Graph Pattern Mining for Business Decision Support

André Petermann
supervised by Erhard Rahm
Database Research Group
University of Leipzig
petermann@informatik.uni-leipzig.de

ABSTRACT

To which extent can graph pattern mining enrich business intelligence? This question was the seed whose sprout became my PhD research. To find an answer, I investigated graph-based data integration, the calculation of business measures from graphs and suitable data mining techniques based thereon. The latter should identify correlations between occurrences of specific graph patterns and values of business measures. Finally, interesting patterns should be presented to decision makers. With real world applications in mind, I additionally considered the requirements of big data scenarios at all stages. In this paper, I summarize my recent contributions and give an outlook on the work required to finally answer the motivating question.

1. INTRODUCTION

To make good decisions, enterprises have a permanent desire to understand the reasons for certain values of business measures. In a classical business intelligence development lifecycle a domain expert is choosing potential impact factors and the analytical model is tailored to evaluate measures by these factors. However, this approach often leads to oversimplified models and, thus, unexpected patterns may remain hidden. Hence, the use of graph models for business intelligence is a promising approach for two reasons: First, some patterns are too complex to be represented using tuples. In particular, this applies to patterns where most of the information is about relationships.

Second, graphs can loosen the coupling of experts' bias and analytical results because data represented by rich graph models like the property graph model [20] allows not only to evaluate instance data but also metadata occurrence, i.e., schema-related information is part of the result and must no be specified in a query. For example, to reveal patterns between objects of classes A and B, ideally analysts just want to ask *"Which patterns typically connect As and Bs?"* and expect an answer like *"Mostly via a sequence of Cs and Ds, but sometimes only via Es"*. In contrast, using a structured model like common data warehouse models, they need to ask several questions like *"Are As and Bs frequently connected via Ds?"* and get simple *"Yes"* or *"No"* answers.

Summarized, wrapping the schemas of data sources into a graph super-model enables more generic queries and mining of self-descriptive patterns. In my PhD research, I developed the BIIIG approach (**B**usiness **I**ntelligence with **I**ntegrated **I**nstance **G**raphs) to enable such flexible graph-based analyses of business data. Figure 1 provides an overview of the approach. In the remainder of this paper, I will give a brief overview of my past and future work.

2. CONTRIBUTIONS

In the following, I will provide an overview of the contributions made during my past PhD research.

2.1 Graph Representation of Business Objects

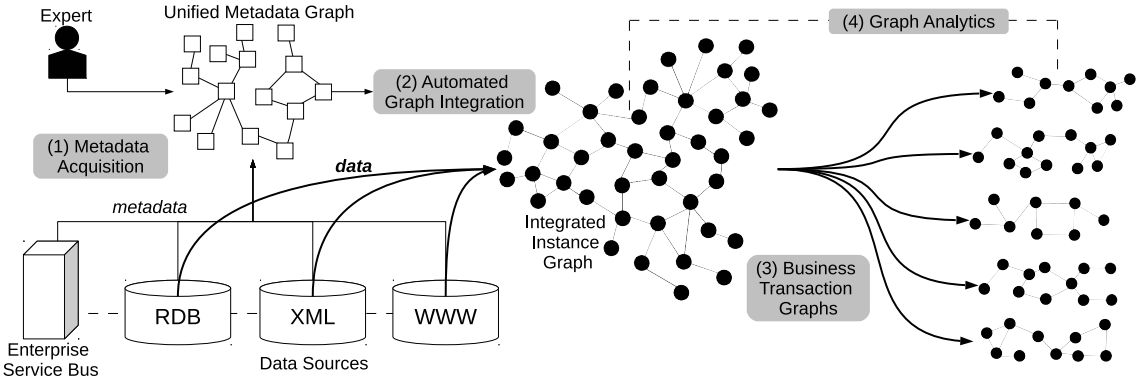
Business data of a company implicitly describes a graph but is typically stored in one or more business information systems based on relational databases. Thus, I first had to consider the process of turning data organized in tables into graphs. In [16], I proposed a semi-automated solution to this problem and implemented a prototype based on a productive graph database [19]. The approach was evaluated using real and synthetic data. In the following, I will briefly discuss my approach to graph-based data transformation and integration but, due to limited space, only for relational databases.

In the initial step (step 1 of Figure 1) metadata of one or more data sources is acquired, stored in a graph model (*unified metadata graph*) and enriched by a domain expert. In this graph, every vertex represents a class of domain objects (class-like tables) and every edge an association between classes (foreign keys or m:n tables). Both, vertices and edges, further contain information about their source system, semantic type, keys and attributes.

In the second step (step 2 of Figure 1), vertices and edges of the metadata graph are interpreted to generate SQL statements. These are used to query instances (data objects and relationships) from the source databases. Afterwards, all data objects are transformed into vertices and all relationships into edges of a so-called *integrated instance graph*. I decided to use the property graph model [20], i.e., a directed labeled multigraph with named attributes (properties). For both, vertices and edges, labels represent their semantic type and all attributes are stored using properties. Another popular model to represent such graphs is the resource description framework (RDF) [10]. However, RDF is

Proceedings of the VLDB 2017 PhD Workshop, August 28, 2017. Munich, Germany.

Figure 1: Overview of the BIIIG approach [16]



more general and provides no dedicated structures for logical relationships, labels and properties. In consequence every attributed relationship must be represented by a subgraph [7] and the total number of edges would be much higher.

Besides model transformation, the second step may also include data integration. Every vertex has a globally unique *source identifier* composed from identifiers for source system, class and record. Thus, the approach supports relationships across data sources. Such relationships may exist for two reasons: First, data objects of different systems may reference each other, for example, a ticket of a customer issue tracking system may reference an invoice stored in an accounting system. Second, certain master data is held redundantly and copies refer to a global business key (e.g., customer number). For the latter case, I proposed *vertex fusion*, a strategy to automatically merge the resulting vertices and to redirect their relationships.

2.2 Business Transaction Graphs

Data warehouse models use a schema (e.g., star schema) that needs to be defined in advance to link facts and dimensions. Data mining techniques based thereon can evaluate the co-occurrence of certain dimensional values (e.g., feature vectors). The major aim of the BIIIG approach was to enable an additional evaluation of the relationship structure among interrelated facts as well as between facts and dimensions. Analyzing such structural patters is promising, for example, to reveal interaction patterns between customers and certain employees that lead to high sales profit. Here, the first challenge was to find a suitable abstraction to enable such analyses. For this reason, I introduced the concept of *business transaction graphs* [16] as the base for measure aggregation (Section 2.3) and graph pattern mining (Section 2.5). A business transaction graph represents, for example, a single execution of a business process like trading or manufacturing goods.

I proposed a domain-specific algorithm to automatically extract a collection of such graphs from the integrated instance graph (step 3 in Figure 1). Figure 2 shows four example business transaction graphs of a sales process. For sake of ease, edge types are omitted. The algorithm is based on the observation that *transactional data* (e.g., `Email`, `Quotation`, `SalesOrder`) only link each other in the case of a causal connection. Here, *causally connected* means object B (e.g., an invoice) would not exist without the prior existence of object A (e.g., a quotation). Thus, the algorithm first identifies connected components of transactional data and, afterwards, adds all *master data* (e.g., `Customer`, `Employee`, `Product`) that is directly connected to one of the component's vertices. In consequence, every transactional vertex belongs to exactly one graph while master data instances may be part of many graphs. The algorithm's only requirement is the categorization of vertices to represent either master or transactional data. This categorization is done by a domain expert at the class level and taken over by their instances.

Due to the bad availability of datasets from real business information systems, I designed and implemented FoodBroker [17], a data generator based on business process simulation. The generated data's schema is inspired by real business information systems. Further on, every master data object has a quality criterion and will, if participating, influence the process execution positively or negatively. For example, the more poor master data objects interact in a process the higher is the chance for a bad process outcome like financial loss. Thus, data generated by FoodBroker is suitable to evaluate the BIIIG approach.

2.3 Business Measure Aggregation

To analyze graph collections, first, measures need to be calculated on the graph-level. For this reason, I proposed the *graph aggregation* operation [14, 16]. Aggregation derives a scalar value from an input graph's vertices and edges including labels and properties, e.g., to count contained vertices of a certain type or to sum all values of a specified property. The actual calculation is specified by a user-defined function γ that is executed for every graph of a collection. For example, the attributes `isClosed` and `soCount` attached to the graphs of Figure 2 represent the results of two different aggregation functions $\gamma_{\texttt{isClosed}}$ and $\gamma_{\texttt{soCount}}$. While $\gamma_{\texttt{soCount}}$ counts vertices of type `SalesOrder`, $\gamma_{\texttt{isClosed}}$ will check, if the graph contains a closed sales quotation, i.e., if the sales process is finished. The result of an aggregation function can be used to filter a graph collection. In our example, only graphs with $\gamma_{\texttt{isClosed}} = true$ were selected to apply $\gamma_{\texttt{soCount}}$. Since vertices of type `SalesOrder` only exist in the case of a confirmed (won) `Quotation`, this aggregation result can be used to categorize graphs into won ($\gamma_{\texttt{soCount}} > 0$) and lost ($\gamma_{\texttt{soCount}} = 0$) ones.

2.4 Scalable Frequent Subgraph Mining

To find correlations between certain business measures values and graph patterns, pattern frequencies need to be computed. This primitive operation is the well known problem of frequent subgraph mining [5]. Since the problem is NP-complete and graph collections in business applications can be very large I required a massive parallel solution to minimize total response times. There are three distributed

approaches to (exact and complete) frequent subgraph mining based on MapReduce [4, 11, 12]. However, none of these approaches is capable to mine directed multigraphs.

Thus, I discussed an extension of the popular gSpan algorithm [22] to support directed multigraphs in [15] and proposed DIMSpan [18], the first approach to frequent subgraph mining based on distributed in-memory dataflow systems like Apache Spark [23] or Apache Flink [2]. In comparison to the existing MapReduce based approaches, DIMSpan not only requires fewer disk access but also shuffles less data over the network and can reduce the total number of expensive isomorphism resolutions to a minimum. In experimental evaluations I have shown that a lightweight data structure as well as effective and fast compression techniques based thereon are key techniques for good scalability in big data scenarios. Figure 3 shows example evaluation results of DIMSpan. The chart on the left hand side shows a perfect scalability for increasing input data volume, since computation time for a portion of 100K graphs is decreasing for a growing number of graphs at different minimum support thresholds s_{min}. The chart on the right hand side shows good speedup for an increasing cluster size.

2.5 Category Characteristic Patterns

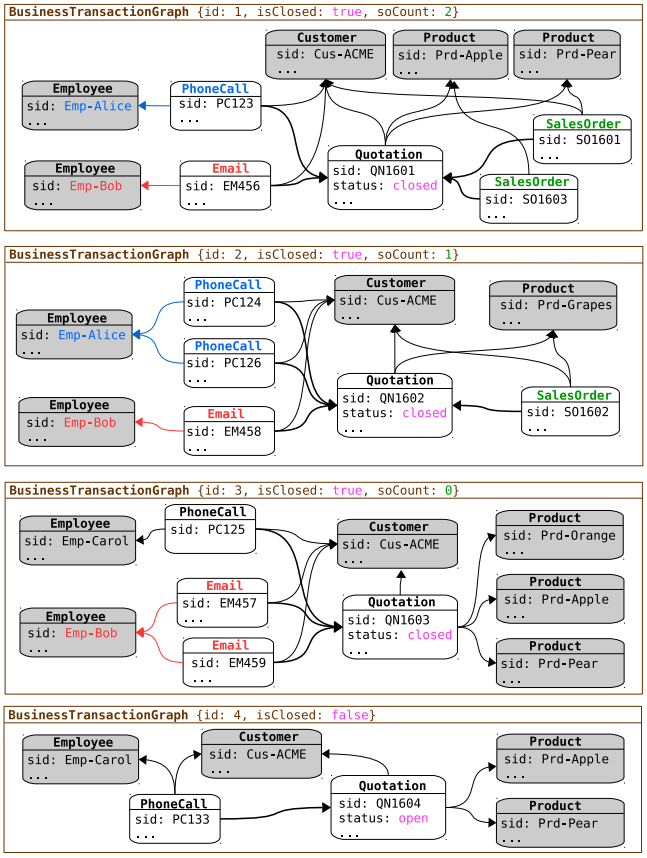
Since we are able to categorize graphs based on aggregated measures and can compute pattern frequencies, we can also mine correlations between categories and certain patterns. In [14] I proposed an analytical workflow to identify such *category characteristic patterns*. Figure 2 shows four example graphs where the top 3 represent finished executions of a sales process ($\gamma_{\texttt{isClosed}} = true$) categorized into won ($\gamma_{\texttt{soCount}} > 0$) and lost ones ($\gamma_{\texttt{soCount}} = 0$). Blue and red color are used to highlight example patterns. The pattern in blue color represents *'a phone call made by Alice'* and the one in red color *'an email sent by Bob'*. To enable the extraction of patterns combining labels and values of certain properties I additionally use a specific transformation between categorization and mining.

In contrast to basic frequent subgraph mining, I require patterns not just to be frequent but to be characteristic for a measure category. For example, the blue pattern is interesting, as it occurs in all of the won cases but not in the lost one. By contrast, the red pattern occurs in all graphs of both categories and, thus, is considered to be trivial. Therefore, I use an *interestingness measure* comparing a pattern's frequency in different categories. The measure is a function that evaluates the relative support of a pattern within a category in relation to its average relative support in all categories. Based on this measure, the analyst sets an *interestingness threshold* to prune patterns by minimum interestingness. Additionally, there is a *candidate threshold* to specify the minimum support of a pattern inside a category to be considered as a candidate. This parameter is used to save computations in exchange for result completeness.

2.6 Framework Integration

The implementation of the initial prototype [19] only covered data integration and simple analytical queries. To find a suitable platform for complex workflows including measure calculation and pattern mining, I performed an in-depth comparison of graph databases [7] and examined the suitability of different graph processing technologies [14]. I found out that none of the existing systems could satisfy my

Figure 2: Example Business Transaction Graphs [14]



requirements, especially they miss support for graph collections and graph properties. Thus, I joined the development of Gradoop [9], a scalable framework supporting complex workflows [15] of multiple operations on both graphs and graph collections.

The aggregation and vertex fusion operators proposed in [16] became part of Gradoop's *extended property graph model*. Additionally, DIMSpan [18] as well as the algorithms to extract business transaction graphs [16] and the one to identify category characteristic patterns [14] were implemented to fit a dedicated interface for plug-in algorithms and are part of Gradoop's open source repository[1]. Besides operators related to the BIIIG approach, the framework provides further valuable analytical operators such as *graph grouping* [8] and *graph pattern matching* [6].

3. PROBLEMS AND FUTURE WORK

In first evaluations of mining characteristic patterns from FoodBroker data I found out that the expected result was returned but the number of patterns quickly became very large and may overwhelm analysts. However, I was already able to identify two particular "data science" problems and their potential solutions: First, the method described in Section 2.5 eliminates trivial patterns for each category but not combinations of trivial and characteristic patterns. Thus, I'll investigate ranking results using a fast analytical method of graph p-value calculation [13]. Based thereon most significant patterns should be presented first.

[1] `www.gradoop.com`

Second, patterns should contain different levels of dimensional attributes. To provide a simple example, on the one hand an analyst won't be interested in the pattern *bread* and *butter*, if there are more specific patterns like *wholegrain bread* and *butter*. On the other hand, if *bread* and *butter* is not returned, the more general pattern of *bakery products* and *butter* could be. Thus, I will extend the DIMSpan algorithm to mine dimensional attributes across multiple levels. This approach has already been studied for itemsets [3] but not for graphs.

Finally, I will evaluate BIIIG in a real world scenario in cooperation with a large-scale enterprise. The evaluation will be based on GRADOOP and cover all steps of my approach. The company will not only provide real business data but also valuate analytical results and scalability.

4. SUMMARY

In my past PhD research, I contributed to the fields of graph data management and graph data mining. In contrast to other graph-based approaches to business intelligence [1, 21], BIIIG covers all steps from data integration to analytical results and requires no advance definition of an analytical schema. To the best of my knowledge, I proposed the first approach to integrate data from multiple source into a single instance graph and the first one using metadata-driven automation. Further on, I was the first who discussed the usage of graph collections to analyze the structure of interrelated business objects and to enable novel data mining techniques based thereon. Additionally, I presented the first horizontally scalable approach to transactional frequent subgraph mining using a distributed in-memory dataflow system and the first supporting directed multigraphs. To finish my PhD research, I will improve applicability by returning cross-level results and ranking them by significance.

5. ACKNOWLEDGMENTS

This work is partially funded within the EU program *Europa fördert Sachsen* of the European Social Fund and by the German Federal Ministry of Education and Research under project ScaDS Dresden/Leipzig (BMBF 01IS14014B).

6. REFERENCES

[1] D. Bleco and Y. Kotidis. Business intelligence on complex graph data. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pages 13–20. ACM, 2012.

[2] P. Carbone et al. Apache flink: Stream and batch processing in a single engine. *Data Eng.*, 38(4), 2015.

[3] T. Eavis and X. Zheng. Multi-level frequent pattern mining. In *International Conference on Database Systems for Advanced Applications*, pages 369–383. Springer, 2009.

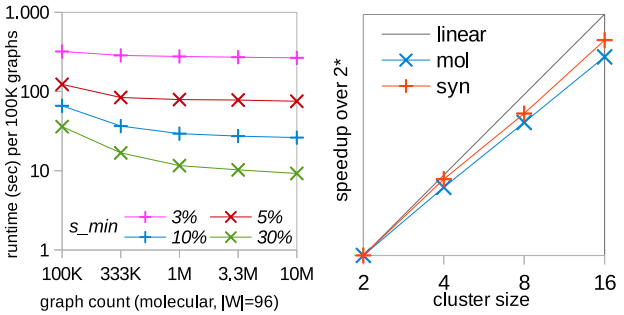
[4] S. Hill et al. An iterative mapreduce approach to frequent subgraph mining in biological datasets. In *Conference on Bioinformatics, Computational Biology and Biomedicine*, pages 661–666. ACM, 2012.

[5] C. Jiang, F. Coenen, and M. Zito. A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, 28(01):75–105, 2013.

[6] M. Junghanns, M. Kieling, A. Averbuch, A. Petermann, and E. Rahm. Cypher-based graph pattern matching in gradoop. In *Proc. 5th Int. Workshop on Graph Data Management Experiences and Systems*. ACM, 2017.

[7] M. Junghanns, A. Petermann, N. Neumann, and E. Rahm. Management and analysis of big graph data: Current systems and open challenges. *Big Data Handbook, Springer*, 2017.

Figure 3: DIMSpan: Scalability for increasing data volume (left) and cluster size (right) [18]



[8] M. Junghanns, A. Petermann, and E. Rahm. Distributed grouping of property graphs with gradoop. In *17th Conference on Database Systems for Business, Technology, and Web (BTW)*, 2017.

[9] M. Junghanns, A. Petermann, N. Teichmann, K. Gómez, and E. Rahm. Analyzing extended property graphs with apache flink. In *1st SIGMOD Workshop on Network Data Analytics*, page 3. ACM, 2016.

[10] G. Klyne and J. J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. 2006.

[11] W. Lin, X. Xiao, and G. Ghinita. Large-scale frequent subgraph mining in mapreduce. In *Int. Conf. on Data Engineering (ICDE)*, pages 844–855. IEEE, 2014.

[12] W. Lu, G. Chen, A. Tung, and F. Zhao. Efficiently extracting frequent subgraphs using mapreduce. In *Int. Conf. on Big Data*, pages 639–647. IEEE, 2013.

[13] G. Micale, R. Giugno, A. Ferro, M. Mongioví, D. Shasha, and A. Pulvirenti. Fast analytical methods for finding significant colored graph motifs. *To be published in Data Mining and Knowledge Discovery*, 2017.

[14] A. Petermann and M. Junghanns. Scalable business intelligence with graph collections. *it-Information Technology*, 58(4):166–175, 2016.

[15] A. Petermann, M. Junghanns, S. Kemper, K. Gómez, N. Teichmann, and E. Rahm. Graph mining for complex data analytics. In *Int. Conf. on Data Mining Workshops (ICDMW)*, pages 1316–1319. IEEE, 2016.

[16] A. Petermann, M. Junghanns, R. Müller, and E. Rahm. BIIIG: Enabling Business Intelligence with Integrated Instance Graphs. In *Int. Conf. on Data Engineering Workshops (ICDEW)*, pages 4–11. IEEE, 2014.

[17] A. Petermann, M. Junghanns, R. Müller, and E. Rahm. Foodbroker-generating synthetic datasets for graph-based business analytics. In *Workshop on Big Data Benchmarks*, pages 145–155. Springer, 2014.

[18] A. Petermann, M. Junghanns, and E. Rahm. Dimspan-transactional frequent subgraph mining with distributed in-memory dataflow systems. *arXiv preprint arXiv:1703.01910*, 2017.

[19] A. Petermann et al. Graph-based Data Integration and Business Intelligence with BIIIG. *PVLDB*, 7(13), 2014.

[20] M. A. Rodriguez and P. Neubauer. Constructions from dots and lines. *Bulletin of the American Society for Information Science and Technology*, 36(6):35–41, 2010.

[21] Z. Wang et al. Pagrol: Parallel graph olap over large-scale attributed graphs. In *30th Int. Conf. on Data Engineering (ICDE)*, pages 496–507. IEEE, 2014.

[22] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *International Conference on Data Mining (ICDM)*, pages 721–724. IEEE, 2002.

[23] M. Zaharia et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proc. of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2, 2012.