Semiring Provenance over Graph Databases

TaPP - King's College London

July 12, 2018

Yann Ramusat

Joint work with
Silviu Maniu, Pierre Senellart

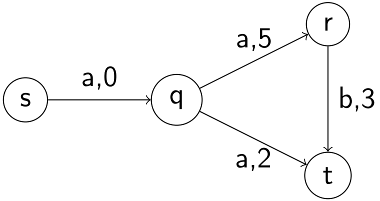
École normale supérieure

Inria

LRI

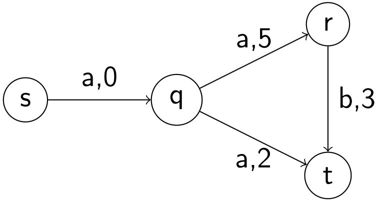
Contents

Working example



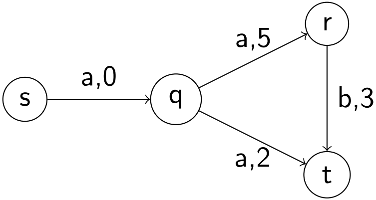
What if these integers represent time to move between two vertices and we want the minimum travel time between s and t?

Working example



What if these integers represent time to move between two vertices and we want the minimum travel time between s and t? 2

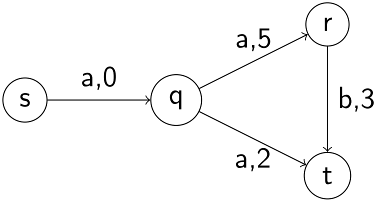
Working example



What if these integers represent time to move between two vertices and we want the minimum travel time between s and t? 2

And now if we only want to consider paths going through a b edge?

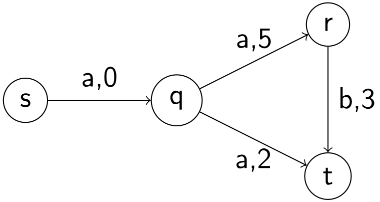
Working example



What if these integers represent time to move between two vertices and we want the minimum travel time between s and t? 2

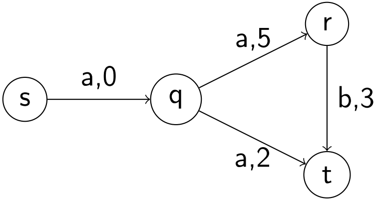
And now if we only want to consider paths going through a b edge? 8

Working example



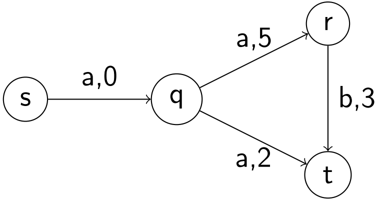
And if these integers represent security restrictions? What is the minimum security clearance needed to go from s to t?

Working example



And if these integers represent security restrictions? What is the minimum security clearance needed to go from s to t? 2

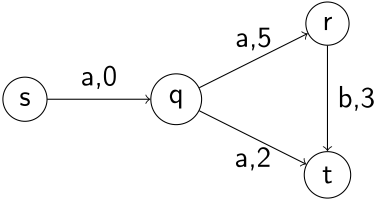
Working example



And if these integers represent security restrictions? What is the minimum security clearance needed to go from s to t? 2

Considering only paths avoiding b edges?

Working example



And if these integers represent security restrictions? What is the minimum security clearance needed to go from s to t? 2

Considering only paths avoiding b edges? 2

Overview

We propose several algorithms able to solve:

1. routing information when probabilistic information is present (e.g., road closures, uncertain travel time);
2. top-k relevant paths; or
3. accessibility under security restrictions;

by computing semiring-based provenance of graph queries.

Each of these algorithms yields a tradeoff between generality and time complexity.

Overview

We propose several algorithms able to solve:

1. routing information when probabilistic information is present (e.g., road closures, uncertain travel time);
2. top-k relevant paths; or
3. accessibility under security restrictions;

by computing semiring-based provenance of graph queries.

Each of these algorithms yields a tradeoff between generality and time complexity.

Overview

We propose several algorithms able to solve:

1. routing information when probabilistic information is present (e.g., road closures, uncertain travel time);
2. top-k relevant paths; or
3. accessibility under security restrictions;

by computing semiring-based provenance of graph queries.

Each of these algorithms yields a tradeoff between generality and time complexity.

Overview

We propose several algorithms able to solve:

1. routing information when probabilistic information is present (e.g., road closures, uncertain travel time);
2. top-k relevant paths; or
3. accessibility under security restrictions;

by computing semiring-based provenance of graph queries.

Each of these algorithms yields a tradeoff between generality and time complexity.

Overview

We propose several algorithms able to solve:

1. routing information when probabilistic information is present (e.g., road closures, uncertain travel time);
2. top-k relevant paths; or
3. accessibility under security restrictions;

by computing semiring-based provenance of graph queries.

Each of these algorithms yields a tradeoff between generality and time complexity.

Overview

We propose several algorithms able to solve:

1. routing information when probabilistic information is present (e.g., road closures, uncertain travel time);
2. top-k relevant paths; or
3. accessibility under security restrictions;

by computing semiring-based provenance of graph queries.

Each of these algorithms yields a tradeoff between generality and time complexity.

Graph Database with Provenance Indication

Definition (Graph Database)

A *graph database* G over Σ is a pair (V, E), where V is a finite set of node ids and $E \subseteq V \times \Sigma \times V$.

Annotations are given by a weight function, $w : E \to \mathbb{S}$ for $(\mathbb{S}, \oplus, \otimes, \bar{0}, \bar{1})$ a semiring.

Graph Database with Provenance Indication

Definition (Graph Database)

A *graph database* G over Σ is a pair (V, E), where V is a finite set of node ids and $E \subseteq V \times \Sigma \times V$.

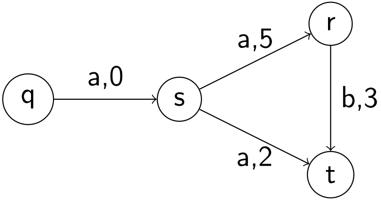
Annotations are given by a weight function, $w : E \to \mathbb{S}$ for $(\mathbb{S}, \oplus, \otimes, \bar{0}, \bar{1})$ a semiring.

Graph Database with Provenance Indication

Definition (Graph Database)

A *graph database* G over Σ is a pair (V, E), where V is a finite set of node ids and $E \subseteq V \times \Sigma \times V$.

Annotations are given by a weight function, $w : E \to \mathbb{S}$ for $(\mathbb{S}, \oplus, \otimes, \bar{0}, \bar{1})$ a semiring.



Basic Semiring Theory

An element $a \in \mathbb{K}$ is idempotent if $a \oplus a = a$. $\mathbb{K}$ is said to be idempotent when all elements are.

Definition (k-closed Semiring)

Let $k \geq 0$ be an integer. A semiring $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ is k-closed if $\forall a \in \mathbb{K}, \bigoplus_{n=0}^{k+1} a^n = \bigoplus_{n=0}^{k} a^n$.

Definition (Star semiring)

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

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

Basic Semiring Theory

An element $a \in \mathbb{K}$ is idempotent if $a \oplus a = a$. $\mathbb{K}$ is said to be idempotent when all elements are.

Definition (k-closed Semiring)

Let $k \geq 0$ be an integer. A semiring $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ is k-closed if $\forall a \in \mathbb{K}, \bigoplus_{n=0}^{k+1} a^n = \bigoplus_{n=0}^{k} a^n$.

Definition (Star semiring)

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

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

Basic Semiring Theory

An element $a \in \mathbb{K}$ is idempotent if $a \oplus a = a$. $\mathbb{K}$ is said to be idempotent when all elements are.

Definition (k-closed Semiring)

Let $k \geq 0$ be an integer. A semiring $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ is k-closed if $\forall a \in \mathbb{K}, \bigoplus_{n=0}^{k+1} a^n = \bigoplus_{n=0}^{k} a^n$.

Definition (Star semiring)

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

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

Examples of Semirings

Tropical: $\mathbb{T} = (\mathbb{R}_+ \cup \{\infty\}, \mathtt{min}, +, \infty, 0)$ to compute shortest-distance.

kTropical: $\mathbb{T}_k$ is the set of k-tuples in $(\mathbb{R}_+ \cup \{\infty\})^k$ ordered for the natural order of $\mathbb{R} \cup \{\infty\}$):

$$\mathbb{T}_k = \{(a_1, \ldots, a_k) \in (\mathbb{R}_+ \cup \{\infty\})^k : 0 \leq a_1 \leq \cdots \leq a_k\}.$$

$$a \oplus_k b = \mathtt{min}_k((a_i)_i \cup (b_j)_j) \text{ and } a \otimes_k b = \mathtt{min}_k((a_i + b_j)_{i,j})$$

used to compute top-k shortest-distance.

Security: $\mathbb{S}_n$ is then $(\{0, \ldots, n+1\}, \mathtt{min}, \mathtt{max}, n+1, 0)$. Integers correspond to levels such as *Top Secret*, *Secret*, *Restricted*, etc... to compute under security restrictions.

Counting: $(N \cup \{\infty\})$ together with a star operation: $0^* = 1$ and $\forall a \in \mathbb{N}, a^* = \infty$ to compute number of paths between two nodes.

Examples of Semirings

Tropical: $\mathbb{T} = (\mathbb{R}_+ \cup \{\infty\}, \mathtt{min}, +, \infty, 0)$ to compute shortest-distance.

kTropical: $\mathbb{T}_k$ is the set of k-tuples in $(\mathbb{R}_+ \cup \{\infty\})^k$ ordered for the natural order of $\mathbb{R} \cup \{\infty\}$):

$$\mathbb{T}_k = \{(a_1, \ldots, a_k) \in (\mathbb{R}_+ \cup \{\infty\})^k : 0 \leq a_1 \leq \cdots \leq a_k\}.$$

$$a \oplus_k b = \mathtt{min}_k((a_i)_i \cup (b_j)_j) \text{ and } a \otimes_k b = \mathtt{min}_k((a_i + b_j)_{i,j})$$

used to compute top-k shortest-distance.

Security: $\mathbb{S}_n$ is then $(\{0, \ldots, n+1\}, \mathtt{min}, \mathtt{max}, n+1, 0)$. Integers correspond to levels such as *Top Secret*, *Secret*, *Restricted*, etc... to compute under security restrictions.

Counting: $(N \cup \{\infty\})$ together with a star operation: $0^* = 1$ and $\forall a \in \mathbb{N}, a^* = \infty$ to compute number of paths between two nodes.

Examples of Semirings

Tropical: $\mathbb{T} = (\mathbb{R}_+ \cup \{\infty\}, \mathtt{min}, +, \infty, 0)$ to compute shortest-distance.

kTropical: $\mathbb{T}_k$ is the set of k-tuples in $(\mathbb{R}_+ \cup \{\infty\})^k$ ordered for the natural order of $\mathbb{R} \cup \{\infty\}$):

$$\mathbb{T}_k = \{(a_1, \ldots, a_k) \in (\mathbb{R}_+ \cup \{\infty\})^k : 0 \leq a_1 \leq \cdots \leq a_k\}.$$

$$a \oplus_k b = \mathtt{min}_k((a_i)_i \cup (b_j)_j) \text{ and } a \otimes_k b = \mathtt{min}_k((a_i + b_j)_{i,j})$$

used to compute top-k shortest-distance.

Security: $\mathbb{S}_n$ is then $(\{0, \ldots, n+1\}, \mathtt{min}, \mathtt{max}, n+1, 0)$. Integers correspond to levels such as *Top Secret*, *Secret*, *Restricted*, etc... to compute under security restrictions.

Counting: $(N \cup \{\infty\})$ together with a star operation: $0^* = 1$ and $\forall a \in \mathbb{N}, a^* = \infty$ to compute number of paths between two nodes.

Examples of Semirings

Tropical: $\mathbb{T} = (\mathbb{R}_+ \cup \{\infty\}, \mathtt{min}, +, \infty, 0)$ to compute shortest-distance.

*k*Tropical: $\mathbb{T}_k$ is the set of k-tuples in $(\mathbb{R}_+ \cup \{\infty\})^k$ ordered for the natural order of $\mathbb{R} \cup \{\infty\}$):

$$\mathbb{T}_k = \{(a_1, \dots, a_k) \in (\mathbb{R}_+ \cup \{\infty\})^k : 0 \leq a_1 \leq \dots \leq a_k\}.$$

$$a \oplus_k b = \mathtt{min}_k((a_i)_i \cup (b_j)_j) \text{ and } a \otimes_k b = \mathtt{min}_k((a_i + b_j)_{i,j})$$

used to compute top-k shortest-distance.

Security: $\mathbb{S}_n$ is then $(\{0, \dots, n+1\}, \mathtt{min}, \mathtt{max}, n+1, 0)$. Integers correspond to levels such as *Top Secret*, *Secret*, *Restricted*, etc... to compute under security restrictions.

Counting: $(N \cup \{\infty\})$ together with a star operation: $0^* = 1$ and $\forall a \in \mathbb{N}, a^* = \infty$ to compute number of paths between two nodes.

Regular Path Queries

Querying graph databases with Regular Path Queries (RPQ).

Definition (Regular Path Query)

RPQs have the form $RPQ(s, t) \coloneqq (s, L, t)$

L = a*b

Regular Path Queries

Querying graph databases with Regular Path Queries (RPQ).

Definition (Regular Path Query)

RPQs have the form $RPQ(s, t) \coloneqq (s, L, t)$

L = a*b

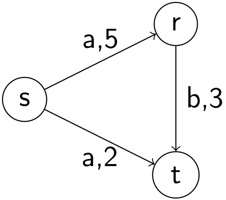
Regular Path Queries

Querying graph databases with Regular Path Queries (RPQ).

Definition (Regular Path Query)

RPQs have the form $RPQ(s, t) \coloneqq (s, L, t)$

L = a*b



Concept of Provenance

For an answer to a query we want to be able to answer to such questions:

- How is this answer produced?
- What score should this answer receive given initial annotations over edges?

We consider the following problem: given a graph database G with annotations over $\mathbb{K}$, a RPQ Q and s and t the source and target nodes, compute the provenance of the RPQ.

Definition (Provenance of an RPQ)

$$\text{prov}_{\mathbb{K}}^{Q}(G)(s,t) := \bigoplus_{\substack{\pi \in P_{st}(G), \\ \rho(\pi) \in L_Q}} w[\pi].$$

Concept of Provenance

For an answer to a query we want to be able to answer to such questions:

- How is this answer produced?
- What score should this answer receive given initial annotations over edges?

We consider the following problem: given a graph database G with annotations over $\mathbb{K}$, a RPQ Q and s and t the source and target nodes, compute the provenance of the RPQ.

Definition (Provenance of an RPQ)

$$\text{prov}^{Q}_{\mathbb{K}}(G)(s,t) := \bigoplus_{\substack{\pi \in P_{st}(G), \\ \rho(\pi) \in L_Q}} w[\pi].$$

Concept of Provenance

For an answer to a query we want to be able to answer to such questions:

- How is this answer produced?
- What score should this answer receive given initial annotations over edges?

We consider the following problem: given a graph database G with annotations over $\mathbb{K}$, a RPQ Q and s and t the source and target nodes, compute the provenance of the RPQ.

Definition (Provenance of an RPQ)

$$\mathrm{prov}_{\mathbb{K}}^{Q}(G)(s,t) := \bigoplus_{\substack{\pi \in P_{st}(G),\\ \rho(\pi) \in L_Q}} w[\pi].$$

Concept of Provenance

For an answer to a query we want to be able to answer to such questions:

- How is this answer produced?
- What score should this answer receive given initial annotations over edges?

We consider the following problem: given a graph database G with annotations over $\mathbb{K}$, a RPQ Q and s and t the source and target nodes, compute the provenance of the RPQ.

Definition (Provenance of an RPQ)

$$\text{prov}_{\mathbb{K}}^{Q}(G)(s,t) := \bigoplus_{\substack{\pi \in P_{st}(G), \\ \rho(\pi) \in L_Q}} w[\pi].$$

Concept of Provenance

For an answer to a query we want to be able to answer to such questions:

- How is this answer produced?
- What score should this answer receive given initial annotations over edges?

We consider the following problem: given a graph database G with annotations over $\mathbb{K}$, a RPQ Q and s and t the source and target nodes, compute the provenance of the RPQ.

Definition (Provenance of an RPQ)

$$\mathrm{prov}_{\mathbb{K}}^{Q}(G)(s,t) := \bigoplus_{\substack{\pi \in P_{st}(G),\\ \rho(\pi) \in L_Q}} w[\pi].$$

Shortest-path and Shortest-distance Problems

Shortest-path: initially with tropical semiring, without any label.

Can be generalized for star-semirings and is then called Shortest-distance, still not using labels.

Shortest-distance is precisely provenance with $L_Q = \Sigma^*$.

Shortest-path and Shortest-distance Problems

Shortest-path: initially with tropical semiring, without any label.

Can be generalized for star-semirings and is then called Shortest-distance, still not using labels.

Shortest-distance is precisely provenance with $L_Q = \Sigma^*$.

Shortest-path and Shortest-distance Problems

Shortest-path: initially with tropical semiring, without any label.

Can be generalized for star-semirings and is then called Shortest-distance, still not using labels.

Shortest-distance is precisely provenance with $L_Q = \Sigma^*$.

Generalized Floyd-Warshall Algorithm

Partial values we compute are defined as follows

$$l_{ij}^{(k)} := \bigoplus_{\pi \in P_{ij}^{(k)}(G)} w[\pi]$$

Partial values can be computed recursively

$$l_{ij}^{(k)} = l_{ij}^{(k-1)} \oplus \left(l_{ik}^{(k-1)} \otimes (l_{kk}^{(k-1)})^* \otimes l_{kj}^{(k-1)} \right).$$

And the initialisation is

$$l_{ij}^{(0)} = \begin{cases} \lambda(i,j) & \text{if } i \neq j, \\ \bar{1} \oplus \lambda(i,j) & \text{if } i = j. \end{cases}$$

The complexity of this approach is $\mathcal{O}(|V|^3(\mathrm{T}_\oplus + \mathrm{T}_\otimes + T_*))$.
If $\mathbb{K}$ is a k-closed semiring, $\mathrm{T}_* \leq k \times (\mathrm{T}_\oplus + \mathrm{T}_\otimes)$.

Generalized Floyd-Warshall Algorithm

Partial values we compute are defined as follows

$$l_{ij}^{(k)} := \bigoplus_{\pi \in P_{ij}^{(k)}(G)} w[\pi]$$

Partial values can be computed recursively

$$l_{ij}^{(k)} = l_{ij}^{(k-1)} \oplus \left(l_{ik}^{(k-1)} \otimes (l_{kk}^{(k-1)})^* \otimes l_{kj}^{(k-1)} \right).$$

And the initialisation is

$$l_{ij}^{(0)} = \begin{cases} \lambda(i,j) & \text{if } i \neq j, \\ \bar{1} \oplus \lambda(i,j) & \text{if } i = j. \end{cases}$$

The complexity of this approach is $\mathcal{O}(|V|^3(\mathrm{T}_\oplus + \mathrm{T}_\otimes + T_*))$.
If $\mathbb{K}$ is a k-closed semiring, $\mathrm{T}_* \leq k \times (\mathrm{T}_\oplus + \mathrm{T}_\otimes)$.

Generalized Floyd-Warshall Algorithm

Partial values we compute are defined as follows

$$l_{ij}^{(k)} := \bigoplus_{\pi \in P_{ij}^{(k)}(G)} w[\pi]$$

Partial values can be computed recursively

$$l_{ij}^{(k)} = l_{ij}^{(k-1)} \oplus \left(l_{ik}^{(k-1)} \otimes (l_{kk}^{(k-1)})^* \otimes l_{kj}^{(k-1)} \right).$$

And the initialisation is

$$l_{ij}^{(0)} = \begin{cases} \lambda(i,j) & \text{if } i \neq j, \\ \bar{1} \oplus \lambda(i,j) & \text{if } i = j. \end{cases}$$

The complexity of this approach is $\mathcal{O}(|V|^3(\mathrm{T}_\oplus + \mathrm{T}_\otimes + T_*))$.
If $\mathbb{K}$ is a k-closed semiring, $\mathrm{T}_* \leq k \times (\mathrm{T}_\oplus + \mathrm{T}_\otimes)$.

Generalized Floyd-Warshall Algorithm

Partial values we compute are defined as follows

$$l_{ij}^{(k)} := \bigoplus_{\pi \in P_{ij}^{(k)}(G)} w[\pi]$$

Partial values can be computed recursively

$$l_{ij}^{(k)} = l_{ij}^{(k-1)} \oplus \left(l_{ik}^{(k-1)} \otimes (l_{kk}^{(k-1)})^* \otimes l_{kj}^{(k-1)} \right).$$

And the initialisation is

$$l_{ij}^{(0)} = \begin{cases} \lambda(i,j) & \text{if } i \neq j, \\ \bar{1} \oplus \lambda(i,j) & \text{if } i = j. \end{cases}$$

The complexity of this approach is $\mathcal{O}(|V|^3(\mathrm{T}_\oplus + \mathrm{T}_\otimes + T_*))$.
If $\mathbb{K}$ is a k-closed semiring, $\mathrm{T}_* \leq k \times (\mathrm{T}_\oplus + \mathrm{T}_\otimes)$.

Generalized Floyd-Warshall Algorithm

Partial values we compute are defined as follows

$$l_{ij}^{(k)} := \bigoplus_{\pi \in P_{ij}^{(k)}(G)} w[\pi]$$

Partial values can be computed recursively

$$l_{ij}^{(k)} = l_{ij}^{(k-1)} \oplus \left(l_{ik}^{(k-1)} \otimes (l_{kk}^{(k-1)})^* \otimes l_{kj}^{(k-1)} \right).$$

And the initialisation is

$$l_{ij}^{(0)} = \begin{cases} \lambda(i,j) & \text{if } i \neq j, \\ \bar{1} \oplus \lambda(i,j) & \text{if } i = j. \end{cases}$$

The complexity of this approach is $\mathcal{O}(|V|^3(\mathrm{T}_\oplus + \mathrm{T}_\otimes + T_*))$.
If $\mathbb{K}$ is a k-closed semiring, $\mathrm{T}_* \leq k \times (\mathrm{T}_\oplus + \mathrm{T}_\otimes)$.

Mohri's Algorithm

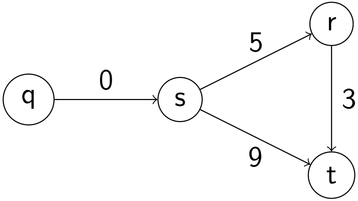
Generalization from Mohri [1998] of the classical relaxation technique used in Bellman–Ford algorithm.

Modifications to work with non-idempotent semirings. All k-closed semirings.

No polynomial bound over the number of relaxations of a vertex.

Mohri's Algorithm

Generalization from Mohri [1998] of the classical relaxation technique used in Bellman–Ford algorithm.

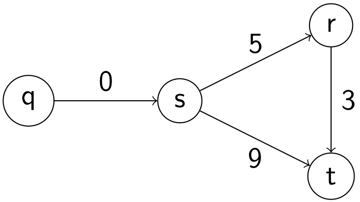


Modifications to work with non-idempotent semirings. All k-closed semirings.

No polynomial bound over the number of relaxations of a vertex.

Mohri's Algorithm

Generalization from Mohri [1998] of the classical relaxation technique used in Bellman–Ford algorithm.

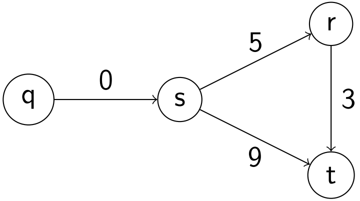


Modifications to work with non-idempotent semirings. All *k*-closed semirings.

No polynomial bound over the number of relaxations of a vertex.

Mohri's Algorithm

Generalization from Mohri [1998] of the classical relaxation technique used in Bellman–Ford algorithm.



Modifications to work with non-idempotent semirings. All *k*-closed semirings.

No polynomial bound over the number of relaxations of a vertex.

Contents

Product Graph

Idea: reduce the problem of provenance of RPQ to the shortest-distance problem, eliminate the RPQ and the labels.

Product of the initial graph and a deterministic automaton L_Q representing the language of the RPQ.

Product Graph

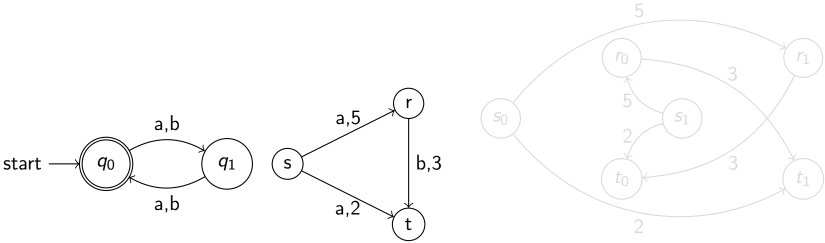
Idea: reduce the problem of provenance of RPQ to the shortest-distance problem, eliminate the RPQ and the labels.

Product of the initial graph and a deterministic automaton L_Q representing the language of the RPQ.

Product Graph

Idea: reduce the problem of provenance of RPQ to the shortest-distance problem, eliminate the RPQ and the labels.

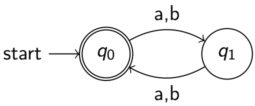
Product of the initial graph and a deterministic automaton L_Q representing the language of the RPQ.

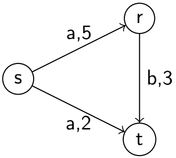


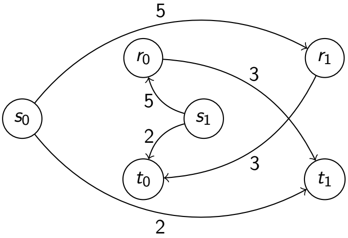
Product Graph

Idea: reduce the problem of provenance of RPQ to the shortest-distance problem, eliminate the RPQ and the labels.

Product of the initial graph and a deterministic automaton L_Q representing the language of the RPQ.







Proof of Correctness

Idea: the label of a valid path leads to an accepting state starting from the initial one.

Theorem

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

Proof of Correctness

Idea: the label of a valid path leads to an accepting state starting from the initial one.

Theorem

$$\mathrm{prov}_{\mathbb{K}}^{Q}(G)(x,y) = \bigoplus_{s_F \in F} \mathrm{prov}_{\mathbb{K}}^{\mathrm{R}}(P_{G \times \mathcal{A}_Q})((x,s_0),(y,s_F))$$

Applying Mohri's Algorithm

We apply it on the product graph to compute single-source provenance. The semiring needs to be k-closed.

Size of the product graph bounded by $n \cdot |G|$ (n size of the automaton).

Theoretical complexity exponential.

Applying Mohri's Algorithm

We apply it on the product graph to compute single-source provenance. The semiring needs to be k-closed.

Size of the product graph bounded by $n \cdot |G|$ (n size of the automaton).

Theoretical complexity exponential.

Applying Mohri's Algorithm

We apply it on the product graph to compute single-source provenance. The semiring needs to be k-closed.

Size of the product graph bounded by $n \cdot |G|$ (n size of the automaton).

Theoretical complexity exponential.

Node Elimination Algorithm (1)

Inspired by the construction of Brzozowski and McCluskey [1963].

Works for star-semirings (such as the counting semiring).

Variant of Floyd-Warshall when we are only interested in a subset of all pairs.

Node Elimination Algorithm (1)

Inspired by the construction of Brzozowski and McCluskey [1963].

Works for star-semirings (such as the counting semiring).

Variant of Floyd-Warshall when we are only interested in a subset of all pairs.

Node Elimination Algorithm (1)

Inspired by the construction of Brzozowski and McCluskey [1963].

Works for star-semirings (such as the counting semiring).

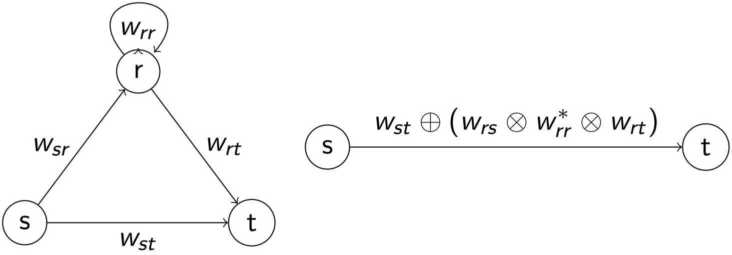
Variant of Floyd-Warshall when we are only interested in a subset of all pairs.

Node Elimination Algorithm (1)

Inspired by the construction of Brzozowski and McCluskey [1963].

Works for star-semirings (such as the counting semiring).

Variant of Floyd-Warshall when we are only interested in a subset of all pairs.



Node Elimination Algorithm (2)

Computing only one time the star for each vertex we can obtain a worst-case complexity in $\mathcal{O}(|V|T_* + |V|^3(T_\oplus + T_\otimes))$.

Using heuristics and sparsity we can improve practical complexity.

Can handle multiple source/targets.

Node Elimination Algorithm (2)

Computing only one time the star for each vertex we can obtain a worst-case complexity in $\mathcal{O}(|V|T_* + |V|^3(T_\oplus + T_\otimes))$.

Using heuristics and sparsity we can improve practical complexity.

Can handle multiple source/targets.

Node Elimination Algorithm (2)

Computing only one time the star for each vertex we can obtain a worst-case complexity in $\mathcal{O}(|V|T_* + |V|^3(T_\oplus + T_\otimes))$.

Using heuristics and sparsity we can improve practical complexity.

Can handle multiple source/targets.

Node Elimination Algorithm (2)

Computing only one time the star for each vertex we can obtain a worst-case complexity in $\mathcal{O}(|V| T_* + |V|^3 (T_\oplus + T_\otimes))$.

Using heuristics and sparsity we can improve practical complexity.

Can handle multiple source/targets.

Study of Dijkstra Algorithm

Natural order: $a \leq_{\mathbb{K}} b := a \oplus b = a$ is an order when $\oplus$ is idempotent.

0-closedness implies idempotence.

In the following we restrict to 0-closed semirings with $\leq_{\mathbb{K}}$ being a total order.

Including for instance tropical and security semirings.

Study of Dijkstra Algorithm

Natural order: $a \leq_{\mathbb{K}} b := a \oplus b = a$ is an order when $\oplus$ is idempotent.

0-closedness implies idempotence.

In the following we restrict to 0-closed semirings with $\leq_{\mathbb{K}}$ being a total order.

Including for instance tropical and security semirings.

Study of Dijkstra Algorithm

Natural order: $a \leq_{\mathbb{K}} b := a \oplus b = a$ is an order when $\oplus$ is idempotent.

0-closedness implies idempotence.

In the following we restrict to 0-closed semirings with $\leq_{\mathbb{K}}$ being a total order.

Including for instance tropical and security semirings.

Study of Dijkstra Algorithm

Natural order: $a \leq_{\mathbb{K}} b := a \oplus b = a$ is an order when $\oplus$ is idempotent.

0-closedness implies idempotence.

In the following we restrict to 0-closed semirings with $\leq_{\mathbb{K}}$ being a total order.

Including for instance tropical and security semirings.

Study of Dijkstra Algorithm (2)

We show Dijkstra's algorithm for shortest path can be generalized to semirings having these specific properties.

Höfner and Möller [2012] already generalized this algorithm but not giving explicitely these requirements.

Idea:

$c \oplus (c \otimes e) = c$ because of 0-closedness.

Study of Dijkstra Algorithm (2)

We show Dijkstra's algorithm for shortest path can be generalized to semirings having these specific properties.

Höfner and Möller [2012] already generalized this algorithm but not giving explicitely these requirements.

Idea:

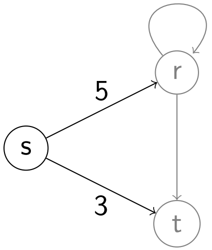
$c \oplus (c \otimes e) = c$ because of 0-closedness.

Study of Dijkstra Algorithm (2)

We show Dijkstra's algorithm for shortest path can be generalized to semirings having these specific properties.

Höfner and Möller [2012] already generalized this algorithm but not giving explicitely these requirements.

Idea:



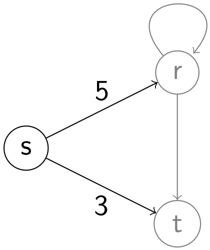
$c \oplus (c \otimes e) = c$ because of 0-closedness.

Study of Dijkstra Algorithm (2)

We show Dijkstra's algorithm for shortest path can be generalized to semirings having these specific properties.

Höfner and Möller [2012] already generalized this algorithm but not giving explicitely these requirements.

Idea:



$c \oplus (c \otimes e) = c$ because of 0-closedness.

Study of Dijkstra Algorithm (3)

We used Fibonacci Heap in the implementation, which give a total complexity in $\mathcal{O}(\mathrm{T}_{\oplus}|V|\log|V|) + |E|(T_{\oplus} + T_{\otimes})$.

Counter example where the natural order is not total:

consider the semiring $(\mathbb{D}_{30}, \wedge, \vee, 30, 1)$ of the divisors of 30.

Study of Dijkstra Algorithm (3)

We used Fibonacci Heap in the implementation, which give a total complexity in $\mathcal{O}(\mathrm{T}_\oplus|V|\log|V|) + |E|(T_\oplus + T_\otimes)$.

Counter example where the natural order is not total:

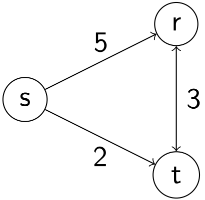
consider the semiring $(\mathbb{D}_{30}, \wedge, \vee, 30, 1)$ of the divisors of 30.

Study of Dijkstra Algorithm (3)

We used Fibonacci Heap in the implementation, which give a total complexity in $\mathcal{O}(\mathrm{T}_{\oplus}|V|\log|V|) + |E|(T_{\oplus} + T_{\otimes})$.

Counter example where the natural order is not total:

consider the semiring $(\mathbb{D}_{30}, \wedge, \vee, 30, 1)$ of the divisors of 30.



Contents

STIF Network

Public transit data for trains, buses, subways, and trams within the Paris region.

"Syndicat des Transports d'Ile-de-France"

Encoded using the General Transit Feed Specification (GTFS).

STIF Network

Public transit data for trains, buses, subways, and trams within the Paris region.

"Syndicat des Transports d'Ile-de-France"

Encoded using the General Transit Feed Specification (GTFS).

STIF Network

Public transit data for trains, buses, subways, and trams within the Paris region.

"Syndicat des Transports d'Ile-de-France"

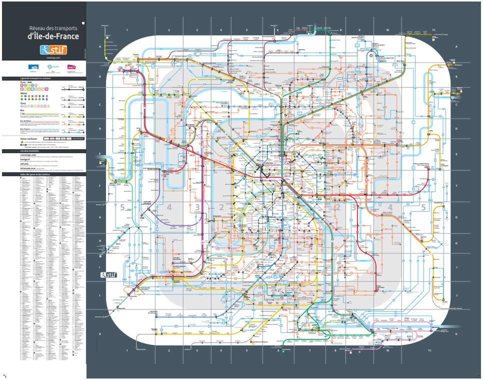
Encoded using the General Transit Feed Specification (GTFS).

STIF Network

Public transit data for trains, buses, subways, and trams within the Paris region.

"Syndicat des Transports d'Ile-de-France"

Encoded using the General Transit Feed Specification (GTFS).



Extraction of the Data

Use tables *stops*, *stop_times*, *trips*, and *routes*.

Labels are lines type and numbers.

Extracted graph contains 16,369 nodes and 41,448 edges.

Subway subgraph contains 302 nodes and 705 edges.

Extraction of the Data

Use tables *stops*, *stop_times*, *trips*, and *routes*.

Labels are lines type and numbers.

Extracted graph contains 16,369 nodes and 41,448 edges.

Subway subgraph contains 302 nodes and 705 edges.

Extraction of the Data

Use tables *stops*, *stop_times*, *trips*, and *routes*.

Labels are lines type and numbers.

Extracted graph contains 16,369 nodes and 41,448 edges.

Subway subgraph contains 302 nodes and 705 edges.

Extraction of the Data

Use tables *stops*, *stop_times*, *trips*, and *routes*.

Labels are lines type and numbers.

Extracted graph contains 16,369 nodes and 41,448 edges.

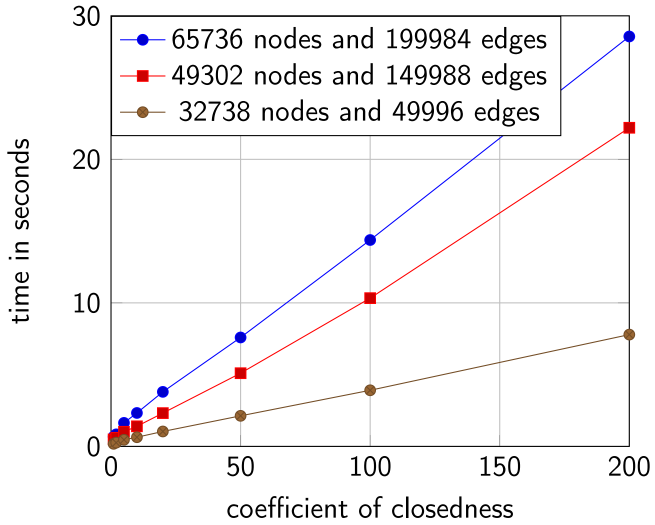
Subway subgraph contains 302 nodes and 705 edges.

Mohri's Algorithm

Graph:	full graph,
Semiring:	k-Tropical,
Request:	paths of length mod 2, 3 and 4,
Nodes:	s, t random,

Avg. of 3 runs.

Mohri's Algorithm

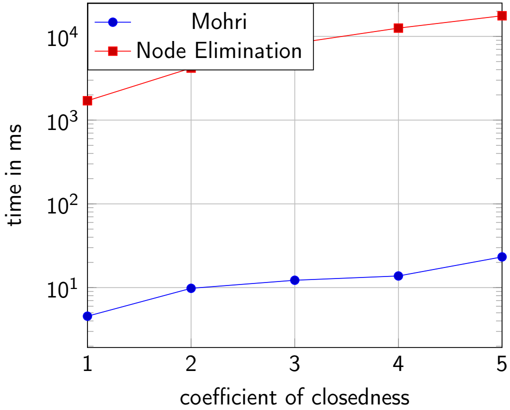


Comparison between Mohri and Node Elimination

Graph:	subway graph,
Semiring:	k-Tropical with description,
Request:	reachability query,
Nodes:	s, t random,

Avg. of 3 runs.

Comparison between Mohri and Node Elimination



Time (in milliseconds) to compute single-source provenance with Mohri (blue) and Node Elimination (red) depending on the coefficient of closedness.

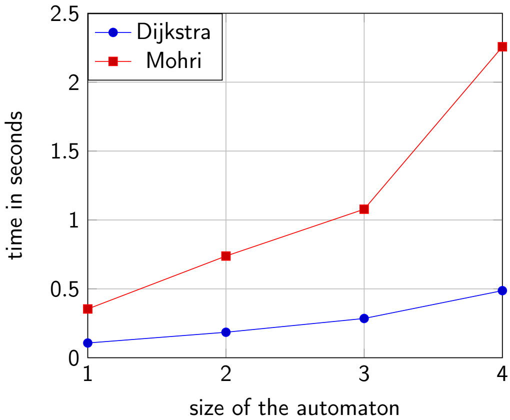
Comparison between Mohri and Dijkstra

For this purpose we used a graph with random security numbers (0–1000) over edges.

Graph:	full graph,
Semiring:	Security,
Request:	paths of length mod 2, 3, 4 and 5,
Nodes:	s, t random,

Avg. of 3 runs.

Comparison between Mohri and Dijkstra



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

Contents

Conclusion

We reduced the computation of the provenance of an RPQ to the computation of shortest-distance by using a graph product.

We sum-up different algorithms proposed:

Name	Framework	Complexity
Floyd-Warshall	star	$\mathcal{O}(\lvert V\rvert^3(\mathrm{T}_\oplus + \mathrm{T}_\otimes + T_*))$
Mohri	k-closed	Exp.
Node Elimination	star	$\mathcal{O}(\lvert V\rvert T_* + \lvert V\rvert^3(T_\oplus + T_\otimes))$
Dijkstra	0-closed total ordered	$\mathcal{O}(\mathrm{T}_\oplus\lvert V\rvert \log\lvert V\rvert + \lvert E\rvert(T_\oplus + T_\otimes))$

Conclusion

We reduced the computation of the provenance of an RPQ to the computation of shortest-distance by using a graph product.

We sum-up different algorithms proposed:

Name	Framework	Complexity
Floyd-Warshall	star	$\mathcal{O}(\lvert V\rvert^3(\mathrm{T}_\oplus + \mathrm{T}_\otimes + T_*))$
Mohri	k-closed	Exp.
Node Elimination	star	$\mathcal{O}(\lvert V\rvert T_* + \lvert V\rvert^3(T_\oplus + T_\otimes))$
Dijkstra	0-closed total ordered	$\mathcal{O}(\mathrm{T}_\oplus\lvert V\rvert \log\lvert V\rvert + \lvert E\rvert(T_\oplus + T_\otimes))$

Perspectives

- Understand practical efficiency of these algorithms.
- Use graph structures and specific properties of transport networks to improve efficiency:
  - contraction hierarchies,
  - low treewidth,
  - sparsity,
  - low highway dimension.

Thank you for your attention!

Perspectives

- Understand practical efficiency of these algorithms.
- Use graph structures and specific properties of transport networks to improve efficiency:
  - contraction hierarchies,
  - low treewidth,
  - sparsity,
  - low highway dimension.

Thank you for your attention!

Perspectives

- Understand practical efficiency of these algorithms.
- Use graph structures and specific properties of transport networks to improve efficiency:
  - contraction hierarchies,
  - low treewidth,
  - sparsity,
  - low highway dimension.

Thank you for your attention!

Perspectives

- Understand practical efficiency of these algorithms.
- Use graph structures and specific properties of transport networks to improve efficiency:
  - contraction hierarchies,
  - low treewidth,
  - sparsity,
  - low highway dimension.

Thank you for your attention!

Perspectives

- Understand practical efficiency of these algorithms.
- Use graph structures and specific properties of transport networks to improve efficiency:
  - contraction hierarchies,
  - low treewidth,
  - sparsity,
  - low highway dimension.

Thank you for your attention!

Perspectives

- Understand practical efficiency of these algorithms.
- Use graph structures and specific properties of transport networks to improve efficiency:
  - contraction hierarchies,
  - low treewidth,
  - sparsity,
  - low highway dimension.

Thank you for your attention!

Bibliography I

J. A. Brzozowski and E. J. McCluskey. Signal Flow Graph Techniques for Sequential Circuit State Diagrams. *Electronic Computers, IEEE Transactions on*, EC-12(2):67–76, April 1963. doi: 10.1109/PGEC.1963.263416. URL http://dx.doi.org/10.1109/PGEC.1963.263416.

Peter Höfner and Bernhard Möller. Dijkstra, Floyd and Warshall meet Kleene. *Formal Aspects of Computing*, 24(4-6):459–476, July 2012. ISSN 0934-5043, 1433-299X. doi: 10.1007/s00165-012-0245-4. URL http://link.springer.com/10.1007/s00165-012-0245-4.

M. Mohri. General algebraic frameworks and algorithms for shortest-distance problems. *Technical Memorandum 981210-10TM*, 1998.

Bibliography II

M. Mohri. Semiring frameworks and algorithms for shortest-distance problems. *J. Autom. Lang. Comb.*, 7(3): 321–350, January 2002. ISSN 1430-189X. URL http://dl.acm.org/citation.cfm?id=639508.639512.