Short Paper

Hierarchical Clustering for Property Graph Schema Discovery

Angela Bonifati
Lyon 1 University & LIRIS CNRS
France
angela.bonifati@univ-lyon1.fr

Stefania Dumbrava
ENSIIE & Inst. Polytech. de Paris
France
stefania.dumbrava@ensiie.fr

Nicolas Mir
ENSIIE
France
nicolas.mir@ensiie.fr

ABSTRACT

The property graph model is becoming increasingly popular among users and is currently employed by several open-source and commercial graph database systems. Although property graphs are widely adopted, there is a lack of understanding of their underlying schema structure. In particular, the *schema discovery* problem consists of extracting the schema concepts from a property graph. A property graph schema helps build a concise description of the data it represents, to make it more digestible for humans and interactive processes, as well as usable for query optimization purposes. In this paper, we address the property graph schema discovery problem and introduce the GMMSchema method based on hierarchical clustering using a Gaussian Mixture Model, which accounts for both label and property information on nodes. We experimentally analyze the accuracy and performance of GMMSchema, compared to those of its closest competitor, and showcase its superiority on several commonly used datasets, including real-world ones, such as the Covid19 knowledge graph, as well as the Fib25 and Mb6 NeuPrint graphs.

1 INTRODUCTION

Graphs are a natural abstraction for representing interconnected data. These have become increasingly pervasive in a wide variety of contexts, ranging from social networks to scientific repositories and the Semantic Web. To address the need to store, process, and analyze graph-shaped data, a novel type of NoSQL stores, namely *graph databases*, have been developed. Their most expressive underlying data model is the *property graph* one along with its variants [1, 3, 16], in which lists of properties can be attached to both the nodes and the edges of a directed, labeled, multi-graph.

Due to this rich formalism and to the fact that graph databases do not impose rigid schema constraints a priori, these can be readily employed to compactly capture complex graphs, integrating heterogeneous data sources. As graph datasets are large and rapidly evolving in practice, this flexibility is desirable for supporting scalable processing. However, the lack of a schema also has numerous drawbacks, as it makes data integration error-prone and hinders query optimization, meta-data management, and the extraction of type-based graph features needed in various machine learning applications.

To address this, we introduce a novel *schema discovery* method, leveraging a *hierarchical clustering* algorithm, based on *Gaussian Mixture Models*. To the best of our knowledge, ours is the first such approach to take into account labeling and property information simultaneously and, more generally, the first to perform schema discovery for property graphs using statistical methods. Schema inference for property graphs based on the MapReduce paradigm has been addressed in the past [14]. However, our method differs from previous work in that it is purely statistical and overcomes previous approaches in terms of both accuracy and performance.

The paper is organized as follows. Section 2 presents the related work in the area. Section 3 introduces preliminary concepts. Section 4 gives an overview of our method and illustrates the underlying GMMSchema algorithm. Section 5 evaluates the quality of the discovered schema and the overall performance of our approach, compared to those of its closest competitor.

2 RELATED WORK

Several works [10, 12] have investigated schema discovery for semi-structured data, to improve querying and data quality.

In the context of RDF datasets, popular methods use clustering techniques to group similar entities into clusters corresponding to distinct types. As such, similarity measures are defined to capture the extent to which entities share the same properties. In [5], *density-based clustering* (DBSCAN) is applied to extracted structural patterns, instead of the original dataset, as in [13]. However, as DBSCAN is parametric, the method exhibits relatively large variance in accuracy. This is improved upon by the *parameter-free* StaTIX method in [15], based on the Louvain community-detection algorithm. While these approaches are *hierarchical* and *soft* (i.e., support overlapping types), similarly to GMMSchema, our approach is tailored to the more complex model of property graphs and ultimately produces a schema graph. In the context of JSON datasets, a MapReduce method is introduced by [2]. This combines individual type inference (Map operation) and type reduction (Reduce operation) to merge types based on equivalence relations. However, this approach is not directly applicable to property graphs, does not explicitly support overlapping types, and only accounts for limited type hierarchies.

This work has been re-purposed for property graphs in [14]. This performs schema inference by considering either a label-based or a property-based heuristic for merging nodes. The former might lead to losing property co-occurrence information and, thus, to missing node types. The latter might result in inferring spurious types. To palliate the relative drawbacks of each approach, GMMSchema combines both label and property information into base types with respect to which data is clustered.

3 PRELIMINARIES

Assume that $\mathcal{L}$, $\mathcal{K}$, $\mathcal{D}$, $\mathcal{T}$ are pairwise disjoint sets of labels, property names (keys), values, and data types. For a set X, let $\mathcal{P}(X)$ be the set of all its finite subsets.

Definition 3.1 (Property Graph). A *property graph* $\mathcal{G}$ is a tuple $(\mathcal{V}, \mathcal{E}, \rho, \lambda, \sigma)$, where $\mathcal{V}$ and $\mathcal{E}$ are disjoint finite sets of vertices and edges, the incidence function $\rho : \mathcal{E} \to (\mathcal{V} \times \mathcal{V})$ is a total function associating each edge with a pair of nodes, the labeling function $\lambda : (\mathcal{V} \cup \mathcal{E}) \to \mathcal{P}(\mathcal{L})$ is a partial function[1] associating a vertex/edge with a set of labels, and $\sigma : (\mathcal{V} \cup \mathcal{E}) \times \mathcal{K} \to \mathcal{P}(\mathcal{N})$ is a partial function associating vertex/edges with properties and, for each property, assigning a set of values from $\mathcal{D}$.

[1] We denote its extension to element sets as $\bar{\lambda} : (\mathcal{P}(\mathcal{V}) \cup \mathcal{P}(\mathcal{E})) \to \mathcal{P}(\mathcal{L})$.

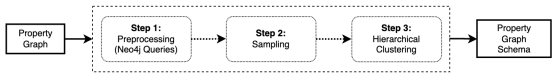


Figure 1: GMMSchema Method

Let us henceforth fix a property graph $\mathcal{G}$. For lack of a standard, we base our definition of a schema for $\mathcal{G}$ on [4].

Definition 3.2 (Base Types). The set of *base types* $\mathcal{BT}$ consists of *element types*. Each element type b is a 4-tuple (L, K, O, E_b), where $L \in \mathcal{L}$ is a set of labels, $K \in \mathcal{K}$ is a set of property names, $O \subseteq K$ is a subset of optional property names, and $E_b \subset \mathcal{BT}$ is the set of element types that b extends.

Definition 3.3 (Schema Graph). The *schema graph* of $\mathcal{G}$ is the tuple $(\mathcal{VT}, \mathcal{ET}, \rho_{\mathcal{T}}, \lambda_{\mathcal{T}}, \sigma_{\mathcal{T}})$, where $\mathcal{VT}$ and $\mathcal{ET}$ are disjoint finite sets of vertex and edge types, $\rho_{\mathcal{T}} : \mathcal{ET} \to (\mathcal{VT} \times \mathcal{VT})$ is a total function associating each edge with a pair of nodes, $\lambda_{\mathcal{T}} : (\mathcal{VT} \cup \mathcal{ET}) \to \mathcal{BT}$ is a total function associating a vertex/edge with a base type, and $\sigma_{\mathcal{T}} : (\mathcal{VT} \cup \mathcal{ET}) \times \mathcal{K} \to \mathcal{T}$ is function associating vertex/edges with properties and, for each property, assigning its corresponding data type.

Example 3.4 (Social Network Forums). Consider a social network graph, similar to those generated by the LDBC Social Network Benchmark [9]. This models a property graph instance, in which persons know each other and create post in various forums. Let us take the below node instance representing a post.

```
{'Post': {
'creationDate': 2015-06-24T12:50:35.556+01:002,
'locationIP': 42, 'browser' : 'Chrome',
'length' : 10, 'language' : 'lat.',
'content' : 'Lorem ipsum'}}
```

Listing 1: Dictionary storing a node instance.

Its base type would be $(\{Post\}, K, \{language, content\}, \emptyset)$, where $K = \{creationDate, locationIP, browser, length\}$ contains mandatory properties, $\{language, content\}$ - its optional ones, and the type has no parent, as in Figure 2.

Gaussian Mixture Models. Our method uses the *Gaussian Mixture Model (GMM)* [8], implementing an Expectation Maximization (EM) algorithm to fit a mixture of multi-variate Gaussian distributions. Given a set of node base types $\chi = \{x_1, \ldots, x_n\}$, we fit χ as a GMM parametrized by $\theta = [\theta_1, \ldots, \theta_k]$, where $\theta_i = (w_i, \mu_i, \Sigma_i)$ is the i-th mixture component. This represents a Gaussian $\mathcal{N}(\mu_i, \Sigma_i)$, with prior probability w_i, such that $\sum_{i=1}^{k} w_i = 1$. For every label L, we consider the set C of L-labeled nodes and run the GMM algorithm to fit a mixture of two normal distributions. We use the resulting model to split C-nodes into two clusters, depending on their similarity to the reference base type. The procedure is then re-iterated to discover further sub-clusters. At the end, we will have computed the set $\mathcal{H}$ of all discovered types and also constructed a dictionary $C_{\mathcal{H}}$, associating every discovered type to the clusters representing its sub-types. This can be directly used to build the schema graph, as will be illustrated.

4 HIERARCHICAL CLUSTERING

In this section, we present the GMMSchema algorithm for property graph schema discovery. It partitions the dataset, using unsupervised clustering, such that each computed cluster corresponds to a node type, determined by an unique combination of labels and properties. The method is outlined in Figure 1.

Algorithm 1 Schema Graph Computation

input : $\mathcal{G}$ - property graph, n - threshold
output : $\mathcal{H}$– node type clusters
function GMM_SCHEMA($\mathcal{G}, n$)
  $\mathcal{L}_{\mathcal{G}} \leftarrow \bigcup_{v \in \mathcal{V}} \lambda(v)$ ▷ Set of node labels
  $\mathcal{H} \leftarrow \emptyset$ ▷ Initial set of all discovered clusters
  $C_{\mathcal{H}} \leftarrow \emptyset$ ▷ Initial (base type, sub-clusters) dictionary
  for $L \in \mathcal{L}_{\mathcal{G}}$ **do** ▷ Iterate through node labels
    $C \leftarrow \{v \in \mathcal{V} \mid L \in \lambda(v)\}$
    $\mathcal{H}, C_{\mathcal{H}} \leftarrow$ HIERARCHICAL_CLUSTERING($C, \mathcal{H}, C_{\mathcal{H}}, n$)
  return $\mathcal{H}, C_{\mathcal{H}}$

First, we *pre-process* the graph instance $\mathcal{G}$ and collect statistics regarding property key frequency, per node label. As described in Algorithm 1, for each label L in the set of all node labels $\mathcal{L}_{\mathcal{G}}$, we compute its corresponding node types and sub-types. Hence, we call the hierarchical inference algorithm on the set C of all nodes whose set of labels contain L. For each such node set C, the HIERARCHICAL_CLUSTERING algorithm proceeds iteratively, by first constructing a reference base type b_{ref}. This intuitively represents the most general type that all nodes in C extend, i.e., the parent cluster for all sub-clusters in C. To build it, we consider the the set of all cluster labels $\mathcal{L}_C$, as well as the n most frequent properties in C. Next, we separate the nodes in C in two clusters, based on how similar their base type is to b_{ref}. To this end, for each node, we use the Dice metric to compare the set of its labels and properties to that corresponding to all labels and property keys in $\mathcal{L}_C$ and K_C, respectively. These similarities are then regrouped into a feature vector d. By fitting a GMM model using d and estimating its parameters with the EM algorithm, we obtain the mixture components θ_1 and θ_2. The prediction step allows us to classify the considered data samples and determine whether they fit b_{ref}. This amounts to 2-clustering and splitting C into C_L^1 and C_L^2. If the property keys of C_L^1 and C_L^2 overlap, we assign the intersection set to b_{ref}. We update the hierarchy dictionary $C_{\mathcal{H}}$ with the information that the base types of C_L^1 and C_L^2 extend b_{ref}. For each sub-cluster, we re-iterate the procedure, recompute a reference node, and use it to discover new sub-types.

Algorithm 2 Cluster Hierarchy Computation

input : C - cluster, $\mathcal{H}$ - set of all discovered clusters, $C_{\mathcal{H}}$ - (base type, sub-clusters) dictionary, n - threshold
output : $\mathcal{H}$– updated set of discovered clusters
function HIERARCHICAL_CLUSTERING($C, \mathcal{H}, C_{\mathcal{H}}, n$)
  $\mathcal{L}_C \leftarrow \bar{\lambda}(C)$ ▷ Current cluster node labels
  $K_C \leftarrow$ set of the n most frequent node properties in C
  $b_{ref} \leftarrow (\mathcal{L}_C, K_C, \emptyset, \emptyset)$ ▷ Reference base type for C nodes
  for $v \in C$ **do** ▷ Iterate through cluster nodes
    $b \leftarrow \lambda_{\mathcal{T}}(v)$ ▷ Store element type of v
    $d(v) \leftarrow Dice(b, b_{ref})$ ▷ Store similarity metric
  $\vec{\theta} \leftarrow \text{fit}(d)$, where $\vec{\theta} = [\theta_1, \theta_2]$
  $C_L^1, C_L^2 \leftarrow \text{predict}(\vec{\theta})$
  if $\bar{\lambda}(C_L^1) \cap \bar{\lambda}(C_L^2) \neq \emptyset$ **then**
    $b_{ref} \leftarrow (\mathcal{L}_C, \bar{\lambda}(C_L^1) \cap \bar{\lambda}(C_L^2), \emptyset, \emptyset)$
  if $j \in \{1, 2\} \wedge C_L^j \notin \mathcal{H}$ **then**
    $C_{\mathcal{H}}[b_{ref}] \leftarrow C_{\mathcal{H}}[b_{ref}] \cup C_L^j$
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{C_L^1, C_L^2\}$
    $\mathcal{H}, C_{\mathcal{H}} \leftarrow$ HIERARCHICAL_CLUSTERING($C_L^j, \mathcal{H}, C_{\mathcal{H}}, n$)
  return $\mathcal{H}, C_{\mathcal{H}}$

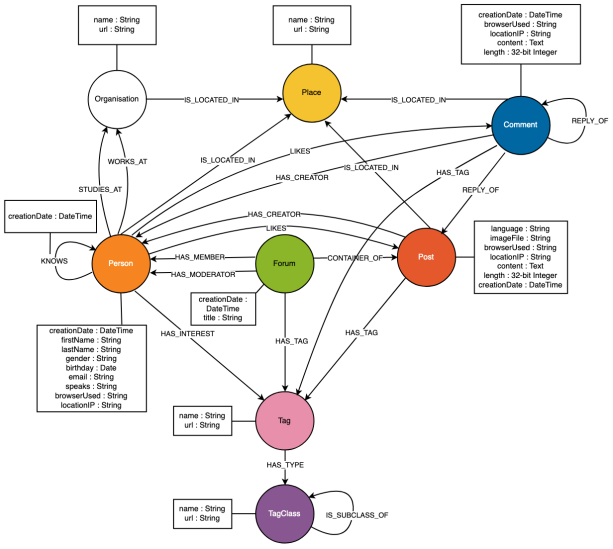


Figure 2: Ground-Truth Schema for LDBC

The runtime efficiency of our algorithm could be improved by fitting multiple Gaussians during one iteration, instead of just performing 2-clustering. Optimization issues regarding this hyperparameter are left for future work.

Example 4.1 (Social Network - Schema Graph). We focus on the discovery of the sub-types corresponding to nodes labeled *Post*. Hence, we run the GMM_Schema algorithm over the graph instance, taking the value $n = 2$, for example[2]. The corresponding reference node is $b_{ref} = (\{Post\}, \{creationDate, locationIP\}, \emptyset, \emptyset)$, as *creationDate* and *locationIP* are common to all cluster nodes. Using the GMM model, we classify *Post* nodes into 2-clusters. Their property key intersection gives us the base type of the parent node $b = (\{Post\}, K, \emptyset, \emptyset)$, where

$$K = \{creationDate, locationIP, browser, length\}.$$

Repeating the procedure in each sub-cluster does not infer new types, as all nodes in each share the same properties. Their reference nodes are: $b_1 = (\{Post\}, K, \{language, content\}, \{b\})$ and $b_2 = (\{Post\}, K, \{imageFile\}, \{b\})$, each representing a new subtype, i.e., *Post*1 and *Post*2 in Figure 3. Each of these newly discovered node types will then have the same connectivity characteristics, i.e., attached in/out-going edge types, as their parent node, *Post*. The discovered schema graph will additionally include two edge subtype edges from *Post*1 and, respectively, *Post*2 to *Post*.

Note that, for ease of exposition, nodes in the running example have single labels only. Our approach, however, covers multiple labels, thus supporting the full expressiveness of the property graph model.

5 EXPERIMENTAL EVALUATION

Our schema inference method is implemented in Python 3 and is compatible with Neo4j 4.3.4. The code base is readily available online [3]. All experiments were performed on an Openstack Virtual Machine with 12 2GHz 64-bits Intel Xeon CPUs, 62 GB of memory, and a 1.5 TB hard drive.

[2]Note that the value of n does not impact the overall precision, but only the convergence rate of the method.

[3]https://github.com/naussicaa/pg-schemainference

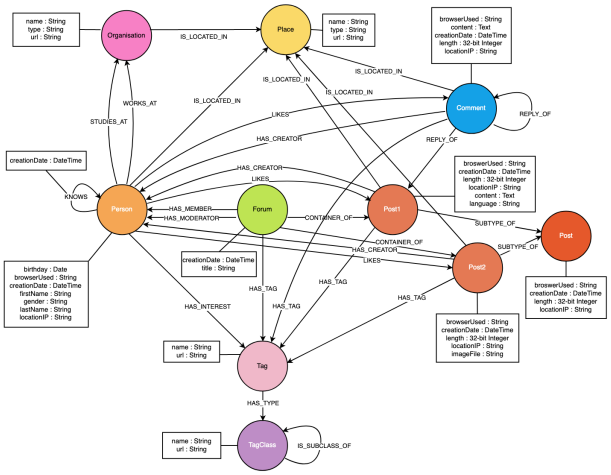


Figure 3: GMMSchema for LDBC

Dataset	Nodes	Edges	Node Labels	Edge Labels	Unlabeled Nodes
LDBC	1,577,397	8,179,418	7	14	0
Mb6	486,267	961,571	10	3	0
Fib25	802,473	1,625,428	10	3	0
Covid19	36,025,729	59,768,373	121	168	474

Figure 4: Dataset characteristics prior to schema discovery.

5.1 Datasets and Metrics

Datasets. We experimentally analyze our approach on several datasets (see Figure 4), described as follows. The LDBC Social Network Benchmark (**LDBC**) [9] is a synthetic graph, corresponding to the topology of a realistic social network and comes together with a ground-truth schema, capturing single-labeled nodes and type hierarchies. The **Mb6** [19] and **Fib25** [18] datasets model NeuPrint connectome data and correspond to the mushroom body and, respectively, medulla neural networks in the fruit fly brain. Both these datasets also contain ground-truth schema with diverse properties, as well as multi-labeled nodes. Finally, **Covid19** corresponds to the Covid-19 Knowledge Graph [7], which integrates data from the Genotype Tissue Expression (GTEx) the Covid-19 Disease Map (Cord-19), and the bibliographical ArXiv, BioRxiv, MedRxiv, PubMed, and PubMed Central repositories. The graph has no ground-truth schema and is continuously evolving. Hence, the reported results consider the August 2021 version.

Metrics. To assess the quality of the discovered schema, we employ several metrics, as follows. The Adjusted Random Index (**Rand Index**) measures the similarity between two clusterings, considering all sample pairs and counting those assigned in the same/different clusters in the predicted and true clusterings. The Adjusted Mutual Index (**AMI**) also measures the similarity between two clusterings, but accounts for potentially unbalanced clusters in the ground truth clustering. **Precision** captures the proportion of identified types part of the ground truth, while **recall** measures the performance of binary classification. The **F1-score** provides an average of precision and sensitivity.

5.2 Evaluation and Discussion

We now present and discuss the results of evaluating GMMSchema on the **LDBC**, **Mb6**, **Fib25**, and **Covid19** datasets.

Schema Quality. As a first indicator of the quality of our schema discovery method, we first analyze the changes in dataset characteristics it generates. Looking at Figure 4 and 5, we notice that,

Dataset	Node Types	Edge Types	Subtype Edges	Hierarchy Depth
LDBC	17	36	9	2
Mb6	19	27	14	4
Fib25	26	106	21	6

Figure 5: Dataset characteristics with GMMSchema discovery.

Dataset	Node Types	Edge Types	Subtype Edges	Hierarchy Depth
LDBC	17	72	51	5
Mb6	68	795	786	9
Fib25	47	427	418	8

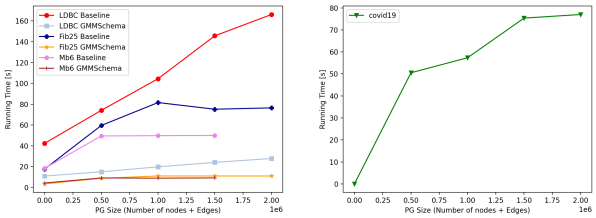
Figure 6: Dataset characteristics (property-based inference).

Dataset	Node Types	Edge Types	Subtype Edges	Hierarchy Depth
LDBC	7	21	0	0
Mb6	5	10	1	1
Fib25	5	10	1	1

Figure 7: Dataset characteristics (label-based inference).

Dataset	Rand Index	AMI	Precision	Recall	F1-score
LDBC	0,96	0,91	1,0	1,0	1,0
Mb6	0,79	0,49	1,0	1,0	1,0
Fib25	0,75	0,41	1,0	1,0	1,0
Covid19	0,94	0,71	NaN	NaN	NaN

Figure 8: GMMSchema clustering quality estimates.



(a) LDBC, Fib25, Mb6 **(b) Covid19**

Figure 9: GMMSchema vs. Baseline total avg. runtimes.

Dataset	Pre-processing	Sampling	Clustering	Total	Baseline
LDBC	22,15	3,27	22,02	47,44	830
Mb6	1,03	0,55	2,95	4,53	48
Fib25	5,33	1,03	5,83	12,19	76
Covid19	44,12	132	204	380,12	266

Figure 10: Component-wise and total runtimes (s) for GMMSchema and the baseline on LDBC, Mb6, Fib25, Covid19.

on average, 2-3 types are discovered per node label, while in the case of edge labels, GMMSchema leads to up to 3 orders of magnitude more edge types (for the **fib25** dataset). This is due to the high hierarchy depth of the dataset, which leads to the creation of ≈ 20 new subtype edges. Comparing these numbers to those corresponding to the property-oriented schema inference method in [14] (Figure 6), we observe that, the baseline can infer up to 3 times more node types, up to 3 orders of magnitude more edge types, up to 7 orders of magnitude more subtype edges (for the **mb6** dataset), and leads to a hierarchies that can be double the depth of those discovered by GMMSchema. However, looking at the perfect accuracy of our schema discovery method, as given by the precision, recall, and F1-score in Figure 8, we can conclude that many of the inferred nodes are actually redundant and that GMMSchema helps address this by also leveraging the information in node labels. Also, comparing with the numbers reported by the label-oriented heuristic in [14] (Figure 7), we can observe GMMSchema also addresses the missing types due to only considering node label information, by also taking into account their corresponding properties. Hence, our schema discovery method achieves a good balance between these two approaches.

In order to assess the quality of our underlying hierarchical clustering method based on Gaussian Mixture models, we compute its corresponding **Rand Index** and **Adjusted Mutual Information (AMI)** scores. To this end, we consider the HDBSCAN [6] clustering algorithm as our baseline, since it is the closest to our approach. As shown in Figure 8, we notice that the two methods agree the most on the largest datasets (**Covid19** and **LDBC**) under both metrics. Also, note that the **AMI** score is better suited for hard clustering (i.e., assigning a given node to a single cluster), whereas both GMMSchema and HDBSCAN perform soft clustering. Moreover, in the LDBC and Covid19 datasets, it is easier to determine the specific cluster corresponding to a node, as the hierarchy is very shallow and there is little overlap in the labels/properties representing type. Conversely, for **Mb6** and **Fib25**, as there are many such overlaps and as the datasets are smaller, being forced to choose a single "right" cluster for a node is harder and more penalizing.

Scalability. Figure 9a captures the overall runtime performance of our GMMSchema method, compared to that of the baseline in [14]. When benchmarking its performance for graphs of up to 2M nodes and edges, GMMSchema exhibits better scalability. For **Mb6**, GMMSchema is up to 5 times more efficient, while for LDBC and **Fib25**, it is up to 8 times faster. One explanation for this is that, unlike in our approach, the pre-processing stage of the baseline method requires writing all nodes and edges in a JSON file. Given that **Covid19** is rapidly evolving and that the version we considered in our experiments does not match that used by the baseline method, we plotted the total average runtime performance of GMMSchema on this dataset separately (Figure 9b). As can be seen, despite the high heterogeneity of the dataset (121 node labels and 168 edge labels), the performance of GMMSchema is up to twice faster than that displayed by the baseline on the much smaller and less diverse LDBC dataset.

Looking at the performance breakdown for the pre-processing, sampling, and clustering components of GMMSchema (Figure 10), we notice that the most expensive step is generally the clustering one. This is to be expected, as the core HIERARCHICAL_CLUSTERING function can call itself recursively. Each such time, a Gaussian Mixture Model has to be retrained, in order to split the considered dataset portion into two clusters, by deciding, for each node, whether its base type matches that of the reference base type.

6 CONCLUSION AND FUTURE WORK

This paper introduces the novel GMMSchema algorithm for schema discovery in the property graph setting. To the best of our knowledge, this is the first approach that targets property graphs and that simultaneously takes into account both the label and property information on nodes. This addresses the limitations with respect to incomplete/spurious node inference of its closest competitor, whose method is based on MapReduce inference. Preliminary experiments show the superiority of our approach, in terms of accuracy and performance, and highlight the promise of employing statistical methods for schema discovery.

Future extensions consists of further optimizing the approach, inferring cardinality information, and supporting incremental maintenance to preserve schema validity under updates. Also, integrating topological information through graph embeddings could benefit scalability. Finally, investigating how data analysis methods, such as formal concept analysis [17], or recent grammatical inference methods [11] can be applied to property graphs and comparing them to our work opens interesting perspectives.

REFERENCES

[1] Renzo Angles. 2018. The Property Graph Database Model. In *AMW (CEUR Workshop Proceedings)*, Vol. 2100. CEUR-WS.org.

[2] Mohamed Amine Baazizi, Dario Colazzo, Giorgio Ghelli, and Carlo Sartiani. 2019. Parametric Schema Inference for Massive JSON Datasets. *VLDB J.* 28, 4 (2019), 497–521.

[3] Angela Bonifati, George H. L. Fletcher, Hannes Voigt, and Nikolay Yakovets. 2018. *Querying Graphs.* Morgan & Claypool Publishers.

[4] Angela Bonifati, Peter Furniss, Alastair Green, Russ Harmer, Eugenia Oshurko, and Hannes Voigt. 2019. Schema Validation and Evolution for Graph Databases. In *ER (Lecture Notes in Computer Science)*, Vol. 11788. Springer, 448–456.

[5] Redouane Bouhamoum, Kenza Kellou-Menouer, Stéphane Lopes, and Zoubida Kedad. 2018. Scaling Up Schema Discovery for RDF Datasets. In *ICDE Workshops*. IEEE Computer Society, 84–89.

[6] Ricardo J. G. B. Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-Based Clustering Based on Hierarchical Density Estimates. In *PAKDD (2) (Lecture Notes in Computer Science)*, Vol. 7819. Springer, 160–172.

[7] CovidGraph. 2021. COVID-19 Knowledge Graph. https://covidgraph.org/.

[8] Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)* 39, 1 (1977), 1–22.

[9] Orri Erling, Alex Averbuch, Josep Lluís Larriba-Pey, Hassan Chafi, Andrey Gubichev, Arnau Prat-Pérez, Minh-Duc Pham, and Peter A. Boncz. 2015. The LDBC Social Network Benchmark: Interactive Workload. In *SIGMOD Conference*. ACM, 619–630.

[10] Silvio Normey Gómez, Lorena Etcheverry, Adriana Marotta, and Mariano P. Consens. 2018. Findings from Two Decades of Research on Schema Discovery using a Systematic Literature Review. In *AMW (CEUR Workshop Proceedings)*, Vol. 2100. CEUR-WS.org.

[11] Benoît Groz, Aurélien Lemay, Slawek Staworko, and Piotr Wieczorek. 2022. Inference of Shape Graphs for Graph Databases. In *ICDT (LIPIcs)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 7:1–7:20.

[12] Kenza Kellou-Menouer, Nikolaos Kardoulakis, Georgia Troullinou, Zoubida Kedad, Dimitris Plexousakis, and Haridimos Kondylakis. 2021. A Survey on Semantic Schema Discovery. *VLDB J.* (2021).

[13] Kenza Kellou-Menouer and Zoubida Kedad. 2015. Schema Discovery in RDF Data Sources. In *ER (Lecture Notes in Computer Science)*, Vol. 9381. Springer, 481–495.

[14] Hanâ Lbath, Angela Bonifati, and Russ Harmer. 2021. Schema Inference for Property Graphs. In *EDBT*. OpenProceedings.org, 499–504.

[15] Artem Lutov, Soheil Roshankish, Mourad Khayati, and Philippe Cudré-Mauroux. 2018. StaTIX - Statistical Type Inference on Linked Data. In *IEEE BigData*. IEEE, 2253–2262.

[16] Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Renzo Angles, Walid G. Aref, Marcelo Arenas, Maciej Besta, Peter A. Boncz, Khuzaima Daudjee, Emanuele Della Valle, Stefania Dumbrava, Olaf Hartig, Bernhard Haslhofer, Tim Hegeman, Jan Hidders, Katja Hose, Adriana Iamnitchi, Vasiliki Kalavri, Hugo Kapp, Wim Martens, M. Tamer Özsu, Eric Peukert, Stefan Plantikow, Mohamed Ragab, Matei Ripeanu, Semih Salihoglu, Christian Schulz, Petra Selmer, Juan F. Sequeda, Joshua Shinavier, Gábor Szárnyas, Riccardo Tommasini, Antonino Tumeo, Alexandru Uta, Ana Lucia Varbanescu, Hsiang-Yun Wu, Nikolay Yakovets, Da Yan, and Eiko Yoneki. 2021. The Future is Big Graphs: a Community View on Graph Processing Systems. *Commun. ACM* 64, 9 (2021), 62–71.

[17] Rudolf Wille. 2005. Formal Concept Analysis as Mathematical Theory of Concepts and Concept Hierarchies. In *Formal Concept Analysis (Lecture Notes in Computer Science)*, Vol. 3626. Springer, 1–33.

[18] Shin ya Takemura and et al. 2015. Synaptic circuits and their variations within different columns in the visual system of Drosophila. *Proceedings of the National Academy of Sciences* 112 (2015), 13711 – 13716.

[19] Shin ya Takemura and et al. 2017. A Connectome of a Learning and Memory Center in the Adult Drosophila Brain. *eLife* 6 (2017).