# Quantum-PG-HIVE: Schema Discovery for Property Graphs Using Quantum Computing

Emmanouil Limnaios[1], Sophia Sideri[1,2] and Haridimos Kondylakis[1,3]

[1]CSD, University of Crete, Heraklion, Greece
[2]LIPADE, Université Paris Cité, Paris, France
[3]FORTH-ICS, Heraklion, Greece

**Abstract**

Property graphs (PGs) are widely adopted across domains due to their flexible, schema-free nature. However, the absence of explicit schemas complicates understanding, integration, validation, and analytics. PG-HIVE introduced a hybrid, incremental framework for schema discovery based on semantic embeddings and Locality-Sensitive Hashing (LSH) clustering. While effective and scalable, LSH remains inherently probabilistic, occasionally producing fragmented clusters and requiring non-trivial post-processing. In this demonstration, we present Quantum-PG-HIVE, an optimization-based extension of PG-HIVE that replaces the LSH clustering step with a Quadratic Unconstrained Binary Optimization (QUBO) formulation. Schema discovery is reformulated as a balanced minimum cut problem over a sparse similarity graph derived from pattern embeddings. The resulting QUBO energy function is minimized via simulated annealing and is directly compatible with quantum annealing hardware.

## 1. Introduction

The rapid adoption of property graph databases across domains such as social networks, neuroscience, and knowledge graphs has intensified the need for automated and scalable schema discovery techniques. Property graphs are inherently flexible and often schema-less, enabling rapid data ingestion and evolution but complicating data understanding, query formulation, and downstream analytics [1, 2]. As graphs scale to millions of nodes and edges with heterogeneous labels and properties, manually engineered schemas become infeasible, motivating principled, data-driven approaches to infer latent structural regularities [3].

PG-HIVE [4] is a state-of-the-art framework for hybrid incremental schema discovery in property graphs, combining pattern extraction, embedding, clustering, and schema inference into a scalable pipeline. At its core, PG-HIVE relies on Locality-Sensitive Hashing (LSH) to cluster node and edge patterns in an embedding space, enabling approximate similarity grouping at scale.

**The problem.** While LSH offers attractive computational properties, it is fundamentally probabilistic: similar patterns may be separated into different buckets, while dissimilar patterns may collide. In practice, this leads to fragmented clusters, spurious schema types, and non-trivial post-processing overhead—especially in graphs with complex, skewed, or noisy structures.

**The solution.** This work introduces Quantum-PG-HIVE, a quantum-inspired clustering module that replaces the LSH-based clustering step in PG-HIVE with an optimization-based approach grounded in Quadratic Unconstrained Binary Optimization (QUBO) [5]. A QUBO is a combinatorial optimization formulation in which the objective is to minimize a quadratic function of binary decision variables, making it equivalent to finding the ground state of an Ising model [6] and directly compatible with both classical heuristic solvers and quantum annealing hardware. Instead of approximating similarity neighborhoods via hashing, Quantum-PG-HIVE formulates schema discovery as a balanced minimum cut problem on a sparse similarity graph derived from pattern embeddings. By explicitly optimizing a global objective function, Quantum-PG-HIVE produces cleaner, more interpretable schemas while maintaining scalability and compatibility with existing PG-HIVE infrastructure.

*Published in the Proceedings of the Workshops of the EDBT/ICDT 2026 Joint Conference (March 24-27, 2026), Tampere, Finland*
*Corresponding author.

In this demonstration, we showcase the full interactive experience of Quantum-PG-HIVE. Users can load a property graph from a backend store (e.g., Neo4j), explore clustering options (manual or adaptive), inspect intermediate and final schema elements, visualize discovered node and edge types, inspect property constraints, and compare schema extraction results under different parameter settings or noise levels, and baseline approaches. The demo highlights Quantum-PG-HIVE's ability to (i) discover types in challenging scenarios, (ii) incrementally update schemas, and (iii) give users full control and transparency over the inference process through a lightweight, intuitive web interface, going beyond classical clustering approaches in both efficiency and effectiveness.

The rest of this paper is structured as follows: Section 2 presents the architecture of the system ans Sectionr̃efsec:scenario presents a demonstration scenario. Finally Section 4 concludes this paper and presents directions for future work.

## 2. Architecture

Quantum-PG-HIVE adopts a modular and extensible architecture that augments the original PG-HIVE pipeline with an optimization-based clustering backend, while preserving the preprocessing, embedding, and schema inference components. The system is designed to maintain architectural continuity with PG-HIVE, enabling controlled comparisons between probabilistic LSH-based clustering and global QUBO-based optimization under identical upstream and downstream conditions. The architecture follows a layered processing model in which graph data flows from storage to representation, from representation to similarity modeling, from similarity modeling to clustering, and finally to schema inference and interactive exploration. The overall architecture of the system is shown in Figure 1.

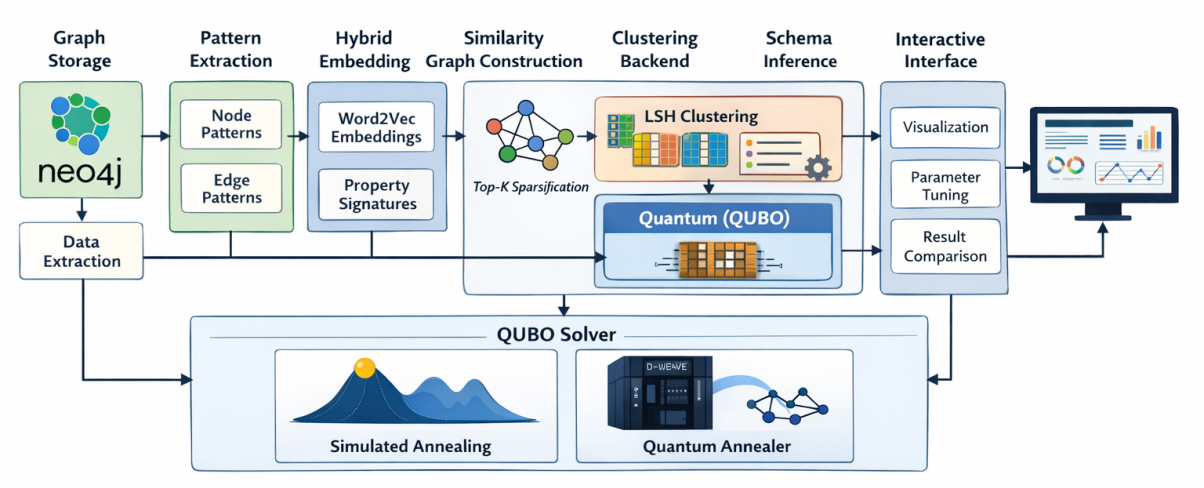


**Figure 1:** Quantum-PG-HIVE high-level architecture.

### 2.1. Architectural Overview

At a conceptual level, the system consists of interconnected layers responsible for data acquisition, feature construction, similarity modeling, clustering, and schema synthesis. These layers are loosely coupled, allowing the clustering backend to be replaced without affecting the rest of the pipeline. The primary architectural enhancement introduced by Quantum-PG-HIVE is the insertion of a QUBO-based optimization engine between similarity modeling and schema inference.

The overall workflow proceeds as follows. The system first connects to a property graph backend and extracts structural information. This information is transformed into pattern-based representations. The patterns are embedded into a hybrid vector space that captures both semantic and structural information. A sparse similarity graph is then constructed over these embeddings. At this stage, either the LSH-based clustering module or the QUBO-based optimization module can be invoked. The clustering output is

subsequently passed to the schema inference engine, which generates formal node and edge types, property constraints, and cardinalities. Finally, results are rendered through an interactive web interface that exposes both intermediate artifacts and final schema representations.

## 2.2. Graph Storage and Data Access Layer

The architecture interfaces with a property graph storage system, typically Neo4j, although the design remains backend-agnostic provided the store supports labeled property graph semantics and query-based extraction. Upon user authentication, the system issues structured queries that retrieve nodes, edges, labels, and associated properties in a uniform format. Extraction is performed in a way that ensures that all structural information relevant to schema discovery is captured consistently across datasets of varying size and complexity.

To support scalability and interactive exploration, the data access layer incorporates batch processing and optional sampling mechanisms. For very large graphs, users may specify sampling limits that preserve type diversity while reducing memory consumption. Sampling occurs before embedding and clustering, ensuring that subsequent computations remain tractable without compromising schema quality.

## 2.3. Pattern Extraction and Hybrid Feature Construction

Following extraction, raw graph elements are transformed into abstract patterns. A node pattern is defined by its combination of labels and the set of properties it contains. An edge pattern is defined by its relationship type, the label combinations of its source and target nodes, and its own property signature. This abstraction decouples schema discovery from individual graph instances and shifts the focus toward recurring structural motifs.

Each pattern is then encoded into a hybrid feature vector. The semantic component of this vector is generated using a Word2Vec model trained on label and property co-occurrence contexts. This embedding captures latent semantic proximity between patterns, even when their surface-level labels differ. The structural component is represented as a binary or sparse vector indicating the presence or absence of properties. The concatenation of semantic embeddings and structural signatures yields a hybrid representation capable of capturing both contextual similarity and structural distinctiveness.

This hybrid representation plays a crucial role in enabling the recovery of latent types in scenarios with incomplete labeling or heterogeneous property distributions. Importantly, this stage remains identical regardless of the clustering backend selected, ensuring that differences in final schema quality are attributable solely to clustering strategy.

## 2.4. Sparse Similarity Graph Construction

Instead of directly applying clustering algorithms to the embedding vectors, Quantum-PG-HIVE constructs an intermediate similarity graph. For each pattern group, pairwise cosine similarities between hybrid embeddings are computed. To prevent quadratic growth in similarity computations, the architecture employs a Top-$K$ sparsification strategy. For each pattern, only its $K$ most similar neighbors are retained, forming a sparse weighted graph $G = (V, E, w)$ where vertices correspond to patterns and edges are weighted by similarity scores.

This sparsification serves two purposes. First, it drastically reduces the number of quadratic interactions that must be encoded in the QUBO formulation, improving computational efficiency. Second, it preserves the most informative similarity relationships, ensuring that optimization focuses on semantically meaningful connections. The value of $K$ is configurable and can be tuned based on dataset density and desired clustering granularity.

## 2.5. Clustering Layer

The clustering layer is the central architectural differentiation between PG-HIVE and Quantum-PG-HIVE. The system supports two interchangeable backends: the baseline LSH module and the optimization-based QUBO module.

When the LSH backend is selected, hybrid vectors are grouped into buckets using Euclidean or MinHash Locality-Sensitive Hashing. Adaptive parameter selection estimates bucket width and the number of hash tables based on sample statistics of embedding distances and property sparsity. This approach offers expected linear-time performance and enables scalable approximate similarity grouping. However, its probabilistic nature may lead to fragmentation or collisions.

When the QUBO backend is selected, clustering is reformulated as a balanced minimum cut optimization problem on the sparse similarity graph. Binary decision variables are introduced to represent partition assignments. The objective function combines a cut term that penalizes separation of highly similar patterns and a balance term that discourages trivial partitions. The resulting quadratic energy function is minimized using simulated annealing. The architecture supports both a high-performance Scala-based implementation integrated into the Spark pipeline and a Python reference implementation compatible with D-Wave's Ocean SDK.

To obtain multiple clusters, recursive bipartitioning is applied. After each optimization step, the graph is partitioned into two subsets, and the process is repeated until stopping criteria based on cut quality or minimum cluster size are satisfied. This hierarchical decomposition yields a dendrogram-like clustering structure. An adaptive balancing mechanism can be enabled to estimate optimal partition ratios in datasets exhibiting skewed type distributions, using spectral heuristics to guide the balance term.

## 2.6. Schema Inference Engine

Once clustering is completed, the resulting pattern groups are passed to the schema inference engine. This component consolidates clusters into formal schema types. Clusters sharing identical label sets are merged directly, while unlabeled clusters are compared to labeled ones using structural similarity metrics. If similarity exceeds predefined thresholds, clusters are merged; otherwise, abstract types are created.

The engine then infers mandatory and optional properties based on frequency analysis, determines property datatypes through lightweight inspection of observed values, and computes relationship cardinalities by analyzing in-degree and out-degree distributions. The final schema is serialized into PG-Schema representations in both STRICT and LOOSE variants, ensuring compatibility with downstream tools and validation frameworks.

In incremental mode, newly discovered clusters are merged with existing schema elements without recomputing the entire dataset. This supports monotonic schema evolution as graphs grow over time.

## 2.7. Interactive Exploration Layer

The final architectural layer consists of a browser-based graphical interface. The interface exposes intermediate artifacts such as similarity graphs, cluster assignments, energy convergence plots in the QUBO mode, and final schema graphs. Users can dynamically switch between clustering backends, adjust hyperparameters such as $K$, $\lambda$, or simulated annealing iterations, and immediately observe changes in schema fragmentation, merge overhead, and inferred cardinalities.

This tight integration between backend computation and frontend visualization ensures transparency of the optimization process and enables interactive experimentation. The architecture thus supports both research-oriented evaluation and real-world deployment scenarios.

### 2.8. Architectural Properties

The modularity of Quantum-PG-HIVE ensures extensibility and experimental flexibility. Because clustering is encapsulated within a dedicated layer, additional optimization solvers or hybrid approaches can be incorporated without altering preprocessing or schema inference logic. The system leverages distributed data processing through Apache Spark for scalability, while optimization subproblems remain sufficiently localized to maintain tractable runtimes. The architecture is therefore both scalable and future-proof, providing immediate compatibility with classical simulated annealing solvers and a direct pathway toward quantum hardware acceleration as it becomes practically viable.

## 3. Demonstration Scenario

The demonstration presents an interactive walkthrough of Quantum-PG-HIVE, guiding participants through the complete workflow of optimization-enhanced schema discovery and enabling direct comparison between the baseline LSH clustering and the QUBO-based clustering backend. The scenario is structured into the following steps.

**1. Dataset Selection and Configuration.** The user connects to a Neo4j property graph instance and selects a dataset. The interface displays basic statistics, including node and edge counts, distinct labels, and relationship types. The user then chooses the clustering backend (LSH or Quantum) and configures execution parameters. For LSH, this includes hash tables and bucket settings; for Quantum, this includes Top-$K$ sparsification, balance penalty $\lambda$, target partition ratio, and simulated annealing iterations. Execution can be performed in static or incremental mode.

**2. Pattern Extraction and Embedding.** The system extracts node and edge patterns based on labels and property signatures. Each pattern is encoded into a hybrid embedding that combines semantic Word2Vec representations with structural property indicators. The interface reports the number of discovered patterns and summarizes structural diversity. This stage is identical for both clustering backends to ensure fair comparison.

**3. Similarity Graph Construction.** A sparse similarity graph is built by computing cosine similarities between embeddings and retaining only the Top-$K$ nearest neighbors per pattern. The interface presents sparsity metrics and similarity distributions, offering insight into structural skew and clustering difficulty.

**4. Clustering Execution.** In LSH mode, hybrid vectors are grouped into hash buckets to produce candidate clusters. In Quantum mode, clustering is formulated as a balanced minimum cut encoded as a QUBO energy function and solved via simulated annealing with recursive bipartitioning. The interface displays cluster counts, balance diagnostics, and (in Quantum mode) energy convergence information.

**5. Schema Inference and Comparison.** The resulting clusters are consolidated into schema types through label-based merging and structural similarity checks. The system infers mandatory and optional properties, datatypes, and relationship cardinalities, and renders the final schema in PG-Schema format. A side-by-side comparison view highlights differences in raw cluster counts, merge overhead, and final node and edge types between LSH and Quantum modes.

**6. Robustness and Incremental Exploration (Optional).** Participants can simulate noise or partial labeling to evaluate robustness and observe how clustering quality evolves under perturbation. In incremental mode, new graph batches are introduced, and the system updates the schema without recomputing from scratch, illustrating monotonic schema evolution.

Through these steps, the demonstration highlights how optimization-based clustering enhances schema cleanliness, reduces fragmentation, and maintains competitive performance while preserving the interactive and incremental capabilities of the original PG-HIVE framework.

## 4. Conclusion

In this demonstration, we presented Quantum-PG-HIVE, an optimization-based extension of the PG-HIVE framework for schema discovery in property graphs. By reformulating the clustering phase as a balanced minimum cut problem expressed through a Quadratic Unconstrained Binary Optimization (QUBO) model, the system replaces probabilistic hashing with a principled global optimization strategy. This shift enables more coherent partitioning of pattern embeddings, reduces fragmentation artifacts, and lowers post-processing overhead, particularly for relationship types where hash-based approaches are more susceptible to instability.

The architecture preserves the hybrid embedding and schema inference pipeline of PG-HIVE while introducing a modular clustering backend that supports both classical simulated annealing and compatibility with quantum annealing hardware. Through the interactive interface, users can inspect intermediate similarity graphs, monitor optimization behavior, tune hyperparameters, and directly compare LSH-based and QUBO-based clustering outcomes under identical conditions. The demonstration highlights improvements in edge-type cleanliness, robustness under noisy or skewed data distributions, and competitive runtime performance at practical graph scales.

Overall, Quantum-PG-HIVE illustrates how global optimization techniques can enhance schema discovery in flexible graph data models without sacrificing scalability or interpretability. By bridging property graph data management with QUBO-based optimization, the system not only improves current clustering quality but also provides a forward-looking pathway toward quantum-enabled graph analytics as quantum hardware matures.

## Declaration on Generative AI

The authors have not employed any Generative AI tools.

## References


[1] A. Bonifati, S. Dumbrava, H. Kondylakis, G. Troullinou, G. Vassiliou, Progressive querying on knowledge graphs, in: A. Simitsis, B. Kemme, A. Queralt, O. Romero, P. Jovanovic (Eds.), Proceedings 28th International Conference on Extending Database Technology, EDBT 2025, Barcelona, Spain, March 25-28, 2025, OpenProceedings.org, 2025, pp. 106–118. URL: https://doi.org/10.48786/edbt.2025.09. doi:10.48786/EDBT.2025.09.

[2] G. Troullinou, G. Agathangelos, H. Kondylakis, K. Stefanidis, D. Plexousakis, DIAERESIS: RDF data partitioning and query processing on SPARK, Semantic Web 15 (2024) 1763–1789. URL: https://doi.org/10.3233/SW-243554. doi:10.3233/SW-243554.

[3] H. Kondylakis, S. Dumbrava, M. Lissandrini, N. Yakovets, A. Bonifati, V. Efthymiou, G. Fletcher, D. Plexousakis, R. Tommasini, G. Troullinou, E. Ymeralli, Property graph standards: State of the art & open challenges, Proc. VLDB Endow. 18 (2025) 5477–5481. URL: https://www.vldb.org/pvldb/vol18/p5477-kondylakis.pdf. doi:10.14778/3750601.3750698.

[4] S. Sideri, G. Troullinou, E. Ymeralli, V. Efthymiou, D. Plexousakis, H. Kondylakis, Pg-hive: Hybrid incremental schema discovery for property graphs, EDBT, 2026, p. (to appear).

[5] F. W. Glover, G. A. Kochenberger, R. Hennig, Y. Du, Quantum bridge analytics I: a tutorial on formulating and using QUBO models, Ann. Oper. Res. 314 (2022) 141–183. URL: https://doi.org/10.1007/s10479-022-04634-2. doi:10.1007/S10479-022-04634-2.

[6] A. Lucas, Ising formulations of many NP problems, CoRR abs/1302.5843 (2013). URL: http://arxiv.org/abs/1302.5843. arXiv:1302.5843.