VeNoM: Approximate Subgraph Matching with Enhanced Neighbourhood Structural Information

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

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

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

Abstract

Subgraph matching is an important research problem in the area of graph mining. Over the years, researchers have made significant headway in this direction with many state-of-the-art algorithms aimed at providing an efficient solution for approximate subgraph matching (ASM). Although many studies have been conducted to compare these and highlight their respective advantages and differences, little analysis has been done on how varying the different aspects of an ASM approach including the depth and breadth of neighborhood affect the performance. In this paper, we propose *VeNoM*, a variant of a state-of-the-art ASM algorithm VELSET, and present different extensions of it by parameterizing the breadth and depth of the neighborhood considered. We discuss the effects of these neighborhood parameters and other graph parameters on performance through empirical results over diverse datasets. We also compare the VeNoM instances against VerSaChI, a two-hop neighborhood similarity based ASM approach. The empirical results suggest that increasing the depth of a neighborhood can increase the accuracy of ASM significantly, although it requires a much longer running time. The breadth of the neighborhood for a vertex does not matter much as long as it is more than a single edge.

CCS Concepts

• **Information systems** → **Data mining**.

Keywords

Subgraph Similarity; Approximate Matching; Statistical Significance; Chi-Square; Labeled Graph

ACM Reference Format:
Shubhangi Agarwal, Sourav Dutta, and Arnab Bhattacharya. 2024. VeNoM: Approximate Subgraph Matching with Enhanced Neighbourhood Structural Information. In *7th Joint International Conference on Data Science & Management of Data (11th ACM IKDD CODS and 29th COMAD) (CODS-COMAD 2024), January 4–7, 2024, Bangalore, India.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3632410.3632459

1 Introduction

Graphs provide a strong foundation for a multitude of domains with a variety of problems and applications. Over the years, various querying and mining techniques have been devised for the same. An extremely important and popular area of research in graph mining is *subgraph querying*, where the problem is to retrieve subgraphs from a large graph that are similar to a query graph. This is useful in many domains such as question-answering [14], criminal network analysis [27], fraud detection [30], etc. If the end goal is retrieval of an exact match, it is referred to as *subgraph isomorphism* and is known to be NP-complete [8]. Hence, heuristic paradigms have been explored [9, 31]. However, these algorithms are not always scalable [16]. Furthermore, real-world graphs can be noisy and incomplete. In such cases, an approximate match is more appropriate.

Many *approximate subgraph matching (ASM)* algorithms [1, 10, 17] have been developed. Different factors affect the algorithms, such as degree of nodes, label distribution of graph, missing edges, etc. One of the most important aspects is, perhaps, the size of the neighborhood considered when comparing two nodes. Despite the considerable amount of research done in approximate subgraph querying methods, there is still room for a more in-depth study of the effect of these factors on the subgraph querying methods.

1.1 Contributions

We propose *VeNoM (Vertex Neighbor Matching)*, an enhancement over VELSET [10], a state-of-the-art ASM approach for deterministic graphs. To analyse the effect of size of neighborhood considered during matching, we instantiate four versions of VeNoM, namely, *VeNoM-(1,1)*, *VeNoM-(2,1)*, *VeNoM-(3,1)* and *VeNoM-(2,2)*. We also highlight that statistical significance measure assumes exact matches to be rarer in nature; however, under certain conditions this may not be the case, leading to lower accuracy. The experiments also provide an understanding of the effects of different graph parameters on performance (runtime and accuracy) of the individual instances. This would help in tuning and comparing the approximate subgraph matching algorithms, for example, the trade-off between a smaller (respectively, larger) running time and lesser (respectively, higher) matching accuracy. Extensive experiments, along with comparisons with other state-of-the-art algorithms (VELSET [10] and VerSaChI [1]) show that accuracy improves as the depth of the neighborhood comparison is increased at cost of a larger runtime, while breadth has a negligible effect as long as it is more than a single vertex, i.e., a single edge is the unit of comparison.

1.2 Related Work

The subgraph isomorphism problem deals with the problem of finding subgraphs in the data graph which are isomorphic to the query graph. An experimental evaluation of five subgraph isomorphism algorithms: VF2 [9], SPath [40], Turbo$_{ISO}$ [12], BoostIso [23] and RI [4]; is presented in [21]. Methodology, features and applications of different subgraph isomorphism algorithms are discussed in [28].

CODS-COMAD 2024, January 4–7, 2024, Bangalore, India

ACM ISBN 979-8-4007-1634-8/24/01…$15.00
https://doi.org/10.1145/3632410.3632459

SGMatch [25] is an isomorphic subgraph matching based on graph decomposition and set cover, solved using ILP. A subgraph-indexing based exact matching algorithm is proposed by [33]. In [32], subgraph matching algorithms are compared on different aspects rather than the absolute performance alone. A short review of various subgraph isomorphism algorithms is provided in [29]. A single framework for subgraph isomorphism methods is given in [19]. However, since subgraph isomorphism is computationally complex, inexact matching is favored due to its practicality. A better understanding of fundamentals of exact and approximate graph matching is presented by [24]. Many approximate subgraph querying techniques have been developed in past; SAPPER [39] is based on graph edit distance, while an efficient graph indexing based method is presented in gIndex [36]. A set-cover based approach is explored in SIGMA [22], and APGM [15] uses a method to mine useful patterns from noisy graph databases. NeMa [17] and SIM-T [18] are based on neighborhood search while VELSET and NAGA are based on statistical analysis [10]. PBSM [6] uses a filter and verification approach while DAF [11] is based on DAG ordering and dynamic programming. VerSaChI [1] is another ASM approach employing statistical analysis and Chebyshev's inequality to quantify two-hop neighborhood similarity. Other notable subgraph matching techniques are TALE [34], FG-Index [7], iGraph [13], Grafil [37], Gcoding [41], GPTree [38], cIndex [5] and G-Finder [20]. A comprehensive survey of graph matching is provided in [35]. Although various ASM approaches have been explored, not much is known of impact of neighborhood size as parameters on their performance. We attempt to bridge this gap for a deeper understanding.

2 The Framework

We first present a succinct summary of VELSET and outline the scenarios where its effectiveness is limited (Sec. 2.2). We next present our algorithm, VeNoM, which is a variant of VELSET, in Sec. 2.3. Different extension flavors, based on depth and breadth of a neighborhood, are discussed in Sec. 2.5-Sec. 2.7. Fig. 1 shows an overview of VeNoM.

2.1 Notations.

Consider a *target graph* $G(V_G, E_G, \mathcal{L}_G)$, with vertex set V_G, set of edges E_G and a function $\mathcal{L}_G : V_G \rightarrow \Sigma_G$ that assigns labels to vertices, where Σ_G is a finite set of labels. A *query graph* $Q(V_Q, E_Q, \mathcal{L}_Q)$ is searched for in G, where V_Q and E_Q represent the set of query vertices and edges, respectively, and $\mathcal{L}_Q : V_Q \rightarrow \Sigma_G$ is the query vertex label function. We use the notation $\mathcal{N}_G(v)$ and $\mathcal{N}_Q(q)$ to denote the set of one-hop neighbors of $v \in V_G$ and $q \in V_Q$, respectively. Additionally, $\langle q, v\rangle$ denotes a candidate vertex pair, where $v \in V_G$ is a candidate match for $q \in V_Q$. A candidate match is any target vertex v that satisfies the condition $\mathcal{L}_Q(q) \approx \mathcal{L}_G(v)$, i.e., the label of the query vertex q is "similar" to that of v. The similarity is based on a user-defined similarity metric, which may be exact or approximate, e.g., Levenshtein distance, Jaccard similarity, etc.

2.2 Overview of VELSET

VELSET is an ASM framework for deterministic labeled graphs based on statistical analysis. To compute similarity between two nodes, it compares the 1-hop neighborhood of the individual vertices. For this purpose, a set of *triplets* is created for each vertex of target graph and query graph. A triplet for a vertex $q \in V_Q$ is of the form $\langle \mathcal{L}_Q(q_i), \mathcal{L}_Q(q), \mathcal{L}_Q(q_j)\rangle$, where $q_i, q_j \in \mathcal{N}_Q(q)$. Triplets for target vertices are defined in the same way. The list of triplets of a vertex represents the label distribution in its neighborhood.

In VELSET, the candidate vertex pairs are formed based on the Jaccard similarity of the labels. To characterize the neighborhood similarity of a vertex pair $\langle q, v\rangle$, the triplets of q are matched against those of v. Triplet matches are categorized into three categories: s_0, when no neighbor label is found to be similar; s_1, when only one of the neighbor labels is similar; and s_2, when both the neighbor labels are similar. The categories are ordered as $s_2 \succ s_1 \succ s_0$, i.e., s_2 is preferred over s_1, and s_1 is preferred over s_0.

To quantify the match similarity of a candidate pair, the *Pearson's chi-squared statistical significance test* is used. It evaluates the likelihood that the deviation of observed from expected is a chance occurrence. Mathematically, $\chi^2 = \sum_i \frac{(O_i - E_i)^2}{E_i}$, where E_i and O_i denote the expected and the observed value, respectively, for each s_i. The observed value for a vertex pair $\langle q, v\rangle$ is the count of the categorical matches obtained by matching triplets. To compute the expected value of the categorical matches, VELSET assumes uniform label distribution and argues that the probability of a label (from query triplet) not matching is $(1 - 1/L)^{d_v}$, where L is number of unique labels and d_v is the degree of vertex $v \in V_G$. Based on this, the probability for each of the categories is defined as:

$$\begin{aligned} P_v(s_0) &= \left((1 - 1/L)^{d_v}\right)^2 \\ P_v(s_1) &= 2 \cdot (1 - 1/L)^{d_v} \cdot \left(1 - (1 - 1/L)^{d_v}\right) \\ P_v(s_2) &= \left(1 - (1 - 1/L)^{d_v}\right)^2 \end{aligned} \tag{1}$$

The expected values are computed by scaling the probability values appropriately for a vertex pair and is used in chi-squared value computation to capture its similarity.

In the final step, highest scoring candidate pair is greedily expanded until a match is found or the search space of candidate neighbor pairs is exhausted. The interested readers can refer to the original paper [10] for more details.

2.3 VeNoM-(2,1)

The expected value calculation in VELSET is influenced by two factors, the number of unique labels and the degree of target graph vertices. The intuition is that expected value for the best match category would be very low and any vertex pair exhibiting a *perfect* triplet match would thus get a higher chi-square value. We verify this by assuming different values of L and d_v in Eq. 1. We also assume a query vertex q of degree 3 and enumerate four possible scenarios based on number of labels that find a match to analyze the chi-square trends (observed values of categories are denoted as $(\#s_0, \#s_1, \#s_2)$):

- m_0: when none of the neighbor labels match, $(3, 0, 0)$
- m_1: when exactly one neighbor label matches, $(1, 2, 0)$
- m_2: when exactly two of the neighbor labels match, $(0, 2, 1)$
- m_3: when all the neighbor labels match, $(0, 0, 3)$

Fig. 2a shows the probability of different match categories for a fixed number of unique labels, $L = 10$ and, different degrees of the vertex. It is observed that $P_v(s_2) < P_v(s_0)$ for lower degrees of v and, for higher degrees of the vertex, the expected values for the s_2 category were higher than for s_0. Due to this reason an inversion in the chi-square values can be seen in Fig. 2b. Note that m_3 is the best

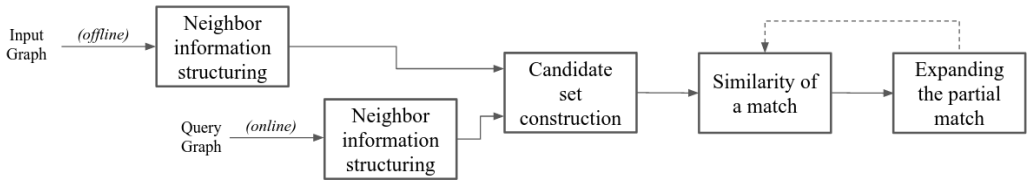
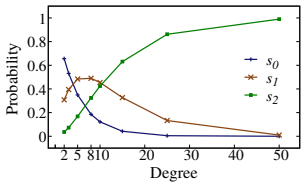
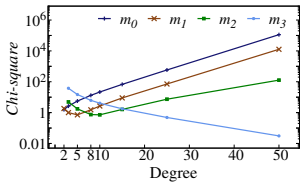


Figure 1: Generalized flowchart used in VeNoM



(a) **Probability trends of s_0, s_1 and s_2**



(b) **χ^2 values for match scenarios m_0-m_3 for query degree 3**

Figure 2: Expected value and χ^2 trends for VELSET for $L = 10$.

possible matching scenario for q, followed by m_2, m_1 and m_0 in that order. It can be seen that for the χ^2 for the best scenario decreases with increase in the input vertex degree. This essentially penalizes a good input vertex match for having a higher number of connections.

In general, the inversion is observed when the label set size is much smaller compared to the vertex degree (roughly, a degree to label ratio of > 0.65). To overcome this issue, we propose VeNoM, which modifies the expected value calculation by replacing the input vertex degree d_v in Eq. 1 with d_q. This is useful since, in general, the query graphs are small and have lower degrees. Thus, with $p_q = (1 - 1/L)^{d_q}$, the probability of the match categories in VeNoM are computed as

$$P_q(s_0) = (p_q)^2; \qquad P_q(s_1) = 2 \cdot p_q \cdot (1 - p_q); \qquad P_q(s_2) = (1 - p_q)^2 \tag{2}$$

For ease of discussing different extensions of VeNoM in this article, we define two terms, *unit* and *group*. Note that both the terms are defined with respect to a vertex.

Definition 2.1. **Unit**. An ordered collection of neighbor labels of a vertex which form a path of length h.

Definition 2.2. **Group**. A set of k units of a vertex along with its label where the units correspond to different neighbors.

The terms *unit* and *group* help us in extending the triplet format to extensions of VeNoM. As discussed previously, a triplet consists of *two* 1-hop neighbor labels of a vertex. In the proposed terminology, we refer to each of these neighbor labels as a *unit*. A triplet is then essentially a *group of two units* along with the label of the vertex. Hence, we denote the modified version of VELSET, as an instance VeNoM-(2,1), in the VeNoM framework. The first number in the bracket denotes the total number of units in a group, while the second number denotes the number of vertices in a unit.

We use the same notation scheme VeNoM-(k,h) for extensions and reductions of VeNoM-(2,1). Essentially, h controls the depth (or hops) of the neighborhood considered while k controls the breadth of its neighborhood that is considered during a match.

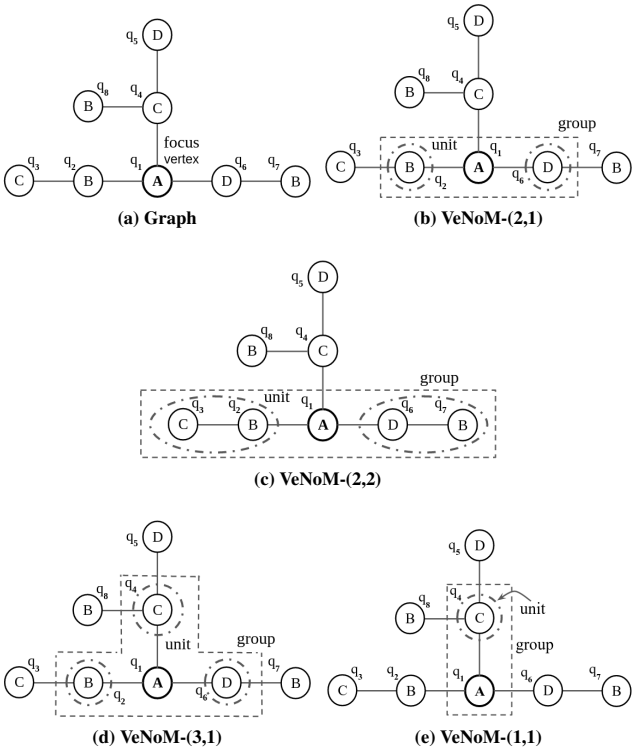


Figure 3: Units and groups in different instances of VeNoM, for vertex q_1. Units are represented by dash-dotted elliptical lines and groups by dashed rectangular lines.

2.4 Extensions

We describe two methods through which VeNoM-(2,1) can be extended to create a family of algorithms: (1) *multi-hop*, where the depth of a unit is increased to include farther neighbors (Sec. 2.5); (2) *multi-neighbor* (Sec. 2.6), the number of units in a group is increased. Additionally, we also discuss how VeNoM-(2,1) can be reduced to a single vertex unit and group version, (Sec. 2.7). Fig. 3 shows the difference between the coverage of neighborhoods in different instances of VeNoM. The dash-dotted line depicts a unit, while the dotted rectangular line outlines a group. For example, for vertex q_1 with label A, in VeNoM-(1,1), VeNoM-(2,1) and VeNoM-(3,1), labels of only its one-hop neighbors are considered as units, i.e., the size of each unit of q_1 is 1. Hence, in Fig. 3 (b), (d) and (e), q_1 has three units, ($\langle B\rangle$, $\langle C\rangle$ and $\langle D\rangle$). While for VeNoM-(2,2) a unit is an ordered pair of two vertex labels, $\langle$1-hop vertex label, 2-hop vertex label$\rangle$. Likewise, in Fig. 3 (c), q_1 has four units, each composed of two vertices: $\langle B, C\rangle$, $\langle C, D\rangle$, $\langle C, B\rangle$ and $\langle D, B\rangle$.

In a group, both VeNoM-(2,1) and VeNoM-(2,2) use two units, VeNoM-(3,1) groups are composed of three units while groups in VeNoM-(1,1) contain a single unit. Under VeNoM-(2,1), in Fig. 3(b), there are three possible groups for vertex q_1: $\langle A, \langle B\rangle, \langle D\rangle\rangle$, $\langle A, \langle B\rangle, \langle C\rangle\rangle$ and $\langle A, \langle C\rangle, \langle D\rangle\rangle$. We discuss the structure of units and groups for other instances in the subsequent sections.

2.5 VeNoM-(2,2)

The VeNoM-(2,1) approach can be extended to h-hops by increasing the depth of the neighborhood considered for each vertex to h-hops, thereby increasing the number of entities in a unit. For brevity, we describe the framework for $k = 2$ and $h = 2$.

In VeNoM-(2,2), a unit for $q \in V_Q$ is a tuple of the form $\langle \mathcal{L}_Q(q_i^1), \mathcal{L}_Q(q_j^2)\rangle$, where $q_i^h \in V_Q$ is the i^{th} node exactly h hops away from q. Note that $q_j^2 \in \mathcal{N}_Q(q_i^1) \setminus \{q\}$, i.e., q_j^2 is a neighbor of q_i^1 other than q. A similar notation is used corresponding to the target vertex.

A group in VeNoM-(2,2) is constructed for each vertex using two of the units of the vertex and the label of the vertex itself. We restrict a group to have units that correspond to different 1-hop neighbors of the vertex to ensure a better structural match. A group for q is of the form $\langle \mathcal{L}_Q(q), \langle \mathcal{L}_Q(q_i^1), \mathcal{L}_Q(q_j^2)\rangle, \langle \mathcal{L}_Q(q_k^1), \mathcal{L}_Q(q_l^2)\rangle\rangle$, with $i \neq k$.

In Fig. 3 (c), q_1 has five valid groups: $\langle A, \langle B, C\rangle, \langle D, B\rangle\rangle$, $\langle A, \langle B, C\rangle, \langle C, B\rangle\rangle$, $\langle A, \langle B, C\rangle, \langle C, D\rangle\rangle$, $\langle A, \langle C, B\rangle, \langle D, B\rangle\rangle$, and $\langle A, \langle C, D\rangle, \langle D, B\rangle\rangle$. An example of an *invalid* group of q_1 is $\langle A, \langle C, B\rangle, \langle C, D\rangle\rangle$, since both the units $\langle C, B\rangle$ and $\langle C, D\rangle$, correspond to the same neighbor q_4. The units and groups are similarly constructed for the target vertices.

2.5.1 Similarity of a Vertex Pair. To compute the similarity of a vertex pair $\langle q, v\rangle$, we match each group of q against an unmatched group of v. The similarity of two groups is defined based on the similarity of their respective constituent units. In VeNoM-(2,2), there are three unit-match categories possible, with $u_2 \succ u_1 \succ u_0$:

- u_2: When both the labels of the query unit have an exact match.
- u_1: When exactly one of the labels of the query unit matches.
- u_0: When none of the labels of the query unit find a match.

Similarly, five group-match levels, s_0-s_4, are defined. The level s_0 corresponds to the case when none of the constituent neighbor labels of the query group find a match; s_1 when exactly one of them matches and so on. Again, to maximize overlap the group-match categories are ordered, $s_4 \succ s_3 \succ s_2 \succ s_1 \succ s_0$, i.e., s_4 is the best match level. The group-match categories can be defined based on the match categories assigned to the constituent units. For instance, a group match would be assigned to the category s_0 when none of the constituent units find a match, i.e., both the pairs are assigned to the category u_0. Similarly, a match is assigned to the category s_1 when one of the two pairs results in a u_1 match while the other in u_0. Mathematically,

$$\begin{gathered} s_0 : (u_0 \wedge u_0)\ ; \quad s_1 : (u_0 \wedge u_1)\ ; \quad s_2 : (u_0 \wedge u_2) \vee (u_1 \wedge u_1)\ ; \\ s_3 : (u_1 \wedge u_2)\ ; \quad s_4 : (u_2 \wedge u_2) \end{gathered} \tag{3}$$

Here, $\wedge$ represents the AND operation and $\vee$ represents the OR operation among the match events. Observe that the five events (s_0-s_4) are exclusive and exhaustive.

To compute the similarity of a vertex pair $\langle q, v\rangle$, we compute its chi-square value, which calculates the deviation of the observed counts of the symbols s_0-s_4, from their expected counts. The observed counts of these events are computed by comparing the groups of q against the groups of v. A group associated with v once matched is not considered for match with any other group of q. The expected counts of the group-match levels can be computed from the probability distribution of the group-match levels. We first compute the probabilities of the unit-match levels and then using Eq. 3 compute the probability distribution for the group-match levels.

2.5.2 Probability Distribution of Unit Symbols, $P_q(u_i)$. The probability that a label does not match depends on the number of unique labels L in the target graph. Assuming a uniform distribution of labels, the probability that a label does not match is $(1 - 1/L)$. Then the probability that none of the 1-hop neighbors of a query node q with degree d_q match the label is $p_q = (1 - 1/L)^{d_q}$. The probability that the second hop label in the query unit does not find a match is $\beta_q = (1 - 1/L)^{b_q}$, where $b_q = \sum_{i=1}^{d_q}(d_{q_i^1} - 1)$ is the number of 2-hop neighbors of q (d_{q^i} is the degree of q_i^1). The value $d_{q_i^i}$ is reduced by 1 to avoid counting q as a 2-hop neighbor of itself. A unit will be assigned the symbol u_1 when exactly one of the two vertices in the unit match. This can happen in two ways, either the 1-hop label matches and the 2-hop label does not match or the 2-hop label matches but the 1-hop label does not. In the first case, the 2-hop label is absent in the neighborhood of the matching 1-hop label. Therefore, the probability of mismatch of 2-hop label is conditioned upon the 1-hop label, which is computed as $\delta_q = (1 - 1/L)^{avg_d(q)}$, where $avg_d(q) = \sum_{i=1}^{d_q}(d_{q_i^1} - 1)/d_q$ is the average number of 2-hop neighbors of q per its 1-hop neighbor. Thus,

$$\begin{gathered} P_q(u_0) = p_q \cdot \beta_q\ ; \qquad P_q(u_1) = (\bar{p}_q \cdot \bar{\delta}_q) + (p_q \cdot \bar{\beta}_q)\ ; \\ P_q(u_2) = \bar{p}_q \cdot \bar{\delta}_q \end{gathered} \tag{4}$$

where $\bar{p}_q = (1 - p_q)$, $\bar{\beta}_q = (1 - \beta_q)$ and $\bar{\delta}_q = (1 - \delta_q)$.

2.5.3 Probability Distribution of Group Symbols, $P_q(s_i)$. The conditions for symbols s_1-s_3, as given in the Eq. 3, and can be achieved in more than one way. For instance, in VeNoM-(2,2) for a group to be assigned the match symbol s_1, the best match found for one of its units has to be u_1 and u_0 for the other. The combination of the match categories u_0 and u_1 can manifest in two ways. Such scenarios are accounted for when computing the probabilities. Considering such scenarios, the probabilities of the group symbols are computed as,

$$P_q(s_0) = P_q(u_0) \cdot P_q(u_0) = (p_q \cdot \beta_q)^2 \tag{5a}$$

$$P_q(s_1) = 2 \cdot P_q(u_0) \cdot P_q(u_1) = 2 \cdot (p_q \cdot \beta_q) \cdot \Big((\bar{p}_q \cdot \bar{\delta}_q) + (p_q \cdot \bar{\beta}_q)\Big) \tag{5b}$$

$$P_q(s_2) = 2 \cdot (p_q \cdot \beta_q) \cdot (\bar{p}_q \cdot \bar{\delta}_q) + \Big((\bar{p}_q \cdot \delta_q) + (p_q \cdot \bar{\beta}_q)\Big)^2 \tag{5c}$$

$$P_q(s_3) = 2 \cdot P_q(u_1) \cdot P_q(u_2) = 2 \cdot \Big((\bar{p}_q \cdot \delta_q) + (p_q \cdot \bar{\beta}_q)\Big) \cdot (\bar{p}_q \cdot \bar{\delta}_q) \tag{5d}$$

$$P_q(s_4) = P_q(u_2) \cdot P_q(u_2) = (\bar{p}_q \cdot \bar{\delta}_q)^2 \tag{5e}$$

Observe that both the unit-match categories and group-match categories are exhaustive, i.e., $\sum_{i=0}^{2} P_q(u_i) = 1$ and $\sum_{i=0}^{4} P_q(s_i) = 1$.

2.5.4 Expected Distribution of Symbols. Multiple units and groups can be generated for a vertex, using different 1-hop and 2-hop neighbors in combination. The total number of units associated with query node q can be computed as $D_q = \sum_{q_j^1 \in \mathcal{N}_Q(q)}(d_{q_j^1} - 1)$. Again, the degree $d_{q_j^1}$ is reduced by 1 to avoid counting q as the 2-hop neighbor of itself. From D_q units, $\binom{D_q}{2}$ groups can be constructed. Of these,

$\sum_{q_j^1 \in \mathcal{N}_Q(q)} \binom{d_{q_j^1}-1}{2}$ units correspond to the same 1-hop neighbors of q, and are, therefore, invalid. Assuming that all the groups are independent of each other, the total number of valid groups that q can exhibit, is computed as, $\eta_q = \binom{D_q}{2} - \sum_{q_j^1 \in \mathcal{N}_Q(q)} \binom{d_{q_j^1}-1}{2}$.

The probability of symbols s_0-s_4 is defined for a single query group, and is multiplied by the total number of groups of by q to compute their expected values for v,

$$E_q(s_i) = P_q(s_i) \times \eta_q \tag{6}$$

2.6 VeNoM-(3,1)

In the multi-neighbor extension scheme, instead of matching a deeper neighborhood structure, a broader neighborhood is matched. In other words, units under this scheme consist of 1-hop neighbors only, as in VeNoM-(2,1); but, the number of units in a group is increased to 3. The two extension schemes can be used in conjunction with each other, i.e., an instance of the type VeNoM-(3,2) is also possible. However, for ease of understanding, we discuss the scheme with $h = 1$ and $k = 3$ units in a group.

2.6.1 Unit and Group Matches. For the same graph setting, as in Sec. 2.5, a group with q as the focus vertex would now look like $\langle \mathcal{L}_Q(q), \mathcal{L}_Q(q_i^1), \mathcal{L}_Q(q_j^1), \mathcal{L}_Q(q_k^1) \rangle$, where $i \neq j \neq k$, with only four vertex labels. For instance, for the graph shown in Fig. 3(d), there is only one group possible for vertex q_1, $\langle A, \langle B \rangle, \langle C \rangle, \langle D \rangle \rangle$.

Since each unit now has only one vertex, there are only two unit-match levels possible, u_0 and u_1, when none or at least one matching unit is found, respectively. As maximum matching is preferred, $u_1 \succ u_0$. Likewise, there are 4 group-match categories, $s_3 \succ s_2 \succ s_1 \succ s_0$, which are defined using the unit-match levels as,

$$\begin{aligned} s_0 &: (u_0 \wedge u_0 \wedge u_0) \; ; \quad s_1 : (u_0 \wedge u_0 \wedge u_1) \\ s_2 &: (u_0 \wedge u_1 \wedge u_1) \; ; \quad s_3 : (u_1 \wedge u_1 \wedge u_1) \end{aligned} \tag{7}$$

Observe that there are three different ways in which the configurations for the symbols s_1 and s_2 can be obtained.

2.6.2 Probability Distribution of Symbols. For q, the probabilities for u_0 and u_1, now, are,

$$P_q(u_0) = p_q \; ; \qquad P_q(u_1) = 1 - p_q = \bar{p}_q \tag{8}$$

The probabilities of the group-match categories based on Eq. 7 and 8, are given by,

$$P_q(s_0) = P_q(u_0) \cdot P_q(u_0) \cdot P_q(u_0) = p_q \cdot p_q \cdot p_q = p_q^3 \tag{9a}$$

$$P_q(s_1) = 3 \cdot (P_q(u_0))^2 \cdot P_q(u_1) = 3 \cdot p_q^2 \cdot \bar{p}_q \tag{9b}$$

$$P_q(s_2) = 3 \cdot P_q(u_0) \cdot (P_q(u_1))^2 = 3 \cdot p_q \cdot \bar{p}_q^2 \tag{9c}$$

$$P_q(s_3) = P_q(u_1) \cdot P_q(u_1) \cdot P_q(u_1) = \bar{p}_q \cdot \bar{p}_q \cdot \bar{p}_q = \bar{p}_q^3 \tag{9d}$$

2.6.3 Expected Distribution of Symbols. The expected values for each symbol is obtained by appropriately scaling their probabilities corresponding to the number of groups exhibited by the query node q, as shown in Eq. 6. For VeNoM-(3,1), the number of groups exhibited by the query node q is $\eta_q = \binom{d_q}{3}$.

2.7 VeNoM-(1,1)

The VeNoM framework is flexible and the VeNoM-(2,1) version can also be reduced to VeNoM-(1,1) configuration. In VeNoM-(1,1), the groups consist of a single unit which is made up of only one 1-hop neighbor. For example, in Fig. 3(e), q_1 has three possible groups: $\langle A, \langle B \rangle \rangle$, $\langle A, \langle C \rangle \rangle$ and $\langle A, \langle D \rangle \rangle$. Once again, only two unit-match levels are defined, u_0 and u_1, with $u_1 \succ u_0$, as in Eq. 8.

Now, since there is only a single unit in a group, unlike previous algorithms, the group-match categories become synonymous with the unit-match levels. To clarify, there are now only two group-match levels: s_0 and s_1, with $s_1 \succ s_0$. The category s_0 is assigned when the unit results in a complete mismatch, i.e, when the single constituent vertex label of the unit does not match. While s_1 denotes the case when the unit is a complete match, i.e., the single constituent vertex label finds a match. Consequently, the group-match probabilities are the same as the unit-match level.

$$P_q(s_0) = P_q(u_0); \qquad P_q(s_1) = P_q(u_1) \tag{10}$$

For computing the expected distribution of the group-match categories, the same process is followed as given in Eq. 6. For the reduced case, the value of $\eta_q = d_q$.

Table 1: Characteristics of real-world datasets

Dataset	#Vertices	#Edges	#Labels
Human	4.6K	86.2K	44
HPRD	9.4K	37K	307
Flickr	80.5K	5.9M	195
PPI	12K	10.74M	2.4K

2.8 Complexity Analysis

Assume a target graph G with n_G vertices and L unique labels uniformly distributed and a query graph Q with n_Q nodes and average degree d_Q. The time complexity for matching a single query group is $O(1)$ with efficient target graph indexing. For a query node with m groups and n_G/L possible candidate target vertices, the complexity of finding the best match is $O(m \cdot n_G/L)$. The computation of m can be broken down to the number of units possible with depth h and k units in a group. The number of units possible with depth h is of the order $O(d_Q^h)$. The number of possible groups is then approximately $\binom{d_Q^h}{k}$ which is of the order $O(d_Q^{h \cdot k})$. Therefore, the complexity of matching the query graph Q with VeNoM-(k, h) is roughly $O(n_Q \cdot d_Q^{h \cdot k} \cdot n_G/L)$. Since query graphs are typically quite small, d_Q is a constant. The time complexity of the entire algorithm is, thus, effectively $O(n_Q \cdot n_G/L)$.

3 Experimental Results

In this section, we analyze the performance of the extension schemes along with the base algorithm on various graph invariants.

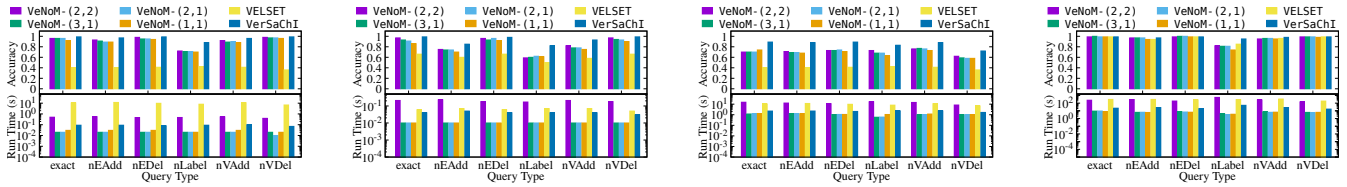
3.1 Experimental Setup

All the algorithms were implemented in C++. The experiments were performed on an Intel(R) Xeon(R) 2.6GHz CPU E5-2697v3 processor with 504GB RAM running CentOS Linux 7.9.2009.[1]

3.1.1 Benchmarks. (1) VELSET [10], since it is the algorithm that we extend, and it also outperformed the then state-of-the-art algorithms NeMa [17] and SIM-T [18]; (2) VerSaChI [1], which is a recent ASM framework that outperformed VELSET. Therefore, we compare against these two algorithms only.

3.1.2 Real Datasets. The Flickr [26] dataset is a large *social network* based on user interactions, while PPI, Human and HPRD [3] are *biological networks* with protein-protein interactions. PPI

[1]The source code is available at https://github.com/shubhangiat/VeNoM.



(a) Human **(b) HPRD** **(c) Flickr** **(d) PPI**

Figure 4: Performance on real-world graphs (runtime in log-scale)

is a randomly extracted small graph from STRING DB (version-10-5.string-db.org). Table 1 summarizes the characteristics of the datasets (including the ratio of average degree of the graph to its label set size).

3.1.3 Synthetic Datasets. We chose the Barabási-Albert (BA) graphs to evaluate scalability of instances of VeNoM, as it is known to be scale-free and approximate real-world graphs [2]. It has two parameters: the number of vertices (n) and the growth factor (m), i.e., the number of edges linking a new node to the existing graph. We introduce a third parameter, the number of unique vertex labels (l). The default values are set to $n = 100K$, $m = 50$ and $l = 150$.

3.1.4 Query Generation. For our experiments, queries of size 9 were created by randomly extracting subgraphs from real-world graph datasets starting with a randomly selected seed vertex. As these queries have an exact match available, we name this query set *exact*. Further, we introduce noises in the *exact* query set to generate different types of noisy queries, (i) addition and deletion of edges (*nEAdd* and *nEDel*, respectively), (ii) alteration of vertex label (*nLabel*) and (iii) addition and deletion of vertices (*nVAdd* and *nVDel*, respectively). The maximum number of perturbations for any exact query graph was capped at 2 while ensuring the connectivity of the query graph. Each query set consisted of 40 instances.

For the synthetic graph experiments, different query sets of size 9 were generated in a similar fashion. For the default synthetic graph, additional queries of sizes 5, 7, 11 and 13, were also created to study the effect of varying query size on different instances of VeNoM.

3.1.5 Metrics. We evaluate the performance of the four algorithms on the following metrics:

(1) *Maximum mean accuracy*: We define accuracy as the ratio of the number of edges in the answer subgraph that match the query subgraph to the number of edges in the query subgraph. To evaluate the quality of the matching subgraphs returned by the algorithms, we choose the answer subgraph with maximum accuracy among the *top-10* highest χ^2 value subgraphs. The average of the maximum accuracy over each query set is reported. We refer to this as *accuracy* in this paper for simplicity.

(2) *Average running time*: To evaluate the efficiency of the algorithms, the time taken to return the *top-10* answer subgraphs is recorded for each query. We compare the running times averaged over a query set, referred to as *runtime*.

3.2 Real-World Graphs

Fig. 4 shows the performance of instances of VeNoM and benchmark algorithms. The accuracy of VELSET was lowest and can be attributed to the apparent inversion of the chi-square values from the expected trend. The improved performance demonstrated by VeNoM indicates that the proposed remodeling was effective.

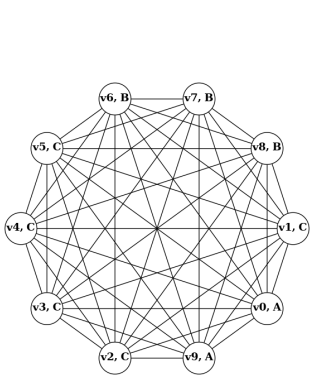
Overall, VeNoM-(2,2) performed slightly better than VeNoM-(2,1) or was comparable. This improvement can be attributed to the two-hop neighborhood match done in VeNoM-(2,2) giving it a *look-ahead* advantage over VeNoM-(2,1) and VeNoM-(3,1). At the same time, VeNoM-(1,1) performed slightly worse than VeNoM-(2,1), as it compares only one neighbor at a time, which reduced its capacity to capture the neighborhood topological similarities. However, for PPI it is observed that VeNoM-(3,1) performed marginally better than VeNoM-(2,2). We explore this further in Sec. 3.3. In general, VerSaChI exhibited relatively higher accuracy which can be due to the two-hop neighborhood overlap-based similarity. However, the run-time of VerSaChI was 5 to 10 times more than that of VeNoM-(x, 1) ($x \in \{1, 2, 3\}$). Due to the depth of the neighbors considered VeNoM-(2,2) showed a higher runtime. As the depth is increased, the number of possible groups increases multi-fold, which adds to the complexity of matching. VeNoM-(2,1) and VeNoM-(3,1) were comparable in terms of both accuracy and runtime.

In general, a significant drop was observed in the accuracy for the *nLabel* query set for all the algorithms. This can be attributed to the underlying semantic relationship between the node labels. During the generation of noisy label queries, the label of some of the vertices was randomly changed to another label from the label vocabulary of the dataset. This did not conform to the underlying label distribution and topology of the real graph, which resulted in partial matches.

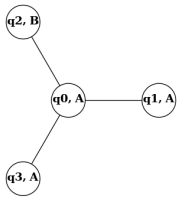
The comparison on real graphs highlights that an edge based neighborhood match, i.e., one vertex at a time, as is the case in VeNoM-(1,1) is insufficient. However, there is no significant improvement when number of vertices being compared is increased from two (VeNoM-(2,1)) to three (VeNoM-(3,1)). Notably, there is no overhead in runtime in a multi-neighbor extension. Between VerSaChI and VeNoM-(2,2), the former is a more holistic approach and performs better, suggesting that as the depth of the neighborhood being compared increases, a more comprehensive view of the neighborhood may be required, by increasing the neighborhood span, for a better comparison.

3.3 VeNoM-(2,2) vs VeNoM-(3,1)

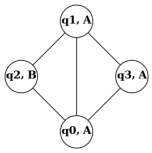
We further analyze the performance difference between VeNoM-(2,2) and VeNoM-(3,1) using toy examples. We choose a complete and labeled graph of size 10 and queries of size 4 (Fig. 5). For query graph 1, (Fig. 5(a)) no performance difference was observed among the algorithms. However, for query graph 2 (Fig. 5(b)), VeNoM-(3,1) exhibited a larger accuracy. On further analysis, it was found this is due to the difference in the selection of candidate pair, based on their χ^2 values (as shown in Fig. 5). Even though the candidate pair $\langle$ q1, v9 $\rangle$ is a better match accuracy-wise, the vertex pair $\langle$ q3, v9$\rangle$ achieves a higher χ^2 value for all instances of VeNoM but VeNoM-(3,1). For VeNoM-(3,1), both the above mentioned vertex



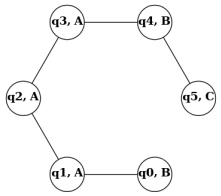
(a) Target Graph



(b) Query Graph 1



(c) Query Graph 2



(d) Query graph 3

Query Graph 2

Algorithms	Best Match	Acc.	χ^2 values ⟨q1, v9⟩	χ^2 values ⟨q3, v9⟩
VeNoM-(2,2)	⟨q0,v0⟩, ⟨q2,v7⟩, ⟨q3,v9⟩	0.40	1.064	1.074
VeNoM-(3,1)	⟨q0,v0⟩, ⟨q1,v9⟩, ⟨q2,v7⟩	0.60	1.272	1.272
VeNoM-(2,1)	⟨q0,v0⟩, ⟨q2,v7⟩, ⟨q3,v9⟩	0.40	0.871	1.025
VeNoM-(1,1)	⟨q0,v0⟩, ⟨q2,v7⟩, ⟨q3,v9⟩	0.40	1.600	1.999

Query Graph 3

Algorithms	2^{nd} and; 3^{rd} Best Match	2^{nd} Best Acc.	3^{rd} Best Acc.
VeNoM-(2,2)	⟨q4, v7⟩, ⟨q5, v2⟩; ⟨q4, 6⟩, ⟨q5, 3⟩	0.2	0.2
VeNoM-(3,1)	⟨ q0, v7 ⟩; ⟨ q5, v2 ⟩	0.0	0.0
VeNoM-(2,1)	⟨ q0, v7 ⟩; ⟨ q5, v2 ⟩	0.0	0.0
VeNoM-(1,1)	⟨ q4, v7 ⟩, ⟨ q5, v5 ⟩; ⟨ q5, v2 ⟩	0.2	0.0

Figure 5: Performance comparison of different instances of VeNoM on sample target and query graphs.

pairs exhibit same statistical significance which gives it a chance to select ⟨ q1, v9 ⟩ over ⟨ q3, v9 ⟩. For such cases VeNoM-(3,1) may be more desirable than VeNoM-(2,2), as the latter also has a higher runtime. All the algorithms reported a partial match with 0.6 accuracy as the best match for the query graph 3 (Fig. 5(c)). However, the overall performance for VeNoM-(2,2) was better with more partial matches in top-3 answers, suggesting that it has the ability to find subgraphs in scenarios where other instances may fail.

3.4 Parameter Study

We study the effects of various graph parameters on the performance of the algorithms.

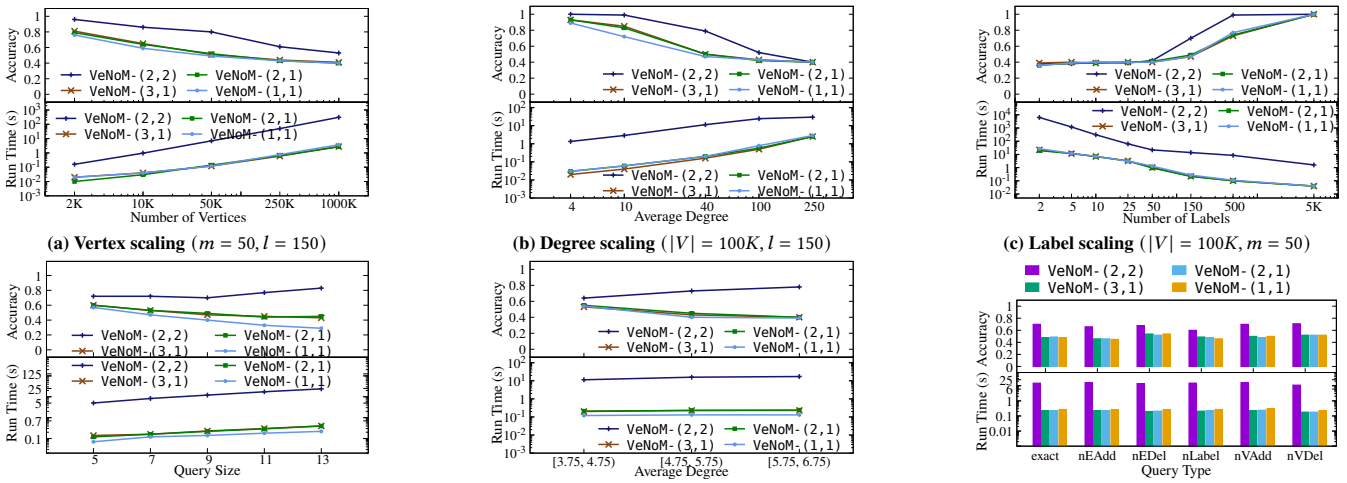
3.4.1 Graph Size. A linear increase in runtime was seen for all the algorithms with increase in the number of vertices (Fig. 6a). VeNoM-(2,2) is the most accurate as compared to its peers which is due to its ability to match neighbors 2-hops away. The performance of VeNoM-(2,1) and VeNoM-(3,1) remain similar with VeNoM-(1,1) being slightly inferior to them due to its lack of coverage. A higher runtime was also reported for VeNoM-(2,2) as with increase in the graph size, the number of candidate match comparison increases.

3.4.2 Average Graph Degree. The parameter m (in BA graphs) is directly proportional to the degree of the graph generated. For all the algorithms, a linear increase in runtime was seen with increase in the average degree of the graph (Fig. 6b). Noticeably, time taken by VeNoM-(3,1) was observed to be marginally lesser than that of VeNoM-(2,2). This is because the number of groups exhibited by VeNoM-(3,1) is lesser than VeNoM-(2,1), implying lesser number of groups to match, which results in a lower runtime. The lower runtime of VerSaChI than VeNoM-(2,2), in the previous section (§3.2) can be attributed to this factor as well. Once again, VeNoM-(2,2) achieved a higher accuracy than others while VeNoM-(1,1) achieved the lowest accuracy for lower graph degrees.

3.4.3 Label Set Size. As the size of the label set was increased, a rapid decrease in the runtime was observed (Fig. 6c). This is a direct consequence of the decrease in the number of candidate matches caused due to label diversity. Another outcome of this effect is an increase in the accuracy of the algorithms. All the algorithms showed a significant rise in accuracy from label set size 50 to 150. This shows that the algorithms are able to distinguish between vertex pairs better after a certain threshold of vertex-per-label ratio is crossed. Another contributing factor towards a lower accuracy, for smaller label sets, is the degree to label ratio, as discussed in Sec. 2.3. Although, VeNoM takes care of inversion in χ^2 values by replacing the target vertex degree with that of the query node, but a graph universe with a very small label set may still lead to value inversion. Once again, VeNoM-(2,2) reached the saturation accuracy faster by utilizing the *look-ahead* advantage for a better performance.

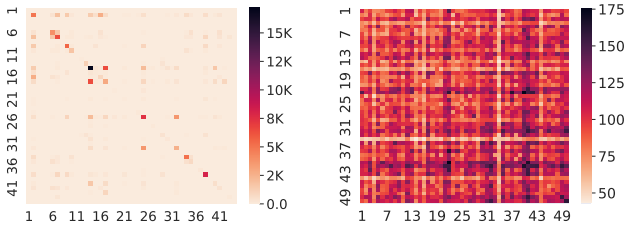
3.4.4 Query Size. With an increase in the number of vertices in the query graph, a near exponential increase is observed in runtime (Fig. 6d). This is because it increases the number of groups in a query. The effect of this is more pronounced in VeNoM-(2,2) due to the increase in depth of the neighborhood considered. However, VeNoM-(2,2) still outperforms the other VeNoM instances, which show a decrease in accuracy with an increase in the query size. VeNoM-(2,2) shows negligible performance change for small query sizes and a slightly higher accuracy for larger queries. This suggests that with higher number of connections, VeNoM-(2,2) is able to create more meaningful groups, making it more robust towards increase in query size. On the other hand, VeNoM-(1,1), with a single vertex per group, suffered heavily and achieved the least accuracy.

3.4.5 Query Degree. To better understand the effects of query degree, the queries of size 9 were binned into three buckets based on average degree of the query graphs: $[3.75, 4.75)$, $[4.75, 5.75)$ and $[5.75, 6.75)$. With increase in query complexity, no significant change in runtime was observed in any of the VeNoM instances (Fig. 6e). However, VeNoM-(2,2) exhibited a steady increase in accuracy and maintained a significant improvement over its counterparts. Meanwhile, the accuracy for VeNoM-(2,1), VeNoM-(3,1) and VeNoM-(1,1) dropped as the average degree of the query graph was increased. The rate of accuracy drop was slightly lower in VeNoM-(2,1) and VeNoM-(3,1), than in VeNoM-(1,1), suggesting slight comparative robustness. The increase in accuracy of VeNoM-(2,2) suggests that with more neighbors, non-trivial groups for a query node could be created, which captured the neighborhood similarity better. Observe that the ratio of the query degree to the label is <0.65 at all times.



(a) Vertex scaling ($m = 50, l = 150$)
(b) Degree scaling ($|V| = 100K, l = 150$)
(c) Label scaling ($|V| = 100K, m = 50$)
(d) Query size ($|V| = 100K, m = 50, l = 150$)
(e) Query degree ($|V| = 100K, m = 40, l = 150$)
(f) Noisy Queries ($|V| = 100K, m = 50, l = 150$)

Figure 6: Performance evaluation on Barabási-Albert graphs (runtime in log-scale)



(a) Human **(b) BA graph** ($|V| = 5K, m = 25, l = 50$)

Figure 7: Heat-map of edge-frequency between labels

3.4.6 Noisy Queries. In experiments with noisy queries, VeNoM-(2,2) performed significantly better than both VeNoM-(2,1) and VeNoM-(3,1), while VeNoM-(1,1) on an average reported a relatively lower accuracy (Fig. 6f). In general, accuracy for *nEDel* and *nVDel* was more than that for *nEAdd* and *nVAdd*, respectively. This is because in the event of deletion a perfect match is still guaranteed to exist, while the same cannot be when an edge or a vertex is added to the query. This aligns with the trend seen in the real-world graphs in Fig. 4. However, in contrast to the performance on real-world graph datasets, negligible accuracy drop was observed for *nLabel* query set in BA graphs. As hypothesized in §3.2, this is caused due to the absence of underlying semantic relations of vertex labels in synthetic graphs. In synthetic graphs, the labels were independent of each other and there was no semantic relationship between them. Thus, complete matches could be found for the *nLabel* queries as well. In real graphs, however, labels of vertices that have connections between them may not be random, and, the connections among labels may form a community. We further analyzed heat-maps Fig. 7 of frequency of connections between different labels for the dataset Human and a similar sized BA graph ($|V| = 5K, m = 25, l = 50$). It was observed that for the Human graph the connections were concentrated among a handful of labels, while in the BA graph the edges were evenly spread over all label combinations. This reinforced our theory of existence of underlying semantic relations among labels in real-world graphs.

3.5 Discussion

Overall, our experiments depicted the following trends:

- Incorporating *multi-hop information* offers significant improvements in accuracy, as shown by VeNoM-(2,2) and VerSaChI. This is potentially due to the extra neighborhood information available during comparison. However, this impacts the runtime negatively.
- The strategy of aggregating *multi-neighbor information* has limited improvement in accuracy. At one-hop level neighborhood match, a single edge based comparison is inadequate and increasing it to two neighboring nodes results in an improvement. However, there is no significant improvement when the capacity is increased to three neighboring vertices from two.
- Performance comparison between VeNoM-(2,2) and VerSaChI suggests that comparison of only two units at a time at two-hop level is insufficient and a larger view of the neighborhood is required for increased accuracy. Further, the number of groups potentially affects the run time of the algorithm, and decreases with larger group size for target graphs with a lower average degree. This causes a slight reduction in the runtime.

The above trade-offs can be considered while subgraph matching algorithms for application-specific requirements are designed.

4 Conclusions

VeNoM, a variant framework for VELSET, is extended to four different instances in a systematic manner, by modulating the depth and breadth of the neighborhood being considered for subgraph match. The experimental analysis shows the effect of both depth and breadth of neighborhood considered on the performance of different algorithms. We also demonstrated that under certain cases the null hypothesis for the statistical significance measure may not hold resulting in lower accuracy. Results with benchmark algorithms show that an integrated approach with breadth and depth of the neighborhood performs better.

In future, other algorithms can be analyzed similarly and a common framework for ASM can be identified.

References

[1] Shubhangi Agarwal, Sourav Dutta, and Arnab Bhattacharya. 2021. VerSaChI: Finding Statistically Significant Subgraph Matches using Chebyshev's Inequality. In *CIKM*. ACM, 2812–2816.

[2] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Rev. Mod. Phys.* 74 (Jan 2002), 47–97. Issue 1.

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

[4] Vincenzo Bonnici and Rosalba Giugno. 2016. On the variable ordering in Subgraph Isomorphism Algorithms. *IEEE/ACM transactions on computational biology and bioinformatics* 14, 1 (2016), 193–203.

[5] Chen Chen, Xifeng Yan, S Yu Philip, Jiawei Han, Dong Qing Zhang, and Xiaohui Gu. 2007. Towards Graph Containment Search and Indexing. In *PVLDB*. 926–937.

[6] Wei Chen, Jia Liu, Ziyang Chen, Xian Tang, and Kaiyu Li. 2018. PBSM: An efficient Top-K subgraph matching algorithm. *International Journal of Pattern Recognition and Artificial Intelligence* 32, 06 (2018), 1850020.

[7] James Cheng, Yiping Ke, Wilfred Ng, and An Lu. 2007. FG-Index: Towards Verification-free Query Processing on Graph Databases. In *SIGMOD*. 857–872.

[8] Stephen A. Cook. 1971. The Complexity of Theorem-Proving Procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing* (Shaker Heights, Ohio, USA) *(STOC '71)*. 8 pages.

[9] Luigi P Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. 2004. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE transactions on pattern analysis and machine intelligence* 26, 10 (2004), 1367–1372.

[10] Sourav Dutta, Pratik Nayek, and Arnab Bhattacharya. 2017. Neighbor-aware search for approximate labeled graph matching using the chi-square statistics. In *WWW'17*. 1281–1290.

[11] Myoungji Han, Hyunjoon Kim, Geonmo Gu, Kunsoo Park, and Wook-Shin Han. 2019. Efficient Subgraph Matching: Harmonizing dynamic programming, adaptive matching order, and failing set together. In *Proceedings of the 2019 International Conference on Management of Data*. 1429–1446.

[12] Wook-Shin Han, Jinsoo Lee, and Jeong-Hoon Lee. 2013. Turboiso: towards ultrafast and robust subgraph isomorphism search in large graph databases. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 337–348.

[13] Wook Shin Han, Jinsoo Lee, Minh Duc Pham, and Jeffrey Xu Yu. 2010. iGraph: A Framework for Comparisons of Disk-based Graph Indexing Techniques. *PVLDB* 3, 1-2 (2010), 449–459.

[14] Sen Hu, Lei Zou, Jeffrey Xu Yu, Haixun Wang, and Dongyan Zhao. 2018. Answering Natural Language Questions by Subgraph Matching over Knowledge Graphs. *IEEE TKDE* 30, 5 (2018), 824–837.

[15] Yi Jia, Jintao Zhang, and Jun Huan. 2011. An Efficient Graph-mining Method for Complicated and Noisy Data with Real-world Applications. *Knowledge and Information Systems* 28, 2 (2011), 423–447.

[16] Foteini Katsarou, Nikos Ntarmos, and Peter Triantafillou. 2015. Performance and Scalability of Indexed Subgraph Query Processing Methods. *Proc. VLDB Endow.* 8, 12 (Aug 2015), 12 pages.

[17] Arijit Khan, Yinghui Wu, Charu C. Aggarwal, and Xifeng Yan. 2013. NeMa: Fast Graph Search with Label Similarity. *Proc. VLDB Endow.* 6, 3 (Jan. 2013), 12 pages.

[18] Segla Kpodjedo, Philippe Galinier, and Giulio Antoniol. 2014. Using local similarity measures to efficiently address approximate graph matching. *Discrete Applied Mathematics* 164 (2014), 161–177.

[19] Jinsoo Lee, Wook-Shin Han, Romans Kasperovics, and Jeong-Hoon Lee. 2012. An In-Depth Comparison of Subgraph Isomorphism Algorithms in Graph Databases. *Proc. VLDB Endow.* 6, 2 (Dec 2012), 12 pages.

[20] Lihui Liu, Boxin Du, Hanghang Tong, et al. 2019. G-Finder: Approximate Attributed Subgraph Matching. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 513–522.

[21] Tinghuai Ma, Siyang Yu, Jie Cao, Yuan Tian, Abdullah Al-Dhelaan, and Mznah Al-Rodhaan. 2018. A comparative study of subgraph matching isomorphic methods in social networks. *IEEE Access* 6 (2018).

[22] Misael Mongiovi, Raffaele Di Natale, Rosalba Guigno, Alfredo Pulvirenti, Alfredo Ferro, and Roded Sharan. 2010. SIGMA: A Set-Cover-Based Inexact Graph Matching Algo. *Journal of Bioinformatics and Computational Biology* 8, 2 (2010), 199–218.

[23] Xuguang Ren and Junhu Wang. 2015. Exploiting vertex relationships in speeding up subgraph isomorphism over large graphs. *Proceedings of the VLDB Endowment* 8, 5 (2015), 617–628.

[24] Kaspar Riesen, Xiaoyi Jiang, and Horst Bunke. 2010. Exact and Inexact Graph Matching: Methodology and Applications. *Managing and Mining Graph Data* (2010), 217–247.

[25] Carlos R Rivero and Hasan M Jamil. 2017. Efficient and scalable labeled subgraph matching using SGMatch. *Knowledge and Information Systems* 51, 1 (2017), 61–87.

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

[27] Theyvaa Sangkaran, Azween Abdullah, and NZ Jhanjhi. 2020. Community Detection Based on Isomorphic Subgraph Analytics in Criminal Network. *IJCSNS* 20, 5 (2020), 94.

[28] Theyvaa Sangkaran, Azween Abdullah, NZ JhanJhi, and Mahadevan Supramaniam. 2019. Survey on isomorphic graph algorithms for graph analytics. *IJCSNS* 19, 1 (2019), 85–92.

[29] Ms Rachna Somkunwar and Vinod Moreshwar Vaze. 2017. Subgraph Isomorphism Algorithms for Matching Graphs: A Survey. *IJETT* 1, 1 (2017).

[30] Junshuai Song, Xiaoru Qu, Zehong Hu, Zhao Li, Jun Gao, and Ji Zhang. 2021. A subgraph-based knowledge reasoning method for collective fraud detection in E-commerce. *Neurocomputing* 461 (2021), 587–597.

[31] Shixuan Sun and Qiong Luo. 2019. Scaling Up Subgraph Query Processing with Efficient Subgraph Matching. In *ICDE*. 220–231.

[32] Shixuan Sun and Qiong Luo. 2020. In-Memory Subgraph Matching: An In-Depth Study. In *ICMD* (Portland, OR, USA) *(SIGMOD)*. 16 pages.

[33] Yunhao Sun, Guanyu Li, Jingjing Du, Bo Ning, and Heng Chen. 2022. A subgraph matching algorithm based on subgraph index for knowledge graph. *Frontiers of Computer Science* 16 (2022), 1–18.

[34] Yuanyuan Tian and Jignesh M Patel. 2008. Tale: A tool for approximate large graph matching. In *2008 IEEE 24th International Conference on Data Engineering*. IEEE, 963–972.

[35] Junchi Yan, Xu-Cheng Yin, Weiyao Lin, Cheng Deng, Hongyuan Zha, and Xiaokang Yang. 2016. A Short Survey of Recent Advances in Graph Matching. In *ICMR* (New York, New York, USA). 167–174.

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

[37] Xifeng Yan, Philip S. Yu, and Jiawei Han. 2005. Substructure Similarity Search in Graph Databases. In *SIGMOD*. 766–777.

[38] Shuo Zhang, Jianzhong Li, Hong Gao, and Zhaonian Zou. 2009. A Novel Approach for Efficient Supergraph Query Processing on Graph Databases. In *International Conference on Extending Database Technology (EDBT)*. 204–215.

[39] Shijie Zhang, Jiong Yang, and Wei Jin. 2010. SAPPER: Subgraph Indexing and Approximate Matching in Large Graphs. *PVLDB* 3, 1-2 (2010), 1185–1194.

[40] Peixiang Zhao and Jiawei Han. 2010. On graph query optimization in large networks. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 340–351.

[41] Lei Zou, Lei Chen, Jeffrey Xu Yu, and Yansheng Lu. 2008. A Novel Spectral Coding in a Large Graph Database. In *International Conference on Extending Database Technology (EDBT)*. 181–192.