. Thirty Years of Graph Matching in Pattern Recognition. *IJPRAI* 18, 3 (2004), 265–298.

[15] S. A. Cook. 1971. The Complexity of Theorem-proving Procedures. In *STOC*. 151–158.

[16] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. 2004. A (Sub)graph Isomorphism Algorithm for Large Graphs. *PAMI* 26, 10 (2004), 1367–1372.

[17] S. Dutta. 2015. MIST: Top-k Approximate Sub-string Mining Using Triplet Statistical Significance. In *European Conference on Information Retrieval (ECIR)*. 284–290.

[18] S. Dutta and A. Bhattacharya. 2010. Most Significant Substring Mining Based on Chi-square Measure. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. 319–327.

[19] S. Dutta and A. Bhattacharya. 2012. Mining Statistically Significant Substrings Based on the Chi-Square Measure. In *Pattern Discovery Using Sequence Data Mining: Applications and Studies*. IGI Global, 73–82.

[20] S. Dutta and J. Lauri. 2019. Finding a Maximum Clique in Dense Graphs via Chi-Square Statistics. In *International Conference on Information and Knowledge Management (CIKM)*. 2421–2424.

[21] S. Dutta, P. Nayek, and A. Bhattacharya. 2017. Neighbor-Aware Search for Approximate Labeled Graph Matching using the Chi-Square Statistics. In *International Conference on World Wide Web (WWW)*. 1281–1290.

[22] B. Gallagher. 2006. Matching Structure and Semantics: A Survey on Graph-based Pattern Matching. In *AAAI*. 45–53.

[23] Rosalba Giugno and Dennis Shasha. 2002. GraphGrep: A Fast and Universal Method for Querying Graphs. *ICPR* 2 (2002), 201–212.

[24] Y. Gu, C. Gao, L. Wang, and G. Yu. 2016. Subgraph Similarity Maximal Allmatching over a Large Uncertain Graph. In *International Conference on World Wide Web (WWW)*. 755–782.

[25] M. Han, H. Kim, G. Gu, K. Park, and W. Han. 2019. Efficient Subgraph Matching: Harmonizing Dynamic Programming, Adaptive Matching Order, and Failing Set Together. In *SIGMOD*. 1429–1446.

[26] W. Han, J. Lee, M. Pham, and J. X. Yu. 2010. iGraph: A Framework for Comparisons of Disk-based Graph Indexing Techniques. *PVLDB* 3, 1-2 (2010), 449–459.

[27] L. Hong, L. Zou, X. Lian, and P. S. Yu. 2015. Subgraph Matching with Set Similarity in a Large Graph Database. *Transactions on Knowledge and Data Engineering* 27, 9 (2015), 2507–2521.

[28] H. Jiang, H. Wang, P. S. Yu, and S. Zhou. 2007. GString: A Novel Approach for Efficient Search in Graph DBs. In *ICDE*. 566–575.

[29] A. Jüttner and P. Madarasi. 2018. VF2++: An Improved Subgraph Isomorphism Algorithm. *Discrete Applied Mathematics* 242 (2018), 69–81.

[30] V. Kassiano, A. Gounaris, A. N. Papadopoulos, and K. Tsichlas. 2016. Mining Uncertain Graphs: An Overview. In *International Symposium on Algorithmic Aspects of Cloud Computing (ALGOCLOUD)*. 87–116.

[31] B. P. Kelley, B. Yuan, F. Lewitter, R. Sharan, B.R. Stockwell, and T. Ideker. 2004. PathBLAST: A Tool for Alignment of Protein Interaction Networks. *Nucleic Acids Research* 32 (2004), 83–88.

[32] A. Khan, Y. Wu, C. C. Aggarwal, and X. Yan. 2013. NeMa: Fast Graph Search with Label Similarity. *PVLDB* 6, 3 (2013), 181–192.

[33] A. Khan, X. Yan, and K. L. Wu. 2010. Towards Proximity Pattern Mining in Large Graphs. In *SIGMOD*. 867–878.

[34] S. Kpodjedo, P. Galinier, and G. Antoniol. 2014. Using Local Similarity Measures to Efficiently Address Approximate Graph Matching. *Discrete Applied Mathematics* 164 (2014), 161–177.

[35] G. Li, L. Yan, and Z. Ma. 2019. An Approach for Approximate Subgraph Matching in Fuzzy RDF Graph. *Fuzzy Sets and Systems* 376 (2019), 106–126.

[36] L. Liu, B. Du, J. Xu, and H. Tong. 2019. G-Finder: Approximate Attributed Subgraph Matching. In *International Conference on Big Data*. 513–522.

[37] L. Livi and A. Rizzi. 2013. The Graph Matching Problem. *Pattern Analysis and Application* 16 (2013), 253–283.

[38] S. Ma, Y. Cao, W. Fan, J. Huai, and T. Wo. 2014. Strong Simulation: Capturing Topology in Graph Pattern Matching. *Transactions on Database Systems (TODS)* 39, 1 (2014), 1–46.

[39] A. Mahmood, H. Farooq, and J. Ferzund. 2017. Large Scale Graph Matching (LSGM): Techniques, Tools, Applications and Challenges. *International Journal of Advanced Computer Science and Applications* 8, 4 (2017), 494–499.

[40] V. Nastase, R. Mihalcea, and D. R. Radev. 2015. A Survey of Graphs in Natural Language Processing. *Natural Language Engineering* 21 (2015), 665–698.

[41] T. Read and N. Cressie. 1988. *Goodness-of-fit Statistics for Discrete Multivariate Data*. Springer Series in Statistics.

[42] C. R. Rivero and H. M. Jamil. 2017. Efficient and Scalable Labeled Subgraph Matching using SGMatch. *Knowledge and Information Systems* 51 (2017), 61–87.

[43] R. A. Rossi and N. K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*. 4292–4293.

[44] M. Sachan and A. Bhattacharya. 2012. Mining Statistically Significant Substrings using the χ^2 Statistic. *PVLDB* 5, 10 (2012), 1052–1063.

[45] H. Shang, Y. Zhang, X. Lin, and J. Yu. 2008. Taming Verification Hardness: An Efficient Algorithm for Testing Subgraph Isomorphism. *PVLDB* 1, 1 (2008), 364–375.

[46] R. Shen and C. Guda. 2014. Applied Graph-Mining Algorithms to Study Biomolecular Interaction Networks. *BioMed Research Int.* 2014, 439476 (2014), 11.

[47] R. Singh, J. Xu, and B. Berger. 2008. Global Alignment of Multiple Protein Interaction Networks with Application to Functional Orthology Detection. *PNAS* 105, 35 (2008), 12763–12768.

[48] S. Sun and Q. Luo. 2019. Scaling Up Subgraph Query Processing with Efficient Subgraph Matching. In *ICDE*. 220–231.

[49] P. Tchebichef. 1867. Des Valeurs Moyennes. *Journal de Mathématiques Pures et Appliquees* 12 (1867), 177–184.

[50] Y. Tian, R. McEachin, C. Santos, D. States, and J. Patel. 2006. SAGA: A Subgraph Matching Tool for Biological Graphs. *Bioinformatics* 23, 2 (2006), 232–239.

[51] Y. Tian and J.M. Patel. 2008. TALE: A tool for approximate large graph matching. In *ICDE*. 963–972.

[52] A. Tversky. 1977. Features of Similarity. *Psychological Review* 84, 4 (1977), 327–352.

[53] J. R. Ullmann. 1976. An Algorithm for Subgraph Isomorphism. *JACM* 23, 1 (1976), 31–42.

[54] F. Vandin, E. Upfal, and B. J. Raphael. 2011. Algorithms for Detecting Significantly Mutated Pathways in Cancer. *JCB* 18, 3 (2011), 507–522.

[55] J. Wang, N. Ntarmos, and P. Triantafillou. 2016. Indexing Query Graphs to Speedup Graph Query Processing. In *EDBT*. 41–52.

[56] J. Yan, X. Yin, W. Lin, C. Deng, H. Zha, and X. Yang. 2016. A Short Survey of Recent Advances in Graph Matching. In *ICMR*. 167–174.

[57] Xifeng Yan, Philip S. Yu, and Jiawei Han. 2005. Graph Indexing Based on Discriminative Frequent Structure Analysis. *TODS* 30, 4 (2005), 960–993.

[58] X. Yan, P. S. Yu, and J. Han. 2005. Substructure Similarity Search in Graph Databases. In *SIGMOD*. 766–777.

[59] Y. Yuan, G. Wang, L. Chen, and H. Wang. 2015. Graph Similarity Search on Large Uncertain Graph Databases. *VLDB Journal* 24, 2 (2015), 271–296.

[60] S. Zhang, J. Li, H. Gao, and Z. Zou. 2009. A Novel Approach for Efficient Supergraph Query Processing on Graph Databases. In *EDBT*. 204–215.

[61] S. Zhang, J. Yang, and W. Jin. 2010. SAPPER: Subgraph Indexing and Approximate Matching in Large Graphs. *PVLDB* 3, 1-2 (2010), 1185–1194.

[62] Qinghua Zou, Shaorong Liu, and Wesley W. Chu. 2004. CTree: A Compact Tree for Indexing XML Data. In *WIDM*. 39–46.