DiscoPG: Property Graph Schema Discovery and Exploration

Angela Bonifati
Lyon 1 University & LIRIS CNRS

Stefania Dumbrava
ENSIIE & Inst. Polytech. de Paris

Emile Martinez
ENS Lyon

Fatemeh Ghasemi
ENS Lyon

Malo Jaffré
ENS Lyon

Pacôme Luton
ENS Lyon

Thomas Pickles
ENS Lyon

ABSTRACT

Property graphs are becoming pervasive in a variety of graph processing applications using interconnected data. They allow to encode multi-labeled nodes and edges, as well as their properties, represented as key/value pairs. Although property graphs are widely used in several open-source and commercial graph databases, they lack a schema definition, unlike their relational counterparts. The property graph schema discovery problem consists of extracting its underlying schema concepts and types. We showcase DiscoPG, a system for efficiently and accurately discovering and exploring property graph schemas. To this end, it leverages hierarchical clustering using a Gaussian Mixture Model, which accounts for both node labels and properties. DiscoPG allows the user to perform schema discovery for both static and dynamic graph datasets. In particular, suitable visualization layouts, along with dedicated dashboards, enable the user perception of the static and dynamic inferred schema on the node clusters, as well as the differences in runtimes and clustering quality. To the best of our knowledge, DiscoPG is the first system to tackle the property graph schema discovery problem. As such, it supports the insightful exploration of the graph schema components and their evolving behavior, while revealing the underpinnings of the clustering-based discovery process.

PVLDB Reference Format:
Angela Bonifati, Stefania Dumbrava, Emile Martinez, Fatemeh Ghasemi, Malo Jaffré, Pacôme Luton, and Thomas Pickles. DiscoPG: Property Graph Schema Discovery and Exploration. PVLDB, 14(1): , 2020.
doi:

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at URL_TO_YOUR_ARTIFACTS.

1 INTRODUCTION

Graphs are natural abstractions for representing interconnected data. They have been adopted in a large array of applications, ranging from social networks to scientific datasets, fraud detection, recommendation systems and the Semantic Web. Their most expressive underlying data model is the *property graph* one, along with its numerous variants [1, 3, 6], in which lists of properties are attached to both the nodes and the edges of a directed, labeled, multi-graph. Property graphs underlie several open-source and commercial systems, out of which graph databases are the most prominent. While these do not impose a priori rigid schema constraints, the rich property graph model enables them to compactly capture complex patterns, while remaining digestible and self-explainable for end users [5]. However, the lack of a schema hinders several applications, such as data exploration, query formulation and optimization, meta-data management, and the extraction of type-based graph features for various machine learning pipelines. Moreover, property graph schemas are considered as relevant abstractions needed in future graph processing systems [6].

To address this, in DiscoPG we tackle the discovery of property graph schemas, leveraging a *hierarchical clustering* algorithm, based on *Gaussian Mixture Models*. Compared to previous schema discovery methods [4], DiscoPG allows to account for both labeling and property information and to build node clusters, reflecting the underlying typing hierarchy of the base dataset. Ours is a purely statistical approach (improving and extending that in [2]), which we have experimentally shown to provide better accuracy and performance than previous analytical methods [4].

DiscoPG allows users to engage in the following scenarios:

- visualization and exploration of the discovered schema, with novel cluster-centered layouts that showcase the size of the discovered node types, their corresponding labels and properties, as well as those of their connecting edges.
- interactive inspection of a schema dashboard, displaying the performance of the underlying hierarchical clustering algorithms and the quality of the obtained schema, across each iteration of the discovery process; this allows analyzing the run-time behavior of the various methods and underpinnings of the graph clustering process.
- analysis of the property graph schema evolution, with custom rendering showing the parts affected by modifications to the base graph instance. Both incremental and recomputation-based schema evolution scenarios are considered.

In our demonstration, in order to showcase DiscoPG, we leverage both real-world datasets and benchmarking datasets, e.g. covidgraph.org and LDBC []. These datasets have different characteristics and range from simple to complex schemas.
The complete code base of DiscoPG is available on Github[1].

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:

[1] https://github.com/PI-Clustering/code

2 SYSTEM OVERVIEW

Given a graph dataset $\mathcal{G}$, DiscoPG implements novel static and dynamic algorithms for discovering the underlying schema of $\mathcal{G}$.

2.1 Schema Discovery Algorithms

2.1.1 Hierarchical Clustering Schema Discovery. In the static case, DiscoPG determines the underlying schema structure of $\mathcal{G}$, by applying the custom `GMM-S` algorithm, which is an optimized version of the GMMSchema method in [2]. `GMM-S` partitions $\mathcal{G}$ using a hierarchical clustering approach, based on fitting a Gaussian Mixture Model (GMM), combining label and property information. Each discovered cluster corresponds to a node type, characterised by an unique combination of labels and properties.

`GMM-S` first collects the set of node labels $\mathcal{L}_{\mathcal{G}}$, recording the number of label occurrences. Next, it processes, in descending order of label frequency, each corresponding set of nodes. For each label L in $\mathcal{L}_{\mathcal{G}}$, `GMM-S` considers the set C of all nodes with label L, and computes the sub-types further determined by their properties.

To this end, it applies an iterative procedure, which outputs a dictionary $C_{\mathcal{H}}$, recording the typing hierarchy induced by the GMM clustering algorithm. The method starts by constructing a reference base type b_{ref} for C, which contains all its node labels, as well as its most frequent properties. Intuitively, this represents the most general type extended by all nodes in C, i.e., the parent cluster for all sub-clusters in C. `GMM-S` then computes a feature vector d, containing the similarity scores between the base types of each node in C and b_{ref}. This is used to fit a GMM model, estimate its parameters with the EM algorithm, and obtain the mixture components $\{\theta_1, \ldots, \theta_n\}$, where n is user-defined parameter, set by default to 2. In the prediction step, nodes are classified into sub-clusters $C_L^1, \ldots, C_L^n$, based on their underlying similarity to b_{ref}.

If there is an overlap of property keys between all sub-clusters, this intersection set is assigned to b_{ref}, as it represents the "core" properties of C. $C_{\mathcal{H}}$ is then updated to record that the base types of the sub-clusters extend b_{ref}. For each sub-cluster, the procedure is re-iterated, as new reference nodes are computed and used to discover new sub-types. The key feature of the schema discovery algorithms of DiscoPG, allowing to capture both node labels and properties at the same time, is that they enable users to obtain more accurate schemas than previous approaches [4].

2.1.2 Schema Discovery for Evolving Graphs. DiscoPG also supports schema discovery in the dynamic case, wherein the dataset $\mathcal{G}$ evolves, upon batch inserting a set of nodes Δ. We outline the novel `I-GMM-D` and `GMM-D` methods, tailored to this end.

The `I-GMM-D` approach takes as input the hierarchy dictionary $C_{\mathcal{H}}$, computed by `GMM-S` in the static case, as well as the updates Δ. Note that all the sub-clusters in $C_{\mathcal{H}}$ correspond to the node types of $\mathcal{G}$'s discovered schema structure. `I-GMM-D` then constructs a similarity vector containing the similarity scores between the base type of each node in Δ and the base reference types of each sub-cluster in $C_{\mathcal{H}}$. Each node in Δ is then assigned to the sub-cluster in $C_{\mathcal{H}}$ with respect to which it has the highest similarity score. If any of the sub-clusters incurs updates accounting for more than a given percentage of their initial size (set as a threshold), full schema recomputation is triggered, as the updates are deemed to potentially affect the structure of the sub-cluster's base type.

The `GMM-D` method takes as input the graph obtained by updating $\mathcal{G}$ with Δ, the desired number n of types to be discovered at each iteration step, and outputs the schema reflecting the evolution in the content of $\mathcal{G}$. The algorithm is an optimized version of `GMM-S`, which uses memoization to avoid unnecessary recursive calls in the sub-clusters that remain unchanged. Specifically, consider the `GMM-S` prediction step, which classifies the nodes of a parent cluster into one of the base types of the n sub-clusters from the previous iteration. While `GMM-S` proceeds to recursively call itself in each of these sub-clusters after this step, `GMM-D` only does so in the sub-clusters to which nodes were currently assigned. To the best of our knowledge, incremental schema discovery for property graphs as enabled by DiscoPG has not been showcased before.

2.2 System Architecture

As depicted in Figure 1, the architecture of the DiscoPG system integrates three modules. These support *discovering* and *exploring* dynamic property graph schemas, as well as *inspecting* and *logging* the performance of the underlying algorithms and the quality of the produced results, through a dashboard. We present each below.

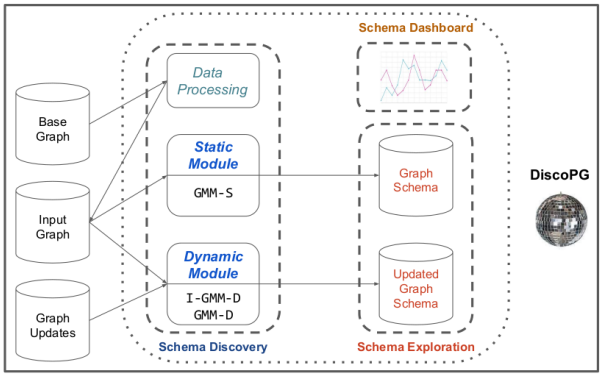

Figure 1: DiscoPG System Architecture

- **Schema Discovery.** The first functionality of the module is that it allows to *preprocess* a given graph dataset and standardise its labels and property keys, by removing potential syntactical inconsistencies and typos. It then applies the `GMM-S` algorithm to *discover* the underlying schema structure of the graph obtained as output during the initial step. Finally, upon receiving updates from the user and a preference regarding whether to apply an *incremental* (`I-GMM-D`) or a *memoization-based recomputation* (`GMM-D`) approach, it computes a novel schema, reflecting the changes.
- **Schema Exploration.** The module allows users to *navigate* the previously produced schema graph. In particular, they can examine the labels and properties associated to both its nodes and edges. Moreover, they can immediately visually *grasp* the proportion of node instances corresponding to each node type, as these are reflected by the depicted cluster sizes. In the dynamic case, the individual impact of the changes to the relevant clusters can also be apprehended through their custom color coding.
- **Schema Dashboard.** Finally, users can *inspect* the performance of DiscoPG's algorithms, as well as the quality

of its produced schemas. As an additional functionality, providing a wider view of the computed metrics, the module enables users to *log* the results obtained with various parameter configurations, across several datasets.

3 DEMONSTRATION OVERVIEW

The audience will be able to use DiscoPG to execute the following steps. In the static setting, one can: load and visualize property graphs, configure the parameters of our `GMM-S` schema discovery method, as well as log and analyze its performance, in terms of runtime and clustering quality, across various datasets. In the dynamic setting, one can modify the initial graph instances, by inserting real or synthetic nodes. Then, one can analyze the updated schemas produced with our dynamic algorithms. These are `I-GMM-D` (which incrementally adjusts the clusters computed in the static case) and `GMM-D` (which dynamically recomputes the node clusters). Finally, one can explore the discovered schemas and visualize their structure: relative cluster sizes, associated labels, properties, and interconnections, as well as the impact incurred due to updates.

3.1 Schema Discovery

Static Case. Users can access DiscoPG through a GUI. The main panel allows users to configure parameters for performing schema discovery using the `GMM-S` algorithm. As depicted in Figure 2, a graph instance can first be loaded, by selecting one from a menu containing the LDBC, NeuPrint's Fib25, and the Covid19 graphs. Note than further datasets can also be added to the system. Once the desired dataset is chosen, DiscoPG supports schema discovery for either the entirety of the dataset or for a custom fraction that can be specified in the field `"Percentage of nodes to consider"`. Note that, in the latter case, the remaining percentage will be used for dynamic case. Additionally, users can specify the number of sub-clusters to be discovered at each iteration (corresponding to the number of fitted Gaussians, as explained in Section 2.1.1). Users can choose whether the discovered schema should include the original edge labels or only the computed subtype relationships, as well as whether to record the evaluation in a log, for further reference. Finally, if a previous schema discovery algorithm has already been run on the chosen dataset, they have the option of considering the most recent result. This can be then used for further processing with dynamic schema discovery algorithms detailed next.

Dynamic Case. Upon following the above steps, a schema is computed and made available for exploration (see Section 3.3). To support the rapid evolution of real-world datasets, DiscoPG allows users to modify the graph instance used for static schema discovery. Configuring the parameters for the dynamic scenario is possible through a dedicated panel (see Figure 3). Users can first choose between the `I-GMM-D` and `GMM-D` algorithms for dynamic schema discovery. Next, they can specify the number of nodes to be inserted and whether these should correspond to real data. In both cases, the update batch is built from the sample $\mathcal{G}_s$ retained for updates, ($(100 - p)\%$ of $\mathcal{G}$, where p is the percentage of data used for static schema discovery). If the user wants to use real data, DiscoPG will randomly select the desired number of nodes from $\mathcal{G}_s$ and add these to the graph instance. If not, DiscoPG will construct the required

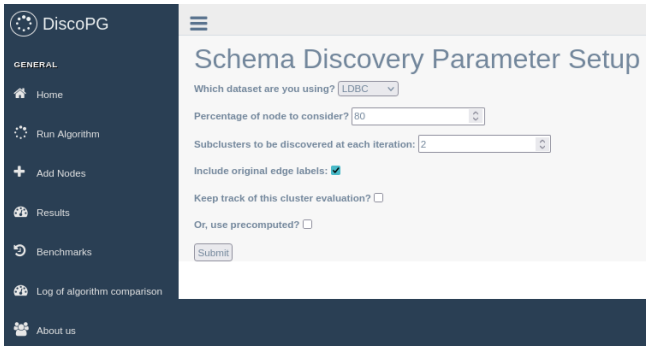

Figure 2: DiscoPG: Schema Discovery Setup (Static Case)

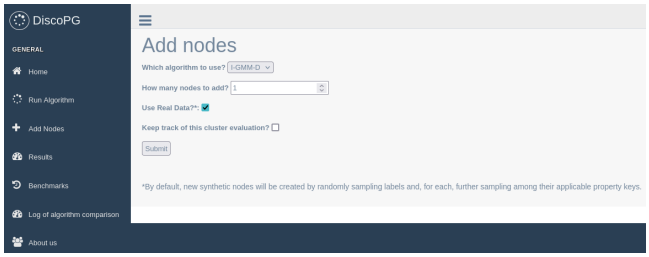

Figure 3: DiscoPG: - Schema Discovery Setup (Dynamic Case)

number of *synthetic nodes*, by first randomly selecting labels from $\mathcal{G}_s$ and, for each, further sampling its set of applicable properties. As before, the users have the option to also log the evaluation.

3.2 Schema Dashboard

Users can use a dedicated `Schema Dashboard` to log and analyze the performance of DiscoPG's algorithm, across various datasets and configuration, in terms of runtime and schema quality.

Performance Metrics. For each logged algorithm, its execution time per iteration is plotted, as captured in Figure 4. Users can zoom in on the portions of the plot they find the most relevant and analyze the comparative behaviour of the algorithms. For example, Figure 4 illustrates this based on the LDBC dataset. We see that the time required to discover new clusters at most iterations is low, keeping under a few microseconds. The few recorded jumps in execution time correspond to the computation of larger clusters and primarily concern the `GMM-S` algorithm, responsible for producing the schema in the static case. In the dynamic setting, we can observe that the incremental algorithm `I-GMM-D` has a steady evolution across a higher number of iterations that its memoization-based couterpart, `GMM-D`. The latter converges much faster, after only 8 iterations, and has the lowest execution times, due to the fact that it does not perform unnecessary computations in the recursive calls.

Quality Metrics. To analyze the quality of the schemas discovered at each iteration, users can observe the behaviour of the algorithms, with respect to the following metrics, measuring the similarity between a pair of clusterings. The Adjusted Mutual Index (**AMI**) accounts for potentially unbalanced clusters in the ground truth. The Adjusted Random Index (**ARI**) considers all sample pairs and

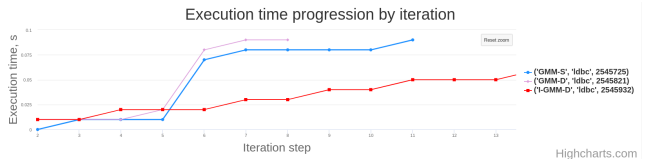

Figure 4: Performance Metrics

Figure 5: Quality Metrics

counts those assigned in the same/different clusters. Both metrics are computed with respect to the HDBSCAN hierarchical clustering algorith. The corresponding plots allow users to visually grasp which algorithm is best suited for schema discovery on the chosen dataset. As they can readily visualize the minimum number of iterations already providing a schema of satisfactory quality, the information can be used for further optimizations. On the LDBC dataset (see Figure 5), we notice that the quality of the schema discovered by `GMM-S` improves with the number of iterations. This trend is also recorded when considering the `GMM-D` algorithm, all the while the convergence rate is much faster, as already observed when analyzing its runtime performance. For the incremental `I-GMM-D` algorithm, we note that it conserves cluster quality, due to its incremental nature, thus providing a more robust alternative to `GMM-D`.

3.3 Schema Exploration

The `Schema Exploration` module enables users to navigate the structure and evolution of the discovered schemas. In the static case, as captured in Figure 6, users can explore the graph structure of the dataset they have chosen in the `Schema Discovery` module (see Section 3.2). For example, when considering the LDBC dataset, as illustrated, users can visualize the clusters corresponding to each of the discovered node types, as well as their labeled inter-connections. Note that the latter are produced based on the information regarding the edges attached to the individual nodes in the clusters. For each of the clusters, users can also inspect their corresponding properties and visualize their respective sizes, in order to understand the relative number of node instances they contain.

In the dynamic case, users can visualize the results of applying the `I-GMM-D` or `GMM-D` algorithms (Figure 7 and, respectively, Figure 8). To identify the impact that updating the base graph has on the discovered schema, DiscoPG provides a custom color coding. The unchanged clusters appear in blue, the newly formed ones - in green, and those that augmented due to node insertions are represented with orange and blue concentric circles.

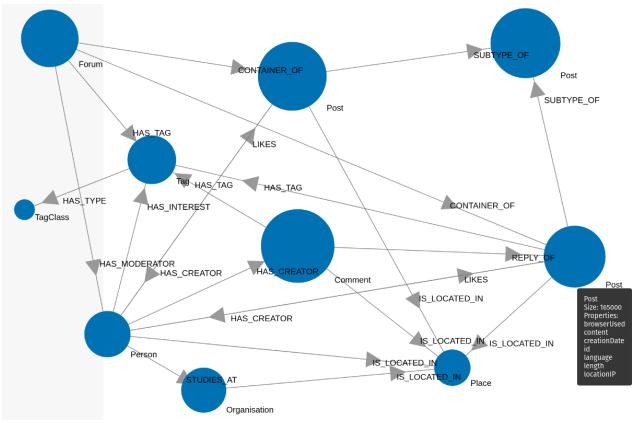

Figure 6: GMM-S Discovered Schema for LDBC

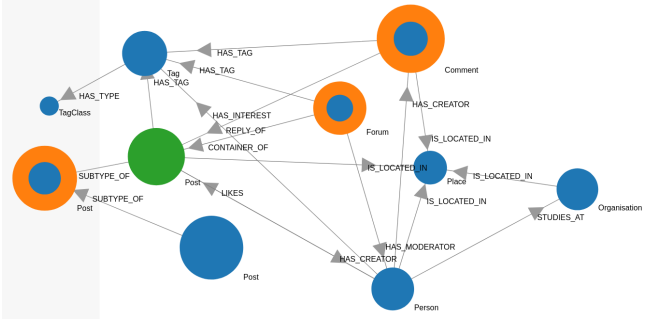

Figure 7: I-GMM-D Discovered Schema for LDBC

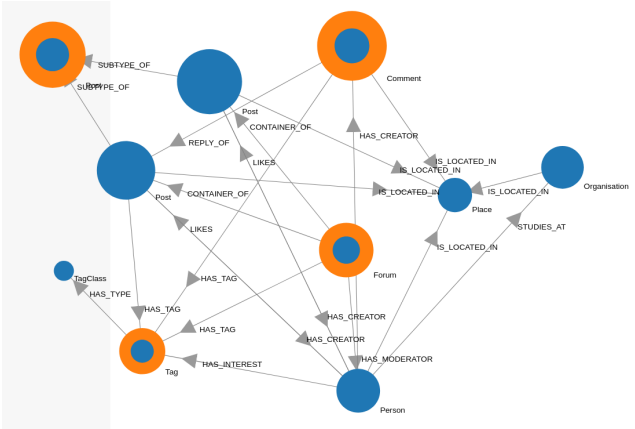

Figure 8: GMM-D Discovered Schema for LDBC

REFERENCES

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

[2] Angela Bonifati, Stefania Dumbrava, and Nicolas Mir. 2022. Hierarchical Clustering for Property Graph Schema Discovery. In *EDBT*.

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

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

[5] Siddhartha Sahu, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, and M. Tamer Özsu. 2020. The ubiquity of large graphs and surprising challenges of graph processing: extended survey. *VLDB J.* 29, 2-3 (2020), 595–618.

[6] Sherif Sakr, Angela Bonifati, Hannes Voigt, et al. 2021. The Future is Big Graphs: a Community View on Graph Processing Systems. *CACM* 64, 9 (2021), 62–71.