




A Novel Data Set for Information Retrieval on the Basis of Subgraph Matching

Kaspar Riesen^{1,2}(✉) , Hans-Friedrich Witschel² , and Loris Grether² 

¹ Institute of Computer Science, University of Bern, Neubrückstrasse 10,
3012 Bern, Switzerland
`riesen@inf.unibe.ch`

² Institute for Informations Systems, University of Applied Sciences Northwestern
Switzerland, Riggbachstrasse 16, 4600 Olten, Switzerland
`{hansfriedrich.witschel,loris.grether}@fhnw.ch`

Abstract. We are facing the challenge of rapidly increasing amounts of data. Moreover, we observe that in many applications the underlying data contains strongly related entities making graphs the most appropriate structure for data modeling. When data is represented by means of a graph, querying corresponds to a graph matching problem. The present paper introduces a novel graph that models information from the medical domain with about 110,000 nodes and 220,000 edges. Additionally we present several basic benchmark queries, i.e. specific subgraphs, from different categories that can be found multiple times in the medical graph. Both the graph and the benchmark can be used to implement, test, and compare novel graph matching algorithms in a real world scenario.

Keywords: Subgraph isomorphism · Graph matching · Graph database

1 Introduction and Related Work

Many of the information repositories available are diverse, large, and often contain strongly related entities. To cope with numerous and arbitrary relations more efficiently, graph based databases are more and more recognized as versatile alternative to relational databases [1]. In fact, in contrast with tabular structures that use foreign keys for relationship modeling, graphs are able to represent not only the values of entities, but can be used to explicitly model structural relations that might exist between different objects by means of edges [2, 3]. Moreover, the user's mental model of the data and the actual data structure stored on a device are fully congruent. Hence, visualizations of graphs typically provide an intuitive and clearly understandable overview of the underlying structures and relationships.

Supported by Innosuisse Project Nr. 26281.2 PFES-ES.

When a graph is employed for the purpose of data storage, the information retrieved in response to a certain query is typically also a graph, which may be, for instance, a subgraph of the underlying database graph [3]. In the present paper, we employ the concept of *subgraph isomorphism* [4] for information retrieval. Subgraph isomorphism indicates that a smaller graph is contained in a larger graph. Let us assume that we represent a query by means of an attributed graph q , termed query graph. Given q and the database graph G , we can check whether the query graph q is contained in the underlying database G .

About a decade ago one of the authors of the present paper introduced a generalized form of graph isomorphism that is particularly well suited for information retrieval from graphs [5]. This generalized subgraph isomorphism methodology allows to mask out attributes in query graphs that are irrelevant for a particular question. Moreover, the algorithm also allows the definition of certain variables in order to retrieve values of predefined attributes as well as the definition of constraints (for example variables that can assume only certain values). In the meantime these basic concepts for subgraph matching in large graphs have been implemented in many commercial software products (e.g. Neo4j or Amazon Neptune, to name just two prominent examples).

Despite the fact that graph based databases have reached a mature level, graph matching [6], information retrieval on graph-like structures [7], or human interaction with graph models [8] are still active fields of research. The major contribution of the present paper in this particular field is twofold. First, we present a novel large graph that models heterogeneous information from the medical domain by means of about 110,000 nodes and 220,000 edges. Second, we present 21 benchmark queries, i.e. specific subgraphs, from seven different categories that can be found multiple times in the large graph. These categories represent important application scenarios and therefore we need to know how efficiently they can be answered. The present paper is similar in spirit to [9–12] where graphs and benchmark tasks for (sub)graph isomorphism or error-tolerant graph matching are presented.

The remainder of the present paper is organized as follows. In Sect. 2 the basic definitions are introduced. Next, in Sect. 3 the novel medical graph is introduced and thoroughly described. Eventually, in Sect. 4, we define the benchmark tasks in the form of subgraphs that can be found in the medical graph together with the respective matching results and run times. Finally, in Sect. 5, we conclude the paper and discuss some future work ideas.

2 Preliminary Definitions

We employ the *property-graph-model* in our approach. Formally, a graph is a 4-tuple $g = (V, E, \mu, \nu)$, where

- V is the finite set of nodes
- $E \subseteq V \times V$ is the set of edges
- $\mu : V \rightarrow \{(t, \mathbf{x}(t)) \mid t \in T_{nodes}, \mathbf{x}(t) \in (D_1(t) \times \dots \times D_{n_t}(t))\}$ is the node property function
- $\nu : V \times V \rightarrow \mathcal{P}(\{t \mid t \in T_{edges}\}) \setminus \emptyset$ is the edge type function.

Through the node property function μ , each node $u \in V$ in a graph is labeled by a (type, property)-pair $(t, \mathbf{x}(t))$. The first component, the type t , is an element of a finite set of node types T_{nodes} . The node types group nodes together and specify the roles they play within the graph. For example, some nodes could represent objects of type **Disease**, while others represent nodes of type **Symptom** or **Treatment**.

In our scenario, nodes also contain *properties* modeled by means of the second component, i.e. $\mathbf{x}(t) = (x_1, \dots, x_{n_t})$. In this property vector each attribute x_i belongs to some domain $D_i(t)$. The dimension of vector $\mathbf{x}(t)$, i.e. the number n_t of attributes, as well as each individual attribute domain $D_i(t)$ depends on the actual type t of the node [5]. Possible properties for nodes of type **Disease** would be, for instance, an ID, the name of the disease, or others.

Formally, edges are pairs of nodes, $(u, v) \in V \times V$ and structure the graph. In our scenario, edges are always directed and always connect exactly one start- with one end-node. In some applications, however, it might be necessary to include more than one edge between the same pair of nodes, because of the existence of multiple relations. In the formal graph model provided above, this can be accomplished by assigning several edge types to the same edge (u, v) by means of the edge type function ν , i.e. $\nu(u, v) = \{t_1, \dots, t_n\}$. Note that the range of function ν is the power set of all edge types from the finite set T_{edges} . Assigning n types $\{t_1, \dots, t_n\}$ to edge (u, v) by means of ν is equivalent to providing n individual edges from node u to node v [5]. In our scenario the edges do not contain any further properties¹.

A possible type of an edge between nodes of type **Disease** and **Symptom** might be, for instance, **causes**. The adjacent nodes, the edge's direction and the label of the edge provide semantic clarity to the relationship.

3 Medical Graph

In the present paper we use the data structure $g = (V, E, \mu, \nu)$ defined in the previous section to model diverse information from the medical domain. To this end, we automatically parse data from the following four public domains:

1. Wikidata: wikidata.org
2. SemMED: skr3.nlm.nih.gov/SemMed/
3. Medline: medlineplus.gov
4. DisGeNET: disgenet.org

From Wikidata we parse entities of five different types, viz. diseases, symptoms, treatments, behaviours (such as tobacco addiction or similar), and diagnostic tests. Next, we complement the set of diseases with entities extracted from SemMED. From the same domain as well as from Medline we extract further diagnostic tests (we select the most frequent diagnostic tests by means of a simple heuristic). Also from SemMED we parse research papers and two patient

¹ This can be generalized in a straightforward manner.

characteristics (gender and age group) that might be typical for certain diseases. The number of research papers is limited to eight papers per disease and we require the title of the paper to include the name of the respective disease. Finally, From DisGeNET we parse genes and proteins (we select genes with a GeneSymbol-Score greater than, or equal to, 0.3).

All of these entities are modeled by means of nodes of different types and with different sets of properties. In total 110,774 nodes of nine different types are built by means of this procedure. In Table 1 the node types, the number of nodes per type and the properties available in the different node types are summarized.

Table 1. The node types, the number of nodes per type and the properties available in the different node types.

Node type $t \in T_{\text{nodes}}$	Count	Properties $\mathbf{x}(t)$
ResearchPaper	60,895	PMID, Abstract
Disease	28,000	CUID, Name, Description, DOID
Protein	9,281	uniproID
Gene	8,490	Name, GeneID, GeneSymbol Score (level of evidence)
Treatment	1,606	CUID, Name
DiagnosticTest	1,384	CUID, Name
Symptom	588	CUID, Name
Behaviour	489	Name
PatientCharacteristic	41	CUID, Name
Total count	110,774	

We use nine different edge types in order to connect the different nodes with each other and build the graph (in total 221,920 edges are inserted). Actually, we build a graph according to the star like scheme with nodes of type **Disease** as most central nodes (see Fig. 1). In Table 2 the edge types, the number of edges per type and the start- and end-node that are connected with the respective edge are summarized. The complete graph is publicly available in a comma separated file at <https://github.com/kaspar-riesen/medical-graph>.

4 Benchmark Tasks and Results

We divide our benchmark queries on the medical graph into the following seven categories or patterns (see also Fig. 2): *Single*, *Double*, *Triple*, *Triangle*, *Growing*

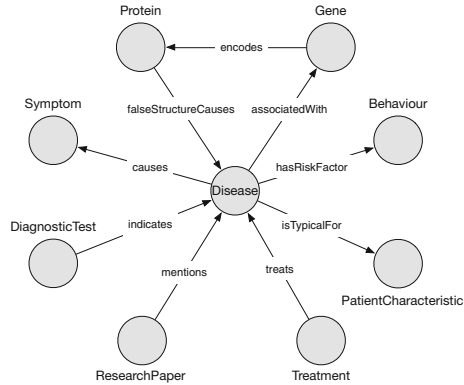


Fig. 1. The nine types of nodes are connected by means of nine different edge types.

Table 2. The edge types, the number of edges per type and the start- and end-node that are connected with the respective edge type.

Edge type $t \in T_{edges}$	Count	From	To
mentions	62,940	ResearchPaper	Disease
falseStructureCauses	43,088	Protein	Disease
associatedWith	42,610	Disease	Gene
indicates	29,702	DiagnosticTest	Disease
isTypicalFor	26,671	Disease	PatientCharacteristic
encodes	9281	Gene	Protein
treats	5523	Treatment	Disease
causes	1358	Disease	Symptom
hasRiskFactor	747	Disease	Behaviour
Total count	221,920		

Star, Top Hub, and Max Overlap. Each category represents a different information need ranging from finding information of a single symptom to differentiation between diseases. For each pattern three different specific queries are defined and described in the next paragraphs.

Single (see Fig. 3 (a)). We search for all nodes of ...

Query 1 ... type `DiagnosticTest` where the property `name` contains ...

Query 2 ... type `Disease` where the property `name` contains ...

Query 3 ... any type where any of the available properties contain ...

...the search string `liver`.

Double (see Fig. 3 (b)). We search for all nodes of type...

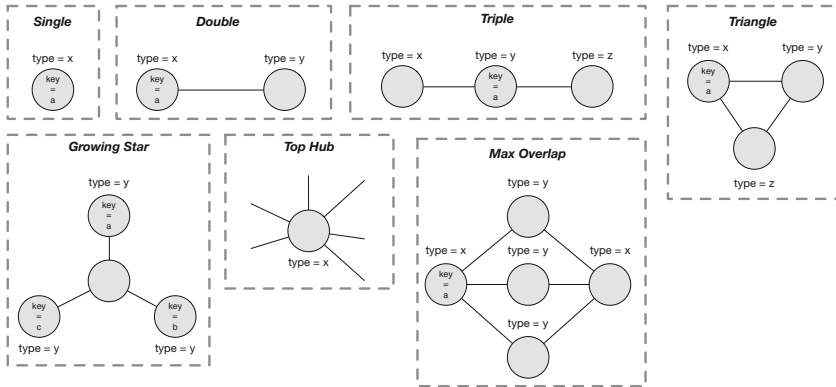


Fig. 2. The seven types of queries.

Query 4 ... Treatment that are connected with an edge `treats` ...

Query 5 ... DiagnosticTest that are connected with an edge `indicates` ...

Query 6 ... Gene that are connected with an edge `associatedWith` ...

...to a node of type `Disease` with the property `name` that contains the search string `breast cancer`.

Triple (see Fig. 3 (c)). We search for all nodes of both types `DiagnosticTest` and ...

Query 7 ... Symptom that are indirectly connected with two edges `indicates` and `causes` via a node of type `Disease` whose property `name` contains the search string `deficiency`.

Query 8 ... Treatment that are indirectly connected with two edges `indicates` and `treats` via a node of type `Disease` whose property `name` contains the search string `periodontitis`.

Query 9 ... Behaviour that are indirectly connected with two edges `indicates` and `hasRiskFactor` via a node of type `Disease` whose property `name` contains the search string `arteriosclerosis`.

Triangle (see Fig. 4 (a)). We search for all nodes of both types `Protein` or `Gene` that are directly connected with an edge `encodes` with each other and simultaneously connected via edges `associatedWith` and `falseStructureCauses` with a node of type `Disease` whose property `name` contains the search string ...

Query 10 ... type 1 diabetes.

Query 11 ... hypothermia.

Query 12 ... skin cancer.

Growing Star (see Fig. 4 (b)). We search for all nodes of type `Disease` that are directly connected via edge `causes` with at least ...

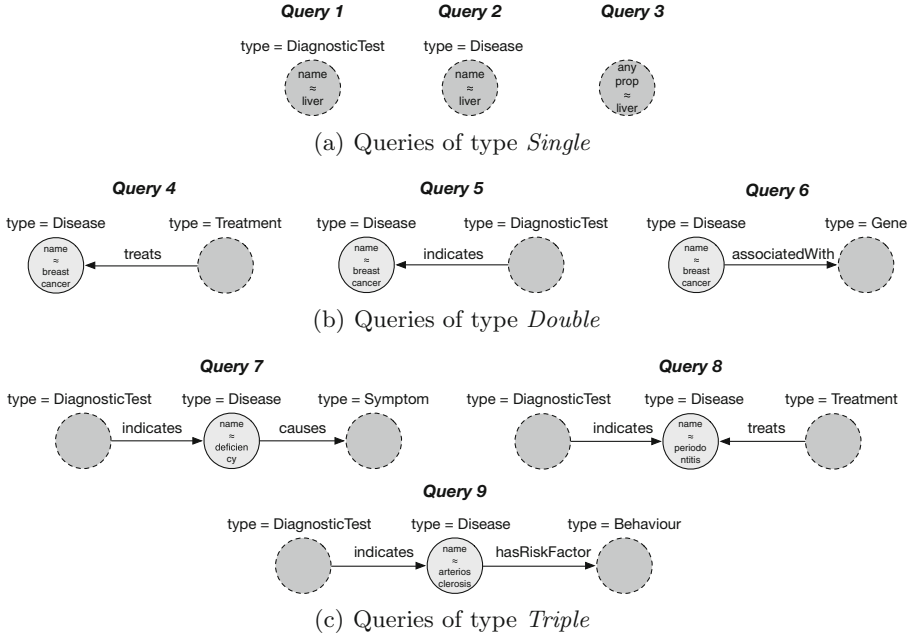


Fig. 3. Queries of different types. Grey colored nodes represent the nodes being searched for.

Query 13 ... two nodes of type *Symptom* where the property name of the first and second symptom contain the search string **fever** and **fatigue**, respectively.

Query 14 ... three nodes of type *Symptom* where the property name of the first, second, and third symptom contain the search string **fever**, **fatigue**, and **anorexia**, respectively.

Query 15 ... four nodes of type *Symptom* where the property name of the first, second, third, and fourth symptom contain the search string **fever**, **fatigue**, **anorexia**, and **diarrhea**, respectively.

Top Hub (see Fig. 4 (c)). We search for the five nodes of type *Disease* that have the most edges of type ...

Query 16 ... *treats* (to nodes of type *Treatment*).

Query 17 ... *indicates* (to nodes of type *DiagnosticTest*).

Query 18 ... *associatedWith* (to nodes of type *Gene*).

Max Overlap (see Fig. 4 (d)). We search for maximum five nodes of type *Disease* that share the most ...

Query 19 ... symptoms with a node of type *Disease* whose property name contains the search string **gastroenteritis**.

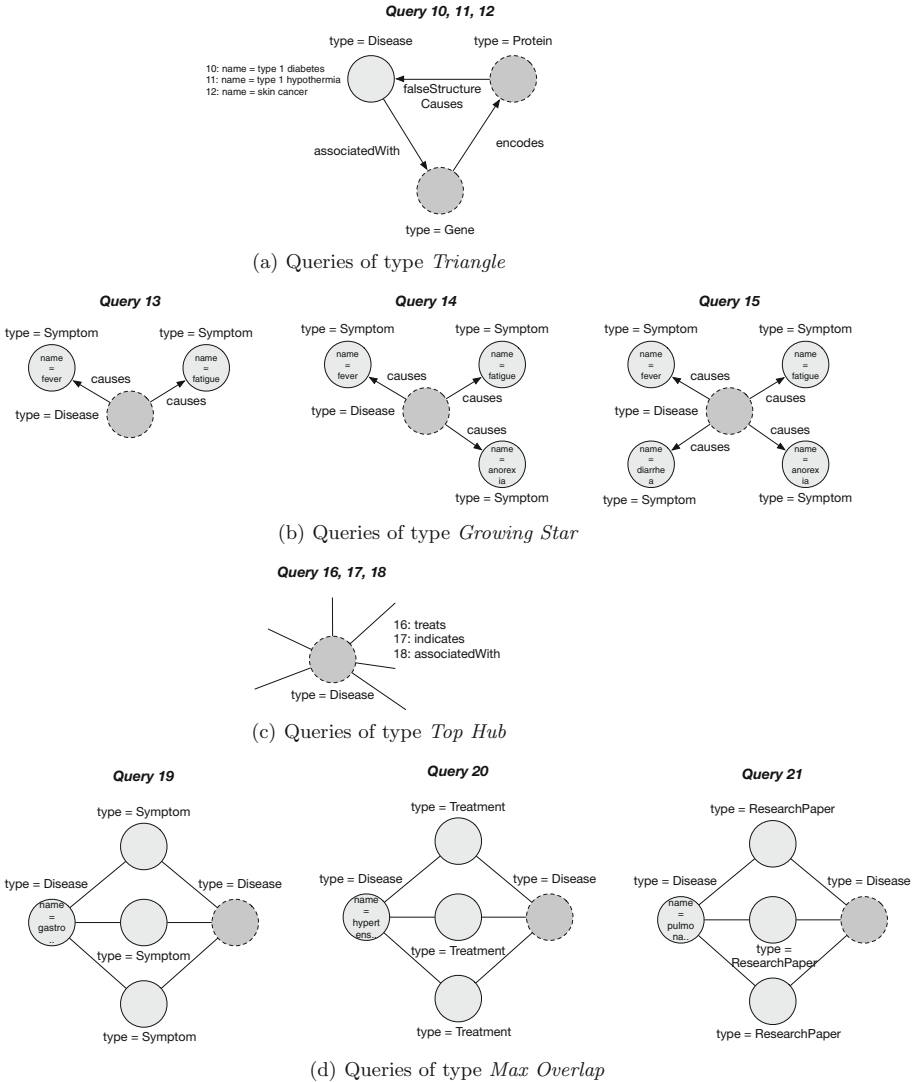


Fig. 4. Queries of different types. Grey colored nodes represent the nodes being searched for.

Query 20 ... treatments with the node of type **Disease** whose property **name** contains the search string **hypertension**.

Query 21 ... mentions in research papers with the node of type **Disease** whose property **name** contains the search string **pulmonary**.

In Table 3 the size of the result, i.e. the number of nodes that match the given subgraphs, as well as the run time for the actual matching are shown for all queries. We run our experiment on a Intel Core i7 with 16 GB RAM and we have

implemented the graph model by means of the graph database implementation `Neo4j's developer edition`². The size of the result sets varies from only one node to more than 1,000 nodes per query. Also for the matching times quite large differences are observable. The ground truth results for these matchings can be found at <https://github.com/kaspar-riesen/medical-graph>.

Table 3. The result size and matching times of queries 1 to 21.

Query	Size of result	Time [ms]
1	7 nodes	3
2	90 nodes	77
3	1066 nodes	834
4	40 nodes	36
5	30 nodes	82
6	1089 nodes	276
7	19 nodes	7
8	45 nodes	13
9	10 nodes	7
10	6 nodes	155
11	14 nodes	114
12	59 nodes	112
13	18 nodes	12
14	7 nodes	4
15	2 nodes	7
16	5 nodes	84
17	5 nodes	7
18	5 nodes	40
19	5 nodes	3
20	5 nodes	12
21	1 node	41

5 Conclusions and Future Work

Several areas in science and industry are facing the challenge of rapidly increasing amounts of data available, making scalable search methods inevitable. The vast majority of efficient search methods are built for traditional, i.e. tabular data

² One can define indexes on properties in `Neo4j` – however, we have omitted this possibility in our evaluation.

representations. Yet, we observe that in many modern applications the underlying data is inherently complex, making this limited representation formalism rather inappropriate. Graphs actually allow us to explicitly model relationships between entities. When data is represented by means of a graph, a search for information, or a pattern, exactly corresponds to a graph matching problem. The present paper introduces a novel graph that models information from the medical domain with about 110,000 nodes and 220,000 edges. Additionally we present several basic benchmark queries, i.e. specific subgraphs, from seven different categories that can be found multiple times in the medical graph.

We see several rewarding avenues to be pursued in future work. First, we invite the research community to test their own algorithms for subgraph isomorphism on the publicly available graph. Second, we see great potential to define more complex and more time-consuming benchmark queries. Last but not least, the medical graph could be substantially increased in the number of nodes and edges by accessing and integrating further repositories.

References

1. Robinson, I., Webber, J., Eifrem, E.: Graph Databases. O'Reilly, Springfield (2015)
2. Kandel, A., Bunke, H., Last, M. (eds.): Applied Graph Theory in Computer Vision and Pattern Recognition. Studies in Computational Intelligence, vol. 52. Springer, Heidelberg (2007). <https://doi.org/10.1007/978-3-540-68020-8>
3. Cook, D., Holder, L.: Mining Graph Data. Wiley-Interscience, Hoboken (2007)
4. Ullmann, J.R.: An algorithm for subgraph isomorphism. *J. ACM* **23**(1), 31–42 (1976)
5. Brügger, A., Bunke, H., Dickinson, P., Riesen, K.: Generalized graph matching for data mining and information retrieval. In: Perner, P. (ed.) *ICDM 2008*. LNCS (LNAI), vol. 5077, pp. 298–312. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70720-2_23
6. Foggia, P., Percannella, G., Vento, M.: Graph matching and learning in pattern recognition in the last 10 years. *Int. J. Pattern Recognit. Artif. Intell.* **28**(1) (2014)
7. Park, C.-S., Lim, S.: Efficient processing of keyword queries over graph databases for finding effective answers. *Inf. Proces. Manag.* **51**(1), 42–57 (2015)
8. Witschel, H.F., Riesen, K., Grether, L.: KvGR: a graph-based interface for explorative sequential question answering on heterogeneous information sources. In: Jose, J.M., et al. (eds.) *ECIR 2020*. LNCS, vol. 12035, pp. 760–773. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45439-5_50
9. Foggia, P., Sansone, C., Vento, M.: A database of graphs for isomorphism and subgraph isomorphism benchmarking. In: *Proceedings of the 3rd International Workshop on Graph Based Representations in Pattern Recognition*, pp. 176–187 (2001)
10. Riesen, K., Bunke, H.: IAM graph database repository for graph based pattern recognition and machine learning. In: da Vitoria Lobo, N., et al. (eds.) *SSPR /SPR 2008*. LNCS, vol. 5342, pp. 287–297. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89689-0_33

11. Neuen, D., Schweitzer, P.: Benchmark graphs for practical graph isomorphism. CoRR, abs/1705.03686 (2017)
12. Solnon, C., Damiand, G., de la Higuera, C., Janodet, J.-C.: On the complexity of submap isomorphism and maximum common submap problems. *Pattern Recogn.* **48**(2), 302–316 (2015)