Graph Constraints in Urban Computing: Dealing with conditions in processing urban data

Laurent d'Orazio, Mirian Halfeld-Ferrari, Carmem Satie Hara, Nadia P Kozievitch, Martin A Musicante

Graph Constraints in Urban Computing:

Dealing with conditions in processing urban data

Laurent D'Orazio*, Mirian Halfeld-Ferrari†, Carmem Satie Hara‡,
Nadia P. Kozievitch§, Martin A. Musicante¶
**Université de Rennes 1 - IRISA - UMR 6074 CNRS, France. Email: laurent.dorazio@univ-rennes1.fr.*
†Université d'Orléans, INSA CVL, LIFO EA - Orléans, France. Email: mirian@univ-orleans.fr.
‡Universidade Federal do Paraná, Curitiba, Brazil. Email: carmem@inf.ufpr.br.
§Universidade Tecnológica Federal do Paraná, Curitiba, Brazil. Email: nadiap@utfpr.edu.br.
¶Universidade Federal do Rio Grande do Norte, Natal, Brazil. Email: mam@dimap.ufrn.br.

***Abstract*—Smart Cities is a world-wide initiative leading to better exploit the resources in a city in order to offer higher level services to people. In this context, urban computing is a process of acquisition, integration, and analysis of big and heterogeneous data generated by a diversity of sources in urban spaces, such as sensors, traffic devices, vehicles, buildings, and humans, to tackle the major issues that cities face, e.g. air pollution, increased energy consumption and traffic congestion. The majority of these information can be represented as graphs, such as the transportation network, in which places (nodes) are connected by some form of public transportation (edges). A vision of the "city of the future", or even the city of the present, rests on the integration of science and technology through information systems. This vision requires a re-thinking of the relationships between technology, government, city managers, business, academia and the research community. This position paper presents our views towards developing techniques for querying and evolving graph-modeled datasets based on user-defined constraints. Our focus is to show how these techniques can be applied to effectively retrieve urban data and have automated mechanisms that guarantee data consistency.**

***Keywords*-Graph Databases; Query Languages; Constraints; Multi-Objective Optimization; Urban Computing.**

I. Introduction

According to [1], it is expected that 70% of the world's population will reside in cities in less than 40 years, bringing new concerns about the sustainability of the infrastructures caused by this massive growth: cities consume 75% of the energy resources and are the cause of 80% of the carbon footprint. With the advent of smart technologies, the population, urban transportation systems, and other elements in the urban environment produce a lot of useful data. Such data can be used to shed light on a number of factors, including user trends and traffic patterns. Integrating multiple heterogeneous data sources into robust and dependable systems to assist people and empower communities involves operating with complex elements that are networked at multiple scales and enabled by various technologies. Such elements often generate massive amounts of dynamic and static data with different levels of abstraction and quality, at multiple temporal and spatial scales, and at real time rates. Dealing with such characteristics, while respecting domain restrictions and mathematics concerns, raises scientific, technological and engineering challenges.

Urban computing proposes the integration of data acquired from a variety of traditional data sources as well as from sensors and other devices, in order to produce knowledge that provide guidance to find solutions to problems in big cities such as pollution, energy consumption, and human mobility. Mobility challenges have already gained attention of the Brazilian Computer Society ([2]) and has also been the subject of several actions of the European Commission ([3]). One of the challenges is the great amount of data from different domains made available. Turning data into knowledge calls for tools capable of dealing with collections of datasets potentially distributed across multiple servers. Traditional database management tools have not been designed to support these new tasks. Indeed, nowadays, data is often associated to the problem of *volume* (referring to the big quantity of data); *variety* (due to its heterogeneity); *velocity* (referring to the continuous production rate) and *veracity* (due to the uncertainty of the data). Besides, data items are usually interrelated and provided with specific semantics. The exploration of such data in decision-making and integration scenarios raises new research challenges. A graph-based data model is appropriate for representing most of datasets involved in urban computing, from traffic information and points of interest, to more conventional data, such as population information and statistics. As an example, Figure 1 shows the bus stops and lines in the city of Curitiba, Brazil, and the City Center in detail. It shows the graph-oriented type of the data and its complexity, considering only a single means of transport. The development of efficient tools for querying and evolving data graphs under different kinds of constraints is essential in this context. Examples of user-defined constraints in this context include routes with a limit on carbon footprint, or that are accessible for wheelchair users ([4]).

Lessons we learn from actions around big data contributed to the emergence of what is now called data science (i.e.,

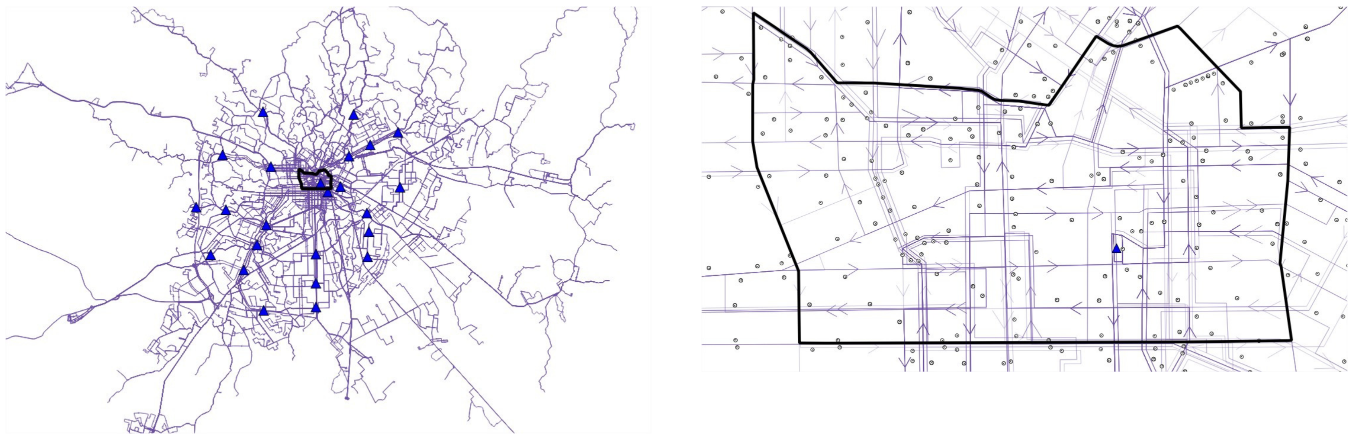


Figure 1: Bus stops and lines in Curitiba (city area of 430 km^2; 1.8 million inhabitants) and its Center.

the extraction of knowledge from data through expertise in disciplines within the fields of mathematics, statistics, and computer science) ([5]). With the evolution towards the cloud with “unlimited” access to resources (computing, storage, memory) it is possible to exploit big data in persistent media (cache, main memory or disk). Dispatching processes, producing and delivering results imply having efficient and well-adapted data management infrastructures. These services are not completely available in existing systems. Thus, it is important to revisit and provide services that cope with big data and semantic characteristics. The key challenge is to hide the complexity for harvesting, storing, curating, accessing and analyzing big data but also to provide interfaces for tuning these functions according to urban computing requirements.

The purpose of this *position* paper is to present our views and efforts towards developing techniques for querying and evolving data graphs, and their application in urban computing. Consistency is the main theme of our investigations. In an upper level, we aim at setting the basis on how to ensure data consistency either in the answers of a querying process or as the result of data and constraint evolution. We consider constraints as a way of personalizing user’s context and emphasize the importance of mechanisms ensuring valid answers independently of distributed sources consistency. At a lower level, we focus on the performance of an interrogation process in a distributed environment.

This paper is organized as follows: Section II motivates the need for new languages capable of dealing with the challenges of querying graph data. Section III presents the challenges of dealing with data constraints, in order to keep a consistent view of the graph database. Section IV shows optimization techniques that may help in processing urban data, in order to obtain reliable information as well as to incorporate user constraints into queries. Section V briefly presents the conclusions of this position paper.

II. Query language for graphs

Graph database systems are becoming popular, mainly because of their ability to represent data of heterogeneous nature.

The Resource Description Framework (RDF) ([6]), a graph-based data model adopted as a general method for the Semantic Web for data publishing and sharing, is now a standard format for data interchange on the Web. Some of the principles of the Semantic Web are also desirable in the urban context. Unique identity for entities facilitates data integration, and linked data promotes knowledge discovery. RDF defines a graph as a set of triples $\langle$subject, predicate, object$\rangle$, where the predicate is seen as an edge linking the subject node to the object node. The *de-facto* standard for querying RDF data is SPARQL ([7]), a declarative query language that uses regular expressions to define paths over the queried graph.

Other languages for processing graph data are Gremlin ([8]) and Cypher ([9]). Gremlin is a query language aimed at graph traversal (in contrast to graph pattern matching), that provides primitives to define and compose paths over the data graph. The Cypher language is inspired by SQL and uses a “pictorially inspired” textual representation ([10]) of patterns, in order to query property graphs (a data model that extends RDF-like graphs by adding valued attributes to nodes and edges).

In order to query graph data, these languages offer ways of defining path expressions that represent graph patterns, to be matched by the data. The most used way of defining these path expressions is by means of regular expressions.

While regular expressions are a well known formalism that allows for efficient implementations, their computational power does not answer to *all* the needs for queries. Examples of this are the "same generation queries" in [11] or the more general sub-graph matching queries in [12]. An important aspect for the maturation of SPARQL is the introduction of

Query on a transportation database

$\mathsf{Connexion}(X_{from}, X_{to}, X_{time}, X_{cfp})$	$\leftarrow$	$\mathsf{Transp}(X_{from}, X_{to}, X_{means}, X_{time}, X_{cfp})$.
$\mathsf{Connexion}(X_{from}, X_{to}, X_{time}, X_{cfp})$	$\leftarrow$	$\mathsf{Transp}(X_{from}, Z, Z_{means}, Z_{time}, Z_{cfp})$, $\mathsf{Connexion}(Z, X_{to}, Y_{time}, Y_{cfp})$, $X_{time} = Z_{time} + Y_{time}$, $X_{cfp} = Z_{cfp} + Y_{cfp}$.
$\mathsf{Q}(X_{from}, X_{to}, a_{totalAvTime}, a_{Avcfp})$	$\leftarrow$	$agg(\mathsf{Connexion}(X_{from}, X_{to}, X_{time}, X_{cfp}), X_{from}, X_{to}$, $a_{totalAvTime} = AVG(X_{time}),\ \ a_{Avcfp} = AVG(X_{cfp}))$.

Constraints defining a context ($\mathcal{C}$)

c_1	$\mathsf{Transp}(X_{from}, X_{to}, X_{means}, X_{time}, X_{cfp})$	$\rightarrow$	$\mathsf{Type}(X_{means}, Rail)$.
c_2	$\mathsf{Transp}(X_{from}, X_{to}, X_{means}, X_{time}, X_{cfp}), (X_{cfp} > 200)$	$\rightarrow$	$\perp$.

Table I: Example of a query and a user's context on a transportation database.

graph analysis functions ([13]).

In this context, defining a query language that is *declarative* and powerful enough to obtain not only the usual information expected from a semantic graph database, but also information built from the analysis of the graph (*e.g.* page rank or mininal path) is *still* a challenge to be met. High-performance graph processing engines have been classified into low- and high-level engines ([14]); the former requiring the user to write imperative code, the latter introducing declarative query languages, most of them inspired on datalog ([11]). Datalog, a well known *recursive* query language, appears as a solution to improve the declarative power and the elegance of writing recursive queries.

Consider, for instance, the means of transport in a city like Paris. A user wants to know the average price and carbon footprint for a journey from home to work. The query on the top of Table I is written in a datalog-like query language we are proposing ([15]).

The transportation network is a graph where each node has the name of a specific place where a user can find a means of transport (a Metro station, a bus stop, a *Batobus* stop, a *Velib* station, etc). Relation Transp stores this networks. Notice that besides attributes indicating an edge (X_{to}, X_{from}) of the transportation graph, the relation stores information about the means of transport (X_{means} – bus, underground, boat, bike, etc), the estimated travel time (X_{time}) and carbon footprint (X_{cfp}) for the graph's edge. The network is thus a multi-graph; also seen as a superposition of different graphs (one for each means of transport). Queries on such a graph can vary from a simple '*Is there a bus from Alesia to Montparnasse?*' to graph analysis such as the minimal path to go from *Gare d'Austerlitz* to *Saint-Lazare*. The query Q in Table I is built from three rules. The first two rules compute all possible connexions between two given nodes. The last rule is an aggregation query that computes the desired answer from results in $\mathsf{Connexion}$.

The elegant and declarative form of expressing recursive queries have recently re-emerged the interest around Datalog for a wide spectrum of knowledge-based applications including distributed programming ([14], [16], [17]). However, as graph analysis becomes very important, it is necessary to extend Datalog ([18], [19]) to allow new types of queries, such as those involving recursive aggregation and convergence testing. For instance, the last part of query Q in Table I takes relation $\mathsf{Connexion}$ and compute the average travel time and average carbon foot print of very two points in the city as follows. It groups $\mathsf{Connexion}$ lines that coincide in their origin and destination (group-by function on attributes X_{to} and X_{from}) and for each group computes the average (AVG) over their attributes X_{time} and X_{cfp}.

Declarative powerful query languages are possible only with improvements on the lower level, responsible for implementing queries on distributed data. Many existing tools can only deal with conjunctive queries or queries slightly more powerful (*i.e.,* large number of restrictions are imposed, *e.g.* queries with just one sub-query, regular queries, among others). We consider these problems as possible extensions of our work in [20]. Solutions and limits of the well-known MapReduce programming model have been discussed (for instance, in [21], [22]).

We are working on the definition of a declarative language (similar to the language proposed in [15]) to support conjunctive context-free queries ([23]). The run-time support for this language is being designed in the form of a query engine that uses an adapted LL(1) algorithm ([24]) in order to look for non-regular paths in the graph database.

We believe that the challenge in proposing new query processing is not only in the association of declarative power and efficient implementation, but also in ensuring data quality. Indeed, in modern scenarios, constraint verification is generally neglected due to their cost; the lack of data quality (Veracity) and the rapidity of data changes (Velocity) being difficult obstacles to overcome. But constraints are the expression of the desired *answer quality* or of a *fixed (or personalised) context.* They settle the validity requirements a user is looking for. Adapting query languages to interact with constraint checkers is the new challenge for those, as us, worried about data quality preservation. In Table I, constraints $\mathcal{C} = \{c_1, c_2\}$ illustrate a context personalization. Our user is only interested in means of transport using railways (c_1). Moreover, he wants paths composed only by edges associated to a carbon footprint less then $200g$ of CO_2 (Table I, denial constraint c_2). Our proposal is to filter answers, rendering to the user only those answers respecting his constraints.

III. Constraints and Updates

The maintenance of data consistency is an essential problem in databases. Consistency maintenance is a fundamental challenge and the dichotomy of database and ontology constraints deserves attention. The so-called ontological constraints ([25]) correspond to inference rules. As an example, consider a constraint stating that all subscribers of the *Autolib* service[1] must have a driving licence. In a traditional database approach, an instance I containing a single fact $\mathsf{Subscribe}(Bob, Autolib)$ is *not* consistent because the constraint requires the existence of a second fact $\mathsf{inCategory}(Bob, LicensedDriver)$. On the other hand, I is consistent when constraints are seen as inference rules because the second is an implicit fact that can be derived from I with the rule. We distinguish constraints from inference rules, envisaging systems that consider both types, although not necessarily on the same level ([15]).

Consider now the problem of updates under the traditional database consistency approach. A database instance is *valid* or *consistent w.r.t.* to a set of constraints $\mathcal{C}$ when all constraints in $\mathcal{C}$ are satisfied by the instance. Whenever the database is updated, these constraints must be checked. If violations are detected, either the update is refused or compensation actions, called side-effects, must be executed in order to guarantee their satisfaction. Usually, there are different sets of side-effects that repair an inconsistent database. The choice is often based on the minimal change criteria: obtain a consistent database that differs from the original inconsistent one in a minimal way.

Constraints in graph databases can be described by *Tuple-Generating Dependencies* (TGDs) and *Denial Dependencies* ([11]), thus incorporating a large body of research on the subject, that was originally developed for the relational database model.

In this context, we are considering two challenges: determinism of updates and chasing graph TGDs.

Determinism of updates. The determinism of updates is an old problem. Should we give the choice to the user at each step? An instance containing the facts $\{\mathsf{User}(Ann)$, $\mathsf{inCategory}(Ann, Student)$, $\mathsf{hasReduction}$ $(Ann,$ $30\%)\}$ satisfies the constraint $\mathsf{User}(X)$, $\mathsf{inCategory}(X, Student) \rightarrow \mathsf{hasReduction}(X, 30\%)\}$ establishing a reduced price condition in Parisian transports. The deletion of $\mathsf{hasReduction}(Ann, 30\%)$ illustrates the non-determinism of the update, since it implies the deletion of either $\mathsf{User}(Ann)$ or $\mathsf{inCategory}(Ann, Student)$ (or both). Although this problem has been the subject of previous work, such as [26], [27], the same questions appear in the context of RDF databases.

[1]Parisien self-service electric cars. https://www.autolib.eu/en/

Chasing Graph TGDs. Commonly used key and foreign-key constraints are not enough for most of RDF applications, and thus we are impelled to consider richer constraints such as tuple-generating (TGD) or denial dependencies as constraints. However, the use of these richer constraints usually renders more difficulty to develop a deterministic update strategy.

Existential variables in TGD constraints give an important flexibility for dealing with RDF data. For instance $\mathsf{Subscribe}(X, Autolib) \rightarrow \exists Y.\mathsf{hasDriverLic}(X, Y)$ imposes the obligation of having a driver licence (identified by its number Y) to users wanting to subscribe to the Autolib service. However, the insertion of $\mathsf{Subscribe}(Bob, Autolib)$ generates as side effects the insertion of $\mathsf{hasDrivingLic}(Bob, N_1)$ where N_1 is a *null* value. The choice of storing or not null values in the database depends on the application. Notice however that the insertion or deletion of facts with null values introduce non-determinism in the update strategy.

On one hand, accepting null information during updates requires dealing with complicate constraint iterations. For instance, consider constraints declaring free of charge means of transport during air pollution alerts: $\mathsf{PollutionAlert}(X_{site}, X_{polluant}, X_{level}) \rightarrow \exists Z.\mathsf{declareFreeTransp}(X_{site}, Z)$ and $\mathsf{declareFreeTransp}(X_{site}, Z) \rightarrow Ecolabel(Z)$. Do all side effects containing null variables should be stored? Solutions for these situations are clearly related to the chase technique ([28]). Intensively studied as a theoretical tool (*e.g.* as in [29], [30], [31]), the chase is now currently used in a number of data management tasks such as data exchange, query answering under constraints or data cleaning (some examples in [32], [33] and a benchmark proposal in [34]). We believe that custom versions of the chase allow improvement in updating RDF data sets ([35]).

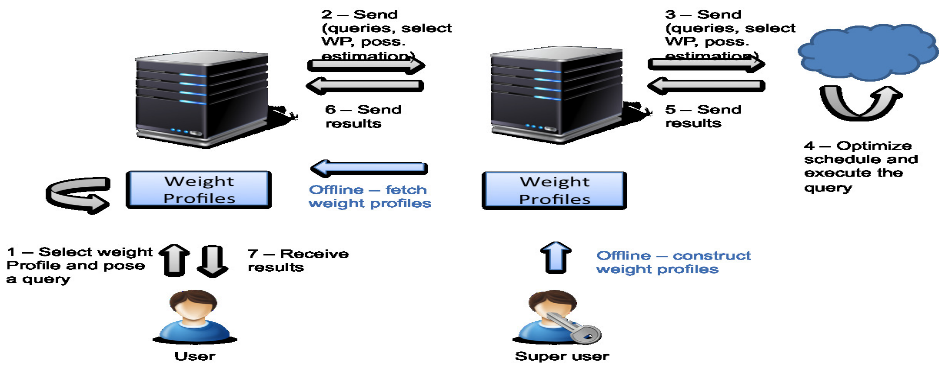
On the other hand, the null value as a place holder can be helpful to stop iterations, during deletions. For example, let $\{\mathsf{touristAttraction}(Louvre), \mathsf{servedByStation}(Louvre, PalaisRoyal)\}$ be the database instance satisfying the constraint $\mathsf{touristAttraction}(X) \rightarrow \exists Y.\mathsf{servedByStation}(X, Y)$. Deleting the fact $\mathsf{servedByStation}(Louvre, PalaisRoyal)$ may just correspond to the insertion of $\mathsf{servedByStation}(Louvre, N_2)$ avoiding all the backward chaining of constraint rules. Our proposal in [36] deals with this aspect.

When discussing updates, questions concerning Belief Revision ([37], [38]) usually appear; the revision viewpoint being popular when dealing with ontologies ([39] as an

Rewritten query on a transportation database

Connexion$(X_{from}, X_{to}, X_{time}, X_{cfp})$	$\leftarrow$	Transp$(X_{from}, X_{to}, X_{means}, X_{time}, X_{cfp})$, Type$(X_{means}, Rail)$, $\neg(X_{cfp} > 200)$
Connexion$(X_{from}, X_{to}, X_{time}, X_{cfp})$	$\leftarrow$	Transp$(X_{from}, Z, Z_{means}, Z_{time}, Z_{cfp})$, Type$(Z_{means}, Rail)$ Connexion$(Z, X_{to}, Y_{time}, Y_{cfp})$, $\neg(Z_{cfp} > 200)$ $X_{time} = Z_{time} + Y_{time}, \quad X_{cfp} = Z_{cfp} + Y_{cfp}$,
Q$(X_{from}, X_{to}, a_{totalAvTime}, a_{Avcfp})$	$\leftarrow$	$agg($Connexion$(X_{from}, X_{to}, X_{time}, X_{cfp}), X_{from}, X_{to}$, $a_{totalAvTime} = AVG(X_{time}), \quad a_{Avcfp} = AVG(X_{cfp}))$

Table II: Query of Table I rewritten to take into account the user's context.



2 – Send (queries, select WP, poss. estimation)
6 – Send results
3 – Send (queries, select WP, poss. estimation)
5 – Send results
4 – Optimize schedule and execute the query
Weight Profiles
Offline – fetch weight profiles
Weight Profiles
1 – Select weight Profile and pose a query
7 – Receive results
Offline – construct weight profiles
User
Super user

Figure 2: Human in the loop interaction.

example). We consider updates as changes in the world; and not as a revision of an initially incomplete knowledge of this world. However, when a model does not include any explicit representation of time it might be interesting to relate update operators to the core principles of Belief Revision. In our opinion, principles guiding update delete (contraction) and insert (revision) operations should match closure, success, inclusion, consistency and vacuity.

IV. Multi-Objective Optimization

We are interested in developing algorithms and techniques for improving the quality of services in urban environments, such as mobility and traffic management. Services of smart cities have increasing demands for better quality, low cost and security. In particular, for urban mobility these requirements pose challenges in the design of products and processes that satisfy all performance goals. Multi-objective Bio-inspired algorithms (MOBAs) ([40]) have been used in several applications and are powerful alternatives to deal with these problems.

MOBAs have been used for controlling LED panels in the city of Quito ([41]) in order to guide drivers and reduce traffic jams. Similar techniques can be used for optimizing public transportation routes and traffic lights coordination, in order to provide sustainable mobility.

In order to provide fast responses to the plethora of applications and users accessing available urban information, the data management system should be able to execute queries on large volumes of data, often stored in distributed servers or in the cloud. Both data constraints and MOBAs can be considered for query optimization.

A. *Query Optimization Based on Constraints*

Semantic query optimization is based on query rewriting, where the equivalence of the resulting queries is defined based on user-defined constraints ([42]) and types ([43]). The chase mechanism used in the optimization is the same we have been investigating to tackle the problem of database consistency under updates. Thus, results for chasing graph data can also be applied for query optimization. Moreover, query re-writing techniques can be used to incorporate user-defined constraints into the body of the query. For instance, Table II shows a rewritten version of the query in Table I where constraints defining the user context are incorporated in the query. Here, challenges include dealing with more sophisticated constraints and overcoming the limits of query evaluation systems for big data, as already mentioned in Section II.

Another important aspect concerns planning the query evaluation. The problem of choosing a plan depends on the ability to estimate the query cost. While the problem of estimating query processing is well known in database management systems ([44]), the cloud computing context makes it more difficult to address. Indeed, optimization must not only consider basic parameters such as response time and data transfers, but must also take into account specific aspects, especially money to be spent with respect to the pay-as-you-go model. The problem is even more challenging due to the variance of the clouds (period of time, type of VM for instance). We address this problem by exploring continuous optimization ([45]), extending current proposal in order to consider multi objectives.

B. Human-in-the-loop Query Optimization

Query optimization in the cloud has to take into consideration several (often contradictory) objectives. That is why multi-objective query processing has been a hot topic recently. Recent contributions have addressed optimization of query processing in the cloud ([46]) or parallelization of the optimization itself ([47]). Nevertheless, to the best of our knowledge existing solutions do not really consider specificities of graph data and they are based on a "Skyline" interaction model, in which the user/administrator is in charge of making a decision visualizing a (potentially large) research space. In addition to address graph databases, our vision is to provide a "human in the loop" approach, as illustrated in Figure 2, with different users (administrators, end-users) making it possible to define some rules offline, which will be used online to reduce the search space.

V. Conclusion

The development of efficient tools for querying and evolving data graphs under different kinds of constraints is *essential* for urban computing. This paper presents the authors' views on some challenges and research directions in this domain.

Acknowledgments

This work is partially funded by EU-BR EUBra-BigSea Project (MCTI/RNP 3rd Coordinated Call), APR-IA GIRAFON and INES, grant CNPq/465614/2014-0.

References

[1] C. Aoun, "The Smart City Cornerstone: Urban Efficiency," in *Schneider Electric White Paper*, 2013.

[2] "Sbc," At http://www.sbc.org.br/documentos-da-sbc/send/141-grandes-desafios/802-grandesdesafiosdacomputaonobrasil.

[3] "Mobility and transport," At https://ec.europa.eu/transport/themes/urban/urban_mobility_en.

[4] R. Minetto, N. P. Kozievitch, R. D. da Silva, L. D. A. Almeida, and J. de Sant, "Shortcut suggestion based on collaborative user feedback for suitable wheelchair route planning," in *Proc. of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 2372–2377.

[5] V. Dhar, "Data science and prediction," *Commun. ACM*, vol. 56, no. 12, pp. 64–73, Dec. 2013. [Online]. Available: http://doi.acm.org/10.1145/2500499

[6] "PRIMER RDF 1.1 primer," 2014. [Online]. Available: https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140225/

[7] "SPARQL 1.1 query language," 2012. [Online]. Available: https://www.w3.org/TR/2012/PR-sparql11-query-20121108/

[8] Apache TinkerPop, "Tinkerpop3," At http://tinkerpop.apache.org/, 2016.

[9] The Neo4j Team, "The neo4j manual v3.0," At http://neo4j.com/docs/stable/, 2016.

[10] R. Angles, M. Arenas, P. Barceló, A. Hogan, J. L. Reutter, and D. Vrgoc, "Foundations of modern graph query languages," *CoRR*, vol. abs/1610.06264, 2016. [Online]. Available: http://arxiv.org/abs/1610.06264

[11] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*. Addison-Wesley, 1995. [Online]. Available: https://books.google.com.br/books?id=HN9QAAAAMAAJ

[12] Z. Sun, H. Wang, H. Wang, B. Shao, and J. Li, "Efficient subgraph matching on billion node graphs," *Proc. VLDB Endow.*, vol. 5, no. 9, pp. 788–799, May 2012. [Online]. Available: http://dx.doi.org/10.14778/2311906.2311907

[13] D. Mizell, K. J. Maschhoff, and S. P. Reinhardt, "Extending SPARQL with graph functions," in *2014 IEEE International Conference on Big Data*, 2014, pp. 46–53.

[14] C. R. Aberger, S. Tu, K. Olukotun, and C. Ré, "Emptyheaded: A relational engine for graph processing," in *Proceedings of the International Conference on Management of Data, SIGMOD Conference 2016, USA, June 26 - July 01, 2016*, 2016, pp. 431–446.

[15] M. Bamha, J. Chabin, M. Halfeld-Ferrari, B. Markhoff, and T. B. Nguyen, "Personalized environment for querying semantic knowledge graphs: a mapreduce solution," submitted.

[16] J. Seo, J. Park, J. Shin, and M. S. Lam, "Distributed socialite: A datalog-based language for large-scale graph analysis," *PVLDB*, vol. 6, no. 14, pp. 1906–1917, 2013.

[17] A. Shkapsky, M. Yang, M. Interlandi, H. Chiu, T. Condie, and C. Zaniolo, "Big data analytics with datalog queries on spark," in *Proceedings of the International Conference on Management of Data, SIGMOD Conference, USA*, 2016, pp. 1135–1149.

[18] M. Mazuran, E. Serra, and C. Zaniolo, "Extending the power of datalog recursion," *VLDB J.*, vol. 22, no. 4, pp. 471–493, 2013.

[19] J. Seo, S. Guo, and M. S. Lam, "Socialite: Datalog extensions for efficient social network analysis," in *29th IEEE International Conference on Data Engineering, ICDE, Australia*, 2013, pp. 278–289.

[20] R. Penteado, R. Schroeder, and C. S. Hara, "Exploring controlled RDF distribution," in *8th IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2016.

[21] J. M. Giménez-Gárcia and M. A. M. Javier D. Fernández, "Mapreduce-based solutions for scalable SPARQL querying," *Open Journal of Semantic Web*, vol. 1, no. 1, pp. 1–18, 2014.

[22] Z. Kaoudi and I. Manolescu, "RDF in the clouds: a survey," *VLDB J.*, vol. 24, no. 1, pp. 67–91, 2015.

[23] J. Hellings, "Conjunctive context-free path queries," in *Proc. 17th International Conference on Database Theory (ICDT), Athens, Greece, March 24-28, 2014*, N. Schweikardt, V. Christophides, and V. Leroy, Eds. OpenProceedings.org, 2014, pp. 119–130.

[24] A. Aho, M. Lam, R. Sethi, and J. Ullman, *Compilers: Principles, Techniques, and Tools*, ser. Alternative eText Formats Series. ADDISON WESLEY Publishing Company Incorporated, 2007. [Online]. Available: https://books.google.com.br/books?id=WomBPgAACAAJ

[25] G. Gottlob, G. Orsi, and A. Pieris, "Ontological queries: Rewriting and optimization," in *Proceedings of the 27th International Conference on Data Engineering, ICDE, Germany*, 2011, pp. 2–13.

[26] F. Bry, "Intensional updates: Abduction via deduction," in *International Symposium on Logic Programming*, 1990.

[27] R. Fagin, J. Ullman, and M. Y. Vardi, "On the semantics of updates in databases," in *PODS*. ACM Press, 1983.

[28] D. Maier, A. O. Mendelzon, and Y. Sagiv, "Testing implications of data dependencies," *ACM Trans. Database Syst.*, vol. 4, no. 4, pp. 455–469, 1979.

[29] A. Deutsch, A. Nash, and J. B. Remmel, "The chase revisited," in *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS, Canada*, 2008, pp. 149–158.

[30] G. Grahne and A. Onet, "On conditional chase termination," in *Proceedings of the 5th Alberto Mendelzon International Workshop on Foundations of Data Management, Santiago, Chile, May 9-12, 2011*, 2011.

[31] A. Onet, "The chase procedure and its applications in data exchange," in *Data Exchange, Integration, and Streams*, 2013, pp. 1–37.

[32] A. Bonifati, I. Ileana, and M. Linardi, "Functional dependencies unleashed for scalable data exchange," in *Proceedings of the 28th International Conference on Scientific and Statistical Database Management, SSDBM, Hungary, 2016*, 2016, pp. 2:1–2:12.

[33] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa, "Data exchange: semantics and query answering," *Theor. Comput. Sci.*, vol. 336, no. 1, pp. 89–124, 2005.

[34] M. Benedikt, G. Konstantinidis, and G. M. et al., "Benchmarking the chase," to appear in Principles of Database Systems (PODS 2017).

[35] M. Halfeld Ferrari and D. Laurent, "Updating RDF/S databases under negative and tuple-generating constraints," LIFO- Université d'Orléans, Tech. Rep., 2017. [Online]. Available: https://www.univ-orleans.fr/lifo/rapports.php?lang=fr&sub=sub3.

[36] M. Halfeld-Ferrari, C. S. Hara, and F. Uber, "RDF updates with constraints," submitted.

[37] H. Katsuno and A. O. Mendelzon, "On the difference between updating a knowledge base and revising it," in *Proc. of the 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'91). Cambridge, MA, USA, April 22-25.*, 1991, pp. 387–394.

[38] "Logic of belief revision," https://plato.stanford.edu/entries/logic-belief-revision/.

[39] N. Nikitina, S. Rudolph, and B. Glimm, "Interactive ontology revision," *J. Web Sem.*, vol. 12, pp. 118–130, 2012.

[40] S. Binitha and S. S. Sathya, "A survey of bio inspired optimization algorithms," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 2, no. 2, May 2012.

[41] D. H. Stolfi, R. Armas, E. Alba, H. Aguirre, and K. Tanaka, "Fine tuning of traffic in our cities with smart panels: The quito city case study," in *GECCO '16: Proceedings of the Genetic and Evolutionary Computation Conference*, 2016, pp. 1013–1019.

[42] A. Deutsch, L. Popa, and V. Tannen, "Query reformulation with constraints," *SIGMOD Rec.*, vol. 35, no. 1, pp. 65–73, Mar. 2006.

[43] M. Meier, M. Schmidt, F. Wei, and G. Lausen, "Semantic query optimization in the presence of types," in *PODS 2010*, 2010, pp. 111–122.

[44] S. Chaudhuri, "An Overview of Query Optimization in Relational Systems," in *Proceedings of the Symposium on Principles of Database Systems (PODS)*, Seattle, Washington, USA, 1998, pp. 34–43.

[45] N. Bruno, S. Jain, and J. Zhou, "Continuous Cloud-Scale Query Optimization and Processing," *Proceedings of the Very Large Data Bases Endowment*, vol. 6, no. 11, pp. 961–972, 2013.

[46] I. Trummer and C. Koch, "An Incremental Anytime Algorithm for Multi-Objective Query Optimization," in *Proceedings of the SIGMOD International Conference on Management of Data*, Melbourne, Victoria, Australia, 2015, pp. 1941–1953.

[47] ——, "Parallelizing Query Optimization on Shared-Nothing Architectures," *Proceedings of the Very Large Data Bases Endowment (PVLDB)*, vol. 9, no. 9, pp. 660–671, 2016.