Semiring Provenance over Graph Databases

Yann Ramusat, Silviu Maniu, Pierre Senellart

Semiring Provenance over Graph Databases

Yann Ramusat
ENS, PSL University
Paris, France
yann.ramusat@ens.fr

Silviu Maniu
LRI, Université Paris-Sud, Université Paris-Saclay
Orsay, France
silviu.maniu@lri.fr

Pierre Senellart
DI ENS, ENS, CNRS, PSL University & Inria Paris & LTCI, Télécom ParisTech
Paris, France
pierre@senellart.com

Abstract

We generalize three existing graph algorithms to compute the provenance of regular path queries over graph databases, in the framework of provenance semirings – algebraic structures that can capture different forms of provenance. Each algorithm yields a different trade-off between time complexity and generality, as each requires different properties over the semiring. Together, these algorithms cover a large class of semirings used for provenance (top-k, security, etc.). Experimental results suggest these approaches are complementary and practical for various kinds of provenance indications, even on a relatively large transport network.

1 Introduction

Graph databases gained a lot of traction due, in part, to their applications to social network analysis [5] or to the semantic web [1]. Graph databases can be queried using several general-purpose navigational query languages and especially *regular path queries* (RPQs) [2]. The *provenance* of a query result is an annotation providing additional meta-information [11], e.g., explaining how or why the answer was produced. There are various useful types of provenance annotations, however calculations performed for different types of provenance are strikingly similar. In a large part, this genericity can be captured using algebraic structures called provenance semirings [6].

In this paper we generalize three existing graph algorithms in order to compute the semiring provenance of RPQs over graph databases. Each of these algorithms corresponds to a trade-off between time complexity and generality, and each needs the semirings to have some additional properties. Implementations of the aforementioned algorithms and experimental results conducted strongly suggest these approaches are complementary and remain practical even for a relatively large transport network and various kinds of provenance semirings. This contribution simplifies the way query answering can be done in a production enrironment. Query answering in different semantics can be done simply by changing the annotation semiring, making the approach generic and still strongly efficient.

TaPP 2018, July 11 - 12, 2018, London, UK

2 Preliminaries

Semirings Following [6], the framework for provenance we consider is based on the algebraic structure of semirings. Here, we only use basics of the algebraic theory of semirings; for further studies about theory and applications, see [7].

A *monoid* is a system $(\mathbb{K}, \oplus, \bar{0})$ such that $\oplus$ is associative and $\bar{0}$ is the identity element. A *semiring* is a system $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ such that: $(\mathbb{K}, \oplus, \bar{0})$ is a commutative monoid, $(\mathbb{K}, \otimes, \bar{1})$ is a monoid, $\otimes$ distributes over $\oplus$, and $\bar{0}$ is an annihilator for $\otimes$.

We are interested in specific classes of semirings, in particular *idempotent*, *k-closed*, and *star* semirings.

Definition 1. *Let $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ be a semiring. An element $a \in \mathbb{K}$ is* idempotent *if $a \oplus a = a$. $\mathbb{K}$ is said to be* idempotent *when all elements of $\mathbb{K}$ are idempotent.*

We now introduce *k-closed semirings*:

Definition 2. *Given $k \geqslant 0$, a semiring $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ is said to be* k-closed *if* $\forall a \in \mathbb{K}, \bigoplus_{n=0}^{k+1} a^n = \bigoplus_{n=0}^{k} a^n$.

Remark 3. *Any 0-closed semiring is idempotent, since, for all a: $a = a \otimes \bar{1} = a \otimes (\bar{1} \oplus \bar{1}) = a \oplus a$.*

The following property of k-closed semirings is standard:

Proposition 4. *Let $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ be a k-closed semiring, then for any integer $l > k$, $\forall a \in \mathbb{K}, \bigoplus_{n=0}^{l} a^n = \bigoplus_{n=0}^{k} a^n$.*

We will also use more complex semirings, called *star semirings* [9]. Some of the algorithms we propose will be able to handle this kind of semirings.

Definition 5. *A* star semiring *is a semiring $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ with an additional unary operator * verifying:*

$$\forall a \in \mathbb{K}, \quad a^* = \bar{1} \oplus (a \otimes a^*) = \bar{1} \oplus (a^* \otimes a).$$

For convenience we define $a^* := \bigoplus_{i=0}^{k} a^i$ for every a in a k-closed semiring. It is easy to verify (using Proposition 4) that any k-closed semiring is in fact a star semiring with this definition of the star operator.

Given an idempotent semiring $\mathbb{K}$, one can define a specific partial order on $\mathbb{K}$ [10]:

Lemma 6. *Let $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ be an idempotent semiring, then the relation $\leqslant_{\mathbb{K}}$ defined by $(a \leqslant_{\mathbb{K}} b) \Leftrightarrow (a \oplus b = a)$ defines a partial order over $\mathbb{K}$, called the* natural order *over $\mathbb{K}$.*

For examples of semirings used for provenance, see [6, 11]. Among classical semirings, note that the *tropical* semiring defined by $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ is 0-closed (and hence idempotent), whose *natural order* coincides with the usual order over reals. It can be extended in a straightforward manner into a k-closed semiring by keeping the k minimum alternatives; we refer to this semiring as the *k-tropical* semiring [10]. The *security* semiring of security clearance levels with $\oplus = \min$, $\otimes = \max$ is also 0-closed. The *counting semiring* $(\mathbb{N} \cup \{\infty\}, +, \times, 0, 1)$ is not k-closed for any k but is a star semiring, with $0^* = 1$ and $\forall a \in \mathbb{N}, a^* = \infty$.

Graph databases with provenance indication Let $\mathcal{V}$ be a countably infinite set, whose elements are called *node identifiers* or *node ids*. Given a finite alphabet Σ, a *graph database* G over Σ is a pair (V, E), where V is a finite set of node ids (i.e., $V \subseteq \mathcal{V}$) and $E \subseteq V \times \Sigma \times V$. Thus, each edge in G is a triple $(v, a, v') \in V \times \Sigma \times V$, whose interpretation is an a-labeled edge from v to v' in G. When Σ is clear from the context, we shall speak simply of a graph database.

A *graph database with provenance indication* (V, E, w) *over* $\mathbb{K}$ is a graph database (V, E) together with a *weight function*, $w : E \to \mathbb{K}$ for $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ a semiring.

Given an edge $e \in E$, we denote by $n[e]$ its third component and we call it its *destination* (or next) vertex, by $p[e]$ its first component and we call it its *origin* (or previous vertex), by $w[e]$ its weight, and by $\rho(e)$ its label. Given a vertex $v \in V$, we denote by $E[v]$ the set of edges leaving v. A path $\pi = e_1 e_2 \cdots e_k$ in G is an element of E^* with consecutive edges: $n[e_i] = p[e_{i+1}]$ for $i = 1, \ldots, k-1$. The label $\rho(\pi) \in \Sigma^*$ is defined as the concatenation of its labels: $\rho(\pi) = \rho(e_1)\rho(e_2)\cdots\rho(e_{k-1})\rho(e_k)$.

We extend n and p to paths by setting $p[\pi] := p[e_1]$, and $n[\pi] := n[e_k]$. A cycle is a path starting and ending at the same vertex: $n[c] = p[c]$. The weight function w can also be extended to paths by defining the weight of a path as the result of the $\otimes$-multiplication of the weights of its constituent edges: $w[\pi] := \bigotimes_{i=1}^{k} w[e_i]$ and can in fact be extended to any finite set of paths by $w[\bigcup_{i=1}^{n} \pi_i] := \bigoplus_{i=1}^{n} w[\pi_i]$.

Let $s \in V$ be a fixed vertex of V called the *source*. We denote by $P_{sv}(G)$ the set of paths from s to $v \in V$. By extension, $P_{ij}(G)$ is the set of paths from i to j.

Regular Path Queries (RPQs) RPQs [2] provide a way to query a graph database using its topology and constitute the basic navigational mechanism for graph databases. RPQs have the form $RPQ(x, y) := (x, L, y)$ where L is a regular language, with the semantics that Q is satisfied iff there exists a least a path having the sequence of its labels' edges in L. From now on, we use the shorthand L_Q for the RPQ $Q = (x, L_Q, y)$.

We now introduce the notion of *provenance of an RPQ* based on the notion of provenance semiring for the language Datalog [6]. On a graph database G with provenance indication over $\mathbb{K}$, such a query Q associates to each pair (x, y) of nodes an element $\mathrm{prov}_{\mathbb{K}}^{Q}(G)(x, y)$ of the semiring called the *provenance of the RPQ between x and y* defined as $\mathrm{prov}_{\mathbb{K}}^{Q}(G)(x, y) := \bigoplus_{\substack{\pi \in P_{xy}(G),\\ \rho(\pi) \in L_Q}} w[\pi]$. Note that this is not always well-defined (since the sum may be infinite). Infinite sums can be simplified using the star operator, so, during this paper, we will restrict to star-semirings. Given a vertex s, the *single-source provenance* problem computes the provenance between s and each vertex of the graph.

3 Provenance of an RPQ

Our contribution consists of the computation of the provenance of an RPQ whose language is non-trivial. That is, given a graph database G over $\mathbb{K}$ and an RPQ Q, we compute the function $\mathrm{prov}_{\mathbb{K}}^{Q}(G)$. For this, we suppose we have a complete deterministic automaton $\mathcal{A}_Q$ to represent L_Q.

Graph transformation We reduce the problem of computing the provenance of a query Q over labeled graph G to the problem of computing shortest distances over an unlabeled graph G'. Let $\mathcal{A}_Q = (S, \Gamma, s_0, F)$ where $\Gamma \subseteq S \times \Sigma \times S$ is the transition function, $s_0 \in S$ is the initial state, and $F \subseteq S$ the set of final states. If $(s, a, q') \in \Gamma$, we note $q \xrightarrow{a} q'$. In order to take into account path restrictions we will transform the graph G by taking the *product graph* $P_{G \times \mathcal{A}_Q}$ between it and the automaton $\mathcal{A}_Q$. The product graph is defined as $P_{G \times \mathcal{A}_Q} = (V \times S, E')$ over $\Sigma' = \{\star\}$, where

$$E' = \{((v, s), \star, (v', s')) \mid \exists a \in \Sigma, (v, a, v') \in E \wedge s \xrightarrow{a} s'\},$$

and the weight function, $w' : E' \to \mathbb{K}$:

$$e = ((v, s), \star, (v', s')) \mapsto \bigoplus_{\substack{a \in \Sigma,\\ s \xrightarrow{a} s'}} w[(v, a, v')].$$

We now show that we can compute the provenance of the query Q, between x and y, in the graph G by computing

the provenance of the reachability query R (the query with underlying language $L_R = \Sigma^*$) over $P_{G\times\mathcal{A}_Q}$ between all pairs having source (x, s_0) and target (y, s_F) for $s_F \in F$ and performing a last step consisting in summing each of them. The next lemma characterizes the kind of paths from the source in $P_{G\times\mathcal{A}_Q}$.

Lemma 7. *If $\mathcal{A}_Q$ is a complete and deterministic automaton, there is a one-to-one mapping c between paths from x in G onto paths from (x, s_0) in $P_{G\times\mathcal{A}_Q}$. Moreover for π with $p[\pi] = x$ and $n[\pi] = y$ we have $n'[c(\pi)] = (y, s)$ with $s_0 \xrightarrow{\rho(\pi)} s$ and $w[\pi] = w'[c(\pi)]$ where n' is the destination for edges in $P_{G\times\mathcal{A}_Q}$ and w' the weight function of this graph.*

Using the above lemma, we can show:

Theorem 8. *The following equality holds:*

$$\text{prov}^Q_{\mathbb{K}}(G)(x, y) = \bigoplus_{s_F\in F} \text{prov}^R_{\mathbb{K}}(P_{G\times\mathcal{A}_Q})((x, s_0), (y, s_F)).$$

Having reduced provenance computation of an RPQ to the provenance computation of the reachability query, we now revisit three classical graph algorithms and apply them to the provenance of reachability: Mohri's algorithm [10], the node elimination algorithm [3], and Dijksta's algorithm [4].

Mohri's algorithm Mohri's algorithm [10] is an approach working over arbitrary k-closed semirings to compute the provenance of the reachability query.

To compute the provenance of a query Q over a graph G with provenance indication over a k-closed semiring, we have just showed we can reduce to the problem of computing the provenance of the reachability query R over the product graph of G with the automaton representing L_Q.

We denote by n the number of states of the automaton. The size of the new graph is bounded by n times the size of the original one. The theoretical complexity of this approach is exponential in the size of the product graph [10]. This is due to the fact the number of simple paths may be exponential in the size of the graph ($|V| + |E|$). But, in practice, as discussed further, experiments exhibit a linear-time behavior in k and the size of the product graph, i.e., proportional to $n \cdot |E| \cdot k$.

Node elimination algorithm Though the framework proposed by Mohri is very general, the theoretical complexity is exponential and the framework does not allow, for example, to count the number of paths between two nodes, since the counting semiring used for this is not k-closed.

While still using the product graph we propose here a new approach inspired by the construction of Brzozowski and McCluskey [3] for obtaining the equivalent language of an automaton, to design an algorithm being able to handle star semirings. Such an algorithm works for every star semiring. Intuitively, this algorithm is very similar to the classical Floyd–Warshall algorithm except that we are interested in only one pair. This algorithm can compute provenance for many pairs if we consider also new sources and targets for each vertex belonging to a pair we are interested in. Each elimination step preserves the provenance between s' and t'.

The complexity highly depends on the heuristic used in the elimination order. For an upper bound, we can consider the worst case where each neighborhood is in $O(|V|)$. Computing only one time the star for each vertex we can obtain a complexity in $O(|V|T_* + |V|^3(T_\oplus + T_\otimes))$ where $T_.$ is the time taken by an elementary operation.

Dijkstra's algorithm Still using the product graph, we propose a generic framework where the very well-known Dijkstra's algorithm [4] can be successfully applied to compute semiring-based provenance.

For this algorithm, we will restrict to 0-closed semirings with $\leqslant_{\mathbb{K}}$ being a total order, which includes for instance the tropical and security semirings. Höfner and Möller already gave an equivalent result in [8]. We present here our proof for the framework we are interested in, using only basic semiring theory and being more compact.

Dijkstra's algorithm maintains a frontier between nodes already seen (P) and unknown nodes. Every path from s to an unknown node goes through an edge, $e \in P \times \bar{P}$. If $\forall c, a, c \oplus (c \otimes a) = c$ then an optimal path between s and an unknown node never uses edges belonging to $\bar{P} \times \bar{P}$. This is precisely the case in a 0-closed semiring ($c \leqslant_{\mathbb{K}} c \otimes a$). The algorithm works by choosing at each step a new vertex to add to P. We just need to be able to compare every path (and thus require $\leqslant_{\mathbb{K}}$ to be total) to choose the next vertex to visit.

Theorem 9. *If the semiring of provenance is 0-closed and $\leqslant_{\mathbb{K}}$ is a total order then a generalization of Dikjkstra's algorithm computes the single-source provenance of the reachability query* R.

The overall combined complexity of maintaining provenance annotations is in $O(|E|(\mathrm{T}_\oplus + \mathrm{T}_\otimes))$. In addition, using a *Fibonacci heap* to maintain the priority queue, it is possible to get a complexity of $O(\mathrm{T}_\oplus \log |V|)$ to extract the minimal element at each iteration of the algorithm, yielding an overall complexity of $O(|E|(\mathrm{T}_\oplus + \mathrm{T}_\otimes) + \mathrm{T}_\oplus |V| \log |V|)$. This theoretical complexity is low compared to the other algorithms, but total order 0-closedness is a strong assumption.

4 Experimental results

We performed experiments on the public transport network made freely available by the STIF organization[1]. This dataset gathers every kind of public transit within the Paris region.

[1] https://opendata.stif.info/explore/dataset/offre-horaires-tc-gtfs-idf/

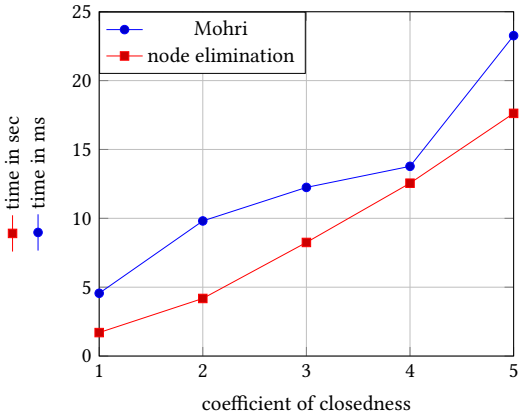

Figure 1. Time to compute single-source provenance with Mohri (blue, milliseconds) and node elimination (red, seconds) depending on the coefficient of closedness.

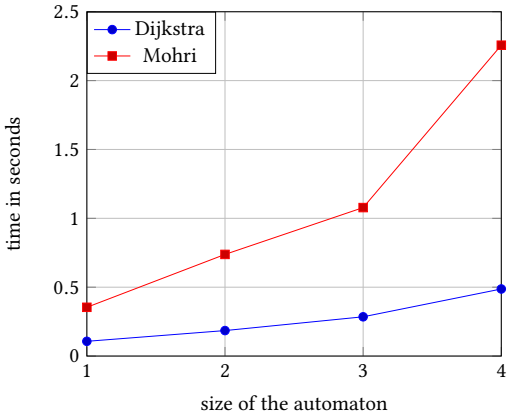

Figure 2. Time (in s) to compute single-source provenance with Mohri (red) and Dijkstra (blue) depending on the size of the product graph (the numbers in the x-axis represent the size of the automaton encoding the query).

We have the data for trains, buses, subways, and trams. Once processed, the graph contains 16,369 nodes, 41,448 edges and labels are line names. We have also extracted a subgraph limited to the subway lines; this is because the whole graph was too large for the node elimination algorithm. This graph contains 302 nodes and 705 edges.

We now study the efficiency of Mohri's algorithm in the STIF network or part of this graph. We first experimented on the complexity of the query (resulting in different sizes of the product graph) and k the coefficient of closedness for the semiring considered (for the k-tropical semiring, which returns the k best shortest distances). The three considered queries restrict the length of the paths to be divisible by 2, 3, and 4, to obtain different size of the product graph. The automaton used to count modulo n needs n states. For the experiments, we have chosen nodes at shortest-distance 10 and computed the average time to complete over 3 launches. Results show that for large graphs the dominant factor is linear in the size of the product graph and in k. Note the size of the graph used by Mohri's algorithm is proportional to the product of the size of the automaton giving the restriction and that of the initial transport network.

We now compare Mohri's algorithm and the node elimination one. We use an ad-hoc provenance semiring that returns a description of the k best paths (length and list of edges) for $k \in \{1, 2, \ldots, 5\}$. We have chosen nodes at shortest-distance 1 and computed the average time to complete for 3 launches. Figure 1 represents the time vs the coefficient of closedness over the graph of subway lines. We observe Mohri's is approximately 1000 times faster and both are linear in k.

The two main semirings that can be handled by both Mohri's and Dijkstra's algorithms are the tropical semiring and the security semiring. Figure 2 represents the time to complete depending on the size of the graph for the security semiring. For this purpose we used a graph with random security levels (as integers between 0 and 1000) over edges. We have chosen nodes at shortest-distance 10 and computed the average time to complete for 3 launches. The implementation of Dijkstra is three times faster than Mohri. Note we used a Fibonacci heap, in its implementation in the Boost C++ library.

5 Conclusion

Our initial results show that theoretically exponential algorithms such as Mohri's can behave better in practice than expected, and hence can be used in some practical settings. Our further work aims to go deeper in both the theoretical analysis (by showing which semirings works for which algorithm) and in practical implementations.

References

[1] Marcelo Arenas and Jorge Pérez. 2011. Querying semantic web data with SPARQL. In *PODS*. ACM, New York, 305–316.
[2] P. Barceló. 2013. Querying Graph Databases. In *PODS*. ACM, New York, 175–188.
[3] J. A. Brzozowski and E. J. McCluskey. 1963. Signal Flow Graph Techniques for Sequential Circuit State Diagrams. *IEEE Trans. Electr. Comp.* EC-12, 2 (1963), 67–76.
[4] Edsger W Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* 1, 1 (1959), 269–271.
[5] Pedro Domingos and Matthew Richardson. 2001. Mining the network value of customers. In *KDD*. ACM, New York, 57–66.
[6] T. J. Green, G. Karvounarakis, and V. Tannen. 2007. Provenance Semirings. In *PODS*. ACM, New York, 31–40.
[7] U. Hebisch and H. J. Weinert. 1998. *Semirings: Algebraic Theory and Applications in Computer Science*. World Scientific, Singapore.
[8] Peter Höfner and Bernhard Möller. 2012. Dijkstra, Floyd and Warshall meet Kleene. *Formal Aspects of Computing* 24, 4-6 (July 2012), 459–476. https://doi.org/10.1007/s00165-012-0245-4
[9] Droste M., Kuich W., and Vogler H. 2009. *Handbook of Weighted Automata*. Springer, Berlin.
[10] M. Mohri. 2002. Semiring Frameworks and Algorithms for Shortest-distance Problems. *J. Autom. Lang. Comb.* 7, 3 (2002), 321–350.
[11] Pierre Senellart. 2017. Provenance and Probabilities in Relational Databases: From Theory to Practice. *SIGMOD Record* 46, 4 (2017).