13TH

ALBERTO MENDELZON INTERNATIONAL WORKSHOP ON FOUNDATIONS OF DATA MANAGEMENT

Asunción, Paraguay June 03 - 07, 2019

CONSEJO NACIONAL DE CIENCIA Y TECNOLOGÍA
CONACYT

PROCIENCIA
PROGRAMA PARAGUAYO PARA EL DESARROLLO DE LA CIENCIA Y TECNOLOGÍA

cds
CENTRO DE DESARROLLO SOSTENIBLE S.A.

Universidad Católica
"Nuestra Señora de la Asunción"

UNIVERSIDAD AUTÓNOMA DE ASUNCIÓN

Proceedings of the 13th Alberto Mendelzon International Workshop on Foundations of Data Management

Asunción, Paraguay

June 2019

ISSN 1613-0073

Diagramming: Ing. Julia Talavera

This publication has been prepared with the support of CONACYT. The content of this proceeding is the exclusive responsibility of the authors and in no case should be considered as the opinion of CONACYT.

La presente publicación ha sido elaborada con el apoyo del CONACYT. El contenido de la misma es responsabilidad exclusiva de los autores y en ningún caso se debe considerar que refleja la opinión del CONACYT.

This event is co-financed by Consejo Nacional de Ciencia y Tecnología - CONACYT with FEEI resources.

Este evento es cofinanciado por el Consejo Nacional de Ciencia y Tecnología - CONACYT con recursos del FEEI.

Table of Contents

Part I

Preface

The Alberto Mendelzon Workshop is an international scientific venue started in 2006 to honor the memory of Alberto Mendelzon, who made a significant contribution to the field of data management. It has been an established forum for top-quality research on the foundations of data management. While focused especially on Latin American students and scholars, the workshop is open to submissions from around the world, and it has so far gathered some of the world's best researchers in the field.

This volume contains papers accepted for the 13th edition of the AMW, held in Asuncion, Paraguay, from June 3 to 7, 2019. As in previous editions, the call for papers of this edition solicited two types of submissions: regular and short papers, where the latter was intended to present ongoing research, results published elsewhere, and applications. The workshop received 28 submissions from around the globe. The vast majority of these submissions were of high quality and reported on intriguing on-going or latest research development on diverse data management topics. Following a rigorous reviewing effort, whereby each submission has been allocated at least three reviewers, the program committee accepted 23 papers to form an excellent workshop program with five sessions: Foundations of Data Management, The Web, Graph Data, Data Mining and Algorithms and Optimisation.

Each accepted paper has a presentation slot at the workshop and may appear in the workshop proceedings. The paper presentations were accompanied by four keynote sessions, featuring Magdalena Ortiz (TU Wien, Austria) on "From Complete to Incomplete Data and Back in Ontology-Enriched Databases", Wang-Chiew Tan (Megagon Labs, USA) on "Enabling Search by Experience", Juan Reutter (Pontificia Universidad Catolica de Chile, Chile) on "Schemas for Graphs and other forms of Semi-Structured Data", and Christoph Koch (EPFL, Switzerland) on "Query Optimisation by Quantifier Elimination".

As in previous editions, AMW 2019 hosted the Alberto Mendelzon Workshop School (AMWS), which consists of four three-hour tutorials given by invited international speakers. The school was held before AMW, on June 5 - 7, 2019. This year's tutorials were on: Deep Learning for Natural Language Processing by Felipe Bravo-Marquez (University of Chile, Chile); Data preparation and data integration: BigGorilla to the rescue by Wang-Chiew Tan (Megagon Labs, USA); Building AI Applications using Knowledge Graphs by André Freitas (University of Manchester, UK); and Languages for Data and Knowledge by Magdalena Ortiz (TU Wien, Austria).

We would like to warmly thank: the authors of the submitted papers; the presenters at the workshop; the keynote speakers; Adriana Marotta (Universidad de la Republica, Uruguay), the General Chair of AMW 2019; the fantastic and professional Program Committee; the AMW School Committee; the AMW Steering Committee; and the local supporting institutions especially to Consejo Nacional de Ciencia y Tecnología (abbreviated CONACYT) . Without their effort, AMW 2019 would not have been successful.

This event is co-financed by Consejo Nacional de Ciencia y Tecnología - CONACYT with FEEI resources.

Este evento es cofinanciado por el Consejo Nacional de Ciencia y Tecnología - CONACYT con recursos del FEEI.

Part II

Keynotes

From Complete to Incomplete Data and Back in Ontology-Enriched Databases

Magdalena Ortiz

Institute of Information Systems, TU Wien, Austria

Abstract. Enriching a database with a background theory expressing domain knowledge, usually called an ontology, has been proposed as a tool to overcome the incompleteness of data. In ontology mediated querying the theory is used to infer answers that may involve implied facts not present in the data. This and other related reasoning problems have been extensively studied over the last decade, mostly for ontologies written in description logics and in dialects of Datalog$^{\pm}$. But the usual first-order semantics used in this setting, which assumes that all data is incomplete, can sometimes be too weak and not give all expected answers. I will discuss some alternatives that have been explored for combining complete and incomplete data in the presence of description logic ontologies, and the challenges that they pose, including increased computational complexity of reasoning and non-monotonicity of the ontology mediated query languages they induce. I will discuss a few interesting reasoning problems that arise in these setting, and some translations from these query languages into variants of Datalog.

Enabling Search by Experience

Wang-Chiew Tan

Megagon Labs

Abstract. Today's online shopping systems enable consumers to sift through a vast amount of information by manipulating combinations of predefined filters. These filters, such as travel dates, price range, and location, are objective attributes that lead to an indisputable set of answers. However, we show that users' search criteria are often subjective and experientially expressed. Hence, to provide consumers with an enhanced search experience, online shopping systems should directly support both subjective and objective search. I will describe how this is done in an experiential search engine that we are currently developing at Megagon Labs; by harnessing information "outside the box", in the text of online reviews or social media, views, and interpreting subjective queries.

Schemas for Graphs and Other Forms of Semi-Structured Data

Juan L. Reutter

IMFD; Pontificia Universidad Católica de Chile

Abstract. Semistructured data is on the rise: Graph databases are now offered by most major database vendors, and JSON is used everywhere on the web. And while these data paradigms are commonly described as "having less structure than relational databases", the industry is also recognising the advantages of pairing the data with some notion of schema, in the form of metadata that would describe both what is in the database and how is the data structured. In this talk we focus on the SHACL/ShEx proposal for RDF graph data and JSON Schema for JSON data, two of the most adopted proposals for semi-structured data. Interestingly, even though these proposals have spanned in two completely different worlds, we show that they share the same foundations and the same spirit. We will also discuss about the challenges that arise from features demanded in these proposals by the community, but that give rise to a number of interesting open problems.

Query Optimization by Quantifier Elimination

Christoph Koch

École Polytechnique Fédérale de Lausanne

Abstract. Many of us who teach database query languages have seen creative students who lack a training in formal logic come up with surprising ways of using aggregation for expressing challenging queries in SQL – ways that do not feel natural to those trained in logic but which nevertheless exact admiration. In this talk, I show how quantifier elimination can be used to optimize SQL queries in surprising ways – ways whose results coincide with and generalize these apparently creative tricks. The new query optimization technique, apart from being potentially useful for practical query engines, suggests a particular way in which the logically untrained mind synthesizes queries (not quantifier elimination, though) – an observation at best based on an amateur's understanding of brain science, but potentially still useful for teaching databases.

Part III

Regular Papers

RDF and Property Graphs Interoperability: Status and Issues

Renzo Angles[1,2], Harsh Thakkar[3], Dominik Tomaszuk[4]

[1] Universidad de Talca, Chile
[2] Millennium Institute for Foundational Research on Data, Chile
[3] University of Bonn, Germany
[4] University of Bialystok, Poland
rangles@utalca.cl[1], thakkar@cs.uni-bonn.de[3], d.tomaszuk@uwb.edu.pl[4]

Abstract. RDF and Property Graph databases are two approaches for data management that are based on modeling, storing and querying graph-like data. In this paper, we present a short study about the interoperability between these approaches. We review the current solutions to the problem, identify their features, and discuss the inherent issues.

1 Introduction

RDF [24] and graph databases [37] are two approaches for data management that are based on modeling, storing and querying graph-like data. Several database systems based on these models are gaining relevance in the industry due to their use in several domains where graphs and network analytics are required [6].

Both, RDF and graph database systems are tightly connected as they are based on graph-oriented database models. On the one hand, RDF database systems (or triplestores) are based on the RDF data model [24], their standard query language is SPARQL [19], and there are languages to describe structure, restrictions and semantics on RDF data (e.g. RDF Schema [13], OWL [18], SHACL [25], and ShEx [11]). On the other hand, most graph database systems are based on the Property Graph (PG) data model [7], there is no standard query language (although there are several proposals [4]), and the notions of graph schema and integrity constraints are limited [32]. Therefore, these two groups of systems (in particular the latter) are dissimilar in data model, schema, query language, meaning and content.

Given the heterogeneity between RDF and graph database systems, it results necessary to study the interoperability among them, i.e. the ability of these systems to exchange data, information (structure and semantics) and knowledge (constraints and business rules).

The main objective of this paper is to present an overview of the research concerning the interoperability between RDF and property graph databases. First, we clarify the notion of database interoperability, identifying three types: syntactic interoperability, semantic interoperability, and query interoperability (Section 2). Second, we present a short review of the current approaches and

works, including data format transformations, data and/or schema exchange, and query translations. (Section 3). Third, we isolate and discuss the main issues and challenges in the topic (Section 4).

2 Database Interoperability

The term "Interoperability" was introduced in the area of information systems, and it could be defined as the the ability of two or more systems or components to exchange information, and to use the information that has been exchanged [3]. In the context of data management, interoperability is concerned with the support of applications which exchange and share information across the boundaries of existing databases [38].

Providing interoperability between database models, systems and applications is a very concrete and pragmatic problem, which stems from the need of reusing existing systems and programs for building new applications [38]. Data and information interoperability is relevant for several reasons, including:

- Promotes data exchange and data integration [30];
- Allows to have a common understanding of the meanings of the data [22];
- Allows the creation of information and knowledge, and their subsequent reuse and sharing [40];
- Facilitates the access to a large number of independently created and managed information sources of broad variety [40];
- Facilitates the reuse of available systems and tools [38];
- Allows to explore the best features of different approaches and systems [31];
- Enables a fair comparison of database systems by using benchmarks [5];
- Supports the success of emergent systems and technologies [38];
- It is a crucial factor for the development of new information systems [29].

One can define several forms of interoperability in information systems [28]. For instance, focusing in the dimension of heterogeneity, Sheth [40] defined four levels of interoperability: system, syntax, structure and semantic. The system level concerns the heterogeneity of computer systems and communications. The syntax level considers machine-readable aspects of data representation (i.e. data formats and serializations). The structure level involves data modeling constructs and schematic heterogeneity. The semantic level requires that the information system understand the semantics of the use's information request and those of information sources.

In the context of Web Languages and Ontologies, the syntactic interoperability means that the applications can take advantage of parsers and APIs providing syntactical manipulation facilities. Additionally, semantic interoperability implies that applications can understand the meaning of representations and thus can setup automatically mappings between different representations by content analysis [33].

In the context of databases, interoperability can be divided into syntactic, semantic and query interoperability. *Syntactic interoperability* refers to the ability of a database system to use data from other database system [23]. It could

means that both database systems are able to exchange information, although they may not being aware of the meaning of such information. *Semantic interoperability* can be defined as the ability of database systems to exchange data in a meaningful way. It implies that the systems have a common understanding of the meanings of the data [22]. *Query interoperability* implies the existence of methods to transform different query languages or data accessing methods between two systems. It means that a query in the source database system can be translated into one that can be directly executed on the target system [48].

3 RDF and Property Graphs interoperability

In general terms, syntactic interoperability between RDF and PG databases means data exchange at the level of serialization formats, semantic interoperability implies the definition of data and schema mappings, and query interoperability implies query translations among SPARQL and property graph query languages. This section presents a review of the approaches and methods proposed for these types of interoperability.

3.1 RDF databases

Every RDF database system is designed to support the Resource Description Framework [24], a W3C standard created to describe web resources, although it could be used in any application domain. RDF defines a data model which is based on the notion of RDF triple. An RDF triple is a tuple formed by a subject, a predicate, and an object. The *subject* denotes a resource, the *predicate* refers to an attribute or relationship of the subject, and the *object* defines the value for such property. A collection of RDF triples could be visualized a graph where the subjects and objects are represented as nodes, and the predicates are represented as edges. An RDF database could be considered as a collection of RDF graphs. RDF reification is a feature which means to create triples about triples, here metadata (e.g. temporal, uncertainty and trust metrics).

The RDF Schema vocabulary [13] provides a simple way to describe the structure of an RDF database. In this case, the schema is described as a collection of a resource classes and property classes. Moreover, the classes could be hierarchically organized by using *subclass* and *subproperty* relationships. More complex restrictions can be expressed in languages like OWL [18], SHACL [25] and ShEx [34]. These languages provide semantic interpretations that allow to infer additional triples. This feature is called RDF(S) entailment.

SPARQL is the standard query language to retrieve and manipulate RDF data. The first version (SPARQL 1.0 [35]) provides basic operators to express graph pattern matching. The second version (SPARQL 1.1 [19]) adds features like explicit negation, path expressions, subqueries and aggregate operators.

3.2 Property graph databases

Most of the current graph database systems have been designed to support the property graph data model. A property graph [7] is a directed labelled multigraph with the special characteristic that each node or edge could maintain a set (possibly empty) of properties. A property is formed by a name and a value.

The notion of schema for a property graph database has not been developed in practice, although some systems use the notions of node types and edge types. Integrity constraints are also in development. In [32], the authors mention three types of integrity constraints: inherent constraints, explicit constraint and implicit constraint. Additionally, we found node/edge/property constraints, path constraints, and graph pattern constraints with property values [12].

In spite of the extensive research on querying graph databases [4], there is no standard query language for property graphs. A recent publication, called the GQL manifest [2], proposes to define and standardize one property graph query language by fuzing the best of three query languages: Cypher [1], PGQL [36] and G-CORE [8].

3.3 RDF-PG Syntactic interoperability

Assume that the syntactic interoperability is given by the facilities to transform data from one format to another. Hence, the main requirement to support syntactic interoperability is the existence of data formats (i.e. a syntax for encoding data stored in a database), over which transformation methods can be implemented.

Turtle, TriG, RDF/XML, RDF/JSON and JSON-LD are data formats for encoding RDF data. In contrast, there is no data format to encode property graphs. Given this restriction, some systems use graph data formats (like GraphML, DotML, GEXF, GraphSON), but none of them is able to cover all the features presented by the property graph data model. YARS-PG [47] is a recent proposal of data format which was designed to be simple, extensible and platform independent, and to support all the features provided by the current database systems based on the property graph data model.

Given a source data format S and a target data format T, the first option to support syntactic interoperability is to define a textual mapping from S to T. Note that the schema of the database is not considered in the transformation. Hence, the structure, semantics and restrictions of the source data could not be preserved by the translated data.

Hartig [20] proposes two transformations between RDF* and property graphs. RDF* is a syntactic extension of RDF which is based on reification. The first transformation maps any RDF triple as an edge in the resulting property graph. Each node has the "kind" attribute to describe the type of a node (e.g. IRI). The second transformation distinguishes data and object properties. The former are transformed into node properties, and latter into edges of a property graph. The limitation of the second transformations is that metadata triples cannot be transformed. The shortcoming of this approach is that RDF* isn't supported by

majority of RDF triplestores (except Blazegraph and the most recent addition, AnzoGraph) and requires conversion of existing RDF data beforehand.

Schätzle et al. [39] propose a mapping which is native to GraphX (a parallel processing system implemented on Apache Spark). The proposed graph model is an extension of the regular graph, but lacking the concept of attributes. The mapping uses an special attribute *label* to store the node and edge identifiers, i.e. each triple $t = (s, p, o)$ is represented using two vertices v_s, v_0, an edge (v_s, v_o) and properties $v_s.label = s$, $v_o.label = o$, $(v_s, v_o).label = p$. The proposed method does not address blank nodes or RDF entailment.

Nyugen et al. [27] proposed *LDM-3N* (labeled directed multigraph-three nodes), a graph model for RDF data. It is an extension of the regular graph, without the concept of attributes, and represents each triple element as separate nodes, thus three nodes (3N) . The *LDM-3N* graph model is used to address the Singleton Property (SP) based reified RDF data.

Tomaszuk [46] presented an approach that uses the *YARS* serialization for transforming RDF data into property graphs. This approach basically performs a transformation between encoding schemes and does not consider the RDF schema and its entailments. This approach has several implementations, eg. neo4j-yars[5] and TTL2YARS[6].

With respect to the methods to transform property graphs into RDF graphs, the literature is very restricted. The current methods [16,20] are based on reification. In the simplest case, for each edge in the property graph there will be a blank node (in the RDF graph) containing at least three nodes (resources or literals) and three edges (properties). Such elements will be necessary to describe all the information of the original edge.

A additional approach to provide syntactic interoperability is the use of an intermediate data format. It is possible to find some tools to transform RDF into other formats and vice versa, eg. Triplify [9] for relational data, GRDDL [14] for XML and CSVW [42] for tabular data. However, to the best of our knowledge, there is not study about the subsequent transformation to property graphs.

3.4 RDF-PG Semantic interoperability

Semantic interoperability between databases means that both, source and target systems, are able to understand the meaning of the data to be exchanged. It implies that both, data and schema must participate of the transformations method.

A common approach to support semantic interoperability is the definition of data and schema transformation methods. The schema transformation method takes as input the schema of the source database, and generates a schema for the target database. Similarly, the data transformation method allows to move the data from the source database to the target database, but taking care of the target schema. The transformation methods can be implemented by using data

[5] `https://github.com/lszeremeta/neo4j-sparql-extension-yars`
[6] `https://github.com/lszeremeta/ttl-to-yars`

formats or data definition languages. To the best of our knowledge, there is no method that support data and schema transformations between RDF and PGs.

A additional approach is the use of a data transformation language. For instance, XSPARQL [10], and SPARQL Template Transformation Language (STTL) [15] are languages that allow data transformation between RDF and other languages or formats. In the opposite direction, RML [17] is a generic language which allows to define mappings from heterogeneous sources to RDF. In a recent article [26], the authors present the Graph to Graph Mapping Language (G2GML) for mapping RDF graphs to property graphs. This language can be processed by an implementation called G2G Mapper (available on `https://github.com/g2gml`). There is no formal definition nor analysis of the features of this transformation language.

3.5 RDF-PG Query interoperability

Query interoperability between RDF and property graph databases is a current issue due to the lack of a standard query language for property graphs. Gremlinator [44,45] is a tool that translates SPARQL queries into Gremlin pattern matching traversals. Gremlin is a popular language used by some graph database systems and graph processing frameworks. Gremlinator [44] has been successfully integrated as a plugin of the famous Apache TinkerPop graph computing framework[7]. Given the above, the openCypher initiative is working on the Cypher to Gremlin translation (`https://github.com/opencypher/cypher-for-gremlin`).

Hartig et al. [21] defined extensions of the SPARQL query language that capture an alternative approach to represent statement-level metadata that can be used in property graphs (see [20]). This proposal, called SPARQL* is an RDF*-aware extension that introduces new features that enable users to directly access metadata triples in queries.

4 Issues and challenges

Based on our literature review about RDF and property graphs interoperability, we identified several issues and challenges which will be discussed in this section.

4.1 Syntactic interoperability

- There is no standard data format for encoding property graphs. This is a crucial issue to support syntactic interoperability.
- The most RDF serializations are triple-centric, while the most PG serializations represent graph as lists of nodes and edges.
- Despite the serializations based on JSON or XML in both models, the syntaxes used are difficult to map.

[7] The `sparql-gremlin` plugin of the Apache TinkerPop framework available on Github – (`https://github.com/apache/tinkerpop/tree/master/sparql-gremlin`)

– The support for multi-values is different in the models. A property graph just support arrays, while RDF provides different types of lists.
– The RDF data model allows metadata about properties, i.e. edges between edges are allowed. Although this feature is not common in real data, a data mapping should be able to manage it. Note that a property graph does not support multi-level metadata.
– RDF reification leads to an explosion in the size of the resulting graph. This can be avoided by implementing a "smart" transformation that is able to recognize a set of triples describing a reification, and map them to a single node in the property graph.

4.2 Semantic interoperability

– The RDF model presents features with special meaning (or semantics) that cannot be modeled by the property graph data model (at least not in a easy way). Blank nodes, reification, and entailment are some of these features.
– Usually, an RDF database contains a mix of data and schema. In such case, it is necessary to decide whether to extract the schema (and transforming it independently), or processing the schema as part of the data.
– Another intrinsic feature of an RDF database is the occurrence of a partial schema. In such case, we must define whether the schema will be used or not. In the first case, it could be necessary to "discover" the schema, and just then transform the data. Hence, such an approach could imply the use of a transformation method that is schema independent, or a combined method that supports data with or without schema.
– A semantic issue is the right and complete interpretation of a reified triple, and its representation in a property graph.
– RDF Schema supports the definition of subclass and subproperty. These features are not supported by current property graph database systems.
– OWL, that is intended to be a layer above RDF Schema, supports more complex constraints for classes (e.g. intersection) and properties (e.g. transitivity). These features are not supported by the property graph model.
– An RDF database could contain semantic information that allows data inference (i.e. to infer new triples based on the existing triples). Current graph database systems have been not designed to support inference.
– Discovering semantic information and resolving mismatches requires the application of human intelligence and judgment. Hence, the semantic interoperability is determined by the power of the translation methods to support data and semantics interpretation.

4.3 Query interoperability

– Unlike the standardisation (via the W3C standards and ISO committees) of query languages for the relational databases (SQL) and the RDF databases (SPARQL), property graph databases do not have a standard query language. This has led to the development of a wide range of vendor-specific

graph query languages (e.g. Cypher for Neo4j and Gremlin for Apache TinkerPop).
- Most of the current property graph query languages do not have a solid formal foundation (semantics, complexity and expressiveness). This raises a critical challenge for supporting query interoperability, since a formal mapping between SPARQL and a property graph language cannot be defined.
- The notion of schema in the context of property graph query languages is not strictly defined, or even absent in some cases due to their NoSQL oriented nature. This creates another challenge when aiming to transform RDF data (which consists of schema information) to property graph data.
- Property graph query languages address two different paradigms: *declarative* and *imperative*. For instance, Cypher is a declarative query language, whereas Gremlin is an imperative graph traversal language that also offers a declarative construct. This adds an additional challenge, since these two different paradigms operate on disparate sets of semantics (i.e. set vs bag semantics), while aiming to support query interoperability.
- There are some on-going efforts, such as [41,43], that advocate consolidating the relational and graph algebras in order to lay a foundation for proving the equivalences between the different transformations and mappings to support *query interoperability* between RDF and Property graphs. Nonetheless, there is still scope for improvement.

Therefore, there is a need to propose a standardized query language for property graph databases. It will facilitate the formal definition and study of query transformation methods.

5 Conclusions

Interoperability is a very important feature that should be supported by any database systems. In this article we concentrate on the interoperability between RDF databases and property graph databases. Our analysis of the available approaches and methods does not cover all the issues and challenges, but shows that there are several problems to deal with.

The interoperability among systems is based on agreements between requesters and providers [22]. Hence, the research on the area must be supported by the development of successful standards (starting with the standardization of the property graph data model and its query language).

Acknowledgments. Renzo Angles was (partially) funded by the Millennium Institute for Foundational Research on Data (IMFD). Harsh Thakkar was funded by the EU H2020 R&I project BOOST4.0 (GA 780732). Dominik Tomaszuk was supported by the National Science Center, Poland (NCN) under research grant Miniatura 2.

References

1. Cypher query language reference, version 9. https://s3.amazonaws.com/artifacts.opencypher.org/openCypher9.pdf
2. The GQL Manifesto. https://gql.today/
3. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. IEEE Std 610 (Jan 1991)
4. Angles, R., Arenas, M., Barceló, P., et al.: Foundations of modern query languages for graph databases. ACM Computing Surveys (CSUR) 50(5) (Sept 2017)
5. Angles, R., Boncz, P., Larriba-Pey, J., Fundulaki, I., Neumann, T., Erling, O., Neubauer, P., Martinez-Bazan, N., Kotsev, V., Toma, I.: The Linked Data Benchmark Council: a Graph and RDF industry benchmarking effort. Sigmod Record 43(1) (March 2014)
6. Angles, R., Gutierrez, C.: An introduction to graph data management. In: Graph Data Management, chap. 1. Data-Centric Systems and Applications, Springer Nature (2018)
7. Angles, R.: The property graph database model. In: Proc. Alberto Mendelzon International Workshop on Foundations of Data Management (AMW) (2018)
8. Angles, R., Arenas, M., Barceló, P., Boncz, P., Fletcher, G., Gutierrez, C., Lindaaker, T., Paradies, M., Plantikow, S., Sequeda, J., van Rest, O., Voigt, H.: G-CORE: A core for future graph query languages. In: Proc. of the International Conference on Management of Data (SIGMOD) (2018)
9. Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., Aumueller, D.: Triplify: Light-weight Linked Data Publication from Relational Databases. In: Proc. of the 18th International Conference on World Wide Web. pp. 621–630. ACM, New York, NY, USA (2009)
10. Bischof, S., Decker, S., Krennwallner, T., Lopes, N., Polleres, A.: Mapping between RDF and XML with XSPARQL. Journal of Data Semantics 1(3) (2012)
11. Boneva, I., Gayo, J.E.L., Hym, S., Prud'hommeau, E.G., Solbrig, H.R., Staworko, S.: Validating RDF with shape expressions. CoRR, abs/1404.1270 (2014)
12. Bonifati, A., Fletcher, G., Voigt, H., Yakovets, N.: Querying Graphs. Synthesis Lectures on Data Management, Morgan & Claypool Publishers (2018)
13. Brickley, D., Guha, R.V.: RDF Schema 1.1, W3C Recommendation (2014)
14. Connolly, D.: Gleaning Resource Descriptions from Dialects of Languages (GRDDL) - W3C Recommendation. https://www.w3.org/TR/grddl/ (September 11 2007)
15. Corby, O., Faron-Zucker, C.: Sttl: A sparql-based transformation language for rdf. In: Proc. of the 11th International Conference on Web Information Systems and Technologies (WEBIST) (2015)
16. Das, S., Srinivasan, J., et al.: A tale of two graphs: Property graphs as RDF in oracle. In: Proc. of the International Conference on Extending Database Technology (EDBT). pp. 762–773 (2014)
17. Dimou, A., Sande, M.V., Colpaert, P., Verborgh, R., Mannens, E., de Walle, R.V.: RML: A generic language for integrated rdf mappings of heterogeneous data. In: Proc. of the Linked Data on the Web Workshop (2014)
18. Group, W.O.W.: OWL 2 Web Ontology Language, Document Overview - W3C Recommendation. https://www.w3.org/TR/owl2-overview/ (December 11 2012)
19. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language - W3C Recommendation. https://www.w3.org/TR/sparql11-query/ (March 21 2013)
20. Hartig, O.: Reconciliation of RDF* and property graphs. arXiv:1409.3288 (2014)

21. Hartig, O., Thompson, B.: Foundations of an alternative approach to reification in rdf. arXiv preprint arXiv:1406.3399 (2014)
22. Heiler, S.: Semantic interoperability. ACM Comput. Surv. 27(2), 271–273 (1995)
23. Joundrey, D.N., Taylor, A.G.: The Organization of Information. Library and Information Science, Libraries Unlimited, fourth edn. (2017)
24. Klyne, G., Carroll, J.: Resource Description Framework (RDF) Concepts and Abstract Syntax. http://www.w3.org/TR/2004/REC-115-concepts-20040210/ (February 2004)
25. Knublauch, H., Kontokostas, D.: Shapes Constraint Language (SHACL) - W3C Recommendation. https://www.w3.org/TR/shacl/ (July 20 2017)
26. Matsumoto, S., Yamanaka, R., Chiba, H.: Mapping RDF graphs to property graphs. In: Proc. of the Fifth International Workshop on Practical Application of Ontology for Semantic Data Engineering (2018)
27. Nguyen, V., Leeka, J., et al.: A formal graph model for rdf and its implementation. arXiv:1606.00480 (2016)
28. Ouksel, A.M., Sheth, A.: Semantic interoperability in global information systems. SIGMOD Rec. 28(1), 5–12 (Mar 1999)
29. Parent, C., Spaccapietra, S.: Issues and approaches of database integration. Commun. ACM 41(5), 166–178 (May 1998)
30. Parent, C., Spaccapietra, S.: Database integration: the key to data interoperability. Advances in Object-Oriented Data Modeling (2000)
31. Park, J., Ram, S.: Information systems interoperability: What lies beneath? ACM Transactions on Information Systems 22(4), 595–632 (Oct 2004)
32. Pokorný, J., Valenta, M., et al.: Integrity constraints in graph databases. Procedia Computer Science 109, 975–981 (2017)
33. Predoiu, L., Zhdanova, A.V.: Semantic Web Languages and Ontologies, p. 7. Encyclopedia of Internet Technologies and Applications, IGI Global (2008)
34. Prud'hommeaux, E., Labra Gayo, J.E., Solbrig, H.: Shape expressions: An RDF validation and transformation language. In: Proceedings of the 10th International Conference on Semantic Systems (SEM). pp. 32–40. ACM, New York, NY, USA (2014)
35. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF - W3C Recommendation. https://www.w3.org/TR/rdf-sparql-query/ (January 15 2008)
36. van Rest, O., Hong, S., Kim, J., Meng, X., Chafi, H.: PGQL: a Property Graph Query Language. In: Proc. of the Int. Workshop on Graph Data Management Experiences and Systems (GRADES) (2013)
37. Robinson, I., Webber, J., Eifrem, E.: Graph Databases. O'Reilly Media, first edn. (June 2013)
38. S.Ceri, Tanca, L., Zicari, R.: Supporting interoperability between new database languages. In: Proc. of the 5th Annual European Computer Conference (CompEuro) (1991)
39. Schätzle, A., Przyjaciel-Zablocki, M., Berberich, T., Lausen, G.: S2X: Graph-parallel querying of RDF with GraphX. In: Biomedical Data Management and Graph Online Querying. pp. 155–168. Springer International Publishing (2016)
40. Sheth, A.P.: Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics, pp. 5–29. Springer US (1999)
41. Szárnyas, G., Marton, J., Maginecz, J., Varró, D.: Reducing property graph queries to relational algebra for incremental view maintenance. arXiv preprint arXiv:1806.07344 (2018)

42. Tandy, J., Herman, I., Kellogg, G.: Generating RDF from Tabular Data on the Web - W3C Recommendation. https://www.w3.org/TR/csv2rdf/ (December 17 2015)
43. Thakkar, H., Punjani, D., Auer, S., Vidal, M.E.: Towards an integrated graph algebra for graph pattern matching with Gremlin. In: International Conference on Database and Expert Systems Applications. pp. 81–91. Springer (2017)
44. Thakkar, H., Punjani, D., Lehmann, J., Auer, S.: Two for one: Querying property graph databases using SPARQL via gremlinator. In: Proc. of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA). pp. 1–5. ACM (2018)
45. Thakkar, H., Punjani, D., et al.: A stitch in time saves nine–SPARQL querying of property graphs using gremlin traversals. arXiv:1801.02911 (2018)
46. Tomaszuk, D.: RDF Data in Property Graph Model. In: Proc. of the 10th International Conference on Metadata and Semantics Research (MTSR). vol. 672 (2016)
47. Tomaszuk, D., Angles, R., Szeremeta, L., Litman, K., Cisterna, D.: Serialization for property graphs. In: Proc. of the 15th International Conference Beyond Databases, Architectures and Structures (BDAS) (2019)
48. Zhan, J., Luk, W.S., Wong, C.: An Object-Oriented Approach to Query Interoperability, pp. 141–153. Springer US (1996)

Trajectory Patterns Based on Segment-Cutting Clustering

Luis Cabrera-Crot[1], Mónica Caniupán[2], Andrea Rodríguez[1], and Diego Seco[1]

[1] Universidad de Concepción, Chile
[2] Universidad del Bío-Bío, Chile
luiscabrera@udec.cl, mcaniupan@ubiobio.cl, andrea@udec.cl, dseco@udec.cl

Abstract. Trajectory patterns characterize similar behaviors among trajectories, which play an important role in applications such as urban planning, traffic congestion control, and studies of animal migration and natural phenomena. In this paper we model trajectories as a sequence of line segments that represent the steady movement of an object along time. We use a segment-clustering process to group trajectories' segments and partial segments based on their temporal and spatial closeness. Then, it defines a *trajectory pattern* that results from the aggregation of segment clusters, aggregation that is not only based on spatial and temporal sequentiality, but also on the compatibility of trajectories in each segment cluster. The experimental assessment shows the effectiveness of the method.

Keywords: Pattern recognition · data mining · trajectories · spatio-temporal databases.

1 Introduction

Due to the current advances in sensor networks, wireless technologies, and RAID-enabled ubiquitous computing, data about moving objects (also called trajectories) is an example of massive data relevant in many real applications [4]. According to [4], a trajectory pattern is a set of individual trajectories that visit the same sequence of places with similar travel times. A classical representation of trajectories is given by a sequence of time-stamped locations in a, typically, 2D space trajectory clustering algorithms aim at grouping similar trajectories (or part of trajectories) according to similarity measures. The main problem of trajectory aggregation is the imprecision of spatio-temporal data [8], which makes more challenging the definition of similarity measures that compare different trajectories. There are several notions of trajectory similarity measures [13] being the Euclidean Distance Measure, in its simplest version, robust for trajectory shifts.

We propose a method to derive *trajectory patterns*, which correspond to patterns of trajectories with similar behavior during a time interval. To obtain the trajectory patterns, we first group complete or partial segments of trajectories that have a similar behavior in space and time. Then we aggregate segment clusters considering a compatibility measure. A similarity measure in terms of a distance function is sensible to the visual comparison, is computationally affordable, and is complemented with strategies to avoid sensitivity to sampling rate and noise. We focus here on a similarity measure

defined in terms of Euclidean distance for free movements, but it is also possible to consider different distance functions when trajectories are constrained to other types of spaces such as networks. Using the Euclidean distance, two segments of trajectories can be considered similar if they coincide (approximately) in their starting and ending points. In addition, two segments can be also considered similar if they coincide in some parts of the segments [6, 12]. But it is not enough to have spatial similarity; temporal similarity must ensure that trajectories also occur close in time to capture the time-sensitivity of the similar behavior.

There exist several proposals for the extraction of trajectory patterns [6, 5, 1, 8, 3, 13, 12, 14]. They mostly vary in terms of their similarity functions and the capacity of making incremental extraction of patterns. They consider similarity between sampling points of trajectories (i.e., segments of trajectories), without considering that, between sampling points, trajectories can be partially similar. According to [14] our work falls in the category of trajectory clustering. We argue in this work that clustering not only whole segments, but also portions of segments, captures similarity between trajectories that could otherwise be underestimated.

We compare our proposal with the work presented in [6] which proposes the *partition and group* framework that allows the discovery of common sub-trajectories from a trajectory database. First, it partitions trajectories into a set of line segments and then it groups similar complete line segments. Trajectories are partitioned according to characteristics points. Then, they generate a representative trajectory for each cluster. The algorithm they propose to obtain clusters is based on a suitable *distance function*, which is composed of the *perpendicular distance*, the *parallel distance*, and the *angle distance*. The algorithm works over a set of line segments and classifies them as part of a cluster or as a noise. Only clusters with a cardinality over a threshold are considered valid. Finally, the algorithm computes a representative trajectory for every cluster. The main difference of this previous work with respect to our proposal is that they cluster complete line segments, while we are able to cluster portions of line segments. In addition, we also include time in the comparison of segments and, when obtaining clusters, we do not exclude a segment as a noise, but we keep portions of segments that do not match a cluster, and use them in future computations of clusters. Only at the end of the process of clustering, we eliminate noise segments. In this way, our implementation is not strict with possible noise segments as the one proposed in [6].

2 Trajectory Pattern

A trajectory of an object is represented based on points with temporal dimension [5]. A point SPoint is a tuple (x, y) on the Euclidean space and a temporal point TPoint is a tuple (SPoint, t) where t is a time instant. For simplicity, we also refer as a time interval a tuple $\mathsf{TT} = [t_s, t_e]$, where $t_s, t_e \in \mathbb{R}$ and $t_e \geq t_s$. A segment S of an object's trajectory is a pair of $\mathsf{TPoints}$ $[\mathsf{P}_s, \mathsf{P}_e]$, where $\mathsf{P}_s = ((x_s, y_s), t_s)$, $\mathsf{P}_e = ((x_e, y_e), t_e)$, and $t_e > t_s$. The definition of trajectory is based in the definition given in [9]. A trajectory T is a tuple composed of an identification and a sequence of consecutive segments $\langle id, \mathsf{S}_1, \mathsf{S}_2, \ldots, \mathsf{S}_m \rangle$ that describes the path of an object and where, for every pair of consecutive segments $\mathsf{S}_i = [\mathsf{P}_s, \mathsf{P}_e]$ and $\mathsf{S}_j = [\mathsf{P}'_s, \mathsf{P}'_e]$ in T, $\mathsf{P}_e.x = \mathsf{P}'_s.x, \mathsf{P}_e.y = \mathsf{P}'_s.y$

and $\mathsf{P}'_s.t - \mathsf{P}_e.t \leq \epsilon$, where$\epsilon$ is the length of the interval in which an object may stop at a particular TPoint. We call *sub-trajectory* Ts to a subset of consecutive segments of an object's trajectory T. A sub-trajectory is also a trajectory. With some abuse of notation, we say that $\mathsf{Ts} \subseteq \mathsf{T}$ if Ts is a sub-trajectory of T.

Notice that a trajectory is a sequence of segments, which differs from the classical definition in terms of a sequence of TPoints. We explicitly define trajectories as a sequence of segments to emphasize the treatment of the segments in the clustering process. Also, a sub-segment is also a segment. At a physical level, however, we can avoid the duplication of endpoints in the definition of segments.

Let T be a trajectory, S, S_1, and S_2 be segments, and P and P' be SPoints, the following are useful operators.

- $\mathsf{SD}(\mathsf{P}, \mathsf{P}')$: it denotes the Euclidian spatial distance between two SPoints P and P'. If points are on a network, then this should be the distance on the network. Overloading this operator, $\mathsf{SD}(\mathsf{S}, \mathsf{P})$ is the shortest Euclidian distance from SPoint P to segment S (i.e., the length of the perpendicular line from P to S).
- $\mathsf{TD}(\mathsf{P}, \mathsf{P}') = |\mathsf{P}.t - \mathsf{P}'.t|$: it is the temporal distance between two TPoints P and P'.
- $\mathsf{ID}(\mathsf{T})$: it returns the identification of a trajectory T.
- $\mathsf{ID}(\mathsf{S})$: it returns the identification of the trajectory to which the segment S belongs to.
- $\mathsf{START}_s(\mathsf{T})$: it returns the starting segment of a trajectory T.
- $\mathsf{END}_s(\mathsf{T})$: it returns the ending segment of a trajectory T.
- $\mathsf{START}_p(\mathsf{S})$: it returns the starting point of a segment S.
- $\mathsf{START}_p(\mathsf{T})$: it returns the starting point of a trajectory T, this is, $\mathsf{START}_p(\mathsf{T}) = \mathsf{START}_p(\mathsf{START}_s(\mathsf{T}))$.
- $\mathsf{END}_p(\mathsf{S})$: it returns the ending point of a segment S.
- $\mathsf{END}_p(\mathsf{T})$: it returns the ending point of a trajectory T.
- $\mathsf{LENGTH}(\mathsf{T})$: it returns the number of segments of T.
- $\mathsf{ANGLE}(\mathsf{S}_1, \mathsf{S}_2)$: it returns the formed angle between segments S_1 and S_2.

2.1 Relations between Trajectory Segments

Intuitively, two segments S_1 and S_2 will be *totally related* if their initial and ending points are close, spatially and temporally, and they have the same direction, this is, the angle between the segments is less than 90°. We formalize this in the following definition.

Definition 1. Let S_1 and S_2 be two trajectory segments from different trajectories. S_1 and S_2 are *totally related* if and only if:

- $\mathsf{SD}(\mathsf{START}_p(\mathsf{S}_1), \mathsf{START}_p(\mathsf{S}_2)) \leq \Delta_s$
- $\mathsf{TD}(\mathsf{START}_p(\mathsf{S}_1), \mathsf{START}_p(\mathsf{S}_2)) \leq \Delta_t$
- $\mathsf{SD}(\mathsf{END}_p(\mathsf{S}_1), \mathsf{END}_p(\mathsf{S}_2)) \leq \Delta_s$
- $\mathsf{TD}(\mathsf{END}_p(\mathsf{S}_1), \mathsf{END}_p(\mathsf{S}_2)) \leq \Delta_t$
- $\mathsf{ANGLE}(\mathsf{S}_1, \mathsf{S}_2) < 90^o$. □

(a) (b) (c) (d)

Fig. 1. Partial relations between two segments S_1 and S_2 from different trajectories

Two segments are *partially related* if they are not totally related, they have the same direction, and one extreme point of a segment is close, spatially and temporally, to some point of the other segment. It is possible to have different forms of partial relations between two segments. We have considered four forms of partial relations.

Definition 2. Let S_1 and S_2 be two trajectory segments that are not totally related. S_1 and S_2 are *partially related* if and only if they satisfy one of the following partial relations, which are illustrated in Figure 1:

- Case 1 (Figure 1(a)). In this case: (i) The starting points of segments S_1 and S_2 are spatially and temporally close to each other. This is, they satisfy the distances established by threshold Δ_s and Δ_t. (ii) The ending point of segment S_1 is not close to any point of segment S_2, but the ending point of segment S_2 is close to some point (different from an endpoint) of segment S_1.
- Case 2 (Figure 1(b)). In this case: (i) The ending points of segments S_1 and S_2 are spatially and temporally close to each other. (ii) The starting point of segment S_1 is not close to any point of segment S_2, but the starting point of segment S_2 is close to some point (different from an endpoint) of segment S_1.
- Case 3 (Figure 1(c)). In this case: Neither the starting and ending point of segment S_1 are close to any point of segment S_2, but the starting and ending point of segment S_2 are close to some point (different from an endpoint) of segment S_1.
- Case 4 (Figure 1(d)). In this case: (i) The starting point of segment S_1 is close to some point of segment S_2. (ii) The ending point of segment S_1 is not close to any point of S_2.(iii) The starting point of segment S_2 is not close to any point of segment S_1. (iv) The ending point of segment S_2 is close to some point (different from an endpoint) of segment S_1. □

2.2 Segment Clustering

We use the concept of *segment cluster* to refer to a set $\mathsf{SC} = \{S_1, S_2, \ldots, S_n\}$ of not necessarily consecutive segments or sub-segments from different trajectories that are totally related to each other. The process of generating segment clusters that are not totally related is as follows. Consider a trajectory T_i with segments $S_1, \ldots, S_n$: (i) If a segment S_i of T_i is not totally related to any of the segments in the already computed segment clusters, but it is partially related to some segment S_j in a cluster c_j, then: (i) We first identify the type of partial relation between segments S_i and S_j, according to Definition 2. (ii) A new segment S_s is generated and corresponds to the projection of

the shortest segment on the longest segment. Note that this implies to calculate a TPoint that splits the original segment. Let us assume that S_i is shorter than S_j then, segment S_s corresponds to the projection of S_i over S_j. (iii) Cluster c_j is updated and contains segments S_s and S_i, a new cluster is generated containing the residual segment $S_j - S_s$.

The dynamic computation of segment clusters refers to the capability of adding new trajectory segments when we had already computed a set of segment clusters, without starting the whole process of segment clustering. The segments of a new trajectory can be added into existing segment clusters or can form new clusters.

We introduce the following notation over segment clusters that is useful in the next definition of trajectory patterns. Let cl be a segment cluster composed of a set S of segments. Then, (i) $\mathsf{CONVEX}(cl)$ denotes the convex hull over the TPoints that form the segments of the cluster; (ii) $\mathsf{TInterval}(cl)$ denotes the time interval $[t_s, t_e]$ such that $t_s = \min_{\forall s_i \in S}\{START_p(s_i).t\}$ and $t_e = \max_{\forall s_i \in S}\{\mathsf{END}_p(s_i).t\}$; and (iii) $\mathsf{IDS}(cl) = \{\mathsf{ID}(s_i) | s_i \in cl\}$ is the set of trajectories's ids that are part of the cluster. Given a segment cluster cl, we call the *pattern* of cl, the tuple of the form (geo, tt, ids), with $geo = \mathsf{CONVEX}(cl)$, $tt = \mathsf{TInterval}(cl)$, and $ids = \mathsf{IDS}(cl)$. Notice that this pattern is essentially a geometric approximation of the segment cluster. For simplicity, we use $cp.a$ with $a \in [geo, tt, ids]$ to access each of the elements of the pattern cp.

2.3 Trajectory Pattern

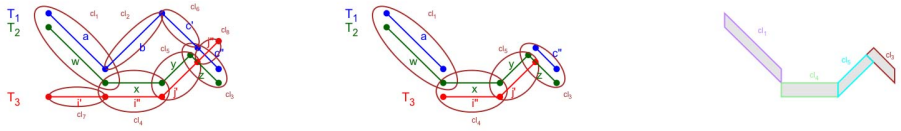
Segment clusters can be aggregated to form *trajectory patterns* if, they have at least two segments, their areas intersect, their share a temporal interval, and they satisfy a compatibility threshold of common trajectories. Trajectory patterns are patterns that combine two or more patterns extracted from single segment clusters. We first introduce the concept of compatible patterns, useful to define trajectory patterns.

Definition 3. Let cp_1, cp_2 be two patterns, with $cp_1 \neq cp_2$. Patterns are compatible if:

1. $cp_1.geo \cap_{spatial} cp_2.geo \neq \emptyset$, where $\cap_{spatial}$ denotes geometric intersection.
2. $cp_1.tt \cap_{time} cp_2.tt \neq \emptyset$, where $\cap_{time}$ denotes temporal intersection.
3. $\frac{|cp_1.ids \cap cp_2.ids|}{|cp_1.ids \cup cp_2.ids|} >= \Delta_j$, where Δ_j is the threshold of the number of common trajectories. □

Definition 4. Let cp_1 and cp_2 be two compatible patterns. A trajectory pattern tp for cp_1 and cp_2 is a tuple of the form (geo, tt, ids), where geo is the geometric union of $cp_1.geo$ and $cp_2.geo$, tt is the time union of $cp_1.tt$ and $cp_2.tt$, and ids the set union of $cp_1.ids$ and $cp_2.ids$. Because trajectory patterns are patterns, we can recursively apply this definition to aggregate more than two patterns. □

The extraction of trajectory patterns starts with an empty set of trajectory patterns TP, a set C of segment clusters and its corresponding patterns C^* and, iteratively, finds compatible clusters to aggregate to trajectory patterns. At the end of this iterative process, TP is empty if it does not find compatible clusters or contains patterns composed of at least two patterns in C^*. This concept is similar to the concept of *macro clusters* presented in [7], where, unlike our proposal, patterns of trajectories are computed considering only complete segments of trajectories and only spatial closeness.



(a) Segment clusters (b) Segment clusters to be aggregated (c) Convex hull

Fig. 2. Aggregation of segment clusters

Example 1. Consider the segment clusters in Figure 2(a). Figure 2(b) shows the segment clusters that can be aggregated from the set of segment clusters, this is, clusters cl_1, cl_3, cl_4 and cl_5. Clusters cl_2, cl_6, cl_7 and cl_8 are discarded because they have only one segment. Since the areas and time intervals of clusters cl_1, cl_3, cl_4 and cl_5 intersect (see Figure 2(b)), they may be aggregated if they satisfy a specific compatibility threshold (Δ_j). □

Intuitively, a set of *trajectory patterns* corresponds to the aggregation of trajectories that have similar behavior and satisfy a compatibility threshold, which indicates a percentage of common trajectories. Note that by applying Definition 4 iteratively, it is possible to get a set of trajectory patterns, each of them with possible different levels of compatibility.

3 Evaluation and Comparison with the State of the Art

To show that our method is effective to find trajectory patterns we use both real and synthetic datasets. To evaluate effectiveness we compare our method with the method, called Traclus, proposed in [6], which does not consider the temporal distance between segments, and also compare complete segments but not sub-segments of trajectories. In this case we use synthetic datasets. We also present a qualitative evaluation of our method that shows some features that are not supported by the baseline. To do so, we use a real dataset that corresponds to truck trajectories in Greece that has been used in [10, 11, 2] to determine trajectory patterns.

3.1 Quantitative Comparison with the State of the Art

We generated 1,000 trajectories with the Brinkhoff generator[3]. Then, we randomly selected 100 of them and created 50 copies of each one. Thus, these 100 trajectories correspond to the ground truth of trajectory patterns that a clustering algorithm should recognize as patterns. Figure 3 shows the comparison between the baseline method presented in [6] and our proposed method.

[3] `http://chorochronos.datastories.org/?q=node/51`

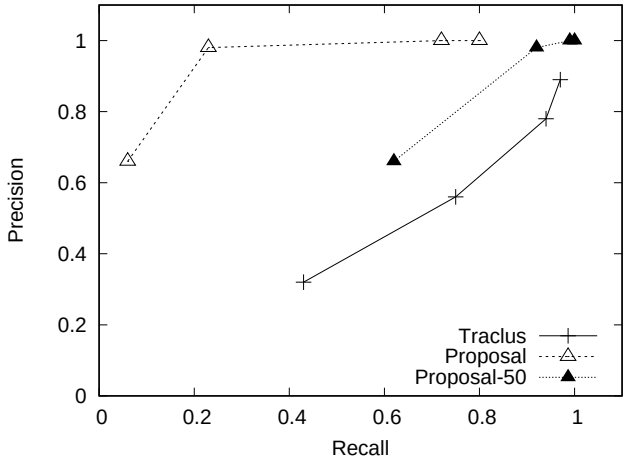


Fig. 3. Comparison with the state of the art

For our method we graphic two lines, one unfiltered and another filtering those segments that have less than 50 elements. For each method we show a line and not a single point because, to consider a segment of the ground truth equal to a segment obtained by an algorithm, a tolerance may be applied to ignore precision issues. For each method, the point located on the leftmost endpoint is the one with tolerance 0, and then it increases to 10, 100 and 250. Obviously, the greater the tolerance, we are considering more distant segments as equal. Even so, they are small tolerance values.

In the chart, we show precision-recall values, which are standard in the information retrieval community. In our domain, precision may be interpreted as the portion of the ground truth that an algorithm can retrieve, whereas recall would be the portion of the retrieved patterns that are indeed in the ground truth. Hence, the ideal method would be the one that gets earlier to the upper right corner (precision 1 and recall 1). As it can be seen, our method with filtering is the one that dominates the others by being the only one to get close to that point. Regarding our unfiltered method, although it obtains a precision of 1, it is at the expense of the recall (that is, it recovers all the reference, but also a lot of noise). The baseline is improving by increasing the tolerance in the comparison, but it does not reach our method with filtering.

3.2 Qualitative evaluation of our proposal

The trucks dataset contains GPS (with reference system GGRS87) positions of 50 trucks transporting concrete in the Athens area between August and September of 2002. This dataset contains temporal information, and the time interval to take the samples is 30 seconds. The set contains 111,927 segments, conforming 276 trajectories, with an average of 406 segments per trajectory. We use a spatial threshold Δ_s of 420 units, a temporal threshold Δ_t of 209,950 units of time, and different com-

patibility thresholds. These thresholds were chosen because they are the values that minimize a ClusterCost function defined for a set of segment clusters C and a set R of isolated segments not included in any $c_i \in C$. Function ClusterCost is defined as: $\mathsf{ClusterCost} = \mathsf{ClusterWeight} + \mathsf{NoiseWeight}$, where:

$$\mathsf{ClusterWeight} = \begin{cases} \sum_{c_i \in C} \frac{\mathsf{AREA}(\mathsf{CONVEX}(c_i))}{|\mathsf{IDS}(c_i)|} & |C| > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathsf{NoiseWeight} = \begin{cases} \sum_{r_i \in R} \mathsf{LENGTH}(r_i)^2 * \pi & |R| > 0 \\ 0 & \text{otherwise} \end{cases}$$

Figure 4(a) shows the 17,855 segment clusters obtained by our algorithms with $\Delta_s = 420$ and $\Delta_t = 209,950$.

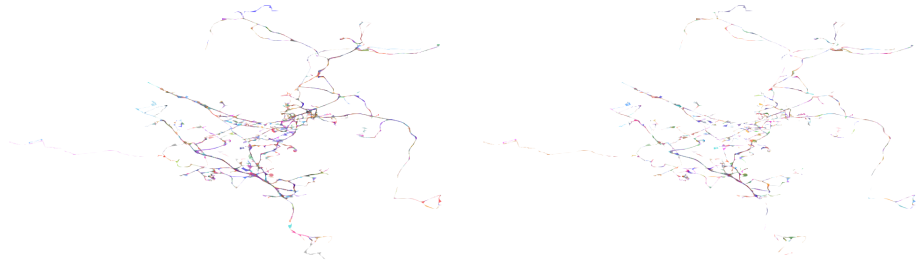
(a) Segment clusters (b) Trajectory patterns for $\Delta_j = 0\%$

Fig. 4. Segment clusters and trajectory patterns for the trucks dataset

Trajectory patterns for the trucks dataset We use different compatibility thresholds to compute trajectory patterns considering the 17,855 segment clusters in Figure 4(a). Consider $\Delta_j = 0\%$, this is, there is not a constraint over a percentage of common trajectories. In this case, all the segment clusters whose areas and time intervals intersect form a unique trajectory pattern. Figure 4(b) shows the result of the experiment, where there are six trajectory patterns and two non-aggregate clusters.

A value for Δ_j greater than 50% provides a lower proportion of trajectory patterns with respect to the total of trajectory patterns and non-aggregate clusters. Figure 5(a) shows the $3,064$ trajectory patterns with $\Delta_j = 60\%$. 5(b) illustrates the result for $\Delta_j = 100\%$, in which case there are $1,963$ trajectory patterns.

Trajectory patterns under different temporal intervals We also compute trajectory patterns for different periods of time. They were obtained considering the following parameters $\Delta_s = 420$ units, $\Delta_t = 209,950$ units of time, and $\Delta_j = 20\%$ (compatibility threshold). We consider different days of the week and weekends, and we can conclude that during the week, trucks remain mostly in the central zone in the afternoons. Figure



(a) $\Delta_j = 60\%$ (b) $\Delta_j = 100\%$

Fig. 5. Trajectory patterns for different Δ_j

6 shows the trajectory patterns for September 11 (Wednesday) to September 14 (Saturday) of 2002. This kind of finding would not be possible with the algorithm in [6] as it does not consider time.

(a) September 11 (b) September 12 (c) September 13 (d) September 14

Fig. 6. Trajectory patterns for days of September of 2002

4 Conclusions and Future Work

We presented a new concept of *trajectory pattern* that corresponds to the aggregation of compatible segment clusters from trajectories. To compute segment clusters, we consider the spatial and temporal distance between segments and sub-segments of trajectories, and also the angle of the segments. Aggregation of segment clusters is possible if the clusters satisfy a compatibility threshold, have more than one segment, and their areas and time intervals intersect. Our method allows the dynamic computation of segment clusters, and process the called *macro clustering* presented [7]. The experimentation we reported shows that the method is effective for computing trajectory patterns. For future work, we would like to optimize the algorithms for segment clustering and trajectory patterns detection. To do so, we can consider indexing structures in space and time.

Acknowledgements

Mónica Caniupán and Luis Cabrera-Crot are partially funded by DIUBB [181315 3/R], and the Algorithms and Databases Research Group [160119/EF]. Andrea Rodríguez is partially funded by Fondecyt [1170497], and the Complex Engineering Systems Institute (CONICYT: FBO16). Diego Seco is partially funded by Fondecyt [1170497].

References

1. Andrienko, N., Andrienko, G.: Spatial generalization and aggregation of massive movement data. IEEE Trans. Vis. Comput. Graph. **17**(2), 205–219 (2011)
2. Frentzos, E., Gratsias, K., Pelekis, N., Theodoridis, Y.: Algorithms for nearest neighbor search on moving object trajectories. Geoinformatica **11**(2), 159–193 (2007)
3. Giannotti, F., Nanni, M., Pedreschi, D.: Efficient mining of temporally annotated sequences. In: Proc. of the Sixth International Conference on Data Mining. pp. 348–359 (2006)
4. Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D.: Trajectory pattern mining. In: Proc. of the 13th International Conference on Knowledge Discovery and Data Mining. pp. 330–339 (2007)
5. Hung, C.C., Peng, W.C., Lee, W.C.: Clustering and aggregating clues of trajectories for mining trajectory patterns and routes. The VLDB Journal **24**(2), 169–192 (2015)
6. Lee, J.G., Han, J., Whangl, K.Y.: Trajectory clustering: a partition-and-group framework. In: Proc. of the SIGMOD Conference. pp. 593–604 (2007)
7. Li, Z., Lee, J.G., Li, X., Han, J.: Incremental clustering for trajectories. In: Proc. of the 15th International Conference on Database Systems for Advanced Applications. pp. 32–46 (2010)
8. Meratnia, N., de By, R.: Aggregation and comparison of trajectories. In: Proc. of the Tenth International Symposium on Advances in Geographic Information Systems. pp. 49–54 (2002)
9. Orlando, S., Orsini, R., Raffaetà, A., Roncato, A., Silvestri, C.: Trajectory data warehouses: Design and implementation issues. Journal of Computing Science and Engineering **1**(2), 211–232 (2007)
10. Panagiotakis, C., Pelekis, N., Kopanakis, I., Ramasso, E., Theodoridis, Y.: Segmentation and sampling of moving object trajectories based on representativeness. IEEE Trans. on Knowl. and Data Eng. **24**(7), 1328–1343 (2012)
11. Pelekis, N., Kopanakis, I., Kotsifakos, E., Frentzos, E., Theodoridis, Y.: Clustering uncertain trajectories. Knowledge and Information Systems **28**(1), 117–147 (2011)
12. Sankararaman, S., Agarwal, P.K., Mølhave, T., Pan, J., Boedihardjo, A.P.: Model-driven matching and segmentation of trajectories. In: Proc. of the 21st International Conference on Advances in Geographic Information Systems. pp. 234–243 (2013)
13. Wang, H., Su, H., Zheng, K., Sadiq, S., Zhou, X.: An effectiveness study on trajectory similarity measures. In: Proc. of the Twenty-Fourth Australasian Database Conference. pp. 13–22 (2013)
14. Zheng, Y.: Trajectory data mining: An overview. ACM TIST **6**(3), 29:1–29:41 (2015)

Spanish Word Embeddings Learned on Word Association Norms

Helena Gómez-Adorno[1 [0000−0002−6966−9912]], Jorge Reyes-Magaña[2,3 [0000−0002−8296−1344]], Gemma Bel-Enguix[2 [0000−0002−1411−5736]], and Gerardo Sierra[2 [0000−0002−6724−1090]]

[1] Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México, Ciudad de México, México
helena.gomez@iimas.unam.mx

[2] Instituto de Ingeniería, Universidad Nacional Autónoma de México, Ciudad de México, México
gbele@iingen.unam.mx, gsierram@iingen.unam.mx

[3] Facultad de Matemáticas, Universidad Autónoma de Yucatán, Mérida, Yucatán
jorge.reyes@correo.uady.mx

Abstract. Word embeddings are vector representations of words in an n-dimensional space used for many natural language processing tasks. A large training corpus is needed for learning good quality word embeddings. In this work, we present a method based on the *node2vec* algorithm for learning embeddings based on paths in a graph. We used a collection of Word Association Norms in Spanish to build a graph of word connections. The nodes of the network correspond to the words in the corpus, whereas the edges correspond to a pair of words given in a free association test. We evaluated our word vectors in human annotated benchmarks, achieving better results than those trained on a billion-word corpus such as, word2vec, fasttext, and glove.

Keywords: word vectors · node2vec · word association norms · Spanish

1 Introduction

The representation of words in a vector space is a very active research area in the latest decades. Computational models like the singular value decomposition (SVD) and the latent semantic analysis (LSA) are capable of modeling word vector representations (*word embeddings*) from the term-document matrix. Both methods can reduce a dataset of N dimensions using only the most important features. Recently, *Mikolov et al.* [19] introduced *word2vec* inspired by the distributional hypothesis establishing that words in similar contexts tends to have similar meanings [22]. This method uses a neural network in order to learn vector representations of words by predicting other words in their context. The vector representation of a word obtained by *word2vec* has the awesome capability of conserving linear regularities between words.

In order to build a model of adequate and reliable vector space, capable of capturing semantic similarity and linear regularities of words, large volumes of text are needed. Although *word2vec* is fast and efficient to train, and pre-trained word vectors are usually available online, it is still computationally expensive to process large volumes of data in non-commercial environments, that is, on personal computers.

Free association is an experimental technique commonly used to discover the way in which the human mind structures knowledge [8]. In free association tests, a person is asked to say the first word that comes to mind in response to a given *stimulus* word. The set of lexical relations obtained with these experiments is called Word Association Norms (WAN). These kinds of resources reflect both semantic and episodic contents [6].

In previous work [4] we learn word vectors in English from a graph obtained from a WAN corpus. The vectors learned from this graph were able to map the contents of semantic and episodic memory in vector space. For this purpose, we used the *node2vec* algorithm [14] which is able to learn node mappings to a continuous vector space from the complete network taking into account the neighborhood of the nodes. The algorithm performs biased random paths to explore different neighborhoods in order to capture not only the structural roles of the nodes in the network but also the communities to which they belong to.

In this paper, we extend previous work of learning word vectors in English[4] by learning vector representations of words from a resource that collects words association norms in Spanish. We build two embedding resources of different dimensions, the first one based on *Normas de Asociación Libre en Castellano* [10] (NALC), and the other using the corpus of *Normas de Asociación de Palabras para el Español de México* [2] (NAP). The obtained embeddings from both resources are available on GitHub, the NALC based embeddings[4] and the NAP based embeddings[5].

The rest of the paper is organized as follows. In section 2, we discuss the related work. In Section 3, we present the corpora of Word Association Norms. In section 4, we describe the methodological framework for learning word vectors from WAN's. Section 5, shows the evaluation of the generated vectors, using a word similarity dataset in Spanish. Finally, in section 6 we draw some conclusions and point out to possible directions of future work.

2 Related Work

Semantic networks [25] are graphs relating words [1] used in linguistics and psycholinguistics not only to study the organization of the vocabulary but also to approach the structure of knowledge. Many languages have corpora of WAN. In the past decades, different association lists were elaborated with the collaboration of a large number of volunteers. However, in recent years, the web has

[4] `https://github.com/jocarema/nalc_vectors`
[5] `https://github.com/jocarema/nap_vectors`

become a natural way to get data to build such resources. *Jeux de Mots*[6] provides an example in French [18], whereas the *Small World of Words*[7] contained datasets in 14 languages at the time of writing.

Sinopalnikova and Smrz [24] showed that WATs are comparable to balanced text corpora and can replace them in case of absence of a corpus. The authors presented a methodological framework for building and extending semantic networks with word association thesaurus (WAT), including a comparison of quality and information provided by WAT vs. other language resources.

Borge-Holthoefer & Arenas [6] used free association information for extracting semantic similarity relations with a Random Inheritance Model (RIM). The obtained vectors were compared with LSA-based vector representations and the WAS (word association space) model. Their results indicate that RIM can successfully extract word feature vectors from a free association network.

In a recent work by *De Deyne et al.* [9] the authors introduced a method for learning word vectors from WANs using a spreading activation approach in order to encode a semantic structure from the WAN. The authors used part of the *Small World of Words* network. The word association-based model was compared with a word embeddings model (word2vec) using relatedness and similarity judgments from humans, obtaining an average of 13% of improvement over the word2vec model.

3 Word Association Norms in Spanish

Many languages have compilations of word association norms. In the past decades, some interesting works have been developed with a large number of volunteers. Among the most well-known English resources accessible on the web are the *Edinburgh Associative Thesaurus*[8] (EAT) [17] and the resource of *Nelson et al.*[9] [21].

For Spanish, there are some corpora of free words association, in this work we used two WAN resources in Spanish: a) *Corpus de Normas de Asociación de Palabras para el Español de México* (NAP) [2] and b) *Corpus de Normas de Asociación Libre en Castellano* [10] (NALC).

The NAP corpus was elaborated with a group of 578 native Mexican speakers young adults, 239 men and 339 women, with ages ranging from 18 to 28 years, and with a range of education of at least 11 years. The total number of tokens in the corpus is 65731, with 4704 different words. The authors used 234 stimulus words, all of them common nouns taken from the *MacArthur word compression and production* [16]. It is important to mention that although the stimuli are always nouns, the associated words are free-choice, that is, the informants can relate to the word stimulus with any word regardless of its grammatical category.

[6] `http://www.jeuxdemots.org/`
[7] `https://smallworldofwords.org/`
[8] `http://www.eat.rl.ac.uk/`
[9] `http://web.usf.edu/FreeAssociation`

For each *stimuli* and its *associates*, the authors computed different measures: time, frequency and association strength.

The NALC corpus includes 5819 *stimuli* words and their corresponding *associates* obtained from the free association responses of a sample of 525 subjects for 247 words, of 200 subjects for 664 words and of 100 for the remaining words. In the compilation of association norms, approximately 1500 university students have participated so far. All the subjects had Spanish as their native language and participated voluntarily in the empirical study. The total number of different words in the corpus is 31207.

4 Learning Word Embeddings on Spanish WANs

The graph that represents a given WAN corpus is formally defined as $G = \{V, E, \phi\}$ where:

- $V = \{v_i | i = 1, ..., n\}$ is the finite set of nodes with size n, $V \neq \emptyset$, which corresponds to *stimuli* words along with its *associates*.
- $E = \{(v_i, v_j) | v_i, v_j \in V, 1 \leq i, j \leq n\}$, is the set of edges, which corresponds to the connections between *stimuli* and *associates* words.
- $\phi : E \rightarrow \mathbb{R}$, is a weighting function over the edges.

We performed experiments with directed and non-directed graphs. In the directed graphs, each pair of nodes (v_i, v_j) follows an established order where the initial node v_i corresponds to the *stimulus* word and the final node v_j to an associated word. For the non-directed graph, all the *stimuli* are connected with their correspondent associates without any order of precedence. We evaluated three edges weighting functions:

Time It measures the seconds the participant takes to give an answer for each *stimulus*.

Frequency It establishes the number of occurrences of each of the associated words with a *stimulus*. In this work we use the inverse frequency (IF):

$$IF = \Sigma F - F$$

where F the frequency of a given associated word, and ΣF is the sum of the frequencies of the words connected to the same *stimulus*

Association Strength Establishes a relation between the frequency and the number of responses for each *stimulus*. It can be calculated as follows:

$$AS_W = \frac{AW * 100}{\Sigma F}$$

where AW is the frequency of a given word associated with a *stimulus*, and ΣF the sum of the frequencies of the words connected the same *stimulus* (the total number of answers). We also used the inverse of the association strength (IAS):

$$IAS = 1 - \frac{F}{\Sigma F}$$

The NAP corpus provides the three weighting functions, however for the NALC corpus only the association strength is available. Thus, in our evaluation we only report results using the association strength for the NALC corpus.

4.1 Node2vec

Node2vec [14] finds a mapping $f : V \rightarrow \mathbb{R}^d$ that transforms the nodes of a graph into vectors of d-dimensions. It defines a neighborhood in a network $N_s(u) \subset V$ for each node $u \in V$ through a S sampling strategy. The goal of the algorithm is to maximize the probability of observing subsequent nodes on a random path of a fixed length.

The sampling strategy designed in *node2vec* allows it to explore neighborhoods with skewed random paths. The parameters p and q control the change between the breadth-first search (BFS) and depth-first search (DFS) in the graph. Thus, choosing an adequate balance allows preserving both the structure of the community and the equivalence between structural nodes in the new vector space.

In this work, we used the implementation of the project *node2vec*, which is available on the web[10] with default values for all parameters. We also examined the quality of vectors with a different number of dimensions.

5 Spanish Word Embeddings Evaluation

There are several evaluation methods for unsupervised word embeddings methodologies [23], which are categorized as extrinsic and intrinsic. In the extrinsic evaluation, the quality of the word vectors is evaluated by the improvement of performance in a given natural language processing tasks (PLN) [12, 13]. Intrinsic evaluation measures the ability of word vectors to capture syntactic or semantic relationships [3].

The hypothesis of the intrinsic evaluation is that similar words should have similar representations. So, we first performed a visualization of a sample of words using the T-SNE projection of the word vectors in a two-dimensional vector space. Figure 1 shows how the words that are related to each other are grouped. We show the word vectors obtained from graphs with the three weighting functions using the NAP corpus only. It is observed that in all cases the vectors illustrate some interesting phenomena. For example, when frequency is taken as weight (the graph below), the word *pájaro* (bird) is drawn very close to *avión* (plane). From this, it is inferred that the feature "fly" is more representative than "animal" for the model. For its part, the word *caballo* (horse), is represented closer to *camioneta* (truck) than to other animals, focusing more on its status as "transportation".

In addition, we evaluated the ability of word vectors to capture semantic relationships through a word similarity task. Specifically, we used two widely

[10] `http://snap.stanford.edu/node2vec/`

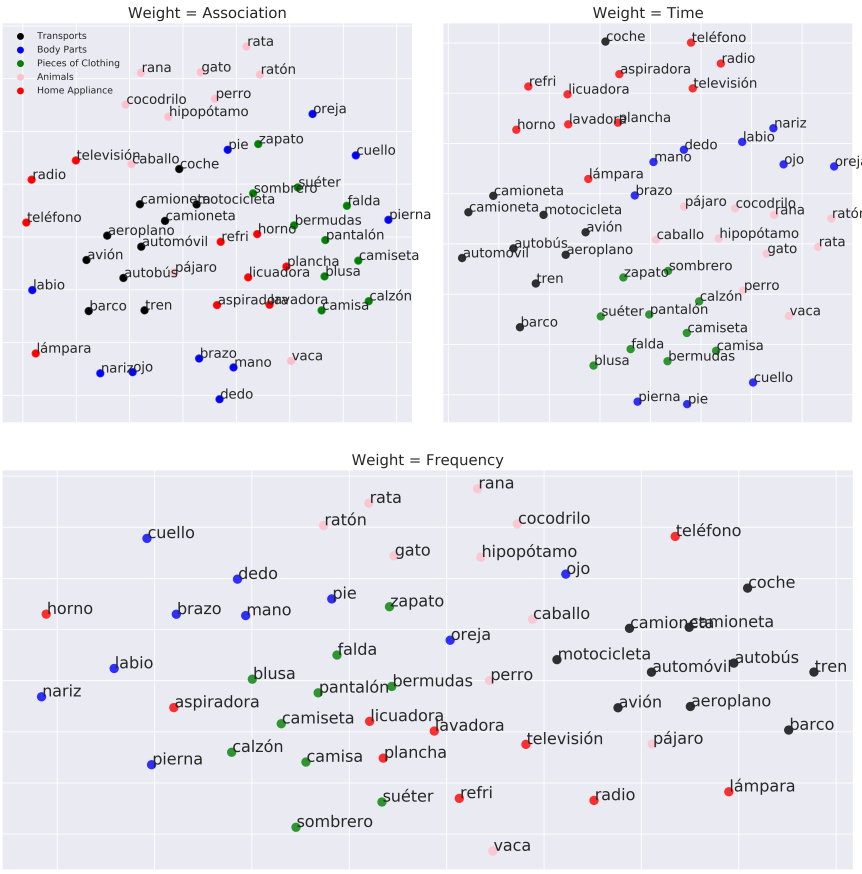


Fig. 1. Projection of the word vectors in 5 semantic groups (of ten words each).

known corpora: a) the corpus *WordSim-353* [11] composed of pairs of terms semantically related with similarity scores given by humans and b) the MC-30 [20] benchmark containing 30 word pairs. Both datasets in its Spanish version [11] [15].

We calculated the cosine similarity between the vectors of word pairs contained in the above mentioned datasets and compare it with the similarity given by humans using the Spearman correlation. To deal with the non-inclusion of every word of the testing data sets in our NALC word association norms, we introduced the concept of overlap in the experiments and calculated the total number of common words between the lists that are being compared. The others are excluded from the evaluation. In principle, having large overlaps is a positive feature this approach. Tables 1 and 2 present the Spearman corre-

[11] `http://web.eecs.umich.edu/~mihalcea/downloads.html`

lation, of the similarity given by human taggers, with the similarity obtained with word vectors (learned from NAP and NALC separately). We also report different dimensions of word vectors learned on the non-directed graphs with different weighting functions. We also report the overlap, which is the number of words that can be found in in both, the given WAN corpus (NAP or NALC) and the evaluation dataset (ES-WS-53 or MC-30).

Table 1. Spearman rank order correlations between Spanish WAN embeddings (based on cosine similarity) and the ES-WS-353 dataset.

	NAP Overlap 140			NALC Overlap 322
Dimension	Inv. Frequency	Inv. Association	Time	Inv. Association
300	0.489	0.463	0.461	0.650
200	0.454	0.456	0.491	0.641
128	0.503	0.463	0.450	0.659
100	0.471	0.478	0.495	**0.664**
50	**0.523**	**0.503**	0.503	0.626
25	0.484	0.478	**0.572**	0.611

Table 2. Spearman rank order correlations between Spanish WAN embeddings (based on cosine similarity) and the MC-30 dataset

	NAP Overlap 11			NALC Overlap 27
Dimension	Inv. Frequency	Inv. Association	Time	Inv. Asociation
300	0.305	**0.563**	0.545	0.837
200	0.468	0.381	0.263	**0.844**
128	**0.545**	0.272	0.300	0.767
100	0.336	0.418	0.372	0.806
50	0.527	0.509	0.272	0.814
25	0.454	0.400	**0.563**	0.788

It can be observed that the word embeddings obtained from the NALC corpus achieved better correlation with the human similarities than the embeddings obtained from the NAP corpus in both datasets, ES-WS-53 and MC-30. The difference in the results can be explained by the size of the vocabulary in both WANs, the NALC corpus has higher overlap with both evaluation datasets than the NAP corpus.

In order to test and compare the quality of the Spanish word vectors, we also performed the experiments with pre-trained Spanish vectors[12]. We selected three word embeddings models: word2vec[13], gloVe[14], and fasttext[15].

Table 3 shows the Spearman rank order correlation between the cosine similarity obtained with word vectors pre-trained in large corpora and the similarity of humans (obtained from *WordSim-353*) and *MC-30* datasets) in comparison with the correlation between NAP embeddings and the humans rated similarities. In the same way, Table 4 shows the same comparison with pre-trained word vectors and the NALC based embeddings.

The highest correlation value was obtained with the vectors trained with the fasttext [5] model. The vectors trained on the Wikipedia in Spanish obtained the best results among the pre-trained models. Our method outperformed the results obtained by the pre-trained vectors when the vectors were learned on the NALC corpus in both evaluation datasets, ES-WS-353 and MC-30.

Table 3. Spearman rank order correlation comparison of NAP embeddings and pre-trained word vectors with the evaluation datasets.

Source	Vector size	MC-30 (Overlap 11)	ES-WS-353 (Overlap 140)
Fasttext-sbwc	300	0.881	0.639
Fasttext-wiki	300	**0.936**	**0.701**
Glove-sbwc	300	0.827	0.532
Word2vec-sbwc	300	0.890	0.634
n2v-Inverse Association	300	0.563	0.463
n2v-Inverse Frequency	300	0.305	0.489
n2v-Time	25	0.563	0.572

6 Conclusions and Future Work

We introduced a method for learning Spanish word embeddings from a Corpus of Word Association Norms. For learning the word vectors, we applied the *node2vec* algorithm on the graph of two WAN corpora, NAP and NALC.

We employ weighting functions on the edges of the graph taking into account three different criteria: time, inverse frequency and inverse associative strength. The best results have been obtained with the association strength, however, the time weighting function also achieved high results. Words with a higher

[12] `https://github.com/uchile-nlp/spanish-word-embeddings`
[13] `https://code.google.com/archive/p/word2vec/`
[14] `https://nlp.stanford.edu/projects/glove/`
[15] `https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md`

Table 4. Spearman rank order correlation comparison of NALC embeddings and pre-trained word vectors with the evaluation datasets.

Source	Vector size	MC-30 (Overlap 27)	ES-WS-353 (Overlap 322)
Fasttext-sbwc	300	0.762	0.613
Fasttext-wiki	300	0.793	0.624
Glove-sbwc	300	0.707	0.482
Word2vec-sbwc	300	0.795	0.624
n2v-Inverse Association	300	0.837	0.650
n2v-Inverse Association	200	**0.844**	**0.664**

association strength usually have a shorter formulation time, which leads to the algorithm to connect more related words in a neighborhood because the node2vec algorithm looks for shorter paths in the graphs.

The results we obtained using the NALC corpus are higher than those obtained with pre-trained word embeddings trained on large corpora. The performance even improves the results achieved with the vectors trained on the Spanish billion words corpus [7]. However, some simple strategies would help improve our results. Some of them would be to adjust the parameters of the algorithm and adapt the system to different types of neighborhoods for the nodes, which could produce different configurations of the vectors. In future work we will perform an extrinsic evaluation these Spanish word vectors, i.e. in some Natural Language Processing task [4].

The evaluations carried out with the vectors learned on the NAP corpus also showed promising results with respect to the similarity and relational indexes. However, due to the low vocabulary length, the results were lower than those obtained on pre-trained embeddings. As future work, we plan to solve this problem by automatically generate word association norms between pairs of words retrieved from a medium-sized corpus. With this process, we will build a new resource that can account for syntactic, semantic and cognitive connections between words.

Acknowledgments

This work was partially supported by the following projects: Conacyt FC-2016-01-2225 and PAPIIT IA401219, IN403016, AG400119.

References

1. Aitchison, J.: Words in the mind: An introduction to the mental lexicon. John Wiley & Sons (2012)
2. Arias-Trejo, N., Barrón-Martínez, J.B., Alderete, R.H.L., Aguirre, F.A.R.: Corpus de normas de asociación de palabras para el espaol de Mxico [NAP]. Universidad Nacional Autnoma de Mxico (2015)

3. Baroni, M., Dinu, G., Kruszewski, G.: Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 238–247 (2014), http://www.aclweb.org/anthology/P14-1023
4. Bel-Enguix, G., Gómez-Adorno, H., Reyes-Magaña, J., Sierra, G.: Wan2vec: Embeddings learned on word association norms. Semantic Web (2019)
5. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Computing Research Repository **arXiv:1607.04606** (2016). https://doi.org/10.1162/tacl_a_00051, https://arxiv.org/abs/1607.04606
6. Borge-Holthoefer, J., Arenas, A.: Navigating word association norms to extract semantic information. In: Proceedings of the 31st Annual Conference of the Cognitive Science Society (2009)
7. Cardellino, C.: Spanish Billion Words Corpus and Embeddings (March 2016), http://crscardellino.github.io/SBWCE/
8. De Deyne, S., Navarro, D.J., Storms, G.: Associative strength and semantic activation in the mental lexicon: Evidence from continued word associations. In: Proceedings of the 35th Annual Conference of the Cognitive Science Society. Cognitive Science Society (2013)
9. De Deyne, S., Perfors, A., Navarro, D.J.: Predicting human similarity judgments with distributional models: The value of word associations. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. pp. 1861–1870 (2016). https://doi.org/10.24963/ijcai.2017/671
10. Fernandez, A., Díez, E., Alonso, M.: Normas de asociación libre en castellano de la universidad de salamanca (2010), http://inico.usal.es/usuarios/gimc/normas/index_nal.asp
11. Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin, E.: Placing search in context: The concept revisited. In: Proceedings of the 10th International Conference on World Wide Web. pp. 406–414. ACM (2001)
12. Gómez-Adorno, H., Markov, I., Sidorov, G., Posadas-Durán, J., Sanchez-Perez, M.A., Chanona-Hernandez, L.: Improving feature representation based on a neural network for author profiling in social media texts. Computational Intelligence and Neuroscience **2016**, 13 pages (2016)
13. Gómez-Adorno, H., Posadas-Durán, J.P., Sidorov, G., Pinto, D.: Document embeddings learned on various types of n-grams for cross-topic authorship attribution. Computing pp. 1–16 (2018)
14. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM International Conference on Knowledge Discovery and Data Mining. pp. 855–864. ACM (2016)
15. Hassan, S., Mihalcea, R.: Cross-lingual semantic relatedness using encyclopedic knowledge. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3. pp. 1192–1201. Association for Computational Linguistics (2009)
16. Jackson-Maldonado, D., Thal, D., Fenson, L., Marchman, V., Newton, T., Conboy, B.: Macarthur inventarios del desarrollo de habilidades comunicativas (inventarios): Users guide and technical manual. Baltimore, MD: Brookes (2003)
17. Kiss, G., Armstrong, C., Milroy, R., Piper, J.: An associative thesaurus of English and its computer analysis. Edinburgh University Press, Edinburgh (1973)
18. Lafourcade, M.: Making people play for lexical acquisition. In Proceedings of the th SNLP 2007, Pattaya, Thaland **7**, 13–15 (December 2007)

19. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. Computing Research Repository **arXiv:1301.3781** (2013), `https://arxiv.org/abs/1301.3781`
20. Miller, G., Charlees, W.: Contextual correlates of semantic similarity. Language and cognitive processes **6**(1), 1–28 (1991). https://doi.org/10.1080/01690969108406936
21. Nelson, D.L., McEvoy, C.L., Schreiber, T.A.: Word association rhyme and word fragment norms. The University of South Florida (1998)
22. Sahlgren, M.: The distributional hypothesis. Italian Journal of Disability Studies **20**, 33–53 (2008)
23. Schnabel, T., Labutov, I., Mimno, D., Joachims, T.: Evaluation methods for unsupervised word embeddings. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 298–307 (2015)
24. Sinopalnikova, A., Smrz, P.: Word association thesaurus as a resource for extending semantic networks. pp. 267–273 (2004)
25. Sowa, J.F.: Conceptual graphs as a universal knowledge representation. Computers & Mathematics with Applications **23**(2), 75–93 (1992). https://doi.org/10.1016/0898-1221(92)90137-7

Part IV

Short Papers

Semantic Width Revisited (Extended Abstract)

Georg Gottlob[1,2], Matthias Lanzinger[1], and Reinhard Pichler[1]

[1] TU Wien, Austria,
[2] University of Oxford, UK
{gottlob, mlanzing, pichler}@dbai.tuwien.ac.at

1 Introduction

Answering conjunctive queries (CQ) and, equivalently, solving constraint satisfaction problems (CSP), are among the most fundamental tasks in computer science. While these problems are NP-complete, there has been much success in identifying "islands of tractability". In this work, we want to further sharpen the image of this complexity landscape. In particular, we are interested in extending the pioneering work by Barceló, Pieris, Romero, and Vardi [1, 2], Dalmau, Kolaitis, and Vardi [4], and Grohe [6] on the deep connections between query minimization and structural decompositions. To this end, the following notion of *semantic widths* will be the central theme of our work. Note that from now on, we only concentrate on CQs. Clearly, all results hold for CSPs as well.

Definition 1. *Let $\mathcal{Q}$ be the class of all conjunctive queries and $w : \mathcal{Q} \to \mathbb{R}^+$ be some property of the query. We define* semantic w *as* sem-$w(q) := \inf\{w(q') \mid q' \simeq q\}$*, where $\simeq$ denotes homomorphic equivalence.*

Such semantic widths can be interpreted as measures of the inherent structural complexity of the underlying question posed by the query, whereas classical notions of width express the complexity of a specific way to pose the question.

So far, semantic notions of acyclicity [2], treewidth (*tw*) [4], and (generalized) hypertree width (*hw* and *ghw*, resp.) [1] have been investigated, while more powerful notions of width such as fractional hypertree width (*fhw*) [7], submodular width (*subw*) [9], and adaptive width (*adw*) [8] have been left unexplored. The goal of this work is to study semantic notions also of *fhw*, *adw*, and *subw*. Recall that the semantic notions of acyclicity, *tw*, and *ghw* can be characterized in terms of the *core* of the CQ. This naturally raises the question if such a characterization is also possible for *fhw*, *subw*, and *adw*. We will give an affirmative answer which, structurally, looks very similar to the previous results. However, some additional machinery will be needed to prove our new results.

In [9], *subw* was introduced to provide an in some sense "complete" characterization of the fixed-parameter tractability (FPT) of CQ Evaluation. Our investigation of the semantic version of *subw* will provide new input for such an FPT-characterization. More specifically, our new notion of sem-*subw* will allow us to identify an FPT-fragment of CQ Evaluation which is *strictly bigger* than the fragment defined via *subw* in [9].

Strongly related to the search for (fixed-parameter) tractable fragments of CQ Evaluation is the complexity of the Check problem which, for given integer $k \geq 1$, is about deciding if a given CQ has width $\leq k$. Among the width notions mentioned above, this problem is tractable for *tw* and *hw* and NP-complete for *ghw* and *fhw*. For *subw* and *adw*, the complexity is expected to be even higher. When moving to semantic notions, the computation of the core introduces a further source of intractability. In case of *ghw* and *fhw*, several structural properties of the hypergraph underlying a CQ have been identified recently [5] to make the Check problem tractable. We will show that these properties may also be helpful for the computation of the semantic variants of *ghw* and *fhw*.

2 Preliminaries

We assume the reader to be familiar with basic concepts such as conjunctive queries (CQs) and their associated hypergraphs. We implicitly extend properties of hypergraphs to CQs through their associated hypergraphs. Due to space limitations, we refer to [9] for definitions of the fractional cover number (ρ^*), generalized hypertree width (ghw), fractional hypertree width (fhw), adaptive width (adw), and submodular width ($subw$).

We are mainly interested in two computational problems here. The first one is the usual query evaluation problem for a class of CQs $\mathcal{Q}$, which we denote as Eval($\mathcal{Q}$). The decision problem of checking, whether a query has width $\leq k$ for width notion w, will be referred to as Check(w, k).

The problem Check(w, k) with $w \in \{ghw, fhw\}$ has been shown NP-complete even for $k = 2$ [5]. On the positive side, also tractable fragments of this problem have been identified in [5] via the following hypergraph properties: for a hypergraph H, the *c-multi-intersection width* of H refers to the maximum cardinality of an intersection of any c distinct edges. For the special case $c = 2$, we simply use the term *intersection width*. The *degree* of H is the maximum number of edges a vertex of H occurs in. The *rank* of H is the maximum edge size in H.

3 Results

The following theorem is the basis of all our further considerations:

Theorem 1. *For every conjunctive query q:*
1. sem-$\rho^*(q) = \rho^*(Core(q))$
2. sem-$fhw(q) = fhw(Core(q))$
3. sem-$adw(q) = adw(Core(q))$
4. sem-$subw(q) = subw(Core(q))$

An interesting consequence of Theorem 1 is that bounded semantic width of a class $\mathcal{Q}$ of CQs, for the notions of width enumerated in the theorem, implies FPT of the Eval($\mathcal{Q}$) problem when parameterized by the query. This is due to the fact that core computation only depends on the query (not the

data). This is of particular interest in the case of bounded sem-*subw*, because it properly generalizes bounded submodular width, and as such "escapes" the FPT-characterization theorem of Marx [9]. To see that the generalization is in fact proper, consider the class of "grid queries" $\mathcal{Q}_{\mathcal{G}_n}$, such that $\mathcal{Q}_{\mathcal{G}_n}$ asks if a given undirected graph contains a grid of size $\geq n$. The core of query $\mathcal{Q}_{\mathcal{G}_n}$ simply asks for the existence of a single (undirected) edge. However, the associated hypergraphs of the family $(\mathcal{Q}_{\mathcal{G}_n})_{n\geq 1}$ include grids of every dimension. Hence, $(\mathcal{Q}_{\mathcal{G}_n})_{n\geq 1}$ has unbounded treewidth. By considering the fractional independent set where every vertex has weight $1/rank(H)$, we get the inequality $subw(H) \geq adw(H) \geq (tw(H)+1)/rank(H)$. It is then clear that $\mathcal{Q}_{\mathcal{G}_n}$ has unbounded *subw*, which is in sharp contrast to sem-$subw(\mathcal{Q}_{\mathcal{G}_n}) = 1$.

Corollary 1. *Let $\mathcal{Q}$ be a class of CQs. Then bounded* sem-fhw, sem-*subw, and* sem-*adw are sufficient conditions for the fixed-parameter tractability of* $\textsc{Eval}(\mathcal{Q})$ *(parameterized by the query). Furthermore, they properly subsume bounded fhw, subw, and adw, respectively.*

In [5], it was shown that the Check problem of *ghw* and *fhw* becomes tractable if the underlying hypergraphs satisfy certain properties such as bounded (multi-)intersection width, bounded degree, and/or bounded rank. Note that all these properties are hereditary in the sense that deletion of vertices and/or edges from a hypergraph does not destroy these properties. Thus, using Theorem 1, we can directly identify tractable fragments of Check for sem-*ghw* and sem-fhw. We present Corollary 2 as an illustrative example for various similar results.

Corollary 2. *For a constant $k > 0$, let $\mathcal{Q}$ be a class of conjunctive queries with bounded fractional hypertree width and bounded degree, then* $\textsc{Check}(\text{sem-}fhw, k)$ *is tractable in $\mathcal{Q}$.*

There exist classes with bounded *fhw* but unbounded *ghw* [7]. However, little is known about the conditions under which this can occur. We show that for classes with either bounded degree or bounded intersection width, the properties of bounded *fhw* and bounded *ghw* (and, therefore, also bounded *hw*) in fact coincide. Furthermore, since sem-fhw and sem-*ghw*(cf. [1]) are characterized by the width of the core, it is easy to generalize the result to the semantic case. As a direct consequence, the promise algorithm presented by Chen and Dalmau in [3] can be adapted to classes with bounded sem-fhw and either bounded degree or bounded intersection width, making evaluation of these classes tractable.

Theorem 2. *Let q be a conjunctive query and let its associated hypergraph have degree d and intersection width i. The following statements hold:*

- $fhw(q) \leq ghw(q) \leq d\, fhw(q)$ *(Implicit in [5])*
- sem-$fhw(q) \leq$ sem-$ghw(q) \leq d\,$sem-$fhw(q)$
- $fhw(q) \leq ghw(q) \leq 2i\, fhw(q)^2 + 2\, fhw(q)$
- sem-$fhw(q) \leq$ sem-$ghw(q) \leq 2i\,$sem-$fhw(q)^2 + 2\,$sem-$fhw(q)$

4 Conclusion

So far, we have given a characterization of the semantic variants of ρ^*, fhw, adw, and $subw$. From this we are able to derive new insights into the complexity of CQs. Figure 1 illustrates our current view of the complexity landscape of CQs.

Many consequences of Theorem 1 are yet to be explored. Of particular interest is the role of sem-$subw$. Marx has shown that bounded $subw$ characterizes those hypergraphs for which evaluation of the associated CQs is fixed-parameter tractable [9]. The natural next step is to characterize precisely those CQs for which the evaluation is fixed-parameter tractable. Semantic submodular width is a natural candidate step in this direction but the question whether it provides a necessary condition for fixed-parameter tractable CQ evaluation remains an important open problem for future work.

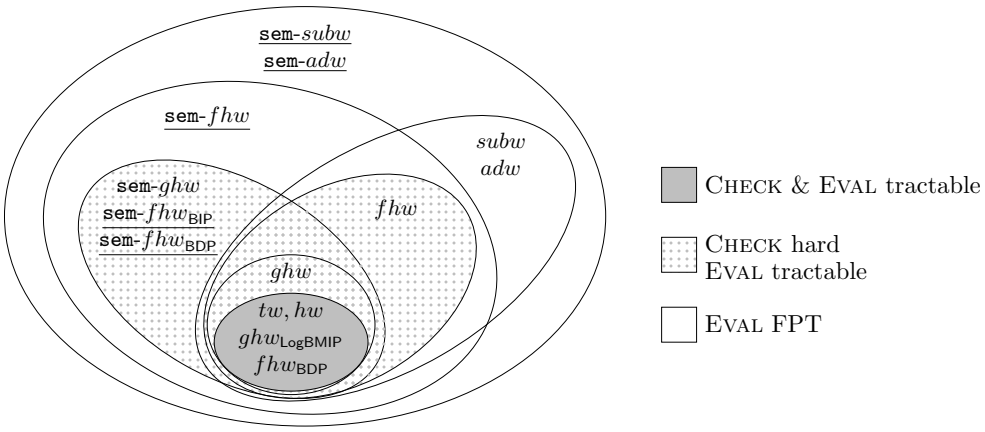


Fig. 1. CQ Complexity Landscape. Contributions presented in this paper are underlined. We write w_P to denote a width notion w on classes constrained to a property P where P is one of bounded degree (BDP), bounded intersection (BIP), or logarithmically bounded multi-intersection (LogBMIP).

Acknowledgements This work was supported by the Austrian Science Fund (FWF):P30930-N35.

References

1. Barceló, P., Pieris, A., Romero, M.: Semantic optimization in tractable classes of conjunctive queries. SIGMOD Rec. **46**(2), 5–17 (Sep 2017)
2. Barceló, P., Romero, M., Vardi, M.Y.: Semantic acyclicity on graph databases. SIAM Journal on Computing **45**(4), 1339–1376 (2016)
3. Chen, H., Dalmau, V.: Beyond hypertree width: Decomposition methods without decompositions. In: Proc. CP 2005. pp. 167–181. Springer (2005)

4. Dalmau, V., Kolaitis, P.G., Vardi, M.Y.: Constraint satisfaction, bounded treewidth, and finite-variable logics. In: Proc. CP 2002. pp. 310–326. Springer (2002)
5. Fischl, W., Gottlob, G., Pichler, R.: General and fractional hypertree decompositions: Hard and easy cases. In: Proc. PODS 2018. pp. 17–32. ACM (2018)
6. Grohe, M.: The complexity of homomorphism and constraint satisfaction problems seen from the other side. Journal of the ACM (JACM) **54**(1), 1 (2007)
7. Grohe, M., Marx, D.: Constraint solving via fractional edge covers. ACM Trans. Algorithms **11**(1), 4:1–4:20 (2014)
8. Marx, D.: Tractable structures for constraint satisfaction with truth tables. Theory of Computing Systems **48**(3), 444–464 (2011)
9. Marx, D.: Tractable hypergraph properties for constraint satisfaction and conjunctive queries. Journal of the ACM **60**(6), 42 (2013)

HyperBench: A Benchmark and Tool for Hypergraphs and Empirical Findings*

Wolfgang Fischl[1], Georg Gottlob[1,2], Davide M. Longo[1], and Reinhard Pichler[1]

[1] TU Wien, *firstname.surname*@tuwien.ac.at
[2] University of Oxford, georg.gottlob@cs.ox.ac.uk

1 Introduction

Answering Conjunctive Queries (CQs) and solving Constraint Satisfaction Problems (CSPs) are classical NP-complete problems of high relevance in Computer Science. Consequently, there has been an intensive search for tractable fragments of these problems over the past decades. We are mainly interested here in tractable fragments defined via decomposition of the underlying hypergraph structure of a given CQ or CSP. The most important decomposition methods are hypertree decompositions (HDs), generalized hypertree decompositions (GHDs), and fractional hypertree decompositions (FHDs) alongside the corresponding width notions hypertree width (*hw*), generalized hypertree width (*ghw*), and fractional hypertree width (*fhw*), respectively.

It has been shown that CQ answering and CSP solving are tractable on every class of CQs/CSPs, if the underlying hypergraphs have bounded $hw(H)$, $ghw(H)$, or $fhw(H)$. Since $fhw(H) \leq ghw(H) \leq hw(H)$ holds for every hypergraph H, bounded *fhw* defines the biggest tractable class of CQ answering and CSP solving. On the other hand, only for *hw*, it is feasible in polynomial time to recognize if a given hypergraph has width $\leq k$ for fixed k. In contrast, for *fhw* and *ghw*, the problem of recognizing low width is NP-complete even for $k = 2$ [7].

In [7], the following properties of the underlying hypergraphs have been identified to ensure tractable computation of GHDs and FHDs of a given width (if they exist) or at least to allow for a good approximation thereof.

Definition 1. *The* intersection width *iwidth*(H) *of a hypergraph* H *is the maximum cardinality of the intersection* $e_1 \cap e_2$ *of any two distinct edges* e_1, e_2 *of* H. *For positive integer* c, *the* c-multi-intersection width *c-miwidth*(H) *of a hypergraph* H *is the maximum cardinality of any intersection* $e_1 \cap \cdots \cap e_c$ *of* c *distinct edges* $e_1, \ldots, e_c$ *of* H. *The* degree $\deg(H)$ *of a hypergraph* H *is the maximum number of edges a vertex of* H *occurs in, i.e.,* $\max_{v \in V(H)} |\{e \in E(H) \mid v \in e\}|$.

We say that a class $\mathcal{C}$ *of hypergraphs has the* bounded intersection property (BIP), *the* bounded multi-intersection property (BMIP), *or the* bounded degree property (BDP) *if there exist constants* i, c, *and* d, *such that every hypergraph in* $\mathcal{C}$ *satisfies* $iwidth(H) \leq i$, *c-miwidth*$(H) \leq i$, *or* $\deg(H) \leq d$, *respectively.*

Indeed, it has been shown in [7] that, if it exists, a GHD of low width can be computed in PTIME for hypergraphs enjoying the BDP, BIP, or BMIP. For

* This is an extended abstract of [6].

FHDs, an exact PTIME algorithm has been presented in case of the BDP; for the BIP, a polynomial time approximation scheme (PTAS) exists.

Despite the appealing theoretical results, little is known in practice about these properties and their interplay with the various notions of width. The goal of this work is to remedy this deficit. More concretely, we investigate questions such as the following: Do the hypergraphs underlying CQs and CSPs in practice, indeed have low degree and (multi-)intersection width? Are these properties non-trivial in the sense that, e.g., low intersection width does not immediately lead to low *hw*, *ghw*, and *fhw*? We also want to get a better understanding of the relationship between *ghw* and *hw*. In general, only the inequality $hw(H) \leq 3 \cdot ghw(H) + 1$ is known. But do these two notions of width indeed differ by factor 3 in practice? And do theoretical tractability results (such as tractable GHD computation in case of the BIP) indeed lead to practically feasible computation?

In order to provide answers to these questions, we have collected a vast amount of CQs and CSPs from concrete applications as well as randomly generated ones, and translated them into a uniform hypergraph format. We have successively performed a series of experiments on these hypergraphs – determining the *hw* and *ghw*, the degree and (multi-)intersection width as well as further metrics relating to the size such as number of vertices, number of edges, and maximum size of edges. All the hypergraphs thus collected together with the results of our experiments are publicly available at http://hyperbench.dbai.tuwien.ac.at/. Below, we give a summary of these results. Moreover, we note that this benchmark has already been profitably applied in the experimental evaluation of decomposition algorithms by other authors [5].

2 Basic Definitions

We assume the reader to be familiar with basic concepts such as CQs and CSPs. The *hypergraph corresponding to a CQ* ϕ is defined as $H = (V(H), E(H))$, where the set of vertices is $V(H) = \mathit{variables}(\phi)$ and the set of edges is $E(H) = \{e \mid \exists A \in \mathit{atoms}(\phi) : e = \mathit{variables}(A)\}$. Due to lack of space, we also have to assume familiarity with the various notions of decompositions and width.

We have already introduced the crucial properties BDP, BIP, and BMIP. By slight abuse of notation, we shall say in the sequel that a hypergraph H has BDP $= d$, BIP $= i$, or c-BMIP $= i$, if H satisfies the conditions $\mathit{iwidth}(H) = i$, $\mathit{c\text{-}miwidth}(H) = i$, or $\deg(H) = d$, respectively.

3 Results

We have collected 3070 CQs and CSPs from different sources and converted them into hypergraphs. Our collection of CQs comprises 535 queries used in practical applications and 500 randomly generated ones using the tool from [11]. The non-random CQs stem from various sources. Queries in [4] come from a huge SPARQL repository comprising over 26 million CQs, from which we have included only the hypergraphs with $hw \geq 2$. Queries in [9] come from a big collection of SQL queries from which we have extracted over 15,000 CQs (in particular, no nested SELECTs). Again, we have only included hypergraphs with $hw \geq 2$ into our benchmark. The remaining non-random CQs come from different benchmarks

such as the Join Order Benchmark (JOB) [10] and TPC-H [12]. Our collection of 2035 CSPs is composed of 1953 instances from [2] and 82 instances used in previous analyses [3,8]. The instances from [2] are divided in two classes: 1090 of them come from applications and the remaining 863 are random instances.

Table 1. Percentage of instances having BDP, BIP, 3-BMIP, 4-BMIP ≤ 5.

	BDP (%)	BIP (%)	3-BMIP (%)	4-BMIP (%)
Application-CQs	81.68	100	100	100
Application-CSPs	53.67	99.91	100	100
Random	10.12	76.82	90.17	93.62

In our first experiment, we computed *BDP, BIP, 3-BMIP, 4-BMIP* for our collection of hypergraphs. The results are presented in Table 1. We group our instances in three classes: application-CQs, application-CSPs, and random instances (both CQs and CSPs). In the first place, we are interested in the percentage of hypergraphs whose values of BDP, BIP, 3-BMIP, and 4-BMIP are small. It turned out that all application-CQs have BIP ≤ 5 and yet smaller 3-BMIP and 4-BMIP. The BDP tends to be bigger, but there are still 81.68% of the application-CQs with BDP ≤ 5. As for the application-CSPs, the BIP and BMIP are also small in general, namely 99.91%, 100% and 100% have BIP ≤ 5, 3-BMIP ≤ 5, and 4-BMIP ≤ 5, respectively. Interestingly, the percentage of application-CSPs with BDP ≤ 5 is rather low (53.67%) compared with application-CQs. The random instances behave differently. Indeed, we have measured 76.82%, 90.17% and 93.62% of the random instances (with very similar behaviour of random CQs and random CSPs) to have small BIP, 3-BMIP, and 4-BMIP, respectively. The percentage of instances with BDP ≤ 5 even falls more dramatically to 10.12% for random instances. To conclude, BIP and BMIP indeed tend to be (very) small for both CQs and CSPs taken from applications and they are still reasonably small for random instances.

As a next step, we have systematically applied the computation of hw [8] to our benchmark. The aim of this experiment was to determine the hw or at least an upper bound thereof for each hypergraph. We have organized the computation in different rounds, each of which has a different value of k, which is initialized with $k = 1$. In each round, we check if $hw(H) \leq k$ holds and, if so, compute a concrete HD with this width. If the program ends with a yes-answer, we have an upper bound on hw. In case of a no-answer, we have a lower bound. No bound is obtained in case of a timeout (which we set at 3,600 seconds). For all the instances with no upper bound (i.e., either no-answer or timeout) for a value of k, we continue with $k := k + 1$ in the next round. We were able to determine that for all application-CQs $hw \leq 3$ holds and to compute concrete HDs of this width. For 694 of all 1172 application-CSPs (59.22%) we have verified $hw \leq 5$. In total, considering also random instances, 1849 (60.23%) out of 3070 instances have $hw \leq 5$. For 1453 of them, we determined exact hw, for the others we only have an upper bound and the actual value of hw could be even less. We conclude that

for the vast majority of CQs and CSPs (in particular those from applications), hw is small enough to allow for efficient CQ answering or CSP solving, respectively.

We have also analysed the correlation between all the hypergraph parameters studied here. BIP and BMIP are obviously highly correlated. More interestingly, we observe a high correlation between any two of the number of vertices, the arity (= maximum edge size), and hw. It is worth underlining that BDP, BIP, 3-BMIP, 4-BMIP have low correlation with hypertree width. That is, low values of these parameters are favourable for GHD computation [7] but they do not imply that also the hw and (as we will see below) the ghw are particularly small.

Finally, we have implemented several algorithms for GHD-computation, which exploit low BIP. For all hypergraphs with $hw \leq k$ and $k \in \{3, 4, 5, 6\}$, we checked whether $ghw \leq k - 1$ holds. To this end, we ran our algorithms with a timeout of 3,600 seconds. If the timeout does not occur, we say that the instance is "solved". We found out that in 98% of the solved cases and 57% of all instances with $hw \leq 6$, hw and ghw have identical values. Actually, we expect that the percentage in case of the solved case is more significant because the GHD computation algorithms usually take longer in case of a no-answers, i.e., we conjecture that most of the unsolved instances also have identical values of hw and ghw.

4 Conclusion

In this work, we have extensively experimented with a big collection of hypergraphs (from both CQs and CSPs). We have thus made several interesting observations, which have been summarized above. For instance, the discrepancy between hw and ghw seems to be much smaller in practice than the theoretical factor 3. Moreover, it has turned out that hypergraph parameters such as BIP, on the one hand, tend to be very small in practice and, on the other hand, low BIP is indeed very helpful for computing concrete decompositions – especially GHDs. This leads us to several directions for future work. Further improvements of our GHD algorithms and implementations are required to increase the number of "solved" instances. The development of algorithms exploiting low 3-BMIP or 4-BMIP seems to be a natural next step, since the latter parameters tend to be yet smaller than BIP. On the more theoretical side, it would be very interesting to settle the open question if bounded BIP also ensures tractable FHD-computation: so far, we only know that BIP allows for a PTAS for the fhw.

Acknowledgement

This work was supported by the Austrian Science Fund (FWF) project P30930-N35. Davide Mario Longo's work was supported by the Austrian Science Fund (FWF) project W1255-N23.

References

1. Arocena, P.C., Glavic, B., Ciucanu, R., Miller, R.J.: The ibench integration metadata generator. Proc. VLDB Endow. **9**(3), 108–119 (2015)

2. Audemard, G., Boussemart, F., Lecoutre, C., Piette, C.: XCSP3: an XML-based format designed to represent combinatorial constrained problems. http://xcsp.org
3. Berg, J., Lodha, N., Järvisalo, M., Szeider, S.: Maxsat benchmarks based on determining generalized hypertree-width. MaxSAT Evaluation 2017 p. 22 (2017)
4. Bonifati, A., Martens, W., Timm, T.: An analytical study of large SPARQL query logs. PVLDB **11**(2), 149–161 (2017)
5. Fichte, J., Hecher, M., Lodha, N., Szeider, S.: An SMT Approach to Fractional Hypertree Width. Proc. CP 2018, pp.109–127 (2018)
6. Fischl, W., Gottlob, G., Longo, D.M., Pichler, R.: Hyperbench: A benchmark and tool for hypergraphs and empirical findings. In: PODS 2019 (to appear) (2019)
7. Fischl, W., Gottlob, G., Pichler, R.: General and fractional hypertree decompositions: Hard and easy cases. In: Proc. PODS 2018. ACM (2018)
8. Gottlob, G., Samer, M.: A backtracking-based algorithm for hypertree decomposition. ACM Journal of Experimental Algorithmics **13** (2008)
9. Jain, S., Moritz, D., Halperin, D., Howe, B., Lazowska, E.: Sqlshare: Results from a multi-year sql-as-a-service experiment. In: Proc. SIGMOD 2016. ACM (2016)
10. Leis, V., Radke, B., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A., Neumann, T.: Query optimization through the looking glass, and what we found running the join order benchmark. The VLDB Journal (2017).
11. Pottinger, R., Halevy, A.: MiniCon: A scalable algorithm for answering queries using views. The VLDB Journal **10**(2-3), 182–198 (Sep 2001)
12. Transaction Processing Performance Council (TPC): TPC-H decision support benchmark. http://www.tpc.org/tpch/default.asp (2014)

Parallel Computation of Generalized Hypertree Decompositions*

Georg Gottlob,[1,2] Cem Okulmus,[1] and Reinhard Pichler[1]

[1]TU Wien, Austria, [2]University of Oxford, UK

1 Introduction

Answering Conjunctive Queries (CQs) and solving Constraint Satisfaction Problems (CSPs) are arguably among the most important tasks in Computer Science. They are classical NP-complete problems. However, they have tractable fragments for instances where the underlying hypergraphs are acyclic. There has been active research for several decades to generalize acyclicity with several notions of decompositions and width [4]. Here we focus on generalized hypertree decompositions (GHDs) and generalized hypertree width (ghw).

Deciding if a given CQ or CSP (strictly speaking, the underlying hypergraph H) has $ghw \leq k$ is NP-complete even for $k = 2$ [3]. However, it was also shown in [3] that the problem of deciding if a given hypergraph has $ghw(H) \leq k$ becomes tractable for fixed k under realistic restrictions. One such restriction is the Bounded Intersection Property (BIP): a hypergraph H has *intersection width* $\leq i$ (denoted as $iwidth(H) \leq i$), if the intersection of any two edges in H has at most i vertices. A class of hypergraphs satisfies the BIP, if there exists a constant i, such that every hypergraph $H \in \mathcal{C}$ has $iwidth(H) \leq i$.

In [2], three different algorithms for checking $ghw(H) \leq k$ (and, if so, computing a concrete GHD of width $\leq k$) were implemented and tested on the HyperBench benchmark [2] comprising over 3,000 hypergraphs derived from CQs and CSPs from various sources. All the algorithms thus implemented rely on the observation that hypergraphs of CQs or CSPs stemming from applications tend to have low *iwidth*. These GHD-computations were purely sequential, even though one of the algorithms seems to lend itself naturally to parallel processing: more precisely, this GHD-algorithm is based on so-called "balanced separators"; at each step of the top-down construction of a GHD, this algorithm searches for a set of at most k edges $\{e_1, \ldots, e_k\}$ (= a "balanced separator"), such that the vertex set $e_1 \cup \cdots \cup e_k$ splits the hypergraph into components whose size (measured in terms of the edges that intersect with each component) is at most half the size of the component processed by the parent node in the GHD.

Given that many of the experiments reported in [2] had high run times or were even stopped due to a time out (with default value 3,600 seconds), we have to look for a different computation strategy. The goal of this work is to provide a parallel computation of GHDs and to move the GHD-computation to a powerful cluster. To this end, we adopt the aforementioned GHD-algorithm

* This work was supported by the Austrian Science Fund (FWF):P30930-N35.

based on balanced separators (referred to as "b-seps", for short, in the sequel) and implement it in the Go programming language [1]. Developed at Google in 2009, it has already seen widespread use by a number of companies such as Dropbox, CloudFare, Netflix and by Google itself.

Below, we describe the challenges that we have faced in designing a parallel implementation of the b-seps approach:

- Finding a good design of the main targets for parallelization, namely the search for the next balanced separator and the recursive calls of the GHD-computation for the resulting components;
- Finding a way to partition the work space equally among CPUs;
- Designing parallel search to support efficient backtracking;
- Splitting resources equally among the search space during recursive calls;
- Introducing *subedges* of the edges in the given hypergraph (which is needed to exploit the low *iwidth*(H)) while avoiding unneeded combinatorial explosion.

Below, we summarize how we have dealt with the above challenges and we report on first, promising experimental results. We will show that the parallel approach is indeed able to significantly speed up the expensive GHD-computations and that it allowed us to solve some cases which were out of reach with the previous, purely sequential GHD-implementations in [2].

2 Preliminaries

We assume the reader to be familiar with basic notions such as conjunctive queries (CQs) and their corresponding hypergraphs. A GHD of a hypergraph $H = (V(H), E(H))$ is a tuple $\langle T, \chi, \lambda \rangle$ where $T = (N, E(T))$ is a tree, and χ and λ are labelling functions, which map to each node $n \in N$ two sets, $\chi(n) \subseteq V(H)$ and $\lambda(n) \subseteq E(H)$. We denote with $B(\lambda(n))$ the set $\{v \in V(H) \mid v \in e, e \in \lambda(n)\}$. The functions χ and λ have to satisfy the following conditions:

1. For each edge $e \in E(H)$ there exists a node $n \in N$ such that $e \subseteq \chi(n)$.
2. For each vertex $v \in V(H)$, $\{n \in N \mid v \in \chi(n)\}$ is a connected subtree of T.
3. For each node $n \in N$, we have that $\chi(n) \subseteq B(\lambda(n))$.

The *width of a GHD* (ghw) is the size of the largest label for λ. It was shown in [2] that for a class of hypergraphs enjoying the BIP, one can add polynomially many subedges of edges in $E(H)$ to ensure $B(\lambda(n)) = \chi(n)$ for every $n \in N$.

For a set of edges $S \subseteq E(H)$, we say two vertices $v_1, v_2 \in V(H)$ are $[S]$-*connected* if there is a path of edges $e_1, \ldots, e_n \in E(H) \backslash S$ such that $v_1 \in e_1$ and $v_2 \in e_n$ and for each pair in the path $e_i, e_{i+1}, i \leq n-1$ we have that e_i and e_{i+1} share a common vertex. We define an $[S]$-*component* to be a maximal set of $[S]$-connected vertices. The *size of an* $[S]$-*component* C is defined as the number of edges $e \in E(H)$ such that $e \cap C \neq \emptyset$. For a hypergraph H and a set of edges $S \subseteq E(H)$, we say that S is a *balanced separator* (b-sep) if all $[S]$-components of H have a size of $\frac{|E(H)|}{2}$ or less .

graph	sequential	parallel	speedup
Dubois-016.xml.hg	668.097 ms	45.39 ms	14.71
rand-25-10-25-87-24.xml.hg	5936.83 ms	879.99 ms	7.84
Nonogram-012-table.xml.hg	73976.08 ms	11057.68 ms	6.69
Pi-20-10-20-30-17.xml.hg	1439.71 ms	200.54 ms	7.17

Table 1. Running times of b-seps algorithm for width 3.

It was shown in [2] that, for every GHD $\langle T, \chi, \lambda \rangle$ of a hypergraph H, there exists a node $n \in N$ such that $\lambda(n)$ is a balanced separator of H. This property can be made use of when searching for a GHD of size k of a hypergraph H: if no such separator exists, then clearly there can be no GHD of H of width k.

3 Results

The Go programming language [1] has a model of concurrency that is based on Hoare's Communicating Sequential Processes [5]. The smallest concurrent unit of computation is called a "goroutine", essentially a light-weight thread.

For the b-seps algorithm, we looked at two key areas of parallelization: the search for b-seps and the recursive calls. For the search, while testing out a number of configurations, we settled ultimately on using two types of goroutines: A number of workers, spawned via a central master goroutine, which starts them and waits on exactly two conditions (whichever happens first): either one of the workers finds a b-sep, or none of them do and they all terminate on their own. In the first case it makes sure all workers terminate and continues the main computation. For the recursive calls, we so far only spawn a single goroutine for each of them and wait for each call to return its result (a GHD of the corresponding subhypergraph).

To split the work equally among the workers during the search, a simple work balancer (each worker receiving jobs from the central master goroutine) would address this nicely. However, we found that this introduces a considerable synchronization delay when compared with splitting up the work beforehand. We assume here that the choice of edges has only small influence on the time needed to check if they form a b-sep. Thus we split the search space of size $M = \binom{N}{k}$ by the number of workers W (and if there is some remainder, we simply increase the workload of the first ($M \mod W$) workers by 1). Each worker has its own iterator, thus minimizing the need for communication during the search.

Our algorithm must support backtracking, i.e., restarting the search and finding another solution (if it exists). If we do not keep track carefully where each worker left off when the parallel search halted earlier, we could be forced to repeat work. We address this by saving the above mentioned iterators, used by each worker, in the main thread so as to reuse them during backtracking.

Another challenge lies in parallelizing the recursive calls in such a way that the resources (CPU cores) are split among them, to reduce unnecessary synchro-

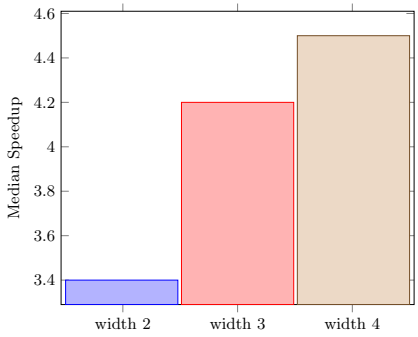


Figure 1. Median speedups for CSP Application instances from [3]

nization delays and not focus too much on one branch of the search tree. We have not yet found a satisfactory solution for this. One idea would be to go back to load balancing, and see if it helps with parallelizing the recursive calls, at the cost of taking away resources from the parallel search of b-seps.

The algorithm needs to compute *subedges* and consider them as possible choices for b-seps, as it would otherwise not be complete. The first and obvious choice is to add all possible subedges in the beginning (adding them globally). This leads to a combinatorial explosion, and more crucially also leads to many useless combinations, such as considering multiple subedges from the same original edge at once. We address this by adopting the local subedge variant from [3] and making it more restricted by first finding a balanced separator among the "original" edges of $E(H)$, and if this leads to a reject case, considering subedge variants of its edges. We also make sure not to repeat the same subedges by caching the vertex sets that generate them.

We used our parallel implementation to look at further instances that can be determined negatively (proving that no GHD of a certain width can exist) or positively (actually producing a GHD of that width). Our test setup was a cluster with 11 machines, each with two 12-core Intel Xeon CPU E5-2650 v4, clocked at 2.20GHz. For the tests, each job ran exclusively on a machine spawning up to 24 goroutines.

Possible speedups of four sample hypergraphs from HyberBench when compared to a sequential execution[1] are showcased in Table 1, where speedup is simply the sequential time divided by the parallel one. Furthermore, when comparing the execution times of the purely sequential version with parallelizing both the search for b-seps and parallelizing the recursive calls, we observe an increase in speedups at higher widths, seen in Figure 1. Additionally, we could produce new results for some previously unsolved instances of the HyperBench benchmark, thus determining their exact *ghw*. We are positive that with further

[1] 'Sequential execution' here refers to running the same general algorithm, but rewritten so that it does not use any goroutines. This version therefore only uses a single CPU core.

improvements, we will be able to "fill out the blank spots" on many hypergraphs of the HyperBench benchmark from [2] and in the process produce a more robust tool to compute GHDs.

4 Outlook

This paper is about work in progress. The next step will be to incorporate further optimizations into our Go implementation such as the following: (1) Applying various heuristics for ordering the hyperedges (to find b-seps faster), (2) caching of previous computations, and (3) looking at hybrid solutions which first apply the recursive splitting into subproblems via the b-seps approach and then (for sufficiently small subproblems) switch to one of the other (sequential) GHD-algorithms in [2]. The ultimate goal is, at least for all hypergraphs in the HyperBench benchmark where an upper bound on *ghw* of at most 6 is known (i.e., slightly more than 1,500 instances), to be able to compute the precise value of *ghw*. Currently, for about half of these instances, the exact *ghw* is still open.

References

1. Golang.org (Feb 2019), https://golang.org/
2. Fischl, W., Gottlob, G., Longo, D.M., Pichler, R.: Hyperbench: A benchmark and tool for hypergraphs and empirical findings. In: PODS 2019 (to appear) (2019)
3. Fischl, W., Gottlob, G., Pichler, R.: General and fractional hypertree decompositions: Hard and easy cases. In: Proc. PODS 2018. pp. 17–32 (2018)
4. Gottlob, G., Greco, G., Leone, N., Scarcello, F.: Hypertree decompositions: Questions and answers. In: Proc. PODS 2016. pp. 57–74 (2016)
5. Hoare, C.A.R.: Communicating sequential processes. Commun. ACM **21**(8), 666–677 (1978)

An Empirical Analysis of GraphQL API Schemas in Open Code Repositories and Package Registries

Yun Wan Kim[1], Mariano P. Consens[1], and Olaf Hartig[2]

[1] University of Toronto, Canada
timyun.kim@mail.utoronto.ca consens@mie.utoronto.ca
[2] Linköping University, Sweden
olaf.hartig@liu.se

Abstract. GraphQL is a query language for APIs that has been increasingly adopted by Web developers since its specification was open sourced in 2015. The GraphQL framework lets API clients tailor data requests by using queries that return JSON objects described using GraphQL Schema. We present initial results of an exploratory empirical study with the goal of characterizing GraphQL Schemas in open code repositories and package registries. Our first approach identifies over 20 thousand GraphQL-related projects in publicly accessible repositories hosted by GitHub. Our second, and complementary, approach uses package registries to find over 37 thousand dependent packages and repositories. In addition, over 2 thousand schema files were loaded into the GraphQL-JS reference implementation to conduct a detailed analysis of the schema information. Our study provides insights into the usage of different schema constructs, the number of distinct types and the most popular types in schemas, as well as the presence of cycles in schemas.

1 Motivation and Approach

The schema of a GraphQL API describes the data and the types of queries supported by the API. An empirical study of the GraphQL schemas used by open source projects, therefore, provides useful information about the characteristics of data interfaces. Currently, there is no comprehensive collection of such schemas or a tool that helps gather schemas from GraphQL APIs. The goal of the work presented in this paper is i) to establish a method to extract schemas into a single collection for analyses and ii) to conduct an empirical analysis of the schemas.

1.1 Data Collection Method

APIs-guru has the most comprehensive list of public GraphQL APIs with links to endpoints and their documentation. By using APIs-guru, combined with manual effort through keyword searching, we collected 67 schemas of distinct APIs. Authentication requirements for most publicly available APIs hindered the efficiency and possible automation of schema extraction. Hence, we decided to take a different approach by extracting schemas from open source repositories from GitHub and used three sources to identify GraphQL repositories.

GitHub API As of June, 2018, there were more than 20,000 repositories on GitHub matching the keyword "graphql" and 2,000 repositories matching the keywords "graphql api".

Libraries.io API Decan et al. [1] explored security vulnerabilities of NPM packages that were dependent on vulnerable packages. Following a similar method, we identified over 37,000 repositories dependent on GraphQL reference implementations.

GHTorrent Archived data of GHTorrent is hosted on Google's Big Query platform. We identified over 5,000 repositories matching the keyword "graphql".

Table 1. Summary of GraphQL repositories identified.

Method	NumRepositories
GitHub API	20,635
Libraries.io API	37,588
GHTorrent	5,188

Table 2. Number of dependent repositories for the most popular implementations.

Package	language	Count
NPM/graphql	JavaScript	12,700
Pypi/graphene	Python	310
Rubygems/graphql	Ruby	470

By using string search for schema for every repository file's full file-path, it was possible to identify exact path of potential schema files and their repository data. Our assumption is that this method returns a considerable portion of actual schemas available such that this portion is representative for the entire population of GraphQL schemas publicly availables. We found that schema files are most often named schema.json, schema.js, and schema.graphql for single-file schemas. For modular schemas, the files are most often separated by types, queries, mutations, and subscriptions but are contained in directories with the name schema or schemas.

After downloading all potential schema files, we tried to load each of them via GraphQL-JS. A successful attempt indicated a valid schema and a failure indicated an invalid schema or an irrelevant file. We identified duplicates through several methods including Levenshtein distance and cosine similarity.

2 Analysis Results

We identified a total of 2,777 valid but non-distinct schemas using the proposed method. 1,880 files were unique JSON-formatted schemas. We also conducted an exhaustive search excluding the "schema" keyword on all GraphQL-related repositories to collect a larger list of 3,949 schemas. The union of the two methods resulted in 4,095 schemas and, by using cosine similarity to filter duplicates, 2,081 schemas were unique. Figure 1 illustrates the number of schemas per source and the overlap of sources. This illustration shows that the different approaches to collect GraphQL schemas are non-redundant.

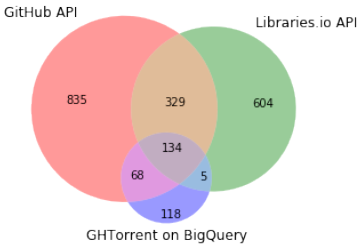


Fig. 1. Number of schemas by sources.

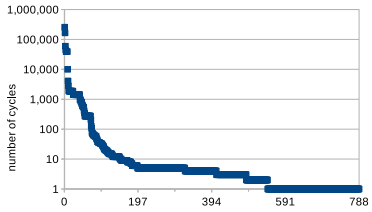


Fig. 2. Number of cycles per schema.

To estimate our recall, we downloaded all .json and .graphql files from all repositories found with the keyword "graphql". By using the 3,949 valid schema counts, the estimated recall of our method is ca. 70% and the precision is 1.8%.

There are five major components of GraphQL schemas that describes the supported operations: Query, Object, Mutation, Subscription, and Directive. While every GraphQL server needs to support queries, which fetch information about data objects, other operations are not necessarily required. Only about 20% of the schemas have the Subscription type that can push information, while about 70% have the Mutation type via which the stored data can be changed.

Table 3. Number of non-empty components in the 2,081 schemas.

Schema components	Frequency
object types	2,079
query type	2,079
directives	2,059
mutation type	1,440
subscription type	414

Table 4. The ten most common object types.

Object type	Frequency
Node	1,009
PageInfo	922
User	879
UserConnection	336
UserEdge	307
BatchPayload	220
Viewer	215
UserPreviousValues	190
Post	182

Object types dictate what information is exchanged between the users and the servers. We find that even after excluding scalar types and type definitions such as Query and Mutation, the most common types are generic types affiliated with reference implementations as shown in Table 4. Node is a reserved interface type for reference implementations such as Apollo and Relay with an identifier field and is the most common.

We traversed each schema in its JSON format recursively to identify their levels of nesting. We find that the median number of levels is 9 and the median number of levels only considering object types is 6. Excluding introspection and scalar type definitions, most schemas have only one level of nesting.

3 Cycles in GraphQL Schemas

Another interesting question is whether the relationships between the types in the schemas form directed cycles, because only if such cycles exist, the data

exposed via a GraphQL API may contain directed cycles and these, in turn, may cause an undesired overhead during query processing [2].

Hence, we analyze *GraphQL schemas as directed graphs*. The vertices in such a graph for a given schema correspond to the object types, the interface types, and the union types in the schema. For every field definition whose value type is based on one such type, the graph contains an edge from the vertex that represents the type in which the field definition appears to the vertex that represents the value type of the field definition. Additionally, there are edges from interface types to their implementing object types and, similarly, from union types to their participating object types. In this paper we focus only on *simple cycles*; that is directed cycles in which repetition of vertices is not allowed.

For the analysis we use a program[3] that loads a schema, generates the corresponding graph representation of this schema, and then enumerates the simple cycles in the generated graph. For the latter step, the program applies a combination of Johnson's algorithm [3] to enumerate the cycles and Tarjan's algorithm [4] to first divide the graph into its strongly connected components, which is a prerequisite of Johnson's algorithm. To run the program for each of the 2,094 schemas we use an ordinary desktop computer with 8 GB of RAM.

We find that 832 of the 2,094 schemas (39.7%) contain at least one simple cycle. For a more detailed analysis of these cycles we can, unfortunately, focus only on 788 of the 832 schemas; the other 44 schemas contain so many simple cycles (at least 10M in each of them) that enumerating these cycles causes the program to crash with an out-of-memory exception.

The distribution of the number of cycles in the remaining 788 schemas is illustrated in Figure 2. As can be observed, the distribution resembles a power law. In more detail, 2 schemas contain more than 100K cycles (that is 0.3% of the 788 schemas), where the maximum is 256,348 cycles; 9 schemas contain more than 10K cycles (that is 1.1%); 41 schemas contain more than 1K cycles (5.2%); 73 contain more than 100 cycles (9.3%); 152 contain at least 10 cycles (19.3%), and 543 contain more than one cycle (68.9%). Hence, 31.1% contain exactly one cycle only.

Moreover, the average length of all cycles within each schema ranges from 2.0 to 20.5, but there is no correlation between this average length and the number of cycles. Similarly, we do not find a correlation between the number of cycles and the number of vertices or edges.

4 Concluding Remarks

This preliminary report describes our approach to collect and analyze thousands of GraphQL schemas from open project repositories. Initial descriptive and structural properties of the collected schemas were presented. The collection has also enabled additional analysis (not included in this contribution) such as temporal characteristics of repository commits and co-committer relationships.

[3] https://github.com/LiUGraphQL/graphql-schema-cycles

Acknowledgements. The authors thank Jonas Lind and Kieron Soames who, as part of their thesis project at Linköping University, have developed the cycle enumeration program and applied it to our collection of schemas. Olaf Hartig's work on this paper has been funded by the CENIIT program at Linköping University (project no. 17.05).

References

1. Decan, A., Mens, T., Constantinou, E.: On the impact of security vulnerabilities in the npm package dependency network. In: Proceedings of the 15th International Conference on Mining Software Repositories. pp. 181–191. MSR '18 (2018), http://doi.acm.org/10.1145/3196398.3196401
2. Hartig, O., Pérez, J.: Semantics and Complexity of GraphQL. In: Proceedings of the 2018 World Wide Web Conference. pp. 1155–1164. WWW '18, Republic and Canton of Geneva, Switzerland (2018), https://doi.org/10.1145/3178876.3186014
3. Johnson, D.B.: Finding all the elementary circuits of a directed graph. SIAM J. Comput. **4**(1), 77–84 (1975). https://doi.org/10.1137/0204007, https://doi.org/10.1137/0204007
4. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972), https://doi.org/10.1137/0201010

Querying APIs with SPARQL

Matthieu Mosser, Fernando Pieressa, Juan Reutter,
Adrián Soto, Domagoj Vrgoč

Pontificia Universidad Católica de Chile

Abstract. Although the amount of RDF data has been steadily increasing over the years, the majority of information on the Web is still residing in other formats, and is often not accessible to Semantic Web services. A lot of this data is available through APIs serving JSON documents. In this work we propose a way of extending SPARQL with the option to consume JSON APIs and integrate the obtained information into SPARQL query answers. Looking to evaluate these queries as efficiently as possible we present an algorithm that produces "worst-case optimal" query plans that reduce the number of requests as much as possible. We also do a set of experiments that empirically confirm the optimality of our approach.

1 Introduction

The Semantic Web provides a platform for publishing data on the Web via the Resource Description Framework (RDF). Having a common format for data dissemination allows applications to access data obtained from different sources. However, the majority of the data available on the Web today is still not published as RDF, which makes it difficult to connect it to Semantic Web services. A huge amount of this data is made available through Web APIs which use a variety of different formats to provide data to the users.

We think that is important to make all of this data available to Semantic Web technologies, in order to create a truly connected Web. We propose an extension of SPARQL that allows us to connect to Web APIs, extending query answers with data obtained from a Web Service, in real time and without any setup.

With the ability of querying endpoints and APIs in real time we face an even more challenging task: How do we evaluate such queries? Connecting to APIs poses an interesting new problem from a database perspective, as the bottleneck shifts from disk access to the amount of API calls. Hence, to evaluate these queries efficiently we need to understand how to produce a query plan for them that minimizes the number of calls to the API.

Work supported by Millennium Institute for Foundational Research on Data (IMFD), Chile. This work was presented in ESWC 2018. [7]

2 SERVICE-to-API Queries

We extended the `SERVICE` operator to allow SPARQL to query Web APIs. We call a query that uses this extended `SERVICE` a **SERVICE-to-API** query. We assume the reader is familiar with the syntax and semantics of SPARQL 1.1 query

language [6]. We concentrate on the so-called REST Web APIs, which communicate via HTTP requests,assuming that all API responses are JSON documents. To illustrate how our extension works we will use the following example:

Example 1. We find ourselves in Scotland in order to do some hiking. We obtain a list of all Scottish mountains using the WikiData SPARQL endpoint, but we would prefer to hike in a place that is sunny. This information is not in WikiData, but is available through a weather service API called `weather.api`. This API implements HTTP requests, so for example to retrieve the weather on Ben Nevis, the highest mountain in the UK, we can issue a GET request with the IRI:

```
http://weather.api/request?q=Ben_Nevis
```

The API responds with a JSON document containing weather information, say of the form

```
{"timestamp": "24/10/2017 11:59:07", "temperature": 3,
"description": "clear sky", "coord": {"lat": 56.79, "long": -5.02}}
```

Therefore, to obtain all Scottish mountains with a favourable weather all we need to do is call the API for each mountain on our list, keeping only those records where the weather condition is `"clear sky"`. One can do this manually, but this quickly become cumbersome, particularly when the number of API calls is large. Instead, we propose to extend the functionality of SPARQL `SERVICE`, allowing it to communicate with JSON APIs such as the weather service above. For our example we can use the following (extended) query:

```
SELECT ?x ?l WHERE {
   ?x wdt:instanceOf wd:mountain . ?x wdt:locatedIn wd:Scotland .
   ?x rdfs:label ?l .
   SERVICE <http://weather.api/request?q={?l}>{(["description"]) AS (?d)}
   FILTER (?d = "clear sky")
  }
```

The first part of our query is meant to retrieve the IRI and label of the mountain in WikiData. The extended `SERVICE` operator then takes the (instantiated) URI template where the variable `?l` is replaced with the label of the mountain, and upon executing the API call processes the received JSON document using an expression `["description"]`, which extracts from this document the value under the key `description`, and binds it to the variable $?d$. Finally, we filter out those locations with undesirable weather conditions. □

We proposed a way to implement the overloaded `SERVICE` operation on top of any existing SPARQL engine. To do so, we partition each query using this operator into smaller pieces, and evaluate these using the original engine whenever possible. The answers given by the engine are extended with the results provided by the APIs. A full specification can be found in [7].

But can we optimize this queries? We discover that for SERVICE-to-API queries the bottleneck are the API calls. This is mostly because HTTP requests are slower that disk access and its something that we cannot control. So if we want to evaluate queries as efficiently as possible we need to do the least amount of API calls as possible. Then, can we reformulate query plans to make sure we are making as few calls as possible?

3 A Worst-case optimal algorithm

What we did is to propose an algorithm that is optimal in the worst case. This algorithm does not make more calls than the number we would need in the worst case over all graphs and APIs of a given size. For matters of space we will not explain all the algorithm and the proofs. The details can be found in [7].

Our algorithm is inspired by the optimal plan exhibited in [1,5] for conjunctive queries. To illustrate it, consider the SERVICE-to-API query of the example1. Note that the query is formed of basic graph patterns and a SERVICE-to-API pattern. We treat each basic graph pattern as a relation and each SERVICE-to-API as a relation with access methods (see e.g. [2,4]).

Then we can think that a SERVICE-to-API query has the form $Q = R_1 \bowtie R_2 \bowtie \ldots \bowtie R_m$. For the sake of presentation we consider that R_j $(1 \leq j \leq m)$ can represent a basic graph pattern or a SERVICE-to-API pattern where its attributes are the variables of the basic graph pattern or the input and output variables of the API. Then, let $A_1, \ldots, A_n$ be an enumeration of all attributes (variables of SPARQL) involved in Q, in order of their appearance. Starting from Q, we construct a query $Q^* = \Delta_n$, where the sequence $\Delta_1, \ldots, \Delta_n$ is defined as:

$$\Delta_1 = \pi_{A_1}(R_1) \bowtie \ldots \bowtie \pi_{A_1}(R_m).$$

$$\Delta_i = \Delta_{i-1} \bowtie \pi_{A_1,\ldots,A_i}(R_1) \bowtie \ldots \bowtie \pi_{A_1,\ldots,A_i}(R_m).$$

The idea is to process the query variable by variable. First we obtain the result of the intersection of all A_1 from the BGPs, and then we extend such BGPs with the values obtained from APIs where their input is the attribute A_1. Then we resolve the join for BGPs with variables A_1 and A_2 (considering the results of the previous iteration) and we extend the results with the output of the APIs with inputs A_1 and A_2. We continue this process until answer the query.

Our main result is the following. Take any SERVICE-to-API query Q, and a database D. Denote by $M_{Q,D}$ the maximum size of the projection of any relation appearing in Q over a single attribute in the database D. Furthermore, let $2^{\rho^*(Q,D)}$ be the AGM bound [1] of the query Q over D, i.e. the maximum size of the output of Q over any relational database having the same number of tuples in each relation as D. Then we can prove the following:

Theorem 1. *Any feasible join query under access methods Q can be evaluated over any database instance D using a number of calls in*

$$O(M_{Q,D} \times 2^{\rho^*(Q,D)}).$$

4 Experiments

We wanted to give empirical evidence that the worst-case optimal algorithm of Section 3 is indeed a superior evaluation strategy for executing queries that use API calls. To construct a benchmark for SERVICE-to-API patterns we reformulate the queries from the Berlin benchmark [3] by designating certain patterns in a query to act as an API call.

Algorithms. We consider three algorithms for `SERVICE`-to-API patterns: (1) *Vanilla*, a naive implementation without optimization; (2) *Without duplicates*, the base algorithm that uses caching to avoid doing the same API twice; and (3) *WCO*, the worst-case optimal algorithm of Section 3.
Results. The number of API calls done for each of the three versions of our algorithm are shown in Table 1. As we see, avoiding duplicate calls reduces the number of calls to some extent, but the best results are obtained when we use the worst-case optimal algorithm. We also measured the total time taken for the evaluation of these queries. Query times range from over 8000 seconds to just 0.7 seconds for the Vanilla version, and in average the use of *WCO* reduces by 40% the running times of the queries.

	Q1	**Q2**	**Q3**	**Q4**	**Q5**	**Q7**	**Q8**	**Q10**	**Q12**	**AVG**
Vanilla	5332	77	5000	5066	2254	15	1	7	1	0%
W/O Duplicates	4990	3	4990	4990	608	15	1	7	1	20%
WCO	2971	0	3284	4571	608	13	0	0	1	53%

Table 1. The number of API call per query for each algorithm. WCO plans average 53% reduction in API calls.

5 Conclusion

In this paper we propose a way to allow SPARQL queries to connect to HTTP APIs returning JSON. We give an intuition of the syntax and the semantics of this extension, discuss how it can be implemented on top of existing SPARQL engines, show an sketch of a worst-case optimal algorithm for processing these queries, and show the usefulness of this algorithm in practice. In future work, we plan to support formats other than JSON, and explore how to support it in public endpoints.

References

1. A. Atserias, M. Grohe, and D. Marx. Size bounds and query plans for relational joins. *SIAM J. Comput.*, 42(4):1737–1767, 2013.
2. M. Benedikt, J. Leblay, and E. Tsamoura. Querying with access patterns and integrity constraints. *PVLDB*, 8(6):690–701, 2015.
3. C. Bizer and A. Schultz. The berlin SPARQL benchmark. *Int. J. Semantic Web Inf. Syst.*, 5(2):1–24, 2009.
4. A. Calì and D. Martinenghi. Querying data under access limitations. In *ICDE 2008*, pages 50–59, 2008.
5. M. Grohe. Bounds and algorithms for joins via fractional edge covers. In *In Search of Elegance in the Theory and Practice of Computation*. Springer, 2013.
6. S. Harris and A. Seaborne. SPARQL 1.1 query language. *W3C*, 2013.
7. M. Mosser, F. Pieressa, J. L. Reutter, A. Soto, and D. Vrgoc. Querying apis with SPARQL: language and worst-case optimal algorithms. In *ESWC 2018*, pages 639–654, 2018.

On Directly Mapping Relational Databases to Property Graphs

Radu Stoica[1], George Fletcher[1], and Juan F. Sequeda[2]

[1] Eindhoven University of Technology, the Netherlands
{r.a.stoica@student., g.h.l.fletcher@}tue.nl
[2] Capsenta, Austin, Texas, USA
juan@capsenta.com

Abstract. Much of the data found in practice resides in relational DBs. However, many contemporary analytical tasks are performed on graphs. Property graphs are currently one of the most prevalent data models for graph data management in industry. Therefore, a key challenge is to understand the fundamental relationships between relational databases and property graph databases. This paper reports our ongoing work towards understanding these relationships by proposing R2PG-DM, a direct mapping of relational databases to property graphs. Given a relational database schema and instance, a direct mapping generates a corresponding property graph instance. The semantics of our mapping is defined using Datalog. Our work is inspired by existing approaches for direct mappings of relational databases into earlier graph data models. Future work is to study our mapping with respect to fundamental properties such as information and query preservation.

1 Introduction

A major class of contemporary data analytics focuses on gaining insights from the rich complex patterns found in graph-structured data collections such as social, communication, financial, biological, and mobility networks [1]. For example, investigative journalists have recently found, through graph analytics, surprising social relationships between executives of companies within the Offshore Leaks financial social network data set, linking company officers and their companies registered in the Bahamas.[3] Property graphs (PG) are currently one of the most prevalent data models for the management of such data in industry [1]. A PG is an edge- and node-labeled directed multigraph where both edges and nodes have associated sets of properties, i.e., key-value pairs.

The Offshore Leaks PG was constructed as a mapping from relational database (RDB) sources.[4] Indeed, much of the data found in practice resides in RDBs. Therefore, a key challenge is to understand the fundamental relationships between RDBs and PGs. A crucial first step in exploratory graph analytics on

[3] *International Consortium of Investigative Journalists.* https://offshoreleaks.icij.org/
[4] https://www.icij.org/blog/2013/06/how-we-built-offshore-leaks-database/

RDBs is to transform the database into a PG. Only then can the basic graph structure be explored and new relationships discovered using contemporary graph database systems.

This motivates the study of direct mappings from RDBs to PGs. A *direct mapping* is a transformation from RDBs to PGs which (1) is domain and schema-independent, i.e., works regardless of the source database schema and instance, and (2) transforms the content of the source instance into a target instance, i.e., given a RDB instance generates a corresponding PG instance.

State of the art. Most approaches to graph analytics over relational data extract graphs as user-specified views on relational data, e.g., [3,8]. This requires, however, that the user already fully understands the desired graph view, which is overly restrictive in the common case of exploratory graph analytics.

The study of direct mappings from relational to property graphs is not as well-developed. Of the small handful of approaches, the focus has been on optimized layout for efficient query processing or mappings which are lossy or obfuscate the input RDB schema [4,5,7,9]. Furthermore, all current direct mappings are defined procedurally (i.e., defined with pseudo-code), making it difficult to formally reason about their correctness and other basic properties.

Our contributions. In this short note, we report on our work-in-progress on R2PG-DM, a declarative direct mapping from RDB to PG. R2PG-DM losslessly transforms both the source instance and schema while also intuitively preserving the original structure of the input RDB. Our work is inspired by the approach of Sequeda et al. to direct mappings from relational to RDF graphs, the W3C standard for sharing graph-structured data on the web [6]. We present R2PG-DM by example and outline the basic research questions we are currently investigating in our study of the connections between RDB and PG.

2 RDBs, PGs, and direct mappings

Let $\mathcal{D}$ be an enumerable set of *data values* containing the special value null, and let $\mathcal{A}$ be an enumerable set of *attribute names* containing the special attribute tid. An RDB *schema* is a triple $\mathbf{S} = (R, att, \Sigma)$, where R is a finite set of *relation names*, att is a function assigning to each $r \in R$ a finite set $att(r) \subseteq \mathcal{A} \setminus \{\mathsf{tid}\}$, and Σ is a finite set of *primary* and *foreign keys* over R and att.[5] For $r \in R$, an *r-tuple* is a function $t : att(r) \cup \{\mathsf{tid}\} \to \mathcal{D}$ such that $t(\mathsf{tid}) \neq \mathsf{null}$. An *instance* I of $\mathbf{S}$ is an assignment to each $r \in R$ of a finite set $\mathsf{I}(r)$ of r-tuples satisfying Σ, such that for distinct $t, t' \in \bigcup_{r \in R} \mathsf{I}(r)$ it holds that $t(\mathsf{tid}) \neq t'(\mathsf{tid})$. An example schema and instance is given in Figure 1 (top left).

A *property graph* is a structure (V, E, e, ℓ, p) where V is a finite set of *vertices*, E is a finite set of *edges*, e is a function assigning an ordered pair of vertices to each edge (i.e., the source and target vertices of the edge, resp.), ℓ assigns a finite set of *labels* to each vertex and edge (from some domain of labels), and p assigns a finite set of *key-value pairs* to each vertex and edge. An example property graph is given in Figure 1 (bottom left).

[5] https://en.wikipedia.org/wiki/Foreign_key

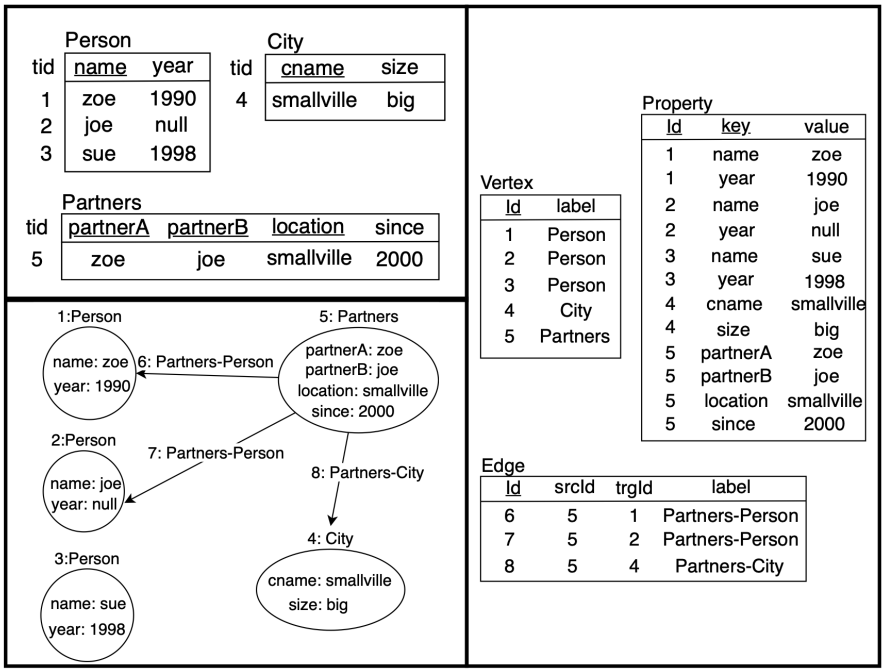


Fig. 1. (top left) An RDB $(\mathbf{S}, \mathsf{I})$, where primary key attributes are underlined, partnerA and partnerB of Partners are foreign keys to name of Person, and location of Partners is a foreign key to cname of City. (bottom left) The property graph R2PG-DM$(\mathbf{S}, \mathsf{I})$ and (right) its representation with predicates $Vertex$, $Edge$, and $Property$.

Let $\mathcal{PG}$ denote the set of all PGs and $\mathcal{RDB}$ denote the set of all pairs $(\mathbf{S}, \mathsf{I})$ where $\mathbf{S}$ is an RDB schema and I is an instance of $\mathbf{S}$. A *direct mapping* is a total function from $\mathcal{RDB}$ to $\mathcal{PG}$.

3 The R2PG-DM direct mapping

We next informally present R2PG-DM, a direct mapping which addresses the shortcomings of current solutions discussed in Section 1. With R2PG-DM, RDB relations are interpreted as classes of "things" (i.e., labeled vertices) and foreign-key relationships are interpreted as edges between these things. In particular:

- each input r-tuple t is represented in the output graph by a vertex v (identified by $t(\mathsf{tid})$), which is labeled with r, the name of the relation to which t belongs; furthermore, v has key-value pair $(a, t(a))$, for each $a \in att(r)$.
- for each foreign key relationship $f \in \Sigma$ (from, say, relation r to relation s), for each pair of tuples $t \in \mathsf{I}(r)$ and $t' \in \mathsf{I}(s)$ participating in f, this relationship is represented by an edge with label r-s from vertex v_t to vertex $v_{t'}$, representing t and t', respectively.

Similarly, as there currently does not exist a standard PG schema language, the input schema is also fully represented in the output PG, e.g., each relation and each attribute of a relation is represented by a vertex, and foreign keys are represented by edges between the relevant attributes of relations, etc. Figure 1 illustrates an RDB database $(\mathbf{S}, \mathsf{I})$ and PG translation R2PG-DM$(\mathbf{S}, \mathsf{I})$. We omit here the translation of $\mathbf{S}$ to PG, and only illustrate the translation of I.

Clearly, each component of the R2PG-DM mapping can be specified by a declarative non-recursive Datalog query [2] over the source DB represented using a fixed set of predicates (see Section 4.1 "Storing relational databases" of Sequeda et al. [6]), with the target PG represented by three predicates $Vertex$, $Edge$, and $Property$ capturing the vertices, edges, and the key-value properties associated with vertices and edges, respectively. This is illustrated in Figure 1 (right).

4 Ongoing research

Our ongoing work proceeds along three lines. First, we are proceeding with a formal study of R2PG-DM, establishing basic properties such as information and query preservation [6]; we hypothesize that R2PG-DM generates a graph that is isomorphic as a graph generated by the direct mapping of [6]. Moreover, we are studying how to extend R2PG-DM in order to generate a graph with properties on edges, taking full advantage of the data model. Second, we are developing practical tools for efficient and scalable R2PG-DM, to be made available as open source code. Third, we aim to extend the R2PG-DM approach to support customized mappings for the cases where there is a schema defined on the target instance (e.g., mapping the relational schema $\mathbf{S}$ to an equivalent PG schema). This builds upon ongoing efforts towards standards for PG schema languages.[6]

References

1. Angela Bonifati, George Fletcher, Hannes Voigt, Nikolay Yakovets. *Querying Graphs.* Morgan & Claypool, 2018.
2. Todd J. Green et al. Datalog and recursive query processing. *Foundations and Trends in Databases* 5(2):105-195, 2013.
3. Mohamed S. Hassan et al. Extending in-memory relational database engines with native graph support. In *EDBT 2018*, pages 25-36.
4. Ognjen Orel, Slaven Zakošek, Mirta Baranović. Property oriented relational-to-graph database conversion. *Automatika* 57(3): 836-845, 2016.
5. Subhesh Pradhan, Sharma Chakravarthy, Aditya Telang. Modeling relational data as graphs for mining. In *COMAD 2009*.
6. Juan F. Sequeda, Marcelo Arenas, Daniel P. Miranker. On directly mapping relational databases to RDF and OWL. In *WWW 2012*, pages 649-658.
7. Roberto De Virgilio, Antonio Maccioni, Riccardo Torlone. R2G: a tool for migrating relations to graphs. In *EDBT 2014*, pages 640-643.
8. Konstantinos Xirogiannopoulos, Amol Deshpande. Extracting and analyzing hidden graphs from relational databases. In *SIGMOD 2017*, pages 897-912.
9. Kang Min Yoo, Sungchan Park, Sang-Goo Lee. RDB2Graph: a generic framework for modeling relational databases as graphs. In *JIST 2014*, pages 148-151.

[6] `https://www.w3.org/Data/events/data-ws-2019/`

Linear Recursion in G-CORE

Valentina Urzua and Claudio Gutierrez

Department of Computer Science, Universidad de Chile and IMFD.

Abstract. G-CORE is a query language with two key characteristics: It is closed under graphs and incoporates paths as first-class citizens. Currently G-CORE does not have recursion. In this paper we propose this extension and show how to code classical polynomial graph algorithms with it.

Keywords: G-CORE, Recursion, Graph Query Language

1 Introduction

G-CORE is a graph database query language designed by a working group belonging to the *Linked Data Benchmark Council*[1]. One of its most novel features is the ability to express paths as first-class citizens. However, path queries are still not enough to express a wide variety of "natural" queries, particularly classical graph algorithms like topological sort, BFS, Eulerian circuit, etc.

We propose to extend G-CORE with (linear) recursive functionalities by introducing a syntax and a semantics that essentially follow the ideas of similar constructs in SQL and SPARQL[1]. What is more novel is that we take the core basic polynomial algorithms in graph theory and show, first, that they cannot be coded in standard G-CORE; and second, how to write queries that code them in G-CORE extended with linear recursion. Finally, we tested the code in an evaluator for G-CORE that is being developed by Roberto García at the University of Talca.

Basic Notions. A *Path Property Graph* (PPG) is an edge and node labeled graph where edges and nodes additionally have property-value pairs. In addition, a PPG may also have a collection of paths, where a path is a concatenation of existing, adjacent, edges in the graph.

A basic *G-CORE query* is an expression of the form:

$$\texttt{CONSTRUCT}\ f\ \texttt{MATCH}\ \gamma\ \texttt{ON}\ G\ \texttt{WHERE}\ \xi \tag{1}$$

where f is a full construct pattern, γ is a full graph pattern and ξ a Boolean condition [2]. The core of a G-CORE query consists in the complete graph pattern γ that defines the content of the `MATCH` clause. The `MATCH` clause is evaluated against the PPG G and returns a set of bindings that are filtered with the `WHERE` clause, to finally build a new PPG H with the `CONSTRUCT` clause.

[1] Ongoing research

Related Work. Adding recursion to database query languages has been extensively studied both from a theoretical [3] as well as from a practical point of view (e.g. SQL, SPARQL [4, 5]). In SQL it was included in the SQL-99 standard and was developed via common table expressions (CTE'S) that have a base SELECT statement and a recursive SELECT statement, that allows to express graph queries like DFS, BFS, topological sort, connected components, etc. Since queries must be linear and many interesting queries are not [6], optimizations have been proposed in [7, 8] (r-sql proposal) that allow only linear recursion and not the explicit negation that is a limitation when implementing recursion [3].

The graph algorithms BFS and DFS were implemented in [9] with the recursive SQL operator, where only trees were allowed as input since otherwise the query would loop infinitely. The topological order was studied in [10], making a BFS starting from the node whose outdegree is zero. In the case of the connected components it is shown in [11] how to obtain them adding recursion.

For SPARQL, Reutter et al [5] proposed a recursion operator based on SQL and make a comparison with property paths. In their study they formalize the syntax of the recursive operator and develop algorithms for evaluating it in practical scenarios. Also, a comparative study of the expressiveness of property paths and the recursive SQL operator was made in [6].

2 Adding a Recursive Operator to G-CORE

Among the main issues when adding recursion to query languages are the complexity of the evaluation, the expressiveness and the efforts to keep the declarative character of queries. From a theoretical point of view, recursive operators are based on the theory of least fixed points [3], that is, the increasing accumulation of results until eventually the evaluation does not add anything else and stops. Our proposal follows these ideas and is based both on the recursive SPARQL operator defined by Reutter et al [5] and the recursive SQL operator [4].

Proposed Syntax of linear recursive queries.

$$\texttt{WITH RECURSIVE } t \texttt{ AS } \{\ q_{base} \texttt{ UNION } q_{rec}\ \}\ q_{out}, \tag{2}$$

where t is a temporary PPG, q_{base} is a usual G-CORE query, q_{rec} is a positive G-CORE query that can use the temporary graph t and q_{out} a recursive G-CORE query itself.

Semantics of recursive queries. First, recall that the queries q_{base} and q_{rec} are usual G-CORE queries, therefore, are of the form (1). Second, recall that the evaluation of a usual query of the form (1) against a PPG G outputs a binding table $M(G)$, and from it the `CONSTRUCT` clause returns the PPG consisting of the union of $f(\ell)$ for each $\ell \in M(G)$. In what follows we will denote by M_b the binding table and by $C_b(\ell)$ the PPG returned by the `CONSTRUCT` clause of over the row ℓ of M_b of the base query q_{base}. Similarly we denote C_r and M_r respectively for the query q_{rec}.

Proposed Semantics of linear recursive queries. Let q be a recursive G-CORE query (like (2)) and G be a PPG. The answer $\text{ans}(q, G)$ of the query corresponds to the least fixed point of the sequence given by:

$$G_0 = G, \quad G_{-1} = \emptyset,$$
$$G_{i+1} = G_i \cup \{\text{ans}(q_{rec}, G + (G_i - G_{i-1}))\},$$

where: (1) the difference $(G_i - G_{i-1})$ is similar to the G-CORE difference operator defined in [2], except that the difference of the sets of nodes is redefined as $(N_i \backslash N_{i-1}) \cup \{a \in N_{i-1} : \ a \text{ is adjacent in } G_i \text{ to a node in } N_i \backslash N_{i-1}\}$; and (2) the semantics of $\text{ans}(q, G + H)$ means the match of q can use G or H or both (note that $\text{ans}(q, G \cup H)$ would not the same).

The answer $\text{ans}(q, G)$ of the recursive G-CORE query q over G is given by the following procedure:

Algorithm 1 Computing the answer of a recursive query in G-CORE

Data: PPG G, Queue of PPG'S $Q = \emptyset$, PPG $t = \emptyset$
for ***each*** *row l of the binding table $M_b(G)$* **do**
  `Insert`$(C_b(l), Q)$
while $Q \neq \emptyset$ **do**
  `Set` $r \leftarrow$ `Extract`(Q)
  $t \leftarrow t \cup r$
  for ***each*** *row l of the binding table $M_r(G + r)$* **do**
    `Insert`$(C_r(l), Q)$
return $ans(q_{out}, t + G)$

The idea is simple: first with the base query we construct the base case. Then with the recursive query we recursively construct a temporary graph which codes the solution that the output query will use.

Thus, first we begin with the queue Q empty and the temporary graph t empty. Then we evaluate the matching clause of q_{base} and for each row of the produced binding table $M_b(G)$, insert $C_b(l)$ into Q. Then we enter the while loop. As long as $Q \neq \emptyset$, we extract the top element of Q and update the temporary graph with it. Then for each row ℓ of the binding table $M_r(G+r)$ obtained from the query q_{rec} against $G+r$ (recall that this means that it could use the original graph G as well as the recently obtained graph r), we insert into the queue Q the new results obtained by the recursive construct $C_r(l)$. When we exhaust the queue Q the temporary graph t is finally ready to be used, and the query q_{out} is evaluated against $t+G$. If q_{out} is a standard G-CORE query the process outputs the result. If q_{out} is a recursive G-CORE query, we proceed as before again.

Notice that the fixed point exists whenever the query is monotone. On the form of query definition in (2) and tis semantics restricts queries to be linear [3–5, 11], that is, you can consult only the new elements added to `t`.

3 An Example: Topological Sort

In graph theory there are problems that have been widely studied and can be solved recursively in polynomial time. Due to space restriction, we will show as an example how the case of topological sort can be coded in recursive G-CORE.

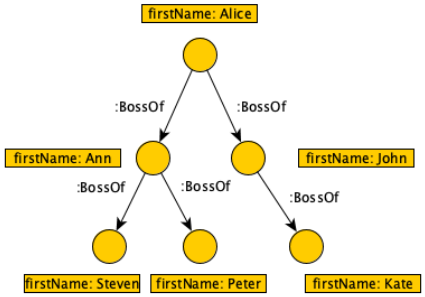


Fig. 1. A PPG G that represents hierarchy of a company

The classic algorithm to obtain the topological sort given an acyclic directed graph is Kahn algorithm (1962). It is not difficult to see that topological sort cannot be expressed in G-CORE. The next query illustrates with the graph of Fig. 1 how to code topological sort in G-CORE:

```
WITH RECURSIVE t AS (
          CONSTRUCT (n) SET n.depth :=0,
          MATCH (n) ON G,
          WHERE NOT EXISTS (CONSTRUCT (y),
                            MATCH (n)-[:BossOf]->(y) ON G)
          UNION{
          CONSTRUCT (x) SET x.depth:=n.depth+1,
          MATCH (x) ON G,
                (n) ON t
          WHERE EXISTS (CONSTRUCT (x),
                        MATCH (x)-[:BossOf]->(n) ON G)}
SELECT z.id,
MATCH (z) ON t,
ORDER BY MAX(z.depth) DESC
```

The above query first looks for those nodes which have no outgoing edges (base case) and are assigned with depth equal to 0 as a property. Then the recursive case comes and the nodes which have out edges to the nodes of the base case are added with property depth equal to 1 and so on until all the nodes in G have been added. Finally, we order by depth and we take the maximum value of the property depth.

References

1. LBDC. http://ldbcouncil.org/.
2. Renzo Angles, Marcelo Arenas, Pablo Barceló, Peter Boncz, George Fletcher, Claudio Gutierrez, Tobias Lindaaker, Marcus Paradies, Stefan Plantikow, Juan Sequeda, Oskar Van, Alex Averbuch, Hassan Chaa, Irini Fundulaki, Alastair Green, Josep Lluis, Larriba Pey, Jan Michels, Raquel Pau, Arnau Prat, Tomer Sagi, and Yinglong Xia. G-CORE A Core for Future Graph query Languages Designed by the LDBC Graph query Language Task Force *. *In: SIGMOD*, 2017.
3. Richard Hull Serge Abiteboul and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
4. Jim Melton and Alan R Simon. *SQL: 1999-Understanding Relational Language Components*. Morgan Kaufmann, 2001.
5. Juan L. Reutter, Adrián Soto, and Domagoj Vrgoč. Recursion in SPARQL. pages 19–35, Berlin, Heidelberg, 2015. Springer-Verlag.
6. N Yakovets, P Godfrey, and J Gryz. Evaluation of SPARQL property paths via recursive SQL. *CEUR Workshop Proceedings*, 1087, 01 2013.
7. Gabriel Aranda, Susana Nieva, Fernando Sáenz-Pérez, and Jaime Sánchez-Hernández. Formalizing a broader recursion coverage in SQL. pages 93–108, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
8. Carlos Ordonez. Optimization of linear recursive queries in SQL. *IEEE Transactions on Knowledge and Data Engineering*, 22, 2010.
9. Devin W. Homan. Transitive Closure in SQL. http://dwhoman.com/blog/sql-transitive-closure.html.
10. FusionBox. Graph Algorithms in a Database . https://www.fusionbox.com/blog/detail/graph-algorithms-in-a-database-recursive-ctes-and-topological-sort-with-postgres/620/.
11. Torsten Grust. Advanced SQL. https://db.inf.uni-tuebingen.de/staticfiles/teaching/ss17/advanced-sql/slides/advanced-sql-05.pdf.

Towards Reconciling Certain Answers and SPARQL: Bag Semantics to the Rescue?*

Sebastian Skritek

TU Wien, skritek@dbai.tuwien.ac.at

1 Introduction

Despite the intention of RDF, the data model for the Semantic Web, to support reasoning about RDF data (as witnessed e.g. by three different semantics for RDF – simple/RDF/RDFS entailment – or accompanying standards like OWL), SPARQL, the standardized query language for RDF, lacks some support of including such reasoning into query answering, as was noted e.g. in [6, 1].

One major cause of this is that SPARQL only makes partially use of *certain answers*, which are the semantics commonly applied in reasoning scenarios. While not fully supporting certain answers might be a reasonable decision when looking at the costs of evaluating and (for a user) understanding a query, it negatively affects the power of SPARQL. As a result, in recent years suggestions have been made how to cover different aspects of reasoning about RDF data (e.g. blank nodes under RDF simple entailment [6]; OWL reasoning [1]) in SPARQL by adopting a certain answer semantics.

However, as observed in [1], just applying the usual definition of certain answers to SPARQL can, in certain cases, be unnecessarily restrictive. To better illustrate these situations, let us first recall the definition of certain answers.

Given an RDF graph G (possibly extended by some additional information), most reasoning formalisms define the semantics of G in terms of an (infinite) set $models(G)$ of RDF graphs implied by G. Query answering in such a setting is then commonly defined in terms of the *certain answer semantic*: for a query Q, the certain answers $certain(Q, G)$ (w.r.t. some reasoning formalism) are

$$certain(Q, G) = \bigcap_{G' \in models(G)} Q(G'),$$

where $Q(G')$ denotes the result of evaluating Q over the RDF graph G'.

While this definition can be immediately applied to SPARQL queries, problems arise e.g. when looking at the `OPTIONAL` operator, or more precisely at the weakly monotone classes of SPARQL queries, like the well-designed queries [10].

Example 1. Consider the SPARQL query Q

`SELECT` ?`auth`, ?`award` `WHERE` {?`auth` `writes` ?`b`} `OPTIONAL` {?`b` `receives` ?`award`}

and an RDF graph $G = \{(a, \texttt{writes}, b)\}$. Assuming that $models(G)$ contains all supersets of G (like e.g. under the RDF simple semantics), $certain(Q, G) = \emptyset$.

* This work was supported by the Austrian Science Fund (FWF): P30930-N35

Given that $(a, \texttt{writes}, b) \in G'$ for every $G' \in \mathit{models}(G)$, this is an unintuitive result. However, the mapping $\mu = \{(?\texttt{auth}, a)\}$ is no certain answer because it is not part of $Q(G')$ for every $G' \in \mathit{models}(G)$. For example, take $G' = G \cup \{(b, \texttt{receives}, p)\}$. Then $Q(G') = \{\{(?\texttt{auth}, a), (?\texttt{award}, p)\}\}$.

What makes this situation unintuitive is that while $\mu \notin Q(G')$, an *extension* of μ is contained in $Q(G')$. In fact, weakly monotone queries are exactly those queries Q where for all pairs of RDF graphs $G \subseteq G'$ the result $Q(G')$ contains a not necessarily proper extension of every mapping in $Q(G)$ [3].

To account for this, based on the notion of subsumption (a mapping μ' subsumes a mapping μ if μ' extends μ), an alternative certain answer semantics was proposed in [1]. One challenge in devising such a semantics is to avoid introducing unjustified subsumed mappings to the certain answers.

Example 2. Consider the query Q from Example 1 and let the RDF graph G_1 be the RDF graph G' from Example 1. Assuming $\mathit{models}(G_1)$ to contain all supersets of G_1, one would expect $\mathit{certain}(Q, G_1) = \{\{(?\texttt{auth}, a), (?\texttt{award}, p)\}\}$, while there is no justification for $\{(?\texttt{auth}, a)\} \in \mathit{certain}(Q, G_1)$. However, for $G_2 = G_1 \cup \{(a, \texttt{writes}, c)\}$, intuitively $\mathit{certain}(Q, G_2) = \{\{(?\texttt{auth}, a), (?\texttt{award}, p)\}, \{(?\texttt{auth}, a)\}\}$. This is because, unlike for G_1, over each graph in $\mathit{models}(G_2)$ at least two different mappings contribute a solution, namely $\{(?\texttt{auth}, a), (?\texttt{b}, b), (?\texttt{award}, p)\}$ and $\{(?\texttt{auth}, a), (?\texttt{b}, c)\}$. However, due to projection, these two mappings may lead to the same solution: for $G_3 = G_2 \cup \{(c, \texttt{receives}, p)\}$ we get (under set semantics) $Q(G_1) = Q(G_3)$. Under bag semantics, $Q(G_1)$ and $Q(G_3)$ still contain the same mapping, but it occurs once in $Q(G_1)$ and twice in $Q(G_3)$.

As a result, these two cases cannot be distinguished under set semantics, which prevents a definition of certain answers that acknowledges the differences between these cases. In [1] this was resolved by excluding all subsumed mappings from the certain answers. E.g., in the above example, $\{(?\texttt{auth}, a), (?\texttt{award}, p)\}$ would be the only certain answer for both, G_1 and G_2. While being a sensible definition, it is nevertheless a little ad hoc.

Given recent advances in SPARQL query answering and reasoning under bag semantics (cf. [6, 9, 2, 4, 5]), in this ongoing work we are revisiting the definition of a certain answer semantics for SPARQL under *bag semantics*, with the final goal to devise a certain answer semantics that (more) adequately describes the certain information returned by weakly monotone queries. This submission does not present new results, but suggests a possible certain answer semantics, (hopefully) showcasing that revisiting certain answer semantics for SPARQL is worthwhile.

2 Subsumption between Bags

A possible way of defining a certain answer semantics that faithfully includes subsumed mappings is to introduce a "subsumption-aware" variant of bag-intersection.

Towards this goal, we fix some notation. A mapping μ is a set of pairs $(?\texttt{x}_\texttt{i}, v_i)$, each pair denoting $\mu(?\texttt{x}_\texttt{i}) = v_i$. A bag M of mappings is a collection of mappings

that may contain each mapping more than once. We write $card_M(\mu)$ to denote the number of times a mapping μ occurs in bag M (if clear, M may be dropped; if we do not specify $card_M(\mu)$, we assume 1 by default). In this submission, we will assume RDF graphs to be sets, while we assume query results to be bags of mappings, a setting sometimes referred to as set-bag semantics in the literature.

Having settled this, we first have to extend the notion of subsumption to bags. For sets L, R of mappings, subsumption $L \sqsubseteq R$ holds when for every mapping $\mu \in L$ there exists a mapping $\mu' \in R$ such that $\mu \subseteq \mu'$. Similar to homomorphisms (cf. [7]), there are several possibilities for extending subsumption to bags L, R of mappings: one could just apply the definition for sets, or one could demand that for every mapping $\mu \in L$ there exists $\mu' \in R$ such that $\mu \subseteq \mu'$ and $card_L(\mu) \leq card_R(\mu')$. While it would be interesting to study the effects of these definitions, for our purpose they are too weak. For example, they cannot resolve the situation described in Example 2: under both defintions, $Q(G_1)$ and $Q(G_2)$ would subsume each other. Thus a subsumption based definition of certain answers could not distinguish $Q(G_1)$ from $Q(G_2)$, despite our intention that $certain(Q, G_2) = Q(G_2)$, but not $Q(G_1)$.

We thus use a stricter definition for subsumption between bags, and say that a bag L is subsumed by a bag R, written $L \sqsubseteq_b R$, if there exists a mapping $h\colon L \to R$ such that $\mu \sqsubseteq h(\mu)$ for all $\mu \in L$ and $card_R(\mu') \geq \sum_{\mu \in L\colon h(\mu)=\mu'} card_L(\mu)$ for all $\mu' \in R$ (this corresponds to the additive homomorphisms in [7]).

Next, we say that a bag M of mappings is $\sqsubseteq_b$ maximal w.r.t. a property Ω if M satisfies Ω and there is no M' satisfying Ω such that $M \sqsubseteq M'$ but $M' \not\sqsubseteq M$.

It is an interesting observation at this point that sets $M_1 \neq M_2$ of mappings may satisfy $M_1 \sqsubseteq M_2$ and $M_2 \sqsubseteq M_1$ (just consider the mappings in Example 2), while (for bags or sets) $B_1 \sqsubseteq_b B_2$ and $B_2 \sqsubseteq_b B_1$ implies $B_1 = B_2$.

The notion of $\sqsubseteq_b$-maximal bags now allows us to define $L \cap_\sqsubseteq R$, a version of bag-intersection that retrieves maximal information from both, L and R: for two bags L, R of mappings, let $L \cap_\sqsubseteq R$ be a $\sqsubseteq_b$-maximal bag M such that $M \sqsubseteq_b L$ and $M \sqsubseteq_b R$. Unfortunately, the result of this operator is not necessarily unique.

Example 3. Let $L = \{\{(x,1),(y,1),(u,1)\},\{(v,1)\}\}$ and $R = \{\{(x,1),(y,1),(v,1)\}, \{(u,1)\}\}$ be two bags of mappings. Then $M_1 = \{\{(x,1),(y,1)\}\}$ and $M_2 = \{\{(u,1)\},\{(v,1)\}\}$ are both $\sqsubseteq_b$-maximal w.r.t. being subsumed by L and R.

However, we will discuss next that in many cases $\cap_\sqsubseteq$, or a slight adaption of it, is nevertheless well-suited to define meaningful certain answers.

3 Certain Answers via $\cap_\sqsubseteq$

Besides not returning a unique bag, another property of $\cap_\sqsubseteq$ needs to be taken care of before it can be used to define certain answers, as illustrated next.

Example 4. Consider a SPARQL query Q

`SELECT` ?`a`, ?`w`, ?`r` `WHERE` {?`a` `isa` `author`} `OPTIONAL` {?`a` `writes` ?`w`.?`a` `reads` ?`r`}

and an RDF graph $G = \{(a, \texttt{isa}, \texttt{author}), (a, \texttt{reads}, b)\}$. Assume that also the

knowledge "every author writes some book" (expressed e.g. in OWL) is given and that every RDF graph $G' \in \mathit{models}(G)$ satisfies this condition. Then $\bigcap_{\sqsubseteq}\{q(G') \mid G' \in \mathit{models}(G)\} = \{\{(?\mathtt{a}, a), (?\mathtt{r}, b)\}\}$.

However, this result does not respect the requirement expressed in the query that a result should contain either a value for both, $?w$ and $?r$, or neither of them. As a result, instead of defining certain answers just as $\bigcap_{\sqsubseteq}\{q(G') \mid G' \in \mathit{models}(G)\}$, following [1], we also restrict the possible domains for the certain answers. For a set $\mathcal{V}$ of sets of variables, we therefore extend $L \cap_{\sqsubseteq} R$ to $L \cap_{\sqsubseteq}^{\mathcal{V}} R$ as being the $\sqsubseteq_b$-maximal bag M such that $M \sqsubseteq_b L$, $M \sqsubseteq_b R$, and $\mathit{dom}(\mu) \in \mathcal{V}$ for all $\mu \in M$.

Finally, for a query Q, let the *admissible solution domains* $\mathit{adsoldom}(Q)$ be the set of possible domains of mappings in $Q(G)$ (for any G). Due to space restrictions we stick to this vague definition; but, for example, for conjunctive queries Q, $\mathit{adsoldom}(Q)$ contains as single element the set of all output variables of Q, for query Q from Example 2 we get $\mathit{adsoldom}(Q) = \{\{?\mathtt{auth}\}, \{?\mathtt{auth}, ?\mathtt{award}\}\}$, and for Q from Example 4, $\mathit{adsoldom}(Q) = \{\{?\mathtt{a}\}, \{?\mathtt{a}, ?\mathtt{w}, ?\mathtt{r}\}\}$.

Definition 1. *Let G be an RDF graph, Q a query, and $\mathit{models}(G')$ the set of graphs entailed by G. Then the* certain answers *of Q are defined as*

$$\mathit{certain}(Q, G) = \bigcap_{\sqsubseteq}^{\mathit{adsoldom}(Q)}\{Q(G') \mid G' \in \mathit{models}(G)\}.$$

While, for arbitrary inputs $L, R, \mathcal{V}$, the result of $L \cap_{\sqsubseteq}^{\mathcal{V}} R$ is not necessarily unique, in most of the settings in which we compute certain answers, L, R, and $\mathcal{V}$ are not arbitrary inputs but adhere to some structure that we can exploit.

For example, when applied to conjunctive queries, $\cap_{\sqsubseteq}^{\mathcal{V}}$ reduces to conventional (set- or bag) intersection, and as a result for these queries we get the "classical" definition of certain answers as a special case of Definition 1.

Similarly, in the special case of $\mathit{models}(G)$ containing a minimal element G (i.e. $G \subseteq G'$ for all $G' \in \mathit{models}(G)$), for weakly monotone SPARQL queries the certain answers are uniquely defined. In fact, applied to the settings in Examples 1 and 2 it produces exactly the intuitive bags of certain answers.

More generally, also for the sets $\mathit{models}(G)$ that exhibit a *canonical model* (i.e. that contain some $G' \in \mathit{models}(G)$ such that for every $G_i \in \mathit{models}(G)$ there exists some homomorphism $h_i \colon G' \to G_i$) we strongly conjecture that for weakly monotone queries, and especially for well-designed SPARQL queries, Definition 1 gives a unique bag of certain answers.

There are, of course, a lot of open question for future and ongoing work. These include the relationship to certain answer semantics from the literature (e.g., while "typical" certain answers for CQs are a special case of Definition 1, this seems not to be the case for the definition in [1]), an investigation of classes that provide a unique bag of certain answers, and of course the costs/complexity of query evaluation under this semantics for specific reasoning formalisms (e.g. OWL). Another line of research is to establish a connection with the work in [8]. Finally, also considering alterations of this definition could be of interest.

References

1. Ahmetaj, S., Fischl, W., Pichler, R., Simkus, M., Skritek, S.: Towards reconciling SPARQL and certain answers. In: Proc. WWW 2015. pp. 23–33. ACM (2015)
2. Angles, R., Gutiérrez, C.: The multiset semantics of SPARQL patterns. In: Proc. ISWC 2016. LNCS, vol. 9981, pp. 20–36. Springer (2016)
3. Arenas, M., Pérez, J.: Querying semantic web data with SPARQL. In: Proc. PODS 2011. pp. 305–316. ACM (2011)
4. Console, M., Guagliardo, P., Libkin, L.: Approximations and refinements of certain answers via many-valued logics. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016. pp. 349–358. AAAI Press (2016)
5. Console, M., Guagliardo, P., Libkin, L.: On querying incomplete information in databases under bag semantics. In: Proc. IJCAI 2017. pp. 993–999. ijcai.org (2017)
6. Hernández, D., Gutierrez, C., Hogan, A.: Certain answers for SPARQL with blank nodes. In: Proc. ISWC 2018. LNCS, vol. 11136, pp. 337–353. Springer (2018)
7. Hernich, A., Kolaitis, P.G.: Foundations of information integration under bag semantics. In: Proc. LICS 2017. pp. 1–12. IEEE Computer Society (2017)
8. Libkin, L.: Certain answers as objects and knowledge. vol. 232, pp. 1–19 (2016)
9. Nikolaou, C., Kostylev, E.V., Konstantinidis, G., Kaminski, M., Grau, B.C., Horrocks, I.: The bag semantics of ontology-based data access. In: Proc. IJCAI 2017. pp. 1224–1230. ijcai.org (2017)
10. Pérez, J., Arenas, M., Gutiérrez, C.: Semantics and complexity of SPARQL. ACM Trans. Database Syst. **34**(3), 16:1–16:45 (2009)

Anomaly Detection in Public Procurements using the Open Contracting Data Standard

Elisabeth Kehler[1] , Julio Paciello[2] and Juan Pane[3]

[1] Universidad Nacional de Asunción, Paraguay, maelikehler@gmail.com
[2] Universidad Nacional de Asunción, Paraguay, julio.paciello@pol.una.py
[3] Universidad Nacional de Asunción, Paraguay, jpane@pol.una.py

Abstract. Public procurement typically presents a potential source of corruption. For this reason, the detection of anomalies in public procurements can improve the quality of purchases, and consequently enable a better quality of life in the country through the correct use of public funds. In this paper, we use as a case study the public contracts of Paraguay, which are in the open data format of the Open Contracting Data Standard (OCDS), for training an unsupervised learning model for anomaly detection, based on the Isolation Forest algorithm. The resulting classification allows to obtain a measurement or scoring of contracts that can be used to identify outliers. Given a local dataset of cases of procurement processes with protests with judgments in favor of the protestant or with citizen complaints, the preliminary results show that the trained model classifies as anomalous more than 45% of the potentially anomalous dataset. A detailed validation considering the public procurements local regulations is needed, with the purpose of building a tool that allows an intelligent sampling of contracts with atypical behavior to review, applicable to Paraguay and other countries that implement the OCDS.

Keywords: Open Contracting Data Standard, Open data, Anomaly detection, Unsupervised learning, Artificial Intelligence.

1 Introduction

Transparency is an important tool to avoid corruption and is essential in a process of public procurements. The Open Contracting Data Standard (OCDS) [1] is a tool that helps to implement the transparency needed and provides the possibility of analyzing the data on a machine learning level. This work uses the publicly available data of the public procurements of Paraguay as a case study. The Public Procurements Office (DNCP) publishes since 2010 contracts in the OCDS open data format. The total number of contracts published by the DNCP since 2010 to 2019 amounts to 311,782.

Anomalies detection in public contracts is especially important in order to find, prevent and take actions of possibly misappropriated funds. These funds can then be redirected to areas with an important social impact, such as education or public health care. The regulatory analysis of the conformity of public procurement processes according to the local legislation is performed manually by a team of public officials of the DNCP, analyzing each procurement process separately. Given the volume of public contracts that are managed annually and the manual work of public officials to perform this task, it is possible to clearly notice that an exhaustive control of all

contracts is not feasible. In addition to the control carried out by the DNCP team, there are also journalistic publications on cases identified as possible frauds. However, journalistic investigations focus mostly on contracts that are potentially more striking for public opinion, which represent a small portion of the total.

Considering the public procurements as a potential source of corruption, and therefore a way for the misuse of the public funds, performing regulatory control to ensure that the processes are aligned with the local legislation represents an important task for a country. In Paraguay, to date, this analysis is done manually by specialized staff of the DNCP. They determine the classification of the data according to the local laws [8] and also considering known fraud schemes, as for example the Red Flags scheme [9]. So, the main problem is that having an ever growing amount of data, a proportional growing number of staff members to analyze the data is required. The use of the OCDS format makes it possible to apply Machine Learning techniques, as unsupervised learning, for anomaly detections of possible outliers to the expected behavior, serving the DNCP as an automated tool for implementing a smart sampling of procurement processes that can require an in-depth verification. In this work, this problem is addressed by proposing the automation of control tasks in a first instance, which allows the DNCP staff to obtain a smartly selected sample of relevant procurement processes for manual review.

The work is organized as follows, in section II the State of Art is mentioned, in section III we explain the proposed solution and in section IV we present the preliminary results and final discussion.

2 State of Art of anomaly detection techniques

Conti and Naldi in [4] present an statistical anomaly detection approach in procurement auctions using an average bid based method evaluating with the detection probability and the false alarm probability. Vaserhelyi and Issa illustrate K-Means Clustering applied to a labeled refund transactions dataset in [5]. Deng and Mei combine Self-Organizing Map (SOM) and K-Means Clustering for an unsupervised approach to detecting Fraudulent Financial Statements in [6]. Panigrahi, Kundu, Sural and Majumdar propose a fusion approach for credit card fraud detection using a rule-based filter, a Dempster-Shafer adder and a Bayesian Lerner in [7].

As we can appreciate there are multiple previous works that address the anomaly detection in certain stages of public procurements, and also in financial transactions. This work proposes to implement a tool that focuses on all stages of public procurement, during the call for bids, the award and contracts, and contract modifications specifically in the format of the OCDS. This approach also differs from the mentioned state of art by applying unsupervised isolation forests to determine the anomaly score of the data points.

Isolation Forest [2, 3] creates a forest of binary trees by randomly selecting a feature and also randomly selecting the split value at each node. The anomaly score is obtained by getting the length of the path to the data point in the isolation tree. It was

chosen for its independence to distance and density, especially because the data sets are high-dimensional, and for its good computational performance.

3 Proposed Solution

The semi-structured version of the data in the OCDS format allows for the implementation of algorithms to analyze it. The goal is to train a unsupervised learning model that separates anomalies from the data for their consequent human analysis to determine if a fraud is taking place.

As seen in Figure 1, first the data had to be cleaned, considering human errors when loading the data. Then a feature selection is done, selecting the variables based on three criteria: a) if the variable has data (is not empty), b) if the data is structured (E.g. no free text or links to text), and c) if the data is part of the local regulations for the procurement process. For an algorithm to be able to analyze the data efficiently, the data needs to be converted to numerical values, in this approach hashing and binary vectors are used to perform these tasks. The data is divided into 3 datasets, a) data in the planning and tender stages of the procurement process, b) contracts, and c) contracts modifications. Finally the data was normalized for use as an input to the algorithm.

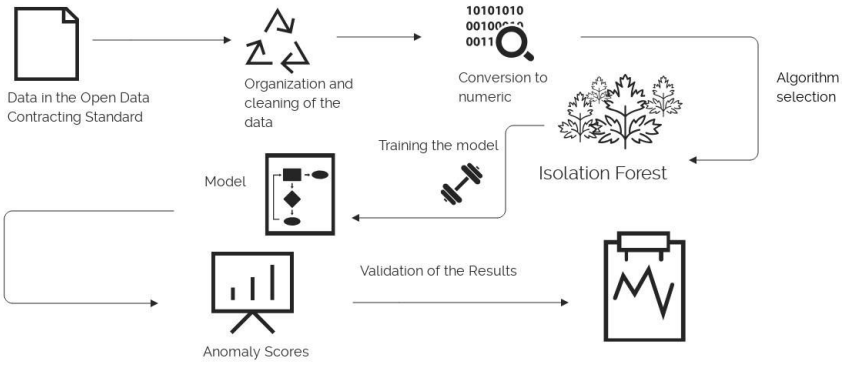


Fig. 1**.** Work flow of the proposed solution.

After training the model and getting the anomaly score for the input data, the DNCP provided to this work a dataset containing potentially anomalous procurement processes from 2010 onwards, in order to obtain a preliminary validation of the effectiveness of the classification model. The dataset includes potentially anomalous procurement processes with protests with judgments in favor of the protestant or with citizen complaints. Protests are internal disputes in the procurement process whereas complaints are external complaints with identity protection about the procurement

process. The scores of these cases obtained with the isolation forest implementation were then analysed to measure the accuracy of the trained models.

4 Preliminary Results and Final Discussion

The results consists of three trained models and the anomaly score for each of the data points. This anomaly score ranges between -1 and 1, where if it is less than 0 it is considered an anomaly and if it is more than 0 is considered as normal. The following Table 1 shows the total data points analysed per dataset, the total data points with protests with judgments in favor of the protestant and with complaints per dataset, the percentage of data points with protests and complaints detected as anomalous and the execution times. The computational platform used was an Intel Core i7, with 16 GB of RAM, running the iforest algorithm implementation of the Python scikit-learn library with a 1000 estimators and 50 samples. The implementation and input/output data can be found at https://gitlab.com/MaEliK/otherframeworks.

Table 1**.** Data points analysed and percentage of anomalous points detected

Data Set	Number of data Points	Protests in dataset	Complaints in dataset	Protests detected as anomalous	Complaints detected as anomalous	Execution Time
Planning and Tender	108470	599	297	48.75%	45.79%	9,119 s
Contracts	137783	1305	457	43.37%	42.45%	42,125 s
Contract Modifica-tions	29406	168	144	16.07%	31.94%	3,538 s

The obtained percentages show the proportion of data classified by the algorithm as anomalous data. It can be noticed that it detects almost half of the known potentially anomalous data, consistently in the planning and tender phases and the contracting phase. Finally, this work proposes an alternative to automate the regulatory control of procurement processes based on data analysis in OCDS format. An unsupervised learning based technique is proposed that could classify in seconds as anomalous more than 45% of the potentially anomalous dataset provided. Next steps are to validate the obtained results with the DNCP staff and check results according to local regulations. Also a better interpretation of the variables that influence high anomalous scores is required.

References

1. OCDS Homepage, http://standard.open-contracting.org/latest/en/, last accessed 2019/03/14.

2. Liu, Fei Tony, Ting, Kai Ming and Zhou, Zhi-Hua. “Isolation forest.” Data Mining, 2008. ICDM‘08. Eighth IEEE International Conference on.
3. Liu, Fei Tony, Ting, Kai Ming and Zhou, Zhi-Hua. “Isolation-based anomaly detection.” ACM Transactions on Knowledge Discovery from Data (TKDD) 6.1 (2012): 3.
4. Conti, Pier Luigi & Naldi, Maurizio. (2009). Detection of Anomalous Bids in Procurement Auctions. SSRN Electronic Journal. 10.2139/ssrn.1493346.
5. Issa, Hussein & Vasarhelyi, Miklos. (2011). Application of Anomaly Detection Techniques to Identify Fraudulent Refunds. SSRN Electronic Journal. 10.2139/ssrn.1910468.
6. Deng, Qingshan & Mei, Guoping. (2009). Combining Self-Organizing Map and K-Means Clustering for Detecting Fraudulent Financial Statements. 2009 IEEE International Conference on Granular Computing, GRC 2009. 126-131. 10.1109/GRC.2009.5255148.
7. Panigrahi, Suvasini & Kundu, Amlan & Sural, Shamik & Majumdar, Arun. (2009). Credit card fraud detection: A fusion approach using Dempster–Shafer theory and Bayesian learning. Information Fusion. 10. 354-363. 10.1016/j.inffus.2008.04.001.
8. Ley 2051/03 “De Contrataciones Públicas”, https://www.contrataciones.gov.py/documentos/download/marco-legal/12760, last accessed 2019/03/14.
9. Development Gateway, Open Contracting Partnership, “Red Flags for integrity: Giving the green light to open data solutions,” in press.

Map-Elites Algorithm for Features Selection Problem

Brenda Quiñonez[a], Diego P. Pinto-Roa [a], Miguel García-Torres[b], María E. García-Diaz [a], Carlos Núñez-Castillo[a] and Federico Divina [b]

[a]Facultad Politécnica - Universidad Nacional de Asunción [1]

[b]Division of Computer Science - Universidad Pablo de Olavide [2]

Abstract

In the High-dimensional data analysis there are several challenges in the fields of machine learning and data mining. Typically, feature selection is considered as a combinatorial optimization problem which seeks to remove irrelevant and redundant data by reducing computation time and improve learning measures. Given the complexity of this problem, we propose a novel Map-Elites based Algorithm that determines the minimum set of features maximizing learning accuracy simultaneously. Experimental results, on several data based from real scenarios, show the effectiveness of the proposed algorithm.

Keywords: Feature Selection, Map-Elites, Combinatorial Optimization, Machine Learning, Data Mining

1 Introduction

Recently, the available data has increased explosively in both the number of samples and the dimensionality in different machine learning applications, such as text mining, artificial vision and bio-medical. Our interest is mainly focused on the high dimensionality of the data. The large amount of high-dimensional data has imposed a great challenge on existing machine learning methods. The presence of noisy, redundant and irrelevant dimensions can make the learning algorithms very slow and can also generate difficulties when interpreting the resulting models [3]. In machine learning and statistics, the feature selection is the process of selecting a subset of relevant characteristics to use in building the model. Attribute selection methods greatly influence the success of data mining processes by reducing computational time and improving learning metrics, for this reason we propose a new attribute technique selection based on Illumination Algorithm [1].

This paper is organized as follows. Section 2 introduces to the features selection problem. Then, we describe the Illumination search algorithms in Section 3, specifically the Map-Elites algorithm. The section 4 contains the proposal of this paper and, finally, in the last section there is a brief discussion of the results obtained so far.

[1]{bquinonez,dpinto,mgarcia,nunez}@pol.una.py

[2]{mgarciat,fdivina}@upo.es

2 The Feature Selection Problem

A feature selection algorithm basically is the combination of a search technique to propose new subsets of features, with an evaluation measure that qualifies the different subsets. The simplest algorithm is to test every possible subset of features to find the one that minimizes the error rate. This is an exhaustive search of space, and it is computationally intractable except for the smallest feature sets; i.e. for n attributes, there are 2^n solutions. The choice of the evaluation metric has a great influence on the algorithm, and they can distinguish among three main categories: wrap methods, filter methods and embedded methods [3]. Wrappers use a search algorithm to search through the space of possible features and evaluate each subset by running a model on the subset. Wrappers can be computationally expensive and have a risk of over fitting to the model. Filters are similar to Wrappers in the search approach, but instead of evaluating against a model, a simpler filter is evaluated. Embedded techniques are embedded in and specific to a model [3].

Many popular search approaches use greedy hill climbing, which iteratively evaluates a candidate subset of features, then modifies the subset and evaluates if the new subset is an improvement over the old. Evaluation of the subsets requires a scoring metric that grades a subset of features.

Search approaches applied to the feature selection include: exhaustive, best first, simulated annealing, genetic algorithm, greedy forward selection, greedy backward elimination, particle swarm optimization, targeted projection pursuit, Scatter Search, Variable Neighborhood Search [2, 4].

Genetic algorithm (GA) [5] method due to the capability to evolve new features of the selected features and a vast exploration of the search space for new fitter solutions. GA includes a subset of the growth-based optimization methods aiming at the use of the GA operators such as selection, mutation and recombination to a population of challenging problem solutions. GA has been effectively applied to several optimization problems such as classification tasks and pattern recognition. The GA's stochastic component does not rule out excitedly dissimilar solutions, which may give the better result. This has the advantage that, given sufficient time and a well bOlli1ded problem, the algorithm can discover a global optimum. It is well suited to feature selection problems because of the above reason. In the next section it will describe the MAP Elites algorithm based on genetic algorithms.

3 MAP Elites

The Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) [1] algorithm illuminates search spaces, which produces a large diversity of high-performing, yet qualitatively different solutions, which can be more helpful than a single, high-performing solution. Interestingly, because MAP-Elites explores more of the search space, it also tends to find a better overall solution than state-of-the-art search algorithms. This is because MAP-Elites illuminates the relationship

between performance and dimensions of interest in solutions. MATP-Elites returns a set of high-performing and improves the state-of-the-art for finding a single, best solution, it will catalyze advances throughout all science and engineering fields.

MAP-Elites is quite simple. First, the user chooses a performance measure f(x) that evaluates a solution x. Second, the user chooses N dimensions of variation of interest that define a feature space of interest to the user. Each dimension of variation is discretized based on user preference or available computational resources. Given a particular discretization, MAP-Elites will search for the highest performing solution for each cell in the N-dimensional feature space. The search is conducted in the search space, which is the space of all possible values of x, where x is a description of a candidate solution [1].

4 MAP-Elites for Feature Selection

In this paper we propose to use the Map-Elites algorithm as an innovative technique for features selection in the automatic learning process. The challenge of this problem is that the inputs variable are binary one, whereas the basic Map-Elites was designed for real numbers. Therefore, the main objective of this proposal will be to create a search space for a MAP-Elites for the binary variables given the feature selection is a combinatorial problem.

To face this challenge, we represent a set of solutions as a vector with the indexes of the selected features. Algorithm 1 shows the pseudo-code of the Combinatorial MAP-Elites proposed. To create the map that allows us to distribute the solutions in the search space, we define the number of cells as the algorithm input parameter *NC*. Subsequently, this parameter is used to calculate a number of fixed features per cell *NFF*, which are used as cell identifiers and help determine which cell of the map each solution will be associated with (1). We also use two more input parameters for the algorithm that are typical of genetic algorithms such as the number of iterations I and the number of initial genomes G.

Algorithm 1 Combinatorial Map-Elites algorithm for feature selection

```
1:  procedure MAP-ELITES(NC, I, G)
2:      NFF ← log2(NC)
3:      MAP ← loadCell(NC, NFF)                    ▷ (1)
4:      for iter = 1 → I do
5:          if iter < G then
6:              MAP ← randomSolution()             ▷ (2)
7:          else
8:              MAP ← randomVariation()            ▷ (3)
9:          end if
10:     end for
11:     return MAP                                 ▷ feature-fitness map
12: end procedure
```

MAP-Elites starts by randomly generating G genomes (solutions coded) and determining the performance and features of each (2). In a random order, those genomes are placed into the cells to which they belong in the feature space (if multiple genomes map to the same cell, the highest-performing one per cell is retained). At that point the algorithm is initialized, and the following steps are repeated until a termination criterion is reached I. A cell in the map is randomly chosen and the genome in that cell produces an offspring via mutation and/or crossover (3). The features and performance of that offspring are determined, and the offspring is placed in the cell if the cell is empty or if the offspring is higher-performing than the current occupant of the cell, in which case that occupant is replaced by the new solution. The algorithm returns a map with the best solution found for each cell along with the corresponding fitness.

5 Discussion

The proposed Map-Elites tries to determine the minimum set of features that maximizes learning accuracy. This preliminary experiment was conducted using different data sets from real scenarios obtained from [2]. Table 1 presents performance of the algorithm where solutions were evaluated using Bayes Classifier and 2-fold cross validation [2]. In this experiment the input parameters for the algorithm were 5,000 iterations (I), a map of 8 cells (NC) and the number of initial genomes (G) equal to 500. In addition, we able to see the accuracy obtained by the proposed algorithm is promissory and at the same time it reduces the number of features.

Table 1: Map-Elites result experiment with Bayes Classifier

Dataset	All features	Fitness	Selected features
ionosphere	34	92.02 ± 2.38	12.6 ± .89
glass	9	70.12 ± 4.27	6.0 ± 1.00
anneal	38	96.44 ± 1.60	7.6 ± 1.34
tokyo1	44	92.91 ± 1.08	10.6 ± 2.51
spambase	57	91.76 ± .60	10.6 ± .89
kr-vs-kp	36	90.43 ± 1.46	3.0 ± .00
corral	6	86.90 ± 2.22	5.0 ± .00
breast-cancer	9	71.34 ± 3.95	3.6 ± .89
hypothyroid	29	96.66 ± .29	1.0 ± .00
labor	16	91.21 ± 11.14	5.0 ± .71
vote	16	95.63 ± 1.26	1.0 ± .00

Currently, the performance of Map-Elites is being tested and compared with the competitive algorithms of the-state-of-the-art.

References

[1] Jean-Baptiste Mouret and Jeff Clune. “Illuminating search spaces by mapping elites”. CoRR. 2015.vol. abs/1504.04909

[2] Félix García López, Miguel García-Torres, Belén Melián Batista, José A. Moreno Pérez, and J. Marcos Moreno-Vegatitle. “Solving feature subset selection problem by a Parallel Scatter Search”. European Journal of Operational Research. 2006.

[3] Miao, Jianyu Niu, Lingfeng. (2016). A Survey on Feature Selection. Procedia Computer Science. 91. 919-926. 10.1016/j.procs.2016.07.111.

[4] M. Garcia-Torres, F. Gomez-Vela, B. Melian, J.M. Moreno-Vega. High-dimensional feature selection via feature grouping: A Variable Neighborhood Search approach, Information Sciences, vol. 326, pp. 102-118, 2016.

[5] Sindhiya, S Selvaraj, Gunasundari. (2015). A survey on genetic algorithm based feature selection for disease diagnosis system. Proceedings of ICCCS 2014 - IEEE International Conference on Computer Communication and Systems. 164-169. 10.1109/ICCCS.2014.7068187.

Bringing Order to Data

Heidar Davoudi[1], Parke Godfrey[2], Lukasz Golab[1], Mehdi Kargar[3],
Divesh Srivastava[4], and Jaroslaw Szlichta[5]

[1]University of Waterloo, Canada, [2] York University, Canada,
[3] Ted Rogers School of Management, Ryerson University, Canada,
[4] AT&T Labs-Research, USA, [5] University of Ontario Institute of Technology, Canada,
hdavoudi@uwaterloo.ca, godfrey@yorku.ca, lgolab@uwaterloo.ca,
kargar@ryerson.ca, divesh@research.att.com,szlichta@uoit.ca

Abstract. Integrity constraints (ICs) are widely used in business intelligence to express and enforce application semantics. However, finding ICs manually is time consuming, requires the involvement of domain experts, and is prone to human error. Thus, techniques have been proposed to automatically find a variety of ICs. We propose an algorithm to automatically discover order dependencies (ODs). Prior work on OD discovery has factorial complexity, is not complete, and is not concise. We propose an algorithm that finds a complete set of ODs with exponential worst-case time complexity in the number of attributes and linear complexity in the number of tuples. We experimentally show that our algorithm is orders of magnitude faster than the prior state-of-the-art.

Keywords: Integrity constraints · Order Dependency · Axiomatization.

1 Introduction

Ordered attributes, such as timestamps and numbers, are common in business data. Order Dependencies (ODs) capture monotonic relationships among such attributes. For instance, Table 1 shows employee tax records in which `tax` is calculated as a `percentage` (`perc`) of `salary` (`sal`). The OD `sal` *orders* `perc` holds: if we sort the table by salary, it is also sorted by percentage. Similarly, `sal orders grp, subg`: if we sort the table by salary, it is also sorted by group with ties broken by subgroup. With interest in data analytics at an all-time high, ODs can improve the consistency dimension of data quality and query optimization [4, 7–9]. ODs can describe business rules (data profiling); and their violations can point out possible data errors. Furthermore, query optimizers can use ODs to eliminate costly operators such as joins and sorts: ordered streams between query operators can exploit available indexes, enable pipelining, and eliminate intermediate partitioning steps. Finally, ODs subsume Function Dependencies (FDs) as any FD can be *mapped* to an equivalent OD by prefixing the left-hand-side attributes onto the right-hand-side [6].

It is time consuming to specify integrity constraints manually, motivating the need for their automatic discovery. However, since ODs are naturally expressed with lists rather than sets of attributes (as in the example above), existing solutions have *factorial* worst-case time complexity in the number of attributes [4]. We describe a more efficient algorithm to discover ODs from data. First, we show that ODs can be expressed with sets of attributes via a *polynomial mapping* into an *equivalent* set-based canonical form. Then, we introduce *sound* and *complete* axioms for set-based canonical ODs,

#	ID	yr	pos	bin	sal	perc	tax	grp	subg
t1	1	16	sec	1	5K	20%	1K	A	III
t2	2	16	dev	2	8K	25%	2K	C	II
t3	3	16	dir	3	10K	30%	3K	D	I
t4	1	15	sec	1	4K	20%	0.8K	A	III
t5	2	15	dev	2	6K	25%	1.5K	C	I
t6	3	15	dir	3	8K	25%	2K	C	II

Table 1: Employee salary data

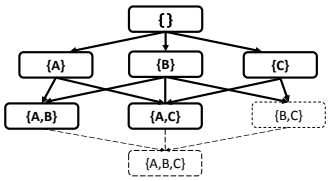


Fig. 1: A set lattice for attributes A, B, C.

which lead to optimizations of the OD discovery algorithm by avoiding redundant computation. This allows us to design a *fast* and *effective* OD discovery algorithm that has *exponential* worst-case complexity, $O(2^{|\mathbf{R}|})$, in the number of attributes $|\mathbf{R}|$, and linear complexity in the number of tuples. We note that this short paper is a summary of our published results in [5, 6, 10].

2 Order Dependency Discovery

2.1 Preliminaries

Order dependencies (ODs) describe relationships among lexicographical orderings of sets of tuples, as in the SQL *order by* statement. Let $\mathbf{X} = [\mathsf{A} \mid \mathbf{T}]$ be a list of attributes, where the attribute A is the *head* of the list, and the list $\mathbf{T}$ is the tail. For two tuples s and t, we write $s \preceq_{\mathbf{X}} t$ *iff* $s_{\mathsf{A}} < t_{\mathsf{A}}$ *or* ($s_{\mathsf{A}} = t_{\mathsf{A}}$ and ($\mathbf{T} = [\,]$ or $s \preceq_{\mathbf{T}} t$)). Given two lists of attributes, $\mathbf{X}$ and $\mathbf{Y}$, $\mathbf{X} \mapsto \mathbf{Y}$ denotes an OD, read as $\mathbf{X}$ *orders* $\mathbf{Y}$. Table $\mathbf{r}$ *satisfies* $\mathbf{X} \mapsto \mathbf{Y}$ *iff*, for all $s, t \in \mathbf{r}$, $s \preceq_{\mathbf{X}} t$ implies $s \preceq_{\mathbf{Y}} t$. Moreover, $\mathbf{X}$ and $\mathbf{Y}$ are *order compatible*, denoted as $\mathbf{X} \sim \mathbf{Y}$ *iff* $\mathbf{XY} \leftrightarrow \mathbf{YX}$. (For example, `month` and `week` of the year in the calendar are order compatible.)

We say that two tuples, s and t, are *equivalent* with respect to an attribute set $\mathcal{X}$ if $s_{\mathcal{X}} = t_{\mathcal{X}}$. Any attribute set $\mathcal{X}$ partitions tuples into *equivalence classes* [3]. We denote the *equivalence class* of a tuple $t \in \mathbf{r}$ with respect to a given set $\mathcal{X}$ by $\mathcal{E}(t_{\mathcal{X}})$; i.e., $\mathcal{E}(t_{\mathcal{X}}) = \{s \in \mathbf{r} \mid s_{\mathcal{X}} = t_{\mathcal{X}}\}$. A *partition* of $\mathbf{r}$ over $\mathcal{X}$ is the set of equivalence classes, $\Pi_{\mathcal{X}} = \{\mathcal{E}(t_{\mathcal{X}}) \mid t \in \mathbf{r}\}$. For instance, in Table 1, $\mathcal{E}(t1_{\{\mathsf{year}\}}) = \mathcal{E}(t2_{\{\mathsf{year}\}}) = \mathcal{E}(t3_{\{\mathsf{year}\}}) = \{t1, t2, t3\}$ and $\Pi_{\mathsf{year}} = \{\{t1, t2, t3\}, \{t4, t5, t6\}\}$.

2.2 Canonical Mapping and Axioms

Expressing ODs in a natural way relies on lists of attributes; e.g., in Table 1, `sal` $\mapsto$ `grp, subg` is not the same as `sal` $\mapsto$ `subg, grp`. In contrast, the order of attributes in an FD does not matter. However, the list representation leads to high complexity when discovering ODs [4]. Thus, we provide a polynomial *mapping* of list-based ODs into *equivalent* set-based canonical ODs [5, 6, 10]. The mapping allows us to develop an efficient OD discovery algorithm that traverses a much smaller set-containment lattice rather than the list-containment lattice used in [4].

The mapping presented in Theorem 1 (below) converts a list-based OD into *canonical* set-based ODs of two types. First, an attribute A is a *constant* within each equivalence class with respect to a set of attributes $\mathcal{X}$, denoted as $\mathcal{X}$: $[\,] \mapsto \mathsf{A}$, if $\mathbf{X}' \mapsto \mathbf{X}'\mathsf{A}$ for any permutation $\mathbf{X}'$ of $\mathcal{X}$ (note that $\mathcal{X}$ functionally determines $\mathcal{Y}$ *iff* $\mathbf{X} \mapsto \mathbf{XY}$, for any list $\mathbf{X}$ over the attributes of $\mathcal{X}$ and any list $\mathbf{Y}$ over the attributes of $\mathcal{Y}$ [6]). Second, two attributes, A and B, are order-compatible within each equivalence class with respect to the set of attributes $\mathcal{X}$, denoted as $\mathcal{X}$: $\mathsf{A} \sim \mathsf{B}$, if $\mathbf{X}'\mathsf{A} \sim \mathbf{X}'\mathsf{B}$. The set $\mathcal{X}$ is called a *context*. For example, in Table 1, `bin` is a constant in the context of `pos`, written as $\{\mathtt{pos}\}$:

1. Reflexivity
$\mathcal{X}$: [] $\mapsto$ A, $\forall$A $\in \mathcal{X}$
2. Identity
$\mathcal{X}$: A $\sim$ A
3. Commutativity
$\dfrac{\mathcal{X}\text{: A} \sim \text{B}}{\mathcal{X}\text{: B} \sim \text{A}}$
4. Strengthen
$\dfrac{\mathcal{X}\text{:[]} \mapsto \text{A} \quad \mathcal{X}\text{A:[]} \mapsto \text{B}}{\mathcal{X}\text{:[]} \mapsto \text{B}}$
5. Propagate
$\dfrac{\mathcal{X}\text{:[]} \mapsto \text{A}}{\mathcal{X}\text{: A} \sim \text{B}}$
6. Augmentation-I
$\dfrac{\mathcal{X}\text{:[]} \mapsto \text{A}}{\mathcal{Z}\mathcal{X}\text{:[]} \mapsto \text{A}}$
7. Augmentation-II
$\dfrac{\mathcal{X}\text{: A} \sim \text{B}}{\mathcal{Z}\mathcal{X}\text{: A} \sim \text{B}}$
8. Chain
$\dfrac{\mathcal{X}\text{: A} \sim \text{B}_1 \quad \forall_{i\in[1,n-1]}, \mathcal{X}\text{: B}_i \sim \text{B}_{i+1} \quad \mathcal{X}\text{: B}_n \sim \text{C} \quad \forall_{i\in[1,n]}, \mathcal{X}\text{B}_i : \text{A} \sim \text{C}}{\mathcal{X}\text{: A} \sim \text{C}}$

Fig. 2: Set-based axiomatization for canonical ODs.

[] $\mapsto$ bin since $\mathcal{E}(t1_{\{\text{pos}\}}) \models [] \mapsto \text{bin}$, $\mathcal{E}(t2_{\{\text{pos}\}}) \models [] \mapsto \text{bin}$ and $\mathcal{E}(t3_{\{\text{pos}\}}) \models [] \mapsto \text{bin}$. Also, {year}: bin $\sim$ salary because $\mathcal{E}(t1_{\{\text{year}\}}) \models \text{bin} \sim \text{salary}$ and $\mathcal{E}(t4_{\{\text{year}\}}) \models \text{bin} \sim \text{salary}$.

Theorem 1. *$\boldsymbol{X} \mapsto \boldsymbol{Y}$ iff $\forall j$, $\mathcal{X}$:[] $\mapsto Y_j$ and $\forall i, j$, $\{X_1, .., X_{i-1}, Y_1, .., Y_{j-1}\}: X_i \sim Y_j$.*

Example 1. The OD [AB] $\mapsto$ [CD] is mapped into the following set of canonical ODs: {A, B}:[] $\mapsto$ C, {A, B}:[] $\mapsto$ D, {}: A $\sim$ C, {A}: B $\sim$ C, {C}: A $\sim$ D, {A, C}: B $\sim$ D.

We present a *sound* and *complete set-based axiomatization* for ODs in Fig. 2 [6]. The set-based axioms allow us to design effective pruning rules for our OD discovery algorithm. For example, OD $\mathcal{X}$:[] $\mapsto$ A is *trivial* if A $\in \mathcal{X}$ by Reflexivity (see also *Example 2*).

Theorem 2. *The axiomatization for canonical ODs in Fig. 2 is sound and complete.*

2.3 Discovery Algorithm

Given the mapping of a list-based OD into equivalent set-based ODs, we present an algorithm, named *FASTOD*, that *efficiently* discovers a complete and minimal set of set-based ODs over a given relation instance. In contrast, the OD discovery algorithm from [4] traverses a lattice of all possible *lists* of attributes, which leads to factorial time complexity. *FASTOD* starts the search from singleton sets of attributes and works its way to larger attribute sets through a set-containment lattice (as in Figure 1), level by level ($l = 0, 1, \ldots$). When the algorithm is processing an attribute set $\mathcal{X}$, it verifies ODs of the form $\mathcal{X} \setminus$ A:[] $\mapsto$ A (let $\mathcal{X} \setminus$ A be shorthand for $\mathcal{X} \setminus \{\text{A}\}$, where A $\in \mathcal{X}$) and $\mathcal{X} \setminus \{\text{A}, \text{B}\}$: A $\sim$ B, where A, B $\in \mathcal{X}$ and A $\neq$ B. Furthermore, an OD $\mathcal{X} \setminus$ A:[] $\mapsto$ A should be minimal, that is, $\nexists\, \mathcal{Y} \subset \mathcal{X}$ such that $\mathcal{Y} \setminus$ A:[] $\mapsto$ A is valid.

The algorithm maintains information about minimal ODs, in the form of $\mathcal{X} \setminus$A:[] $\mapsto$ A, in the *candidate* set $\mathcal{C}_c^+(\mathcal{X})$ [3] (as ODs subsume FDs [7, 9]), where $\mathcal{C}_c^+(\mathcal{X}) = \{\text{A} \in \mathbf{R} \mid \forall_{\text{B}\in\mathcal{X}}\ \mathcal{X} \setminus \{\text{A}, \text{B}\}\text{:[]} \mapsto \text{B}$ does not hold$\}$. Similarly, it stores information about minimal ODs in the form of $\mathcal{X} \setminus \{\text{A}, \text{B}\}$: A $\sim$ B, in the *candidate* set $\mathcal{C}_s^+(\mathcal{X})$, where $\mathcal{C}_s^+(\mathcal{X}) = \{\{\text{A}, \text{B}\} \in \mathcal{X}^2 \mid \text{A} \neq \text{B}$ and $\forall_{\text{C}\in\mathcal{X}}\ \mathcal{X} \setminus \{\text{A}, \text{B}, \text{C}\}$: A $\sim$ B does not hold, and $\forall_{\text{C}\in\mathcal{X}}\ \mathcal{X} \setminus \{\text{A}, \text{B}, \text{C}\}$:[] $\mapsto$ C does not hold$\}$. The following lemmas can be used to prune the search space:

Lemma 1. *An OD $\mathcal{X} \setminus$ A:[] $\mapsto$ A, where A $\in \mathcal{X}$, is minimal iff $\forall_{B\in\mathcal{X}}$ A $\in \mathcal{C}_c^+(\mathcal{X} \setminus B)$.*

Lemma 2. *An OD $\mathcal{X} \setminus \{A, B\}$: A $\sim$ B, where A, B $\in \mathcal{X}$ and A $\neq$ B, is minimal iff $\forall_{C\in\mathcal{X}\setminus\{A,B\}}$ $\{A, B\} \in \mathcal{C}_s^+(\mathcal{X} \setminus C)$, and A $\in \mathcal{C}_c^+(\mathcal{X} \setminus B)$ and B $\in \mathcal{C}_c^+(\mathcal{X} \setminus A)$.*

According to Lemma 1, we do not need to check $\mathcal{X} \setminus \mathsf{A}{:}[\,] \mapsto \mathsf{A}$ if $\mathsf{A} \notin \mathcal{X} \bigcap_{\mathsf{B} \in \mathcal{X}} \mathcal{C}_c^+(\mathcal{X} \setminus \mathsf{B})$, and based on Lemma 2, we do not need to consider $\mathcal{X} \setminus \{\mathsf{A}, \mathsf{B}\}{:}\ \mathsf{A} \sim \mathsf{B}$ if $\mathsf{A} \notin \mathcal{C}_c^+(\mathcal{X} \setminus \mathsf{B})$ or $\mathsf{B} \notin \mathcal{C}_c^+(\mathcal{X} \setminus \mathsf{A})$. Moreover, according to Lemma 3, we can delete nodes from the lattice under the following conditions:

Lemma 3. *Deleting node $\mathcal{X}$ from the l^{th} lattice level, where $l \geq 2$, has no effect on the output set of minimal ODs if $\mathcal{C}_c^+(\mathcal{X}) = \{\}$ and $\mathcal{C}_s^+(\mathcal{X}) = \{\}$.*

Example 2. Let $\mathsf{A}{:}[\,] \mapsto \mathsf{B}$, $\mathsf{B}{:}[\,] \mapsto \mathsf{A}$ and $\{\}{:}\ \mathsf{A} \sim \mathsf{B}$. Since $\mathcal{C}_c^+(\{\mathsf{A}, \mathsf{B}\}) = \{\}$ and $\mathcal{C}_s^+(\{\mathsf{A}, \mathsf{B}\})$ as well as $l = 2$, by the pruning levels rule (Lemma 3), the node $\{\mathsf{A}, \mathsf{B}\}$ is deleted and the node $\{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ is not considered *(*see Figure 1*)*. This is justified as $\{\mathsf{AB}\}{:}[\,] \mapsto \mathsf{C}$ is not minimal by the Strengthen axiom, $\{\mathsf{AC}\}{:}[\,] \mapsto \mathsf{B}$ is not minimal by Augmentation–I, $\{\mathsf{BC}\}{:}[\,] \mapsto \mathsf{A}$ is not minimal by Augmentation–I, $\{\mathsf{C}\}{:}\ \mathsf{A} \sim \mathsf{B}$ is not minimal by Augmentation–II, $\{\mathsf{A}\}{:}\ \mathsf{B} \sim \mathsf{C}$ is not minimal by Propagate, and $\{\mathsf{B}\}{:}\ \mathsf{A} \sim \mathsf{C}$ is not minimal by Propagate.

Note that while we provide theoretical guarantees for *FASTOD* to find a complete set of ODs, the *ORDER* algorithm [4] is not complete.

Theorem 3. *The FASTOD algorithm computes a complete and minimal set of ODs.*

In [10], we extend our algorithm to discover *bidirectional* ODs, which allow a mix of ascending and descending (desc) orders. For example, a student with an alphabetically lower letter grade has a higher percentage grade than another student. We develop additional pruning rules and show that efficiency of discovery of bidirectional ODs remains the same as for one-directional ODs.

We experimentally compare *FASTOD* with previous approaches (*ORDER* [4]). Our algorithm can be orders of magnitude faster. For instance, on the `flight` dataset from `http://metanome.de` with 1K tuples and 20 attributes, *FASTOD* finishes the computation in less than 1 second, whereas ORDER did not terminate after 5 hours. Moreover, *FASTOD*'s candidate sets do not increase in size during the execution of the algorithm (unlike *ORDER*) because of the concise candidate representation (e.g., many ODs that are considered minimal by ORDER are found to be redundant by our algorithm).

Finally, in [2], we show that a recent approach to OD discovery called *OCDDISCOVER* in Consonni et al. [1] is incorrect. We show that their claim of completeness of OD discovery is *not* true. Built upon their incorrect claim, *OCDDISCOVER*'s pruning rules are overly aggressive, and prune parts of the search space that contain legitimate ODs. This is the reason their approach appears to be "faster" in practice than our *FASTOD* discovery algorithm despite being significantly worse in asymptotic complexity. Finally, we show that Consonni et al. [1] misinterpret our set-based canonical form for ODs, leading to an incorrect claim that our FASTOD implementation has an error.

3 Conclusions

We presented an efficient algorithm for discovering ODs. The technical innovation that made our algorithm possible is a novel mapping into a set-based canonical form and an axiomatization for set-based canonical ODs. In future work, we plan to study *conditional* ODs that hold over portions of data.

References

1. Consonni, C., Sottovia, P., Montresor, A., Velegrakis, Y.: Discovering Order Dependencies through Order Compatibility. In: EDBT. pp. 409–420 (2019)
2. Godfrey, P., Golab, L., Kargar, M., Srivastava, D., Szlichta, J.: Errata note: Discovering order dependencies through order compatibility. *Technical Report, 5 pages, available at* https://arxiv.org/abs/1905.02010 (2019)
3. Huhtala, Y., Karkkainen, J., Porkka, P., Toivonen, H.: Efficient Discovery of Functional and Approximate Dependencies Using Partitions. In: ICDE. pp. 392–401 (1998)
4. Langer, P., Naumann, F.: Efficient order dependency detection. VLDB J. **25**(2), 223–241 (2016)
5. Mihaylov, A., Godfrey, P., Golab, L., Kargar, M., Srivastava, D., Szlichta, J.: FASTOD: bringing order to data. In: 2018 IEEE 34th International Conference on Data Engineering (ICDE). pp. 1561–1564. IEEE (2018)
6. Szlichta, J., Godfrey, P., Golab, L., Kargar, M., Srivastava, D.: Effective and Complete Discovery of Order Dependencies via Set-based Axiomatization. PVLDB **10**(7), 721–732 (2017)
7. Szlichta, J., Godfrey, P., Gryz, J.: Fundamentals of Order Dependencies. PVLDB, 5(11): 1220-1231 (2012)
8. Szlichta, J., Godfrey, P., Gryz, J., Ma, W., Qiu, W., Zuzarte, C.: Business-Intelligence Queries with Order Dependencies in DB2. In: EDBT, 750-761 (2014)
9. Szlichta, J., Godfrey, P., Gryz, J., Zuzarte, C.: Expressiveness and Complexity of Order Dependencies. PVLDB 6(14): 1858-1869 (2013)
10. Szlichta, J., Godfrey, P., Golab, L., Kargar, M., Srivastava, D.: Effective and complete discovery of bidirectional order dependencies via set-based axioms. The VLDB Journal **27**(4), 573–591 (2018)

A Non-Uniform Tuning Method for SQL-on-Hadoop Systems

Edson Ramiro Lucas Filho, Renato Silva de Melo, and Eduardo Cunha de Almeida

Universidade Federal do Paraná, Brazil
{erlfilho,rsmelo,eduardo}@inf.ufpr.br

Abstract. A SQL-on-Hadoop query consists of a workflow of MapReduce jobs with a single point of configuration. This means that the developer tunes hundreds of tuning parameters directly in the query source code (or via terminal interface), but the system assumes the same configuration to every running job. The lurking problem is that the system allocates computing resources uniformly to every running job, even if they present different consumption needs (after all the tuning setup is the same). In this paper, we demonstrate that such uniform allocation of resources drives the query to inefficient performance. We claim that applying a non-uniform allocation method to define a customized tuning setup for each job can outperform the uniform allocation. Ultimately, we demonstrate that searching for specific tuning setup is an optimization problem with polynomial cost.

Keywords: Tuning · Self-Tuning · SQL-on-Hadoop · MapReduce.

1 Introduction

Ever since the proliferation of SQL-on-Hadoop engines [2,7,3,23,8,14,22,1] the number of configuration parameters exposed by such systems, and by the underneath MapReduce systems [9,20], has grown considerably, as presented in Figure 1. For instance, the SQL-on-Hadoop system Apache Hive has increased from 96 configuration parameters in its 0.6.0 release up to 989 parameters in the release 3.0.0. The underlying processing engine Apache Hadoop has increased from 104 configuration parameters in its 0.12.0 release up to 530 parameters in the release 3.1.0. The impact on performance of this growing number of tuning parameters has given rise to a successful line of research on tuning MapReduce systems [17,16,21,18,4,13,15,10,5,24,26,11,12,19].

However, tuning SQL-on-Hadoop queries is more complex than tuning MapReduce jobs, because queries span a workflow of jobs with a single point of configuration (i.e., the tuning setup is set in the query source code or via command line). In practice, developers tune the physical resources to be allocated by the query and the query processing engine propagates such set of physical resources to all jobs in the query plan with potential inefficient performance (see Section 4).

In traditional SQL processing, a query plan indicates the execution flow for the query operators. SQL-on-Hadoop processing is no different, but in the dis-

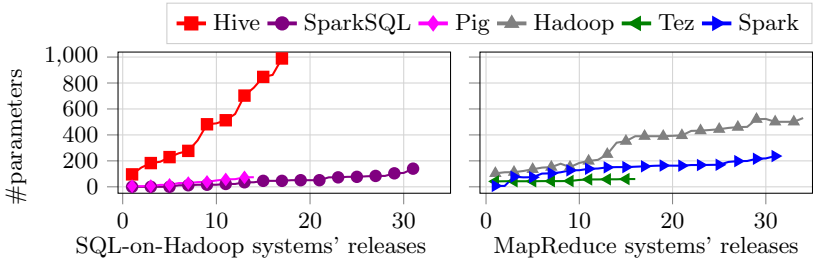


Fig. 1: Number of configuration parameters per release.

tributed environment of MapReduce systems every job of a query plan implements a different set of query operators with different resource consumption needs [6,22]. For instance, while one job requires disk bandwidth due to a *TableScan* operator, another job requires memory throughput due to a *Sort* operator. The lurking problem of tuning SQL-on-Hadoop with a single point of configuration: the propagation of the same set of physical resources to jobs with different resource needs drives the query to inefficient performance.

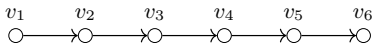
Contribution. In order to avoid such propagation of tuning setup, we propose a novel resource allocation method that assigns a specific tuning setup for each job in the query plan. We formulate the searching for specific tuning setup as a combinatorial optimization problem, highlight its special structure, and show that some special properties preserve the computational efficiency for a particular input of this problem. We also present the shortcoming of tuning SQL-on-Hadoop queries with the current MapReduce tuning advisers and how these queries can achieve better performance when employing our method.

Structure. Section 2 presents the notations and definitions required for presenting the problem. Section 3 presents the problem of tuning SQL-on-Hadoop queries. Section 4 presents the current tuning allocation method and its impact on SQL-on-Hadoop queries. Section 5 presents our method. Section 6 concludes.

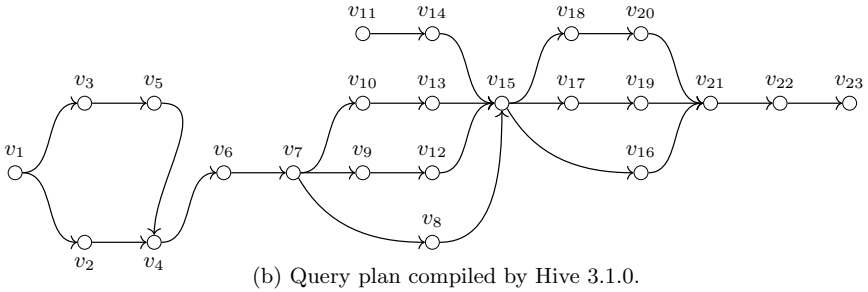
2 Notation and Definitions

Let us define a *query plan* as a directed acyclic graph $G = (V, E)$, where the set of vertices V represents the MapReduce jobs, and the set of edges E denotes the precedence between two jobs. More precisely, a vertex (job) $j \in V$ is a tuple of the form $j = (O_j, T_j, C_j)$ in which O_j is the set of physical *query operators* it executes, T_j is the set of *associated input tables* that are read and processed by j, and C_j is the set of *configurations* used to allocate resources. In this set each $c \in C$ represents a configuration exposed by the SQL-on-Hadoop system that allocates resources to jobs. Each directed edge is an ordered pair of vertices $e = (i, j) \in E$ that connects the jobs i to j, when the execution of i directly precedes j.

Figure 2 illustrates the query plans compiled for TPC-H query 7, in version 0.13.1 and 3.1.0 of Apache Hive to reinforce the growing complexity of SQL-on-



(a) Query plan compiled by Hive 0.13.1.



(b) Query plan compiled by Hive 3.1.0.

Fig. 2: Query plans for TPC-H query 7.

Hadoop queries. Note that the number of MapReduce jobs in each query plan differs considerably across software releases due to the development of different optimization techniques and the addition of new query operators. Although our example sticks to the Apache Hive terminology other SQL-on-Hadoop systems like Spark [3] and Impala [14] also represent the query plans as DAGs.

3 The Tuning Allocation Problem

In SQL-on-Hadoop systems, the task of allocating physical resources through tuning setups mainly comprises two steps: (1) searching for the most efficient tuning setup for a job $j \in V$, and (2) allocating the found tuning setup to the same job j. We define the *most efficient tuning setup* as the configuration that drives the query to decrease response time or minimize resource usage.

Searching for the most efficient tuning setup is a task performed by Tuning Advisers. The common approach is to profile the query execution and, then, to employ heuristics such as Genetic Algorithm [16,17,5], Hill Climbing [15,10], Random Forest [4], Particle Swarm Optimization [13], Exhaustive Search [18], Recursive Random Search [11] and others [21,27]. We define the searching for the most efficient tuning setup as the *Adviser Function*, as follows:

Definition 1 (Adviser Function). *We denote as $f(j)$ the adviser function $f : V \mapsto \mathcal{C}$ which is responsible for always finding the most efficient tuning setup C_j for $j \in V$, where the set $\mathcal{C} = \{C_1, \ldots, C_k\}$ is the collection of possible configurations.*

However, allocating the found tuning setup remains a problem due to the single point of configuration of SQL-on-Hadoop queries. We propose to replace the single point of configuration of queries, exposed to developers, by an automated tuning system able to allocate specific tuning setup per job. Thus, we

define the task of allocating tuning setups C_j for each job $j \in V$ in a query plan G as the problem of assigning the most efficient tuning setup found by the adviser function $f(j)$.

In Definition 2, we formulate the tuning allocation as a combinatorial optimization problem, and use this perspective to develop the reasoning about how to improve the current allocation of configurations to jobs for a query plan. For the problem definition, suppose that we have a computable function $\sigma : \mathcal{C} \times V \rightarrow \mathbb{R}$, where $\sigma(C_i, j)$ determines the computational cost of execute a job j with a tuning setup C_i. We define the problem of assigning the most efficient tuning setup, as follows:

Definition 2 (Tuning Allocation Problem). *Given a directed acyclic graph $G = (V, E)$ representing a query plan, a set $\mathcal{C}$ of possible configurations, and a cost function $\sigma : \mathcal{C} \times V \rightarrow \mathbb{R}$. Find an assignment of tuning setups such that the total cost to process all jobs is minimum, that is, the summation*

$$\sum_{j \in V} \sum_{C_i \in \mathcal{C}} \sigma(C_i, j)$$

is minimum.

Next, we present the current approach employed by SQL-on-Hadoop systems to assign tuning setups to jobs. In Section 5, we present our approach to assign tuning setups to jobs.

4 Uniform Tuning

The methodology employed by the current SQL-on-Hadoop engines allocates the same physical resources to all jobs of a query plan. We name this methodology as the *Uniform Tuning* method, and we treat it as a shortcoming for the query plan. We define the Uniform Tuning method as follows:

Definition 3 (Uniform Tuning). *The Uniform Tuning is the assignment of configurations to jobs such that $C_u = C_v$ for any $u, v \in V$ and $C_u, C_v \in \mathcal{C}$.*

Despite the adviser function $f(j)$ always find the most efficient tuning setup C_j for every job $j \in V$, in the case where only one tuning setup C_j is replicate to all jobs in V (according to Definition 3), C_j may negatively affect the query's performance, as we observe in Figure 3. The uniform tuning problem becomes more severe because SQL-on-Hadoop systems delegate to developers the responsibility to chose one of the available tuning setups C_j to be replicated.

The Fig. 3 illustrates that the Uniform Tuning drives the query processing to inefficient performance in practice through experiments in the Amazon Elastic MapReduce (EMR) cloud environment. We used Starfish [11] to generate the tuning setups. We ran the TPC-H benchmark with Scale Factor of 100GB on Apache Hive (version 0.13.1) and Apache Hadoop (version 0.20.2). We are attached to these versions because they are the versions supported by Starfish.

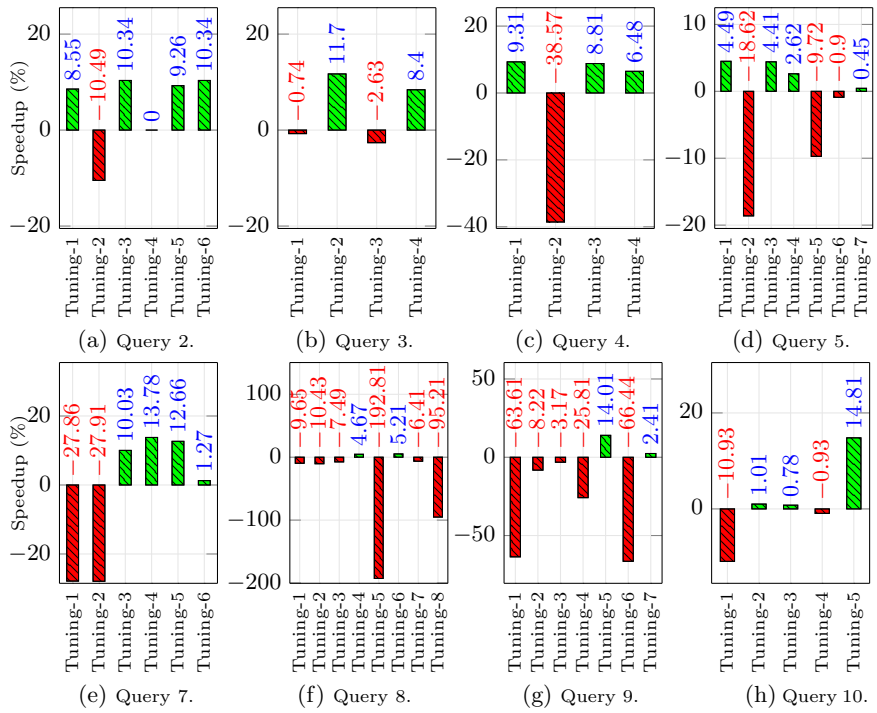


Fig. 3: Speedups compared to default configuration for the uniform tuning. Tuning-{v}, represents a tuning setup generated by Starfish for the job v.

The experiments ran with 6 machines (type c5.2xlarge: 8 of CPU and 16GB of RAM, 100GB of SSD disks). Each runtime is an average of 3 executions.

The x-axis represent the available tuning setups, where tuning-j represents the given C_j generated for a job j. Each C_j was applied uniformly to the query plan, following Definition 3. The y-axis represents the speedups when the execution of the query under the tuning-j is compared to the execution of the default configuration. The default configuration is the configuration bundled with the SQL-on-Hadoop system and is our baseline. According to Definition 3, each tuning setup allocates a different set of resources uniformly to the query, consequently, producing a different outcome. Note that in all queries the performance degrades due to the presence of the uniform allocation of resources, as expected.

We claim that applying a non-uniform allocation method to define a customized tuning setup for each job can outperform the current alternative (uniform tuning), which consists of choosing arbitrarily a tuning setup C_0 and replicate it to all jobs. Therefore, our contribution is a non-uniform allocation method that allows SQL-on-Hadoop systems to apply specific tuning setup per job.

5 Non-Uniform Tuning

For a query plan $G = (V, E)$, every job implements a different set of SQL operators, i.e., for every job $u, v \in V$ we have $O_u \neq O_v$. Consequently, we assume that each job presents different resource consumption needs [6]. For instance, while a job u requires disk bandwidth due to a *TableScan* operator, another job v requires memory throughput due to a *Sort* operator. Next, the rest of this section discuss properties and facts about the particular structure of this problem. The first fact states that the optimum of the problem presented in Definition 2 can be found efficiently when using a non-uniform tuning method. Since this study aims to find a way to do more efficient queries, it is very important to be sure that the allocation of optimal tuning can be made in polynomial time.

The following integer linear programming model describes the optimal solution for the Tuning Allocation Problem. We define the binary decision variable $x_{ij} \in \{0, 1\}$ to indicate whether a tuning setup $C_i \in \mathcal{C}$ is assigned to a job $j \in V$. Let the coefficient $c_{ij} = \sigma(C_i, j)$ be the computational cost for executing a job j with the tuning setup C_i.

$$\text{Minimize} \quad \sum_{j \in V} \sum_{C_i \in \mathcal{C}} c_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{C_i \in \mathcal{C}} x_{ij} = 1 \qquad \forall j \in V \tag{2}$$

$$\sum_{j \in V} x_{ji} = 1 \qquad \forall C_i \in \mathcal{C} \tag{3}$$

$$x_{ij} \in \{0, 1\} \qquad \forall C_i \in \mathcal{C}, \forall j \in V \tag{4}$$

The objective function (1) minimizes the computational cost in processing all the jobs for a given query plan. Constraints in (2) ensure that exactly one tuning setup is assigned to a vertex. Reciprocally, the equality in (3) says that only one vertex $j \in V$ can choose a tuning setup $C_i \in \mathcal{C}$. Finally, the constraints in (4) preserve the decision variable's integrality.

Claim 1. The Non-Uniform Tuning method can find the optimal solution for the Tuning Allocation Problem in polynomial time.

The program in (1)-(4) describes precisely the model of an well known combinatorial optimization problem called Assignment Problem. For clarity of exposition, we present a procedure to transform an instance $\langle G = (V, E), \mathcal{C} \rangle$ of the Tuning Allocation Problem into an instance $\langle G' = (V', E') \rangle$ of the original Assignment Problem as follows. Let G' be a undirected graph such that we create a new vertex i associated to each set $C_i \in \mathcal{C}$. Also add a copy of each $j \in V$ and define U as the set of all i added to G' in this way we have $V' = U \cup V$ with U and V disjoint. Now, we add an edge $\{i, j\}$ between every vertex $i \in U$ and $j \in V$. The cost of assigning a tuning setup C_i to job j is c_{ij} and can be interpreted as an weight on the edge $\{i, j\} \in E'$. Notice that all edges

in E' go between U and V. This implies that G' (the input of the Assignment Problem) is a complete bipartite graph.

Even though the Assignment Problem problem is known to be NP-hard, an optimal solution can be obtained efficiently (within execution time bounded by a polynomial function) when the input graph is bipartite [25]. Therefore, the problem can be solved efficiently, because of the *total unimodularity* of the constraint matrix of the Assignment Problem.

An important observation about the Tuning Allocation Problem is that if we drop the constraints (3) of the linear model, the solution remains feasible for the problem. It means that we can attribute a configuration C_i to more than one job (otherwise, the uniform tuning method would not be feasible). Therefore, the natural formulation for this problem is a relaxation for the Assignment Problem. Actually, the resulting integer linear program can be rewritten as follows:

$$\begin{aligned} \text{Minimize} \quad & \sum_{j \in V} \sum_{C_i \in \mathcal{C}} c_{ij} x_{ij} \\ \text{subject to} \quad & \sum_{C_i \in \mathcal{C}} x_{ij} = 1 && \forall j \in V \\ & x_{ij} \in \{0, 1\} && \forall C_i \in \mathcal{C}, \forall j \in V. \end{aligned}$$

Fortunately, the coefficient matrix does not change, and we still have an incidence matrix of a bipartite graph. In this way, the matrix of coefficients of the problem maintains the total unimodularity. Consequently, the Tuning Allocation Problem also can be solved in polynomial time.

Claim 2. For a query plan $G = (V, E)$, in worst case, the non-uniform method is equal to the uniform method, otherwise the non-uniform method performs better.

Let the tuning setup C_0 be the one assigned to all jobs $j \in V$ of the query plan G using the uniform tuning. Now, consider the case with non-uniform tuning in which for the same query plan G, the most efficient tuning for every job j can be obtained by calling interactively the function $f(j)$ (see Definition 1). Directly from definition of the adviser function $f(j)$, we have the following inequality

$$\sigma(f(j), j) \leq \sigma(C_0, j) \tag{5}$$

for every job $j \in V$. The inequality says that, in the best case for the uniform tuning, the arbitrary tuning setup C_0 can be the most efficient for j, but there are no guarantees that it will happen. While attributing a tuning setup C_i chosen by the function $f(j)$, we know that it is always the most suitable tuning setup for j. Consequently, the total cost required to execute all jobs $j \in V$ of the query plan G using the non-uniform method, can be written, as the following summation:

$$\sum_{j \in V} \sigma(f(j), j).$$

So, from the inequality (5) we can compare the total costs of the two methods, as follows:

$$\sum_{j \in V} \sigma(f(j), j) \leq \sum_{j \in V} \sigma(C_0, j),$$

i.e., the computational cost for executing the query plan G using the non-uniform assignment method is at most equals to the cost for the uniform method.

The advantage of using the uniform tuning method is that it is simple and straightforward, i.e., the SQL-on-Hadoop engine just replicates the chosen tuning setup. On the other hand, there are no guarantees that the chosen tuning setup will perform well during the query's execution. In this sense, our proposal of non-uniform tuning personalizes the tuning setup for each job and the optimal assignment of tuning setups is guaranteed. In other words, the Tuning allocation Problem can be solved optimally with the non-uniform tuning method and Claim 1 shows that it can be done in polynomial time. Furthermore, a direct consequence of the non-uniform tuning method is the improvement in the performance of the query plan, as stated in Claim 2.

6 Conclusion

In this paper we presented the shortcoming of MapReduce tuning advisers when applied to SQL-on-Hadoop systems due to the uniform allocation of physical resources and its consequences on query's performance. After, we modeled the Tuning Allocation Problem in order to solve the uniform allocation of physical resources. We presented our *Non-Uniform Tuning* method and proved that it allows assigning optimal tuning setups to SQL-on-Hadoop queries. We also proved that the optimal set of tuning setups required by a query can be found in polynomial time. For future work, an interesting direction consists of combining multiple tuning advisers to optimize one single query, once different tuning advisers focus on different query operators with different resource usage.

Acknowledgement: This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

1. Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D., Silberschatz, A., Rasin, A.: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. Proceedings of the VLDB Endowment (2009)
2. Apache Software Foundation: Apache Flink: Stateful Computations over Data Streams. Apache.Org (2015)
3. Armbrust, M., Xin, R.S., Lian, C., Huai, Y., Liu, D., Bradley, J.K., Meng, X., Kaftan, T., Franklin, M.J., Ghodsi, A., Zaharia, M.: Spark SQL: Relational Data Processing in Spark
4. Bei, Z., Yu, Z., Zhang, H., Xiong, W., et al.: RFHOC: A Random-Forest Approach to Auto-Tuning Hadoop's Configuration. TPDS (2016)
5. Bei, Z., Yu, Z., Liu, Q., Xu, C., et al.: MEST: A Model-Driven Efficient Searching Approach for MapReduce Self-Tuning. IEEE Access (2017)

6. Boncz, P.A., Neumann, T., Erling, O.: Tpc-h analyzed: Hidden messages and lessons learned from an influential benchmark. In: TPCTC (2013)
7. Chen, S.: Cheetah: a high performance, custom data warehouse on top of MapReduce. Proceedings of the VLDB Endowment (2010)
8. Costea Adrian Ionescu Bogdan, A.R., Micha Switakowski Cristian Bârc, A., Sompolski Alicja Łuszczak Michał Szafrá nski Giel de Nijs Peter Boncz, J.: VectorH: Taking SQL-on-Hadoop to the next level. In: International Conference on Management of Data - SIGMOD (2016)
9. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: OSDI'04 (2004)
10. Ding, X., Liu, Y., Qian, D.: JellyFish: Online Performance Tuning with Adaptive Configuration and Elastic Container in Hadoop Yarn. In: ICPADS (2015)
11. Herodotou, H., Lim, H., Luo, G., Borisov, N., Dong, L.: Starfish: A Self-tuning System for Big Data Analytics. CIDR (2011)
12. Jiang, D.: The Performance of MapReduce : An In-depth Study. Proceedings of the VLDB Endowment (2010)
13. Khan, M., Huang, Z., Li, M., Taylor, G.A., Khan, M.: Optimizing Hadoop parameter settings with gene expression programming guided PSO. Concurrency and Computation: Practice and Experience (2017)
14. Kornacker, M., Behm, A., Bittorf, V., et al.: Impala: A Modern, Open-Source SQL Engine for Hadoop. CIDR (2015)
15. Li, M., Zeng, L., Meng, S., Tan, J., et al.: MRONLINE: MapReduce online performance tuning. In: HPDC (2014)
16. Liao, G., Datta, K., Willke, T.L., Kalavri, V., et al.: Gunther: Search-based autotuning of MapReduce. Euro-Par (2013)
17. Liu, C., Zeng, D., Yao, H., Hu, C., et al.: MR-COF: A Genetic MapReduce Configuration Optimization Framework. In: Theoretical Computer Science (2015)
18. Liu, J., Ravi, N., Chakradhar, S., Kandemir, M.: Panacea: towards holistic optimization of MapReduce applications. In: CHO (2012)
19. Qin, X., Chen, Y., Chen, J., Li, S., Liu, J., Zhang, H.: The Performance of SQL-on-Hadoop Systems - An Experimental Study. In: IEEE International Congress on Big Data (2017)
20. Sakr, S., Liu, A., Fayoumi, A.G.: The Family of MapReduce and Large-Scale Data Processing Systems. ACM Computing Surveys (2013)
21. Shi, J., Zou, J., Lu, J., Cao, Z., Li, S., Wang, C.: MRTuner: A Toolkit to Enable Holistic Optimization for MapReduce Jobs. PVLDB (2014)
22. Tapdiya, A., Fabbri, D.: A Comparative Analysis of State-of-The-Art SQL-on-Hadoop Systems for Interactive Analytics. In: IEEE International Conference on Big Data (2017)
23. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., Murthy, R.: Hive: a Warehousing Solution Over a Map-Reduce Framework. Proceedings of the VLDB Endowment (2009)
24. Van Aken, D., Pavlo, A., Gordon, G.J., Zhang, B.: Automatic Database Management System Tuning Through Large-scale Machine Learning. In: SIGMOD (2017)
25. Wolsey, L.A., Nemhauser, G.L.: Integer and combinatorial optimization. John Wiley & Sons (2014)
26. Yang, H., Luan, Z., Li, W., Qian, D.: MapReduce Workload Modeling with Statistical Approach. Journal of Grid Computing (2012)
27. Zhang, B., Krikava, F., Rouvoy, R., Seinturier, L.: Self-Balancing Job Parallelism and Throughput in Hadoop. In: DAIS (2016)

Datalog-based Reasoning for Knowledge Graphs

Luigi Bellomarini[1,2], Georg Gottlob[1,3], and Emanuel Sallinger[1,3]

[1] University of Oxford
[2] Banca d'Italia
[3] TU Wien

Abstract. This is a short abstract based on the recently published VLDB 2018 paper [6] – the main technical paper describing the Vadalog system. It describes the central motivations and gives useful pointers on the architecture and the main algorithms.

Introduction and motivation. The importance of capitalizing and exploiting corporate knowledge has been clear to decision makers since the late 1970s, when this idea was gradually made concrete in the context of *expert systems*, software frameworks able to harness such knowledge and provide answers to structured business questions. Through *deductive database systems* – database systems that have advanced reasoning capabilities – and in particular the language *Datalog*, the area became of high interest to the database community in the 1980s and 1990s. Nevertheless, even though the importance of harnessing knowledge certainly has grown steadily since then, culminating in today's desire of companies to exploit *knowledge graphs*, the interest in deductive databases has faltered due to immature technology and overly complicated KRR formalisms.

Yet, we are currently assisting to a resurgence of *Datalog* in academia and industry [1, 4, 7–10]: companies like LogicBlox have proven that a fruitful exchange between academic research and high-performance industrial applications can be achieved based on Datalog [1], and companies like LinkedIn have shown that the interest in Datalog permeates industry [16]. Meanwhile, the recent interest in machine learning brought renewed visibility to AI, raising interest and triggering investments in thousands of companies world-wide, which suddenly wish to collect, encapsulate and exploit their corporate knowledge in the form of a knowledge graph.

In this context, it has been recognized that to handle the complex knowledge-based scenarios encountered today, such as reasoning over large knowledge graphs, Datalog has to be extended with features such as existential quantification. Yet, Datalog-based reasoning in the presence of existential quantification is in general undecidable. Many efforts have been made to define decidable fragments. Warded Datalog$^{\pm}$ is a very promising one, as it captures PTIME complexity while allowing ontological reasoning. Yet, so far, no implementation of Warded Datalog$^{\pm}$ was available.

We recently introduced the Vadalog system, a Datalog-based system for performing complex logic reasoning tasks, such as those required in advanced knowledge graphs; it is Oxford's contribution to the VADA research programme [18, 13], a joint effort of the universities of Oxford, Manchester and Edinburgh and around 20 industrial partners such as Facebook, BP, and the NHS (UK national health system). The Vadalog system proposes the first implementation of Warded Datalog$^{\pm}$, a high-performance Datalog$^{\pm}$ system utilising an aggressive termination control strategy. In this paper, we summarise the key aspects of the Vadalog system, while pointing to the original technical paper for the details and a comprehensive experimental evaluation.

Reasoning over knowledge graphs. The term *knowledge graph* (KG) has no standard definition. It can be seen as referring only to Google's Knowledge Graph, to triple-based models, or to multi-attributed graphs, which represent n-ary relations [15, 17]. As shown by Krötzsch [14], in order to support rule-based reasoning on such data structures, it is sometimes necessary to use tuples of arity higher than three at least for intermediate results. In this paper, we adopt a general notion of KGs by allowing relations of arbitrary arity, to support all of these models and modes of reasoning.

Example 1. An example of a simple knowledge graph reasoning setting is given in [15]:

$$\text{Spouse}(x, y, \mathit{start}, \mathit{loc}, \mathit{end}) \rightarrow \text{Spouse}(y, x, \mathit{start}, \mathit{loc}, \mathit{end})$$

This rule expresses that when a person x is married to a person y at a particular location, starting date and end date, then the same holds for y and x. That is, the graph of persons and their marriage relations is symmetric.

As stated in [15], most modern ontology languages are not able to express this example. Beyond this simple example, there are numerous requirements for a system that allows ontological reasoning over KGs. Navigating graphs is impossible without powerful recursion; ontological reasoning is impossible without existential quantification in rule heads [12]. Yet reasoning with recursive Datalog is undecidable in the presence of existential quantification, so some tradeoffs have to be accepted. While a comprehensive analysis of various requirements was given in [5], let us isolate three concrete characteristics for a language that supports reasoning over KGs:

1. **Recursion over KGs**. Should be at least able to express full recursion and joins, i.e., should at least encompass Datalog. Full recursion in combination with arbitrary joins allows to express complex reasoning tasks over KGs. Navigational capabilities, empowered by recursion, are vital for graph-based structures.
2. **Ontological Reasoning over KGs**. Should at least be able to express SPARQL reasoning under the OWL 2 QL entailment regime and set semantics. OWL 2 QL is one of the most adopted profiles of the Web Ontology Language, standardized by W3C.
3. **Low Complexity**. Reasoning should be tractable in data complexity. This is a minimal requirement for allowing scalability over large volumes of data.

Beyond these specific requirements, the competition between powerful recursion, powerful existential quantification and low complexity has spawned fruitful research throughout the community to address the mentioned Datalog undecidability in the presence of existential quantification. This has been done under a number of different names, but which we shall here call *Datalog*$^{\pm}$, the "+" referring to the additional features (including existential quantification), the "-" to restrictions that have to be made to obtain decidability. Many languages within the Datalog$^{\pm}$ family of languages have been proposed and intensively investigated [2, 3, 7–10]. Depending on the syntactic restrictions, they achieve a different balance between expressiveness and computational complexity.

Figure 1 gives an overview of the main Datalog$^{\pm}$ languages. In fact, most of these candidates, including Linear Datalog$^{\pm}$, Guarded Datalog$^{\pm}$, Sticky Datalog$^{\pm}$ and Weakly Sticky Datalog$^{\pm}$ do not fulfil (1). Datalog itself does not fulfil (2). Warded and Weakly Frontier Guarded Datalog$^{\pm}$ satisfy (1) and (2), thus are expressive enough. However,

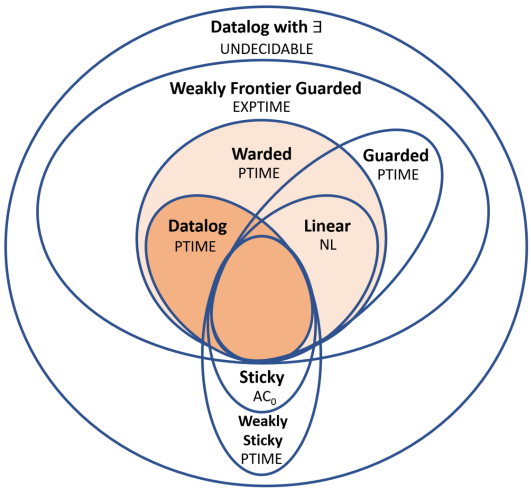


Fig. 1: Syntactic containment of Datalog$^{\pm}$ languages. Annotations (non-bold) denote data complexity. All names that do not explicitly mention Datalog refer to the respective Datalog$^{\pm}$ languages. E.g., "Sticky" refers to "Sticky Datalog$^{\pm}$".

the expressiveness of Weakly Frontier Guarded Datalog$^{\pm}$ comes at the price of it being EXPTIME-complete [3]. Thus it does not fulfil (3). Thus, in total, the only known language that satisfies (1), (2) and (3) is Warded Datalog$^{\pm}$. Yet, while Warded Datalog$^{\pm}$ has very good theoretical properties, the algorithms presented in [12] are alternating-time Turing machine algorithms, far away from a practical implementation.

Before summarising the characteristics of the Vadalog system, let us propose an example of a Datalog program, which we consider representative of the complexity of a typical reasoning setting involving KGs. The example is at the same time close to a classical scenario [11] as well as relevant for an application of the KG of one of our current industrial partners; in particular, for detecting specific risks related to issuers or guarantors for funds that are non-eligible due to their financing arrangement: in Europe there is prohibition on guaranteed debt instruments issued by a "closely-linked entity".

Example 2. Consider a set of rules about ownership relationships among a large number of companies. The extensional knowledge is stored in a database in the form of tuples of a relation Own($comp_1$, $comp_2$, w): company $comp_1$ directly owns a fraction w of company $comp_2$, with $0 \leq w \leq 1$. In addition, with a set of two rules, we intensionally represent the concept of *company control* (also studied in [11]): a company x controls a company y if company x directly owns more than half of the shares of company y or if x controls a set S of companies that jointly own more than half of y. Note that the `msum` operator, i.e., aggregation in the form of *monotonic sum*, is not a standard feature of Datalog, but an extension that some Datalog-based systems support [6].

$$\text{Control}(x, x).$$
$$\text{Own}(x, y, w), w > 0.5 \rightarrow \text{Control}(x, y)$$
$$\text{Control}(x, y), \text{Own}(y, z, w), v = \texttt{msum}(w, \langle y \rangle), v > 0.5 \rightarrow \text{Control}(x, z).$$

Here, for fixed x, the aggregate construct $\texttt{msum}(w, \langle y \rangle)$ forms the sum over all values w such that for some company y, $\text{Control}(x, y)$ is true, and $\text{Own}(y, z, w)$ holds, i.e., company y directly owns fraction w of company z. Now, with the described knowledge graph, many questions can be asked, such as: (i) obtain all the pairs (x, y) such that company x controls company y; (ii) which companies are controlled by x? Which companies control x? (iii) does company x control company y?

The Vadalog system. The Vadalog system is built around the Vadalog language, with Warded Datalog$^{\pm}$ as its logical core. It is currently used as the core deductive database system of the overall *Vadalog Knowledge Graph Management System* described in [5], as well as at various industrial partners, including the finance, security, and media intelligence industries. The **main contributions** are:

- A **novel analysis** of Warded Datalog$^{\pm}$, which culminates in the **first practical algorithm for Warded Datalog$^{\pm}$**. In the Vadalog system, the core principle behind the reasoning process is performing an *aggressive termination control*, which amounts to adopting dynamic programming strategies to preempt the generation of already explored areas of the KG as well as isomorphic ones, guaranteeing at the same time termination (as the fragment is decidable) and efficiency (as all the "informative" KG areas can be explored in PTIME). To achieve this goal, it is essential to recognise these KG areas as early as possible.
  While the naïve solution would imply a full caching of all the generated facts, the challenge here is guaranteeing limited memory footprint. We identify a number of *guide data structures* that closely exploit the underpinnings of Warded Datalog$^{\pm}$ and allow to abstract large KG areas by means of single representative facts (or patterns thereof). More in particular, we propose structures that play a complementary role for exploiting the periodicity of the KG: *warded forest*, which actually allows *vertical pruning* of isomorphic trees based on root isomorphism, and *(lifted) linear forest*, which allows *horizontal pruning* of entire pattern-isomorphic trees.
- A **system and architecture** that implements this algorithm in a relational database-inspired operator pipeline architecture. The pipeline's operators rely on *termination strategy wrappers* which transparently prevent the generation of facts that may lead to non-termination while ensuring the correctness of the result. The system is completed by a wide range of standard and novel optimisation techniques such as the dynamic construction of in-memory indices, stream-optimized "slot machine" joins, monotonic aggregation, materialization points, etc.
- The technical paper also presents a full-scale **experimental evaluation** of the Vadalog system on a variety of real-world and synthetic scenarios that thoroughly validate the effectiveness of our techniques on Warded Datalog$^{\pm}$ in absolute terms and comparatively with the top existing systems, which are outperformed by our reasoner.

Acknowledgements. This work is supported by the EPSRC programme grant VADA EP/M025268/1, the Vienna Science and Technology Fund (WWTF) grant VRG18-013, and the EU Horizon 2020 grant 809965.

References

1. M. Aref, B. ten Cate, T. J. Green, B. Kimelfeld, D. Olteanu, E. Pasalic, T. L. Veldhuizen, and G. Washburn. Design and implementation of the LogicBlox system. In *SIGMOD*, pages 1371–1382, 2015.
2. J. Baget, M. Leclère, and M. Mugnier. Walking the decidability line for rules with existential variables. In *KR*. AAAI Press, 2010.
3. J. Baget, M. Mugnier, S. Rudolph, and M. Thomazo. Walking the complexity lines for generalized guarded existential rules. In *IJCAI*, pages 712–717. IJCAI/AAAI, 2011.
4. P. Barceló and R. Pichler, editors. *Datalog in Academia and Industry - Second International Workshop, Datalog 2.0, Vienna, Austria, September 11-13, 2012. Proceedings*, volume 7494 of *Lecture Notes in Computer Science*. Springer, 2012.
5. L. Bellomarini, G. Gottlob, A. Pieris, and E. Sallinger. Swift logic for big data and knowledge graphs. In *IJCAI*, pages 2–10, 2017.
6. L. Bellomarini, E. Sallinger, and G. Gottlob. The vadalog system: Datalog-based reasoning for knowledge graphs. *PVLDB*, 11(9):975–987, 2018.
7. A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.*, 48:115–174, 2013.
8. A. Calì, G. Gottlob, and T. Lukasiewicz. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.*, 14:57–83, 2012.
9. A. Calì, G. Gottlob, T. Lukasiewicz, B. Marnette, and A. Pieris. Datalog+/-: A family of logical knowledge representation and query languages for new applications. In *LICS*, pages 228–242, 2010.
10. A. Calì, G. Gottlob, and A. Pieris. Towards more expressive ontology languages: The query answering problem. *Artificial Intelligence*, 193:87–128, 2012.
11. S. Ceri, G. Gottlob, and L. Tanca. *Logic programming and databases*. Springer, 2012.
12. G. Gottlob and A. Pieris. Beyond SPARQL under OWL 2 QL entailment regime: Rules to the rescue. In *IJCAI*, pages 2999–3007, 2015.
13. N. Konstantinou, M. Koehler, E. Abel, C. Civili, B. Neumayr, E. Sallinger, A. A. A. Fernandes, G. Gottlob, J. A. Keane, L. Libkin, and N. W. Paton. The VADA architecture for cost-effective data wrangling. In *SIGMOD Conference*, pages 1599–1602. ACM, 2017.
14. M. Krötzsch. Efficient rule-based inferencing for OWL EL. In *IJCAI 2011*, pages 2668–2673, 2011.
15. M. Marx, M. Krötzsch, and V. Thost. Logic on MARS: ontologies for generalised property graphs. In *IJCAI 2017*, pages 1188–1194, 2017.
16. W. E. Moustafa, V. Papavasileiou, K. Yocum, and A. Deutsch. Datalography: Scaling datalog graph analytics on graph processing systems. In *BigData*, pages 56–65. IEEE, 2016.
17. J. Urbani, C. J. H. Jacobs, and M. Krötzsch. Column-oriented datalog materialization for large knowledge graphs. In *AAAI 2016*, pages 258–264, 2016.
18. VADA. Project. `http://vada.org.uk/`, 2016.

Dynamic Pipelining of Multidimensional Range Queries*

Amalia Duch, Daniel Lugosi, Edelmira Pasarella, and Cristina Zoltan

Universitat Politècnica de Catalunya, Spain
{duch,edelmira,zoltan}@cs.upc.edu

Abstract. The problem of evaluating orthogonal range queries efficiently has been studied widely in the data structures community. It has been common wisdom for several years that for queries containing more than 20% of the elements of the dataset a linear scanning of the data was the most efficient solution. In recent experimental works using modern hardware–with main memory and parallelism– the conclusion is that linear scan is preferable for almost every query configuration (even containing a 1% of the data). In this work we propose an alternative approach to evaluate multidimensional range queries based on the dynamic pipeline paradigm –using main memory and concurrency. Our aim is to prove that under this framework, it is possible to beat the performance of linear scanning by the one of hierarchical multidimensional data structures– such as kd trees, quad trees, R trees or similar.

1 Introduction

It is a common task nowadays to ask Google Maps for the closest gas station or TripAdvisor for good restaurants around a specific area. These requests are examples of a computing task –frequent in a wide range of applications– that is formally called the *Associative Retrieval* problem [2, 4]. In associative retrieval we consider a collection $\mathcal{F}$ of n records. Each *record* (or key) is an ordered k-tuple ($k \geq 2$) $x = (x_1, \ldots, x_k)$ of values (the attributes or coordinates of the record) drawn from domain $D = \prod_{1 \leq j \leq k} D_j$, where each D_j is totally ordered.

A *range query* over $\mathcal{F}$ is the retrieval of those of its records that fall inside a given region. Specifically, we consider *orthogonal* range queries Q, in which the region is specified by a sequence of s unidimensional ranges, this is, $Q = (i_1, l_1, u_1), \ldots, (i_s, l_s, u_s)$, where $1 \leq s \leq k$, $i_j \neq i_{j'} \forall j \neq j'$ (with $1 \leq j \leq s$ and $1 \leq j' \leq s$) and every triplet (i_j, l_j, u_j) fix the lower (l_j) and upper (u_j) boundaries of the unidimensional range for coordinate i_j $(1 \leq i_j \leq k)$. If $s = k$ then we say that Q is a *complete* range query. Otherwise, we say that it is *partial*.

In order to efficiently deal with orthogonal range queries the storage of the records in $\mathcal{F}$ should be crucial. Extensive collections of general purpose multidimensional data structures –such as kd-trees, quad-trees or R-trees– have been proposed theoretically as adequate storage methods [2, 4] to support range

* Work supported by grant GRAMM (TIN2017-86727-C2-1-R) and EU FEDER funds.

queries. However, in practice, the usefulness of this approach heavily relies on the selectivity and configuration of the sequence of range queries and, unfortunately common wisdom told that a simple scan beats multidimensional data structures for queries accessing more than 15%-20% of a data collection [5].

Recently, multidimensional range queries as well as the efficiency of hierarchical multidimensional data structures to support them have been revisited under a modern hardware perspective [5]. Moreover, in [5] the authors state that in current machines –using main memory and parallelisation– data structures are useless even for very selective range queries an thus, they conclude that in current machines scanning should be favoured over parallel versions of such data structures.

In this work, we propose a new way to parallelise the multidimensional range query problem using the *dynamic pipeline* model [1, 3]. Our aim is to prove that, with our algorithm, the use of hierarchical multidimensional data structures would be preferable over scanning for range queries containing a sub-linear number of elements of the collection.

2 Dynamic Pipeline Algorithms

We propose an algorithm based on a dynamic pipeline [1, 3] of processes via an asynchronous model of computation, synchronised by means of channels. In general, a *dynamic pipeline* is a data-driven unidimensional and unidirectional chain of stages connected by means of data channels. This computational structure stretches and shrinks depending on the spawning and the lifetime of its stages. A dynamic pipeline is similar to an ordinary pipeline, except that the number of stages that it contains is not fixed but dynamically generated at runtime. In fact, it is self-adaptive to the characteristics of a specific query.

Algorithms under this paradigm must specify four kind of stages: *input*, *output*, *generator* and *filter* stages as well as the number and the type of the I/O unidirectional channels. The input and output stages are the interface of the pipeline, managing the input and output data respectively. Input data is fed to the input stage and the output stage will produce results. The generator is responsible to create the (parameterised) filter stages.

We now describe two algorithms to solve range queries based on the dynamic pipeline model: a naïve algorithm equivalent to a linear scan of the whole dataset followed by the algorithm that we propose, based on a preprocessing of the dataset by means of data structures such as kd trees, quad trees or R trees. We will describe both algorithms for a single range query since the same process is applicable to a sequence of queries iterating on the process for a single one.

Naïve Algorithm. We start by describing a naïve algorithm equivalent to a concurrent approach of the complete scan of the data set.

To answer Q using the pipeline approach it is necessary to have a recursive process that constructs a sequence of filters (processes). Every filter stands for one of the s unidimensional ranges of Q, let us say j ($1 \leq j \leq s$), and it discards

from further consideration all the points of the data set that are outside range (i_j, l_j, u_j), that is, all those points with i_j-th coordinate smaller than l_j or greater than u_j. Since the query has s ranges, the pipe will end up with s processes acting concurrently.

This naïve algorithm starts by setting an initial pipeline consisting of 3 stages –the input, the generator and the output stages, in this sequential order– and two channels –the first carry the sequence of triplets of Q and the second the sequence of points in $\mathcal{F}$.

The process starts by feeding (in sequential order) the input stage with the triplets of Q carried by the first channel. The configuration of the pipe evolves (stretches) as follows. The input stage passes the data from the first channel (a triplet of the query at a time) to its successor neighbour. At the very beginning the triplet (i_1, l_1, u_1) is passed from the input stage to the generator stage. Every time a triplet arrives to the generator a filter stage, standing for this triplet, is created as the stage immediately previous to the generator stage. So, at the very beginning, a filter f_1 for the first range of Q is created. The pipeline consists now of four stages: input, f_1, generator and output, in this order. When triplet (i_j, l_{i_j}, u_{i_j}) passes through the input stage, it passes also through f_1, since $i_j \neq i_1$, it passes through f_2, ..., f_j, up to arrive to the generator where filter f_{j+1} is created. The process continues until the elements carried by the first channel are all treated and the channel is empty. The pipe now, regarding the initial pipe, has s additional stages, one per each triplet of the query.

The next step is to treat the data carried by the second channel, the points. Every point of the data set passes through the pipe. At every filter f_i, as we already mentioned, if the i-th coordinate of the point is outside the range stored at f_i the point is discarded, otherwise it is passed to next stage. Therefore, all the points that arrive to the output stage should be reported as part of Q.

This naïve algorithm will force to read and check every point in the original set, having therefore complexity proportional to n, independently of the configuration of Q.

Proposed Algorithm. To improve the efficiency of the naïve algorithm we propose a preprocessing of the dataset by means of a hierarchical multidimensional data structure. The data structure can be any of the classical ones [2, 4] (such as kd trees, quad trees, R trees, etc.) with the unique requirement that it divides the domain D of the points into a partition of m k-dimensional hyperrectagles that are called *bounding boxes*, where $m \geq 1$ is the number of elements in the partition and depends on the kind of tree used and the number of levels of that tree. Each bounding box BB is defined by a sequence of k ranges, this is, $BB = (1, l_1, u_1), \ldots, (k, l_k, u_k)$. In our preliminar experiments we use quad tries [2, 4] to preprocess the data points and to end up with a sequence $S = BB_1, \ldots, BB_m$ of bounding boxes. It is worth noting that the dynamic pipeline algorithm works identically with any other multidimensional data structure that fits the previous requirement.

Now, instead of directly filtering points, the pipe will filter, first, the sequence S of bounding boxes produced by the data structure (it will have then 3 channels

instead of two, as before). The bounding boxes will pass through the pipe by their specification and not by the points that they contain. Every filter stage, checking for the i-th range of the query, will discard from further consideration all the bounding boxes whose i-th range does not intersect the i-th range of the query and will pass the intersecting bounding boxes to next stage. Additionally, the filtering of bounding boxes divides them into two groups: BBC (the group of bounding boxes which are completely contained into Q) and BBP (the group of bounding boxes that intersect Q but are not contained in it).

The algorithm will output all points that are inside of BBC bounding boxes (since they are all in Q) and it will look and filter all the points in BBP bounding boxes to decide whether they are in Q.

The total number of treated points corresponds to the number of points contained in the BBC and BBP bounding boxes, which can be considerably less than the total number of points in $\mathcal{F}$ (but this highly depends on the configuration of the queries, data points, and chosen data structure). Additionally, the algorithm incurs in the cost of checking the m bounding boxes of S. Our proposal is to maintain this cost negligible compared to the number of points that have to be checked by choosing correctly the number of levels of the tree data structure during the preprocessing of the data.

3 Ongoing Work

We have implemented quad tries in the C++ programming language producing with this program the sequence of m files that containing the points inside the corresponding bounding box in sequence S. Besides, we have implemented the pipeline in the Go programming language (because of its mechanisms of go-routines and channels). Our preliminar experiments show that our proposed algorithm beats the naïve one treating systematically half the points of the data set for queries containing up to a 25% of the points of $\mathcal{F}$. We plan to conduct further experiments according to the following guidelines: (a) Considering huge datasets and allocating their corresponding (tree-like) hierarchical representations in the RAM, we envision that the performance of our algorithm overcomes the results presented in [5] under similar conditions and thus, it overcomes the complexity of linear scanning. We will study, then, the incidence of stressing the population of the memory in order to find insights regarding the percentage of memory that can be used for storage purposes without affecting the performance of the schedule and the memory management of the Go system; (b) Under the premise that the set of points is uniformly distributed, we plan to measure the incidence of the chosen level of the data structure (and thus of the number m of bounding boxes to be filtered) on the performance of our algorithm; (c) The dimensionality of the data increases the parallelism of our algorithm –which depends on the query– so we are interested in studying how, eventually, our algorithm is more suitable than other proposals in high dimensional settings; (d) Finally, in order to study the scalability and real applicability of our model, we plan to conduct our experiments with big real datasets and benchmarks.

References

1. J. Aráoz and C. Zoltan. Parallel triangles counting using pipelining. *CoRR*, abs/1510.03354, 2015.
2. V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.
3. E. Pasarella, M-E. Vidal, and C. Zoltan. Comparing mapreduce and pipeline implementations for counting triangles. *Electronic proceedings in theoretical computer science*, 237:20–33, 2017.
4. H. Samet. *Foundations of Multidimensional and Metric Data Structures.*
5. S. Sprenger, P. Schäfer, and U. Leser. Multidimensional range queries on modern hardware. In *Proceedings of the 30th International Conference on Scientific and Statistical Database Management, SSDBM 2018, Bozen-Bolzano, Italy, July 09-11, 2018*, pages 4:1–4:12, 2018.

AMW 2019

This Event is co-financed by Consejo Nacional de Ciencia y Tecnología - CONACYT with FEEI resources.

Este Evento es cofinanciado por el Consejo Nacional de Ciencia y Tecnología - CONACYT con recursos del FEEI.