Journal of Automata, Languages and Combinatorics **7** (2002) 3, 321–350

SEMIRING FRAMEWORKS AND ALGORITHMS FOR SHORTEST-DISTANCE PROBLEMS

Mehryar Mohri
AT&T Labs – Research
180 Park Avenue, Rm E135, Florham Park, NJ 07932, USA
e-mail: mohri@research.att.com

ABSTRACT

We define general algebraic frameworks for shortest-distance problems based on the structure of semirings. We give a generic algorithm for finding single-source shortest distances in a weighted directed graph when the weights satisfy the conditions of our general semiring framework. The same algorithm can be used to solve efficiently classical shortest paths problems or to find the k-shortest distances in a directed graph. It can be used to solve single-source shortest-distance problems in weighted directed acyclic graphs over any semiring. We examine several semirings and describe some specific instances of our generic algorithms to illustrate their use and compare them with existing methods and algorithms. The proof of the soundness of all algorithms is given in detail, including their pseudocode and a full analysis of their running time complexity.

Keywords: Semirings, finite automata, shortest-paths algorithms, rational power series

Introduction

Classical shortest-paths problems in a weighted directed graph arise in various contexts. The problems divide into two related categories: single-source shortest-paths problems and all-pairs shortest-paths problems. The single-source shortest-path problem in a directed graph consists of determining the shortest path from a fixed source vertex s to all other vertices. The all-pairs shortest-distance problem is that of finding the shortest paths between all pairs of vertices of a graph.

In the classical shortest-paths problems, edge weights may represent distances, costs, or any other real-valued quantity that can be added along a path, and that one wishes to minimize. Edge weights are real numbers (elements of $\mathbb{R}$), and the specific operations used are: addition ($+$) along a path, and minimum (min) applied to path weights.

Classical shortest-paths problems can be generalized to other weight sets, and to other operations. The weights, elements of a set $\mathbb{K}$, may be real numbers, strings, regular expressions, subsets of another set, or any other quantity that can be *multiplied* along a path using an operation $\otimes$, and that can be *summed* using an operation $\oplus$. The weight of a path is obtained by multiplying edge weights along that path using $\otimes$, and the *shortest distance* from a vertex p to a vertex q is the sum of the weights of all paths from p to q using $\oplus$.

We call the problems of finding the shortest distances with this generalized definition of shortest distances *shortest-distance problems*. For these problems to be well-defined, certain restrictions such as the distributivity of $\otimes$ over $\oplus$ are required. More precisely, the algebraic structure that provides the appropriate framework for these problems is that of *semirings*: the system $(\mathbb{K}, \oplus, \otimes)$ is a semiring in the problems that we consider. The notion of *shortest path* is not pertinent anymore for these general problems since for some semirings and weighted graphs there might be no path between two vertices p and q with a weight equal to the *shortest distance* from p to q.

The absence of a unifying framework for single-source shortest paths problems was pointed out by Alfred Aho et al. [2, p. 207]. We define a general algebraic framework for single-source shortest-distance problems based on the structure of semirings. We give a generic algorithm for solving these problems. The algorithm is generic in two senses: it works with any semiring covered by our general framework, and is correct regardless of the queue discipline chosen for the implementation of the algorithm. Classical algorithms such as that of Bellman-Ford [4, 17] are specific instances of this generic algorithm.

We give the proof of the correctness and termination of the algorithm, including a full analysis of its running time complexity with respect to the times to compute the $\oplus$ and $\otimes$ operations. We also give a generic algorithm for solving single-source shortest-distance problems in weighted directed acyclic graphs. The algorithm works with any semiring. The algorithm of Lawler [24] is a specific instance of this algorithm.

We then describe several specific semirings that verify the conditions of our general framework for single-source shortest distance problems, give for various queue disciplines the complexity of our algorithm and compare them with existing methods and algorithms. We show, in particular, that the same generic algorithm can be used to solve classical shortest-paths problems, k-shortest-distance problems, and other problems by using each time the appropriate semiring.

The all-pairs shortest-distance problem is known as the *algebraic path problem* and has been studied by numerous authors in the past with semiring frameworks of varying degrees of generality [17, 18, 19, 4, 9, 35, 16, 41, 42, 7, 3, 2, 25] starting with the work of Stephen Kleene and his proof of the equivalence of finite automata and regular expressions [21] (see [15] for a survey of the algebraic path problem). The axioms of the closed semirings presented by some of these authors were not correct [42, 7, 3, 2]. These inaccuracies were corrected by Daniel Lehmann who also provided a general semiring framework including non-idempotent semirings [25]. We will describe a general framework for all-pairs shortest-distance problems and refer to previous work for the generic algorithms working within that framework which are generalizations of the Floyd-Warshall or Gauss-Jordan elimination algorithms.[1]

[1]See [8] for a modern presentation in the idempotent case, [20, 7, 43, 40] for various applications and specific instances of the algebraic path problem, [33, 34] for a clear summary of the problem and corresponding solutions and applications, and [15] for an extensive bibliography and description of systolic algorithms for the algebraic path problem.

The paper is organized as follows. In Section 1, we give a brief introduction to the algebraic concepts and results needed for the following sections. In Section 2, we define the general single-source shortest-distance problems and introduce our general semiring framework. Section 3 describes in detail a generic algorithm for computing single-source shortest distances for all semirings covered by our framework. Section 4 introduces the generic topological order queue discipline for computing single-source shortest distances and the specific case of acyclic graphs. Section 5 clarifies the relationship between classical shortest-paths algorithms and our generic algorithm. Some problems such as the computation of the k shortest distances from a fixed source to any other vertex of a graph can also be solved using that algorithm. Section 6 introduces the appropriate semirings to use to solve those problems and analyzes the complexity of our algorithm when used with these semirings.

1. Algebraic Foundations

The general frameworks used in later sections are based on the algebraic structure of semirings. Thus, we first briefly review some definitions and properties of semirings and give classical and new theorems that are relevant to the rest of this paper.

1.1. Semirings

Definition 1 *A* semiring *is a system* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *such that*

1. $(\mathbb{K}, \oplus, \overline{0})$ *is a commutative monoid with* $\overline{0}$ *as the identity element for* $\oplus$*,*
2. $(\mathbb{K}, \otimes, \overline{1})$ *is a monoid with* $\overline{1}$ *as the identity element for* $\otimes$*,*
3. $\otimes$ *distributes over* $\oplus$*: for all* a, b, c *in* $\mathbb{K}$*,*

$$(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c),$$
$$c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b),$$

4. $\overline{0}$ *is an annihilator for* $\otimes$*:* $\forall a \in \mathbb{K},\ a \otimes \overline{0} = \overline{0} \otimes a = \overline{0}$.

The Boolean semiring $(\{0, 1\}, \vee, \wedge, 0, 1)$ is the most well-known semiring. The semiring $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ is the underlying algebraic structure of many classical shortest-paths algorithms and is called the *tropical semiring*.

1.2. Properties

In this section, we give some general definitions of semirings and describe some of their properties relevant to the next sections of this paper.[2]

A semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is said to be *commutative* when the multiplicative operation $\otimes$ is commutative:

$$\forall a, b \in \mathbb{K},\ a \otimes b = b \otimes a.$$

[2]See [23] for basic definitions of semirings and their properties.

Definition 2 *Let* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *be a semiring. An element* $a \in K$ *is* idempotent *if* $a + a = a$. $\mathbb{K}$ *is said to be* idempotent *when all elements of* $\mathbb{K}$ *are idempotent.*

Both the tropical semiring and the Boolean semirings are *idempotent*. Given an idempotent semiring $\mathbb{K}$, one can define a specific partial order on $\mathbb{K}$. Under certain conditions, this partial order is unique.

Lemma 1 *Let* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *be an idempotent semiring, then the relation* $\leq_{\mathbb{K}}$ *defined by*

$$(a \leq_{\mathbb{K}} b) \Leftrightarrow (a \oplus b = a)$$

defines a partial order over $\mathbb{K}$*, called the* natural order over $\mathbb{K}$.[3]

Proof. The reflexivity and antisymmetry of $\leq_{\mathbb{K}}$ follow immediately its definition. Let a, b, and c be elements of $\mathbb{K}$, and assume that $a \oplus b = a$ and $b \oplus c = b$. Then $a \oplus c = (a \oplus b) \oplus c = a \oplus (b \oplus c) = a \oplus b = a$. This shows the associativity of $\leq_{\mathbb{K}}$. □

Definition 3 *Let* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *be a semiring, and* $\leq$ *a partial order over* $\mathbb{K}$. $\mathbb{K}$ *is* negative *if* $\overline{1} \leq \overline{0}$*,* positive *if* $\overline{0} \leq \overline{1}$.

An important property of semirings when dealing with shortest paths problems is *monotonicity*. When monotonicity holds, the computation of shortest distances can be factored.

Definition 4 *Let* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *be a semiring, and* $\leq$ *a partial order over* $\mathbb{K}$. *We say that* $\mathbb{K}$ *is* monotonic *if for all* $a, b, c \in \mathbb{K}$ [4]

1. $(a \leq b) \Rightarrow (a \oplus c \leq b \oplus c)$,
2. $(a \leq b) \Rightarrow (a \otimes c \leq b \otimes c)$,
3. $(a \leq b) \Rightarrow (c \otimes a \leq c \otimes b)$.

Note that in a negative idempotent monotonic semiring: $\forall a \in \mathbb{K},\ a \leq \overline{0}$. This justifies the use of the term *negative semiring*.

Lemma 2 *Let* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *be an idempotent semiring, then* $\mathbb{K}$ *provided with the natural order* $\leq_{\mathbb{K}}$ *is negative and monotonic.*

Proof. In view of Lemma 1, $\leq_{\mathbb{K}}$ defines a partial order on $\mathbb{K}$. Since $\overline{1} \oplus \overline{0} = \overline{1}$, $\mathbb{K}$ is negative. Let a, b and c be in $\mathbb{K}$ and assume that $a \leq_{\mathbb{K}} b$, then $a \oplus b = a$. Hence $(a \oplus b) \oplus c = (a \oplus c)$. Since $\mathbb{K}$ is idempotent we also have $(a \oplus b) \oplus (c \oplus c) = (a \oplus c)$, thus $(a \oplus c) \oplus (b \oplus c) = (a \oplus c) \Leftrightarrow (a \oplus c) \leq_{\mathbb{K}} (b \oplus c)$. $(a \otimes c) \leq_{\mathbb{K}} (b \otimes c)$ and $(c \otimes a) \leq_{\mathbb{K}} (c \otimes a)$ directly result from right and left multiplication of both sides of $a \oplus b = a$ by c. □

[3] As always, the reverse order of $\leq_{\mathbb{K}}$ also defines a partial order.

[4] One can define in a similar way *right* and *left monotonicity* by imposing only the first two conditions or the first and third conditions. In all that follows, unless otherwise specified, one can replace *monotonic* by *right monotonic*.

Idempotence combined with monotonicity and negativity makes the partial order over $\mathbb{K}$ unique.

Proposition 1 *Let* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *be a semiring provided with a partial order* $\leq$*. Assume that* $\mathbb{K}$ *is negative, idempotent, and monotonic, then* $\leq$ *coincides with the natural order* $\leq_{\mathbb{K}}$*.*

Proof. Since $\overline{1} \leq \overline{0}$, by monotonicity we have $\forall b,\ b \leq \overline{0}$, and $\forall (a,b) \in \mathbb{K}^2,\ b \oplus a \leq a$. The monotonicity of $\leq$ and the idempotence of $\mathbb{K}$ give: $\forall (a,b) \in \mathbb{K}^2,\ (a \leq b) \Rightarrow (a (= a \oplus a) \leq b \oplus a)$. Since $\leq$ is antisymmetric, these inequalities imply: $\forall (a,b) \in \mathbb{K}^2$, $(a \leq b) \Rightarrow (a = b \oplus a = a \oplus b)$. Thus $\forall (a,b) \in \mathbb{K}^2,\ (a \leq b) \Rightarrow (a \leq_{\mathbb{K}} b)$. Conversely, let a and b be in $\mathbb{K}$ and assume that $a \leq_{\mathbb{K}} b$, thus $a \oplus b = a$. Since $\overline{1} \leq \overline{0}$, $a \leq \overline{0}$, and $(a \oplus b) \leq b$, that is $a \leq b$. Thus: $\forall (a,b) \in \mathbb{K}^2,\ (a \leq_{\mathbb{K}} b) \Leftrightarrow (a \leq b)$. This ends the proof of the proposition. □

A similar proposition holds in the case of positive idempotent semirings with the reverse order of $\leq_{\mathbb{K}}$.

Definition 5 *Let* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *be a semiring.* $\mathbb{K}$ *is* bounded *if* $\overline{1}$ *is an annihilator for* $\oplus$*:* $\forall a \in \mathbb{K},\ \overline{1} \oplus a = \overline{1}$.

The Boolean semiring $\mathcal{B} = (\{0,1\}, \vee, \wedge, 0, 1)$ is an example of a bounded semiring: $1 \vee 0 = 1 \vee 1 = 1$. We will see other examples of bounded semirings such as the *tropical semiring*.

Lemma 3 *Let* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *be a bounded semiring. Then,* $\mathbb{K}$ *is idempotent.*

Proof. Since $\mathbb{K}$ is bounded, $\overline{1} \oplus \overline{1} = \overline{1}$. The multiplication of both sides of this equality by a gives: $\forall a, a \oplus a = a$. □

For a bounded semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ provided with the natural order $\leq_{\mathbb{K}}$ we have $\forall a \in \mathbb{K},\ \overline{0} \leq_{\mathbb{K}} a \leq_{\mathbb{K}} \overline{1}$. This justifies the choice of the term *bounded*.

Our general framework for single-source shortest distance problems is based on *k-closed* semirings.[5]

Definition 6 *Let* $k \geq 0$ *be an integer. A semiring* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *is* k-closed *if*

$$\forall a \in \mathbb{K},\quad \bigoplus_{n=0}^{k+1} a^n = \bigoplus_{n=0}^{k} a^n.$$

When $k = 0$, the previous expression can be rewritten as: $\forall a \in \mathbb{K},\ \overline{1} \oplus a = \overline{1}$. Thus, the notion of 0-closedness coincides with that of boundedness for commutative semirings.

[5]See [13, 14] for the definition of the class of *locally closed semirings* which includes k-closed semirings. Our semiring framework for single-source shortest-distance problems includes locally closed semirings.

Lemma 4 *Let* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *be a* k*-closed semiring, then for any integer* $l > k$ *and* $a \in \mathbb{K}$

$$\forall a \in \mathbb{K}, \quad \bigoplus_{n=0}^{l} a^n = \bigoplus_{n=0}^{k} a^n.$$

Proof. The proof is by induction on l. By definition, the equality holds with $l = k+1$. Assume now that it holds for $l \geq (k+1)$. By distributivity of $\otimes$ over $\oplus$ and by hypothesis $\forall a \in \mathbb{K}$,

$$\bigoplus_{n=0}^{l+1} a^n = \bigoplus_{n=0}^{l-k-1} a^n \oplus a^{l-k} \otimes \left(\bigoplus_{n=0}^{k+1} a^n\right) = \bigoplus_{n=0}^{l-k-1} a^n \oplus a^{l-k} \otimes \left(\bigoplus_{n=0}^{k} a^n\right) = \bigoplus_{n=0}^{l} a^n.$$

This proves the lemma. □

The following definition provides a general framework for all-pairs shortest-distance problems.

Definition 7 *A semiring* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *is* closed *if*

- *for all* $a \in \mathbb{K}$*, the infinite sum* $\bigoplus_{n=0}^{\infty} a^n$ *is well-defined and in* $\mathbb{K}$*;*
- *associativity, commutativity, and distributivity apply to countable sums. Thus, for any three countable sets* $(a_i)_{i \in I}$ *and* $(b_j)_{j \in J}$ *with*

  $$A = \bigoplus_{i \in I} a_i \in \mathbb{K} \qquad \text{and} \qquad B = \bigoplus_{j \in J} b_j \in \mathbb{K}$$

  the following properties hold.

  - *Associativity: for any partitioning of* I *in* I_k*,* $k \in K$*,* $\bigoplus_{i \in I_k} a_i \in \mathbb{K}$ *and* $A = \bigoplus_{k \in K}(\bigoplus_{i \in I_k} a_i)$*, and a similar property holds with* $\otimes$*;*
  - *commutativity: let* I' *be a permutation of* I*, then* $\bigoplus_{i \in I'} a_i \in \mathbb{K}$ *and* $A = \bigoplus_{i \in I'} a_i$*;*
  - *distributivity:* $\bigoplus_{i \in I, j \in J}(a_i \otimes b_j) \in \mathbb{K}$ *and* $A \otimes B = \bigoplus_{(i,j) \in I \times J}(a_i \otimes b_j)$*.*

This definition of closed semirings is more general than the one used by Cormen et al. [8] since it does not assume idempotence. However, idempotence is not necessary for the proof of correctness of the generic all-pairs shortest-distance algorithms of Floyd-Warshall and Gauss-Jordan [25] (see [25, 8, 33] for the presentation of these algorithms). The running time complexity of these algorithms is

$$O(|Q|^3(T_\oplus + T_\otimes + T_*))$$

where $T_\oplus$, $T_\otimes$ and T_* denote the worst cost of $\oplus$, $\otimes$, and closure. The space complexity of these algorithms is $O(|Q|^2)$. Due to these complexities, these algorithms are often impractical for graphs of several hundred million vertices and edges. The single-source shortest-distance algorithms presented in the next section can be used to solve the all-pairs shortest-distance problem more efficiently for sparse graphs in the case of some semirings such as the *tropical semiring* introduced in the next sections.

2. Single-Source Shortest-Distance Problems

In this section, we define the single-source shortest-distance problem. We consider a semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$, and a weighted directed graph $G = (Q, E, w)$ over $\mathbb{K}$, where Q stands for the set of vertices of G, E for the set of edges, and $w : E \to \mathbb{K}$ the weight function mapping edges to elements of the semiring. Given an edge $e \in E$, we denote by $n[e]$ its destination (or next) vertex, by $p[e]$ its origin (or previous vertex), and by $w[e]$ its weight. Given a vertex $q \in Q$, we denote by $E[q]$ the set of edges leaving q.

A *path* $\pi = e_1 e_2 \ldots e_k$ in G is an element of E^* with consecutive edges: $n[e_i] = p[e_{i+1}]$ for $i = 1, \ldots, k-1$. 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 $w[\bigcup_{i=1}^{n} \pi_i] = \bigoplus_{i=1}^{n} w[\pi_i]$. We denote by $P(q)$ the set of paths from s to $q \in Q$.

The classical single-source shortest paths problem is defined by Bellman-Ford equations [4, 17] with real-valued weights and specific operations: the weights are added along the paths using addition of real numbers (+ operation), and the solution of the equation gives the shortest distance to each vertex $q \in Q$ (min operation). We generalize this problem by considering an arbitrary semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$. The weights are then elements of a set $\mathbb{K}$, they are $\otimes$-multiplied along the paths, and the solution of the equations is the $\oplus$-sum of the weights of the paths from the source to each vertex $q \in Q$.

Let $s \in Q$ be a fixed vertex of G called the *source*. For any vertex $q \in Q$, we denote by $\delta(s, q)$ the *shortest distance* from s to q associated to the weighted directed graph G and define it by[6]

$$\begin{cases} \delta(s, s) = \overline{1}, \\ \forall q \in Q - \{s\},\ \delta(s, q) = \displaystyle\bigoplus_{\pi \in P(q)} w[\pi] \end{cases} \tag{1}$$

when all these summations are well-defined and in $\mathbb{K}$. In the following, we will consider a class of semirings for which this assumption holds. Note that in general the notion of *shortest path* is not meaningful since with some semirings there might be no path from s to q with weight $\delta(s, q)$. Thus, in general, we will restrict ourselves to the use of the term *shortest distance*.

Given a graph G, we can extend the definition of k-closed semirings in the following way.

[6]If $P(q) = \emptyset$, the summation is defined to be $\overline{0}$. The first equation ($\delta(s, s) = \overline{1}$) can be omitted. Its presence does not affect the generality of the problem and the algorithm described later however. Indeed, the addition of a new source vertex s connected to the previous source by an edge with weight $\overline{1}$ leads to an equivalent problem where $\delta(s, s) = \overline{1}$.

Definition 8 *Let* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *be a commutative semiring,* $G = (Q, E, w)$ *a weighted directed graph over* $\mathbb{K}$ *and* $k \geq 0$ *an integer.* $\mathbb{K}$ *is* k-closed for G *if for any cycle* π *in* G*:*

$$\bigoplus_{n=0}^{k+1} w[\pi]^n = \bigoplus_{n=0}^{k} w[\pi]^n.$$

For each vertex $q \in Q$, we denote by $P_k(q)$ the set of paths from s to q with at most k occurrences of a cycle. It is not hard to show that the set of paths in $P_k(q)$ is finite. When $k = 0$, $P_k(q)$ is the set of *simple paths* from s to q. By Lemma 4, for a semiring $\mathbb{K}$ k-closed for G, we have

$$\forall\, l \geq k, \bigoplus_{\pi \in P_l(q)} w[\pi] = \bigoplus_{\pi \in P_k(q)} w[\pi].$$

Since $P_\infty(q) = P(q)$, this leads us to define

$$\bigoplus_{\pi \in P(q)} w[\pi] = \bigoplus_{\pi \in P_k(q)} w[\pi]$$

and thereby define the shortest distances $\delta(s, q)$ when $\mathbb{K}$ is k-closed for G. The semiring framework that we consider in the following is that of commutative semirings $\mathbb{K}$ k-closed for G. As an illustration of the choice of our framework, consider the semiring $(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$ used in classical shortest-paths problems. It is not bounded since $\min\{-1, 0\} \neq 0$. But, if the graph G has no negative cycle, this semiring is 0-closed (bounded) for G: the weight $w[c]$ of any cycle c is non-negative, thus $\min\{w[c], 0\} = 0$.

3. Generic Single-Source Shortest-Distance Algorithm

In this section, we present a generic algorithm to compute single-source shortest distances for semirings covered by our framework. The algorithm is generic in two senses: it works with any semiring that falls within this framework and with any choice of the queue discipline.

Our framework includes semirings that are not idempotent. For example, we may consider the ring $(\mathbb{R}^{n\times n}, +, \cdot, 0_n, I_n)$ of square $n \times n$ matrices over $\mathbb{R}$, $n > 0$, which is not idempotent ($I_n + I_n \neq I_n$). If $G = (Q, E, w)$ is a weighted directed graph over this ring such that its cycles are all weighted with powers of a nilpotent matrix N then $(\mathbb{R}^{n\times n}, +, \cdot, 0_n, I_n)$ is n-closed for G since $N^{n+1} = 0$. In Section 6.2, we will present an infinite family of non-idempotent k-closed semirings, *k-tropical semirings*, and their application to the computation of the k-shortest distances from a given source vertex to each vertex of a graph.

Before presenting our generic algorithm, let us first mention that a straightforward extension of the classical algorithms based on a relaxation technique would not produce the correct result for non-idempotent semirings. Indeed, consider the weighted graph of Figure 1. Regardless of the order in which the vertices are visited, the resulting shortest distances will not be correct.

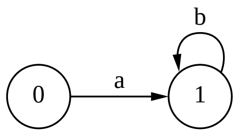

Figure 1: Single-source shortest distances for non-idempotent semirings

The successive values of a tentative shortest distance from the source 0 to the vertex 1 will be: a, then $a \oplus (a \otimes b) = a \otimes (\overline{1} \oplus b)$, then $a \otimes (\overline{1} \oplus b) \oplus a \otimes (\overline{1} \oplus b) \otimes b = a \otimes (\overline{1} \oplus b)^2, \ldots, a \otimes (\overline{1} \oplus b)^n, \ldots$ Thus, assuming that the algorithm converges within a finite number of iterations N, then the result will be: $a \otimes (\overline{1} \oplus b)^N$ which in general could be different from the expected and correct result $a \otimes b^*$, even if $b^N = b^*$ for the semiring considered.

3.1. Proofs and Algorithm

We present a generic algorithm for solving single-source shortest-distance problems. Our algorithm is based on a generalization of the classical *relaxation* technique. As seen earlier, a straightforward extension of the relaxation technique would lead to an algorithm that would not work with non-idempotent semirings. To deal properly with multiplicities in the case of non-idempotent semirings, we keep track of the changes to the tentative shortest distance from s to q after the last extraction of q from the queue. In Figure 2 the pseudocode of the algorithm is presented.

```
Generic-Single-Source-Shortest-Distance (G, s)
for i ← 1 to |Q|
      do d[i] ← r[i] ← 0̄
d[s] ← r[s] ← 1̄
S ← {s}
while S ≠ ∅
        do q ← head(S)
           Dequeue(S)
           r′ ← r[q]
           r[q] ← 0̄
           for each e ∈ E[q]
               do if d[n[e]] ≠ d[n[e]] ⊕ (r′ ⊗ w[e])
                     then d[n[e]] ← d[n[e]] ⊕ (r′ ⊗ w[e])
                          r[n[e]] ← r[n[e]] ⊕ (r′ ⊗ w[e])
                          if n[e] ∉ S
                             then Enqueue(S, n[e])
d[s] ← 1̄
```

Figure 2: Pseudocode of a generic algorithm for solving single-source shortest-distance problems

We use a queue S to maintain the set of vertices whose leaving edges are to be relaxed. S is initialized to $\{s\}$ (Line 4). For each vertex $q \in Q$, we maintain two attributes: $d[q] \in \mathbb{K}$ an estimate of the shortest distance from s to q, and $r[q] \in \mathbb{K}$ the total weight added to $d[q]$ since the last time q was extracted from S. Lines 1–3 initialize arrays d and r. After initialization, $d[q] = r[q] = \overline{0}$ for $q \in Q - \{s\}$, and $d[s] = r[s] = \overline{1}$.

Given a vertex $q \in Q$ and an edge $e \in E[q]$, a *relaxation* step on e is the sequence of instructions of Lines 11–13 and denoted by Relax(q, e). r' is the value of $r[q]$ just after the last extraction of q from S.

Each time through the **while** loop of Lines 5–15, a vertex q is extracted from S (Lines 6–7). The value of $r[q]$ just after extraction of q is stored in r', and then $r[q]$ is set to $\overline{0}$ (Lines 8–9). Lines 11–13 relax each edge leaving q. If the tentative shortest distance $d[n[e]]$ is updated during the relaxation and if $n[e]$ is not already in S, the vertex $n[e]$ is inserted in S so that its leaving edges be later relaxed (Lines 14–15). $r[n[e]]$ is updated whenever $d[n[e]]$ is to keep track of the total weight added to $d[n[e]]$ since $n[e]$ was last extracted from S or since the time after initialization if $n[e]$ has never been extracted from S. Finally, Line 16 resets the value of $d[s]$ to $\overline{1}$.

To prove the correctness and termination of the algorithm, we define for each vertex $q \in Q$ two subsets of $P(q)$, $D(q)$ and $R(q)$, such that at any time during the execution of the algorithm:

$$d[q] = \bigoplus_{\pi \in D(q)} w[\pi] \qquad \text{and} \qquad r[q] = \bigoplus_{\pi \in R(q)} w[\pi].$$

These sets and a working subset R' are defined as follows. Just after each execution of the instruction of

- Line 2: $D(i) \leftarrow R(i) \leftarrow \emptyset$;
- Line 3: $D(s) \leftarrow R(s) \leftarrow \{\epsilon\}$;[7]
- Line 8: $R' \leftarrow R(q)$;
- Line 9: $R(q) \leftarrow \emptyset$;
- Line 10: $D(n[e]) \leftarrow D(n[e]) \cup R'e$;
- Line 13: $R(n[e]) \leftarrow R(n[e]) \cup R'e$.

Since $R(q)$ is set to the empty set after each extraction of vertex q from the queue S and augmented with the set of paths R' whenever $d[q]$ is updated, $R(q)$ represents at any time the set of all paths whose weight has been added to $d[q]$ since the last extraction of q ($R(q) = \emptyset$ before any extraction of q).

Similarly, $D(q)$ represents the set of all paths π such that π belonged to R' at some time, thus the set of all paths whose weight was compared to $d[q]$ via the relaxation of Lines 11–15 and by construction:

$$d[q] = \bigoplus_{\pi \in D(q)} w[\pi].$$

[7]We use the following convention in what follows: $w[\epsilon] = \overline{1}$.

The following lemma shows that the instruction $D(n[e]) \leftarrow D(n[e]) \cup R'e$ is always executed with $R'e \cap D(n[e]) = \emptyset$, in other words that only new paths are added to $D(n[e])$.

Lemma 5 *At any time during the execution of the algorithm, just after extraction of vertex q from queue S (Line 6), for any edge e leaving q: $R(q)e \cap D(n[e]) = \emptyset$.*

Proof. The proof is by induction on the number of queue extractions. The property clearly holds before any queue extraction since only $D(s)$ is non-empty at that point and $R(q)e \cap D(s) = R(q)e \cap \{\epsilon\} = \emptyset$ for any edge e regardless of the value of $R(q)$.

Assume now that the property holds just after any queue extraction before extraction of $q = p[e]$ at time t_0. If $R(q)e \cap D(n[e]) \neq \emptyset$ at time t_0, then there exists a path π_1 such that $\pi_1 e \in R(q)e \cap D(n[e])$. Thus, at time t_0, $\pi_1 \in R(q)$ and $\pi_1 e \in D(n[e])$.

Since $\pi_1 e \in D(n[e])$, there exists a time $t' < t_0$ where edge e has been relaxed with $\pi_1 \in R'$, $R[q] = \emptyset$ and $\pi_1 e \in D(n[e])$. Let t'_0 be the time of the most recent queue extraction before t'. At time t'_0, q has been extracted from the queue with $\pi_1 \in R(q)$, thus $\pi_1 \in D(q)$ at time t'_0.

Since $\pi_1 \in R(q)$ at time t_0 and $R(q) = \emptyset$ at time $t' < t_0$, there exists a time t'' with $t' < t'' < t_0$ where π_1 has been added to $R(q)$. This implies in particular that $\pi_1 \neq \epsilon$, thus there exists a path π_0 and an edge e_1 such that $\pi_1 = \pi_0 e_1$. Let t''_0 be the most recent queue extraction before t'', we have $t'_0 < t''_0 < t'' < t'_0 < t_0$. At time t''_0, $p[e_1]$ has been extracted from the queue with $\pi_0 \in R(p[e_1])$ and $\pi_1 \in R(p[e_1])e_1$. Thus, at time t''_0, $R[p[e_1]]e_1 \cap D(q) \neq \emptyset$, but this contradicts the induction hypothesis. □

Lemma 6 *At any time during the execution of the algorithm and for any vertex $q \in Q$:*

$$\pi = \pi_1\pi_2 \in D(q) \Rightarrow \pi_1 \in D(p[\pi_2]).$$

Proof. By construction of D, a path πe is added to the set $D(n[e])$ only if π is a path in R'. Hence, π is in $R(p[e])$ at the time of the previous extraction of $p[e]$. All the paths in $R(p[e])$ have been added to $D(p[e])$ since the last extraction of $p[e]$, therefore π is in $D(p[e])$. Thus $\pi e \in D(n[e]) \Rightarrow \pi \in D(p[e])$ and a repeated application of this implication proves the lemma. □

Lemma 7 *Assume that the equality $d[p[e]] = d[p[e]] \oplus x$ holds at time t during the execution of the algorithm for some $x \in \mathbb{K}$ and assume that edge e is later relaxed at time $t' > t$, then at any time after that relaxation*

$$d[n[e]] = d[n[e]] \oplus (x \otimes w[e]).$$

Proof. The new value of $d[p[e]]$ at time t' is of the form: $d[p[e]] \oplus y$. Let d_0 (respectively d_1) be the value of $d[n[e]]$ just before (respectively just after) relaxation. By definition of relaxation, we have

$$d_1 = d_0 \oplus ((d[p[e]] \oplus y) \otimes w[e]).$$

Thus

$$\begin{aligned} d_1 \oplus (x \otimes w[e]) &= d_0 \oplus ((d[p[e]] \oplus x \oplus y) \otimes w[e]) \\ &= d_0 \oplus ((d[p[e]] \oplus y) \otimes w[e]) = d_1. \end{aligned}$$

Future modifications of $d[n[e]]$ will only $\oplus$-add some value $z \in \mathbb{K}$ to $d[n[e]]$ and won't affect the equality. This proves the lemma. □

Corollary 1 *Let $\pi = e_1 \ldots e_n$ be a path of G. Assume that the equality $d[p[e_1]] = d[p[e_1]] \oplus x$ holds at time t during the execution of the algorithm for some $x \in \mathbb{K}$ and assume that edges $e_1 \ldots e_n$ are later relaxed respectively at times $t_1 < \cdots < t_n$, $t < t_1$, then at any time after t_n*

$$d[n[\pi]] = d[n[\pi]] \oplus (x \otimes w[\pi]).$$

Proof. The result is a direct consequence of a repeated application of Lemma 7. □

Lemma 8 *Let $\pi_0, \ldots, \pi_n$ be paths in G and c a cycle. Let us assume that we have $\pi_0 c \pi_1 c \ldots \pi_{n-1} c \pi_n \in D(q)$ at some time t during the execution of the algorithm, then $\pi_0 \ldots \pi_n \in D(q)$ or there exists $x \in \mathbb{K}$ such that $d[q] = d[q] \oplus w[\pi_0 \ldots \pi_n] \oplus x$ at any time after t.*

Proof. Let $\pi_1 \pi_2 \ldots \pi_n = e_1 \ldots e_m$. Since c is a cycle and $\pi_0 c \ldots \pi_{n-1} c \pi_n \in D(q)$, edges $e_1, \ldots, e_m$ have been relaxed first (at least once) at times respectively $t_1 < \cdots < t_m$ and by Lemma 6, $\pi_0 \in D(p[c\pi_1 c \ldots \pi_{n-1} c \pi_n]) = D(n[\pi_0])$. Assume that $\pi_0 \ldots \pi_n \notin D(q)$, then there exists $i < m$ maximum such that $\pi_0 e_1 \ldots e_i \in D(n[\pi_0 e_1 \ldots e_i])$. By definition of i, $\pi_0 e_1 \ldots e_i e_{i+1}$ cannot be in $R(n[\pi_0 e_1 \ldots e_i])$ at any time before t. Thus, the test of Line 11 must not have been passed during the relaxation of edge e_{i-1}, and at time t_{i-1} [8]

$$d[n[\pi_0 e_1 \ldots e_i]] = d[n[\pi_0 e_1 \ldots e_i]] \oplus (w[R(n[\pi_0 e_1 \ldots e_{i-1}])] \otimes w[e_i]).$$

Since t_{i-1} is the first relaxation of e_{i-1} after t_{i-2}, we have $\pi_0 e_1 \ldots e_{i-1} \in R(n[\pi_0 e_1 \ldots e_{i-1}])$. Thus, there exists x such that

$$d[n[\pi_0 e_1 \ldots e_i]] = d[n[\pi_0 e_1 \ldots e_i]] \oplus w[\pi_0 e_1 \ldots e_i] \oplus x.$$

By Corollary 1, this implies that

$$d[n[\pi_0 e_1 \ldots e_m]] = d[n[\pi_0 e_1 \ldots e_m]] \oplus (w[\pi_0 e_1 \ldots e_i] \oplus x) \otimes w[e_{i+1} \ldots e_n].$$

Since $n[\pi_0 e_1 \ldots e_m] = n[\pi_0 \pi_1 \ldots \pi_n]$, this proves the lemma. □

Lemma 9 *At any time during the execution of the algorithm, if $R'e \cap P_k(n[e]) = \emptyset$, then the test of Line 11 is not passed.*

[8] Note that since c is a cycle, $n[\pi_0 e_1 \cdots e_i] = n[\pi_0 e_1 \cdots e_{i-1}]$.

Proof. Let π be a path such that $\pi e \in R'e - P_k(n[e])$ at time t. By definition of $P_k(n[e])$, πe is a path with strictly more than k occurrences of a cycle c and there exist paths $\pi_1, \ldots, \pi_{k+1}$ such that

$$\pi e = \pi_1 c \ldots \pi_{k+1} c.$$

Since $\pi \in R(p[e])$, by Lemma 6, $\pi_1 c \ldots \pi_k c \pi_{k+1} \in D(n[e])$. For $i = 1, \ldots, k+1$, let $\Pi_i = \pi_1 c \pi_2 c \ldots \pi_i c \pi_{i+1} \pi_{i+2} \ldots \pi_{k+1}$ and $\Pi_0 = \pi_1 \ldots \pi_{k+1}$. By Lemma 8, at time t, for any i, $\Pi_i \in D(n[e])$ or there exists $x_i \in \mathbb{K}$ such that: $d[n[e]] = d[n[e]] \oplus w[\Pi_i] \oplus x_i$. Thus, at time t just before the relaxation of e, there exists $X \in \mathbb{K}$ such that

$$d[n[e]] = \bigoplus_{i=0}^{k} w[\Pi_i] \oplus X = w[\pi_1 \ldots \pi_n] \otimes \bigoplus_{i=1}^{k} w[c]^i \oplus X.$$

Since $\mathbb{K}$ is k-closed for G:

$$\begin{aligned} d[n[e]] \oplus w[\pi e] &= w[\pi_1 \ldots \pi_n] \otimes \bigoplus_{i=1}^{k} w[c]^i \oplus X \oplus w[\pi_1 \ldots \pi_n] \otimes w[c]^{k+1} \\ &= w[\pi_1 \ldots \pi_n] \otimes \bigoplus_{i=1}^{k} w[c]^i \oplus X = d[n[e]]. \end{aligned} \tag{2}$$

Thus, if $R'e \subseteq P(q) - P_k(n[e])$, then

$$d[n[e]] \oplus (r' \otimes w[e]) = d[n[e]] \oplus w[R'e] = d[n[e]]$$

and the test of Line 11 is not passed. □

Theorem 1 *Let $G = (Q, E, w)$ be a weighted directed graph over $\mathbb{K}$ and s a fixed source vertex. Assume that $\mathbb{K}$ is k-closed for G, then if we run the* Generic-Single-Source-Shortest-Distance *algorithm on the weighted graph G and source $s \in Q$, the algorithm terminates and at termination for any vertex q, $d[q] = \delta(q)$.*

Proof. By Lemma 9, the instructions of Lines 12–15 are executed only if R' contains at least one path in $P_k(n[e])$ and by Lemma 5 that path does not already belong to $D(n[e])$. Thus, for each vertex $q = n[e]$, the instructions of Lines 12–15 can be executed at most $\text{card}(P_k(q))$ times and a vertex q can only be inserted in S a finite number of times. This proves that the algorithm terminates.

By construction, at any time during the execution of the algorithm, for any vertex q, $d[q] = w[D(q)]$, with $D(q) \subseteq P(q)$. The proof of Lemma 9 shows that the paths in $D(q)$ that are not in $P_k(q)$ do not contribute to the value of $d[q]$, thus in fact we can write: $d[q] = w[D'(q)]$, with $D'(q) = D(q) \cap P_k(q)$.

Assume that the algorithm does not compute the correct shortest-distances, then at termination, there exits a vertex q and a path $\pi = e_1 \ldots e_n \in P_k(q) - D'(q)$ such that $d[q] \neq d[q] \oplus w[\pi]$. Let q and π be such that π is the shortest path having this property. π cannot be reduced to just one transition ($n = 1$), since vertex s is extracted at least once and thus e_1 has been relaxed at least once and $e_1 \in D(n[e_1])$ after that relaxation. Thus, $n \geq 2$ and by definition of π

$$d[q_{n-1}] = d[q_{n-1}] \oplus w[e_1 \ldots e_{n-1}] \tag{3}$$

with $q_{n-1} = n[e_1 \ldots e_{n-1}]$, and

$$d[q] \neq d[q] \oplus w[e_1 \ldots e_n]. \tag{4}$$

Eq. (4) implies that $w[e_1 \ldots e_n] \neq \overline{0}$, thus we also have $w[e_1 \ldots e_{n-1}] \neq \overline{0}$ and in view of Eq. (3), we also have $d[q_{n-1}] \neq \overline{0}$. Since $d[q_{n-1}] \neq \overline{0}$, q_{n-1} has been inserted in S at least once and thus e_n has been relaxed at least once. Let t be the time at which e_n was last relaxed before termination. If the Eq. (3) held at time t, then by Corollary 1 the following holds after relaxation and at any time after

$$d[q] = d[q] \oplus w[e_1 \ldots e_{n-1}] \otimes w[e_n]$$

which contradicts Eq. (4). It the Eq. (3) did not hold at time t, then the value of $d[q_{n-1}]$ must have been changed after t and there must have been a later relaxation of the edges leaving q_{n-1} at time $t' > t$ which contradicts the definition of t. □

The proof of the correctness and termination of the algorithm did not require specifying the queue discipline used. In other words, vertices can be extracted from S in any chosen order and this won't affect the correctness of the algorithm or its termination. The algorithm is thus *generic* in that sense and also in the sense that it works for different semirings. The choice of the queue discipline can directly affect the complexity of the algorithm however.

3.2. Complexity

The running time complexity of the Generic-Single-Source-Shortest-Distance algorithm depends on the semiring $\mathbb{K}$ and the operations $\oplus$ and $\otimes$ used. We denote by $T_\oplus$ the time to compute $\oplus$ and by $T_\otimes$ the time to compute $\otimes$.[9] We denote by $N(q)$ the number of times the vertex q is inserted in S when the generic algorithm is run on the directed graph G. We denote by $C(E)$ the worst cost of removing a vertex q from the queue S (Lines 6–7 of the pseudocode) and by $C(I)$ the worst cost of inserting q in S. During a relaxation call, a tentative shortest distance may be updated (Line 12). This may also affect the queue discipline. We denote by $C(A)$ the worst cost of an assignment including the possible cost of reorganizing the queue to perform this assignment.

The initialization step of the algorithm (Lines 1–3) takes $O(|Q|)$ time, each relaxation (Lines 11–13) takes $O(T_\oplus + T_\otimes + C(A))$ time. There are exactly $N(q)|E[q]|$ relaxations at q. The total cost of the relaxations is thus $O((T_\oplus + T_\otimes + C(A)) \cdot |E| \max_{q \in Q} N(q))$. Since each vertex q is inserted in S $N(q)$ times (Line 15), it is also extracted from S $N(q)$ times (Lines 6–7), and the total complexity of the algorithm is

$$O(|Q| + (T_\oplus + T_\otimes + C(A))|E| \max_{q \in Q} N(q) + (C(I) + C(E)) \sum_{q \in Q} N(q)).$$

As mentioned before, the Generic-Single-Source-Shortest-Distance algorithm works with any queue discipline. Some queue disciplines are better than others. The appropriate choice depends on the semiring $\mathbb{K}$ and the specific restrictions

[9]In general, these computation times might heavily depend on the operands. We make the assumption here that they are bounded and that $T_\oplus$ and $T_\otimes$ correspond to the upper bounds.

imposed on G. With a good choice, the maximum number of times a vertex is inserted in S ($\max_{q \in Q} N(q)$) can be limited.[10] The algorithm is then very efficient. In the next sections, we will present some classical algorithms using the following queue disciplines: *topological order*, *shortest-first order*, *first-in first-out order*. In the worst case, since the number of simple paths from s to q may be exponential in the size of the graph ($|Q| + |E|$), the complexity of the algorithm is exponential.

The results presented in this section can be extended to cover the case of non-commutative semirings [28]. Our framework can also be generalized by introducing *right* and *left semirings* [28].[11] A right semiring is an algebraic structure similar to a semiring except that it may lack left distributivity. A left semiring is defined in a similar way. An example of left semiring is the *string semiring* $(\Sigma^* \cup \{\infty\}, \wedge, \cdot, \infty, \epsilon)$ defined on the set of strings over an alphabet Σ [29]. Our generic shortest-distance algorithm can be used with the left semiring in the first step of the minimization of subsequential transducers [29]. Apart from generalizations of this type, it seems that our general framework covers essentially all semirings for which the algorithm works. Indeed, the condition in the definition of k-closed semirings on the convergence after k iterations is necessary for the computation of the weight of the shortest distance in presence of a loop.

Some semirings such as $\mathcal{R} = (\mathbb{R}, +, \cdot, 0, 1)$ do not verify the conditions of the framework for our general single-source shortest-distance algorithm, but are covered by the general framework of the all-pairs shortest-distance algorithm we described in a previous section. The generalized algorithms of Floyd-Warshall or Gauss-Jordan can be used to solve the single-source shortest problem with such semirings but their cubic time complexity makes them impractical for many large graphs of several hundred million edges encountered in many applications. One can decompose the graph into its strongly connected components, use Floyd-Warshall or Gauss-Jordan's algorithms for computing the all-pairs shortest-distances within each strongly connected component and then find all-pairs shortest-distances by considering the acyclic component graph [30]. But the solution remains impractical in presence of large strongly connected components.

One can then have recourse to various approximations of Floyd-Warshall and Gauss-Jordan algorithms, but such algorithms do not fully exploit the sparsity of the graphs. We have devised an approximate single-source shortest-distance algorithm that can be viewed as an alternative and that is orders of magnitude faster in practice to compute single-source shortest distances in such cases. The algorithm is derived from our generic single-source shortest distance algorithm when $\mathbb{K}$ is provided with a metric Δ by replacing the relaxation condition $d[n[e]] \neq d[n[e]] \oplus (r' \otimes w[e])$ by an approximate test

$$\Delta(d[n[e]], d[n[e]] \oplus (r' \otimes w[e])) \leq \epsilon$$

where $0 \leq \epsilon$ is a positive number used for approximation.

[10] This number is a constant or is linear in $|Q|$ in many classical algorithms.

[11] These generalizations tend to lengthen the proofs and the overall presentation, thus we chose not to present them here. The same generic algorithm can be used in those generalized cases.

In the next section, we present a variant of the generic algorithm which guarantees a linear time complexity when used with acyclic weighted directed graphs.

4. Generic Topological Order Single-Source Shortest-Distance Algorithm

As mentioned before, the generic single-source shortest-distance algorithm works with any queue discipline. Here, we restrict the queue discipline so that in the case of acyclic graphs the order be topological.

A weighted directed graph $G = (Q, E, w)$ can be decomposed into *strongly connected components* [39]. The strongly connected components of a directed graph, SCCs in short, are the equivalence classes of its vertices under the relation R defined by $q \; R \; q'$ if there is a path in G from q to q', and a path from q' to q. The corresponding decomposition, the *component graph* of G, is an acyclic directed graph that has one vertex for each SCC of G, and an edge from u to v if there exists an edge from the SCC of G corresponding to u to the SCC of G corresponding to v. Since the component graph is acyclic, its vertices, which correspond to the SCCs of G, can be *topologically sorted*. A topological sort of a graph G' is an ordering of its vertices such that if G' contains an edge e from q to q', then q appears before q' in the ordering. The computation of the SCCs of G and that of the topological sort of the SCCs can all be done in linear time $O(|Q| + |E|)$ [22, 2, 8].

Assume that the SCCs of G are defined and topologically ordered. We can then number them according to the topological order: $S_1, S_2, \ldots, S_N$. The queue discipline used for S in the Generic-Single-Source-Shortest-Distance algorithm can be restricted by imposing that the SCC containing the vertex extracted from S have the lowest number. In other words, as long as S contains vertices that are in S_1 the next vertex extracted must be in S_1, then we proceed with S_2 and so forth. By definition of the topological order, if the vertices contained in G are all in $\bigcup_{i>i_0} S_i$, $1 \leq i_0 \leq N$, at time t, then a vertex of S_{i_0} cannot be inserted in the queue S at any time after t, since there is no edge from a vertex in $\bigcup_{i>i_0} S_i$ to a vertex in S_{i_0}.

This defines a variant of the generic algorithm, Generic-Topological-Single-Source-Shortest-Distance, which consists of computing the shortest distances for the vertices in the SCC S_1, then proceed with S_2, etc. Figures 3–4 give the pseudocode of the algorithm. The arrays d and r are initialized as in the previous generic algorithm (Lines 1–3, Figure 4). Then, for each SCC X of G considered in topological order, the procedure SCC-Single-Source-Shortest-Distance (G, X) is called. Line 1 (Figure 3) inserts in the queue S the set of vertices q in X such that $r[q] \neq \overline{0}$, that is the vertices q whose shortest distances have been modified since the initialization. The pseudocode is then identical to that of the generic algorithm, except from Line 11, which imposes the vertex $n[e]$ to be in X in order to be inserted in S. This corresponds to the restriction over the queue discipline that we defined above which imposes that the vertices of X be considered before those of SCCs that come after X in the topological order. Line 6 resets the value of $d[s]$ to $\overline{1}$ as in the generic algorithm.

Note that within each SCC X an arbitrary queue discipline can be chosen for S.

```
SCC-Single-Source-Shortest-Distance (G, X)
  S ← {q ∈ X : r[q] ≠ 0̄}
  while S ≠ ∅
        do q ← head(S)
           Dequeue(S)
           r ← r[q]
           r[q] ← 0̄
           for each e ∈ E[q]
               do if d[n[e]] ≠ d[n[e]] ⊕ (r ⊗ w[e])
                     then d[n[e]] ← d[n[e]] ⊕ (r ⊗ w[e])
                          r[n[e]] ← r[n[e]] ⊕ (r ⊗ w[e])
                          if n[e] ∉ S and n[e] ∈ X
                             then Enqueue(S, n[e])
```

Figure 3: Single-source shortest-distance algorithm restricted to the set X

Theorem 2 *Let $G = (Q, E, w)$ be a weighted directed graph over $\mathbb{K}$ and s a fixed source vertex. Assume that $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is a semiring k-closed for G, then if we run the* Generic-Topological-Single-Source-Shortest-Distance *algorithm on the weighted graph G and source $s \in Q$, the algorithm terminates, and at termination for any vertex $q \in Q$, $d[q] = \delta(s, q)$.*

Proof. The topological single-source shortest-distance algorithm is a specific case of the generic algorithm where the choice of the queue discipline is more restricted. By Theorem 1, Generic-Single-Source-Shortest-Distance works with any queue discipline. Thus, Generic-Topological-Single-Source-Shortest-Distance terminates and computes exactly the shortest distance to each vertex $q \in Q$. □

When the graph G is acyclic, the algorithm works with any semiring.

Corollary 2 *Let $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ be a semiring. Let $G = (Q, E, w)$ be an acyclic weighted directed graph over $\mathbb{K}$ and s a fixed source vertex. If we run the* Generic-Topological-Single-Source-Shortest-Distance *algorithm on the weighted graph G and source $s \in Q$, the algorithm terminates, and at termination for any vertex $q \in Q$, $d[q] = \delta(s, q)$.*

Proof. When G is acyclic, by definition, any semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is right k-closed for G. □

The algorithm of Lawler [24] is a special case of Generic-Topological-Single-Source-Shortest-Distance algorithm when used with an acyclic graph weighted with real-valued numbers. The Generic-Topological-Single-Source-Shortest-Distance algorithm is useful among other things for computing the coefficients of a rational power series represented by a weighted automaton or a weighted transducer [5, 6, 10, 36, 23].

An efficient implementation of the GENERIC-SINGLE-SOURCE-SHORTEST-DISTANCE algorithm with various queue disciplines including the generic topological order can be found in the latest version of the FSM library [30].

4.1. Complexity

The complexity of the GENERIC-TOPOLOGICAL-SINGLE-SOURCE-SHORTEST-DISTANCE algorithm is not different from that of GENERIC-SINGLE-SOURCE-SHORTEST-DISTANCE in the general case. But we can show that the complexity of the topological single-source shortest-distance algorithm is linear when G is acyclic.

GENERIC-TOPOLOGICAL-SINGLE-SOURCE-SHORTEST-DISTANCE (G, s)
for $i \leftarrow 1$ **to** $|Q|$
  do $d[i] \leftarrow r[i] \leftarrow \overline{0}$
$d[s] \leftarrow r[s] \leftarrow \overline{1}$
for each SCC X considered in topological order
  do SCC-SINGLE-SOURCE-SHORTEST-DISTANCE (G, X)
$d[s] \leftarrow \overline{1}$

Figure 4: Generic topological single-source shortest-distance algorithm

When G is acyclic, each strongly connected component X is reduced to a single vertex q. q cannot be reinserted in S after relaxation of the edges leaving q since G is acyclic. Thus, each vertex q is inserted in S at most once: $\forall q \in Q,\ N(q) \leq 1$. There are exactly $|E[q]|$ relaxations calls made in the procedure SCC-SINGLE-SOURCE-SHORTEST-DISTANCE (G, X) for each vertex q. As mentioned earlier, the topological sort can be done in linear time $O(|Q| + |E|)$. The test of Line 11 (Figure 3) can be done is constant time since X is reduced to a single vertex. The cost of the extraction from S is constant since S contains at most one vertex, $C(E) = O(1)$. The cost of an assignment $C(A)$ is also constant since it does not affect the topological order.

Thus, based on the general complexity formula given in the previous section for the GENERIC-SINGLE-SOURCE-SHORTEST-DISTANCE algorithm, the running time complexity of the GENERIC-TOPOLOGICAL-SINGLE-SOURCE-SHORTEST-DISTANCE algorithm is

$$O(|Q| + (T_\oplus + T_\otimes)|E|).$$

In next sections, we will examine the use of the generic algorithms just presented with various semirings. This will show how the same algorithm can be used in different contexts by just modifying the underlying algebra.

5. Classical Shortest-Distance Algorithms

Classical shortest-distance algorithms such as Dijkstra's algorithm and the Bellman-Ford algorithm are special cases of the generic single-source shortest-distance algorithm. They correspond to the case where $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is the *tropical semiring*.

5.1. Tropical Semirings

The operations used in many optimization problems are min and $+$. Traditional shortest-paths problems used in various applications are specific instances of such general optimization problems. The semirings associated to these operations are called *tropical semirings* due to the extensive work of Imre Simon in Brazil relating to these semirings [38]. See [1] and [32] for more specific presentations and discussions of tropical semirings.

It is not hard to verify that the system $\mathcal{T} = (\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ defines a semiring over the set of non-negative numbers $\mathbb{R}_+$ completed with the infinity element ∞, when the min and $+$ operations are extended in the following way:

$$\forall a \in \mathbb{R}_+ \cup \{\infty\},\ \min\{\infty, a\} = \min\{a, \infty\} = a$$

and

$$\forall a \in \mathbb{R}_+ \cup \{\infty\},\ \infty + a = a + \infty = \infty.$$

We will use the term *tropical semiring* to refer to the semiring $\mathcal{T}$. Note that the natural order over $\mathcal{T}$ is just the usual order of real numbers (see Lemma 1):

$$\forall a, b \in \mathbb{R}_+ \cup \{\infty\}, (\min\{a, b\} = a) \Leftrightarrow (a \leq b).$$

In the same way, we can define other tropical semirings such as

1. $(\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$: *non-negative natural tropical semiring*,
2. $(\mathbb{Z} \cup \{\infty\}, \min, +, \infty, 0)$: *natural tropical semiring*,
3. $(\mathbb{Q} \cup \{\infty\}, \min, +, \infty, 0)$: *rational tropical semiring*,
4. $(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$: *real tropical semiring*,
5. $(\mathbb{N} \cup \{\omega, \infty\}, \min, +, \infty, 0)$: *ordinal tropical semiring*, with the order [26]

   $$0 \leq 1 \leq 2 \leq \cdots \leq \omega \leq \infty,$$

   and the following extension of addition:

   $$\forall a \in \mathbb{N} \cup \{\omega, \infty\},\ a + \omega = \omega + a = \max\{a, \omega\}.$$

These semirings can be extended to include $-\infty$. For example, $(\mathbb{R} \cup \{-\infty, +\infty\}, \min, +, +\infty, 0)$ is also a semiring. From our perspective here, an essential property of many of the tropical semirings is that they are all idempotent and that some such as the tropical semiring $\mathcal{T}$ are bounded:

$$\forall a \in \mathbb{R}_+ \cup \{\infty\},\ \min\{0, a\} = 0.$$

Thus, the generic algorithms presented in previous sections can be used with the tropical semiring. In particular, the classical shortest-path algorithm of Dijkstra [9] is a special case of the GENERIC-SINGLE-SOURCE-SHORTEST-DISTANCE algorithm where the semiring $\mathbb{K}$ is taken as the tropical semiring $\mathcal{T}$ and where the queue discipline is based on the *shortest-first order*.

Similarly, the classical Bellman-Ford algorithm [4, 17] is a special case of the GENERIC-SINGLE-SOURCE-SHORTEST-DISTANCE algorithm where the semiring $\mathbb{K}$ is taken as the tropical semiring $\mathcal{T}$, and where the queue discipline is based on the *first-in first-out order.*

Bellman-Ford's algorithm can also be used with directed graphs with negative weights that have no negative cycles [8]. Indeed, the corresponding algorithm is a special case of our generic algorithm when used with the real tropical semiring $(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$. As noticed earlier (Section 2), this semiring is not bounded, but it is 0-closed (bounded) for a graph with no negative cycle.[12]

6. k-Shortest-Distance Algorithms

Given a weighted graph $G = (Q, E, w)$ over real-valued numbers and a fixed source vertex s, the *k-shortest distance problem* consists of determining the k shortest distances from s to each vertex $q \in Q$. Another slightly different problem is the *k-distinct-shortest distance problem*, which consists of determining the k *distinct* shortest distances from s to each vertex $q \in Q$.

These problems arise in a variety of domains ranging from network routing to speech recognition, where one wishes to find not just the shortest path, but the k shortest paths (see [11, 12] for an extensive bibliography on k-shortest-paths algorithms). Using the k (distinct) shortest distances from s to q, one can easily find the k (distinct) shortest paths from s to q. The k-shortest distance problem is the main problem encountered in most applications such as in speech recognition, or computational biology where one wishes to determine the k most likely alternatives, since two paths with different labels may share the same weight and thus both belong to the k shortest paths. The k-distinct-shortest distance problem is often of less interest in that context.

We will show that the k-shortest-distance problem and the k-distinct-shortest-distance problem can both be viewed as specific instances of the general shortest-distance problem where the semiring considered is a *k-tropical semiring.* We describe k-tropical semirings, and show that the generic single-source shortest-distance algorithm can be used to solve these problems. We also give the complexity of a simple implementation of the algorithm for a specific choice of the queue discipline.

6.1. k-Tropical Semirings

We define two semirings that can be used to solve k-shortest paths problems.

6.2. The k-Tropical Semiring $\mathcal{T}_k$

Let $k \geq 1$ be a positive integer. $\min_k$ is a mapping from the set of m-tuples of $\mathbb{R} \cup \{\infty\}$, $m \geq k$, to $(\mathbb{R} \cup \{\infty\})^k$ such that for any $(x_1, \ldots, x_m) \in (\mathbb{R} \cup \{\infty\})^m$: $\min_k(x_1, \ldots, x_m) = (y_1, \ldots, y_k)$, where $(y_1, \ldots, y_k)$ is the ordered list of the k shortest elements of $(x_1, \ldots, x_m)$ with repetitions using the natural order of $\mathbb{R} \cup \{\infty\}$. As an example: $\min_2(1, 1, 2, 3) = (1, 1)$.

[12] Note that by Theorem 1 any other queue discipline, in particular the shortest-first order used in Dijkstra's algorithm can also be used in that case. This in general does not result in a better time complexity however.

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

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

The following two operations can be defined for all $a, b \in \mathbb{T}_k$:

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

For example, for $k = 2$,

$$(1,2) \oplus_2 (2,3) = \min_2(1,2,2,3) = (1,2) \quad \text{and}$$
$$(1,2) \otimes_2 (2,3) = \min_2(3,4,4,5) = (3,4).$$

It is not hard to verify that $\overline{0}_k$ and $\overline{1}_k$ as defined by

$$\overline{0}_k = (\infty, \ldots, \infty) \qquad \text{and} \qquad \overline{1}_k = (0, \infty, \ldots, \infty)$$

are the identity elements for $\oplus_k$ and $\otimes_k$.

Proposition 2 *The system $\mathcal{T}_k = (\mathbb{T}_k, \oplus_k, \otimes_k, \overline{0}_k, \overline{1}_k)$ is a $(k-1)$-closed commutative semiring. For $k \geq 2$, $\mathcal{T}_k$ is not idempotent.*

Proof. Clearly, for any $a, b \in \mathbb{T}_k$, $a \oplus_k b \in \mathbb{T}_k$, and $\oplus_k$ is commutative. To show the associativity of $\oplus_k$, it suffices to note that

$$\forall a, b, c \in \mathbb{T}_k,\ (a \oplus_k b) \oplus_k c = \min_k((a_i) \cup (b_j) \cup (c_k)).$$

$\overline{0}_k$ is the identity element for $\oplus_k$: $\forall a \in \mathbb{T}_k,\ a \oplus_k \overline{0}_k = \min_k(a_1, \ldots, a_k, \infty) = a$. Thus, $(\mathbb{T}_k, \oplus_k, \overline{0}_k)$ is a commutative monoid. For $k \geq 2$, $\mathcal{T}_k$ is not idempotent. As an example, let x, y be two elements of $\mathbb{R} \cup \{\infty\}$ with $x < y$, and let $a \in \mathbb{T}_k$ be defined by $a = (x, \ldots, x, y)$. Then, if $k \geq 2$, $a \oplus a = (x, \ldots, x) \neq a$. Clearly, for any $a, b \in \mathbb{T}_k$, $a \otimes_k b \in \mathbb{T}_k$, and $\otimes_k$ is commutative. To show the associativity of $\otimes_k$, it suffices to note that

$$\forall a, b, c \in \mathbb{T}_k,\ (a \otimes_k b) \otimes_k c = \min_k((a_i + b_j + c_l)_{i,j,l}).$$

$\oplus_k$ distributes over $\otimes_k$, $\forall a, b, c \in \mathbb{T}_k$:

$$\begin{aligned}(a \otimes_k c) \oplus_k (b \otimes_k c) &= \min_k(\min_k((a_i + c_j)_{i,j}), \min_k((b_j + c_l)_{j,l})) \\ &= \min_k((a_i + c_j)_{i,j}), (b_j + c_l)_{j,l})) \\ &= (a \oplus_k b) \otimes_k c.\end{aligned}$$

$\overline{0}_k$ is an annihilator for $\otimes_k$: $\forall a \in \mathbb{T}_k,\ a \otimes_k \overline{0}_k = \min_k((a_i + \infty)_i) = \overline{0}_k$. Thus, $\mathcal{T}_k = (\mathbb{T}_k, \oplus_k, \otimes_k, \overline{0}_k, \overline{1}_k)$ is a commutative semiring. Let $a = (a_1, \ldots, a_k)$. For $i = 0, \ldots, k$, $a^i = \min_k(A_i)$ where A_i is the set of all sums of i elements chosen from $\{a_1, \ldots, a_k\}$ with repetitions, $(A_0 = \{\overline{1}_k\} = \{(0, \infty, \ldots, \infty)\})$. We have

$$\bigoplus_{i=0}^{k}{}_k\ a^i = \min_k(A_0, A_1, \ldots, A_k).$$

Let $x = a_{i_1} + \cdots + a_{i_k}$ be an element of A_k. Then, the k following sums: 0, and $a_{i_1} + \cdots + a_{i_r}$, with $r = 1, \ldots, k-1$, are elements of $A_0, \ldots, A_{k-1}$ and are all less than or equal to x. Thus, x can be omitted in the computation of $\min_k(A_0, A_1, \ldots, A_k)$, $\min_k(A_0, \ldots, A_k) = \min_k(A_0, \ldots, A_{k-1})$ and $\mathcal{T}_k$ is $(k-1)$-closed. $\square$

The proposition shows that the k-tropical semirings are covered by our general framework for single-source shortest-distance algorithms. We will show that they can be used to solve k-shortest-distance problems. For $k = 1$, $\mathcal{T}_k$ is the more familiar tropical semiring which, as already shown in a previous section, is 0-closed or bounded.

6.3. The k-Tropical-Distinct Semiring $\mathcal{T}'_k$

Let $k \geq 1$ be a positive integer. We define $<_s$ as the strict order over $\mathbb{R}_+$ completed in the following way:

$$\forall x \in \mathbb{R}_+,\ y \in \mathbb{R} \cup \{\infty\}, \quad (x <_s y) \Leftrightarrow (x < y) \quad \text{and} \quad (\infty <_s y) \Leftrightarrow (y = \infty).$$

$\min'_k$ is a mapping from the set of m-tuples of $\mathbb{R} \cup \{\infty\}$, $m \geq k$, to $(\mathbb{R} \cup \{\infty\})^k$ such that for any $(x_1, \ldots, x_m) \in (\mathbb{R} \cup \{\infty\})^m$: $\min'_k(x_1, \ldots, x_m) = (y_1, \ldots, y_k)$, where $(y_1, \ldots, y_k)$ is the ordered list of the k shortest distinct elements of $(x_1, \ldots, x_m)$ without repetitions using the order $<_s$, completed with ∞ when the number of distinct elements is less than k. As an example: $\min_2(1, 1, 2, 3) = (1, 2)$, and $\min_3(1, 1, 1, 2) = (1, 2, \infty)$. We define $\mathbb{T}'_k$ as the set of nonnegative k-tuples in $(\mathbb{R}_+ \cup \{\infty\})^k$ ordered according to $<_s$: $\mathbb{T}'_k = \{(a_1, \ldots, a_k) \in (\mathbb{R}_+ \cup \{\infty\})^k : 0 \leq a_1 <_s \cdots <_s a_k\}$. The following two operations $\oplus'_k$ and $\otimes'_k$ can be defined for all $a, b \in \mathbb{T}'_k$:

$$a \oplus'_k b = \min'_k((a_i)_i \cup (b_j)_j) \qquad \text{and} \qquad a \otimes'_k b = \min'_k((a_i + b_j)_{i,j}).$$

As an example, for $k = 3$,

$$(1, 2, 3) \oplus'_3 (0, 1, 2) = \min'_3(1, 2, 3, 0, 1, 2) = (0, 1, 2),$$
$$(1, 2, 3) \otimes'_3 (0, 1, 2) = \min'_3(1, 2, 3, 2, 3, 4, 3, 4, 5) = (1, 2, 3).$$

$\overline{0}_k$ and $\overline{1}_k$ are the identity elements for $\oplus'_k$ and $\otimes'_k$.

Proposition 3 *The system $\mathcal{T}'_k = (\mathbb{T}'_k, \oplus'_k, \otimes'_k, \overline{0}_k, \overline{1}_k)$ is a $(k-1)$-closed commutative semiring. $\mathcal{T}'_k$ is idempotent. $\mathcal{T}'_k$ provided with its natural order $\leq_{\mathcal{T}'_k}$ is negative and monotonic.*

Proof. Clearly, for any $a, b \in \mathbb{T}'_k$, $a \oplus'_k b \in \mathbb{T}'_k$, and $\oplus'_k$ is commutative. As with $\oplus_k$, we note that

$$\forall a, b, c \in \mathbb{T}'_k,\ (a \oplus'_k b) \oplus'_k c = \min'_k((a_i) \cup (b_j) \cup (c_k))$$

which shows the associativity of $\oplus'_k$. $\overline{0}_k$ is the identity element for $\oplus'_k$:

$$\forall a \in \mathbb{T}'_k,\ a \oplus'_k \overline{0}_k = \min'_k(a_1, \ldots, a_k, \infty) = a.$$

Thus, $(\mathbb{T}'_k, \oplus'_k, \overline{0}_k)$ is a commutative monoid. Furthermore, $(\mathbb{T}'_k, \oplus'_k, \overline{0}_k)$ is idempotent:

$$\forall a \in \mathbb{T}'_k,\ a \oplus'_k a = \min'_k(a_1, a_1, \ldots, a_k, a_k) = a.$$

Clearly, for any $a, b \in \mathbb{T}'_k$, $a \otimes'_k b \in \mathbb{T}'_k$, and $\otimes'_k$ is commutative. To show the associativity of $\oplus'_k$, it suffices to note that

$$\forall a, b, c \in \mathbb{T}'_k,\ (a \oplus'_k b) \oplus'_k c = \min'_k((a_i + b_j + c_l)_{i,j,l}).$$

$\oplus'_k$ distributes over $\otimes'_k$, $\forall a, b, c \in \mathbb{T}'_k$:

$$\begin{aligned}(a \otimes'_k b) \oplus'_k (b \otimes'_k c) &= \min'_k(\min'_k((a_i + c_j)_{i,j}), \min'_k((b_j + c_l)_{j,l})) \\ &= \min'_k((a_i + c_j)_{i,j}), (b_j + c_l)_{j,l})) \\ &= (a \oplus'_k b) \otimes'_k c.\end{aligned}$$

$\overline{0}_k$ is an annihilator for $\otimes'_k$:

$$\forall a \in \mathbb{T}'_k,\ a \otimes'_k \overline{0}_k = \min'_k((a_i + \infty)_i) = \overline{0}_k.$$

Thus, $\mathcal{T}'_k = (\mathbb{T}'_k, \oplus'_k, \otimes'_k, \overline{0}_k, \overline{1}_k)$ is a commutative semiring. Since $\mathcal{T}'_k$ is idempotent, by Lemma 1 it can be provided with a natural order $\leq_{\mathcal{T}'_k}$. $\mathcal{T}'_k$ provided with that order is negative and monotonic (Lemma 2). Furthermore, the order $\leq_{\mathcal{T}'_k}$ is the unique order making $\mathcal{T}'_k$ monotonic and negative (Proposition 1). The proof of the $(k-1)$-closedness of $\mathcal{T}'_k$ is similar to that of $\mathcal{T}_k$ (Proposition 2). □

The proposition shows that the k-tropical distinct semirings also fall within the general framework for single-source shortest-distance algorithms. We will show that they can be used to solve k-distinct-shortest-distance problems. For $k = 1$, $\mathcal{T}'_k$ is the tropical semiring which is bounded, as shown in the previous sections. The semirings $\mathcal{T}'_k$ were first introduced by Schier [37] for solving k-distinct-shortest-distance problems but with an algorithm different from the general algorithm we presented in previous sections. These semirings are also discussed by [34] for solving the algebraic path problem.

6.4. k-Shortest-Distance Algorithm

Let $G = (Q, E, w)$ be a weighted graph over non-negative real-valued numbers, and let s be a fixed source vertex of G. For $q \in Q$, we denote by $\delta_k(s, q) \in \mathbb{T}_k$ the ordered set of the k shortest distances from s to q with repetitions. $\delta_k(s, q)$ is defined by

$$\forall q \in Q,\ \delta_k(s, q) = \min_k(w[\pi] : \pi \in P(q)).$$

The problem of determining $\delta_k(s, q)$, $q \in Q$, can be viewed as a single-source shortest-distance problem with the graph $G_k = (Q, E, w_k)$ weighted over $\mathbb{T}_k$, if we define w_k by: $\forall e \in E, w_k(e) = (w[e], \infty, \ldots, \infty)$. Indeed, we have

$$\forall q \in Q,\ \delta_k(s, q) = \bigoplus_{\pi \in P(q)}{}_k\ w_k[\pi].$$

Thus, the generic single-source shortest-distance algorithm can be used to solve k-shortest-distance problems.

Theorem 3 *Let $G = (Q, E, w)$ be a weighted directed graph over non-negative real-valued numbers, and let s be a fixed source vertex. Then, if we run the* Generic-Single-Source-Shortest-Distance *algorithm on the graph $G_k = (Q, E, w_k)$ weighted over $\mathcal{T}_k$, the algorithm terminates, and at termination, for any vertex $q \in Q$, $d[q] = \delta_k(s, q)$, the ordered list of the k shortest distances from s to q with repetitions.*

Proof. By Proposition 2, $\mathcal{T}_k$ is a k-closed semiring. Thus, the semiring is covered by our general framework and by Theorem 1, the GENERIC-SINGLE-SOURCE-SHORTEST-DISTANCE algorithm computes the k shortest distances from s to each vertex $q \in Q$. $\square$

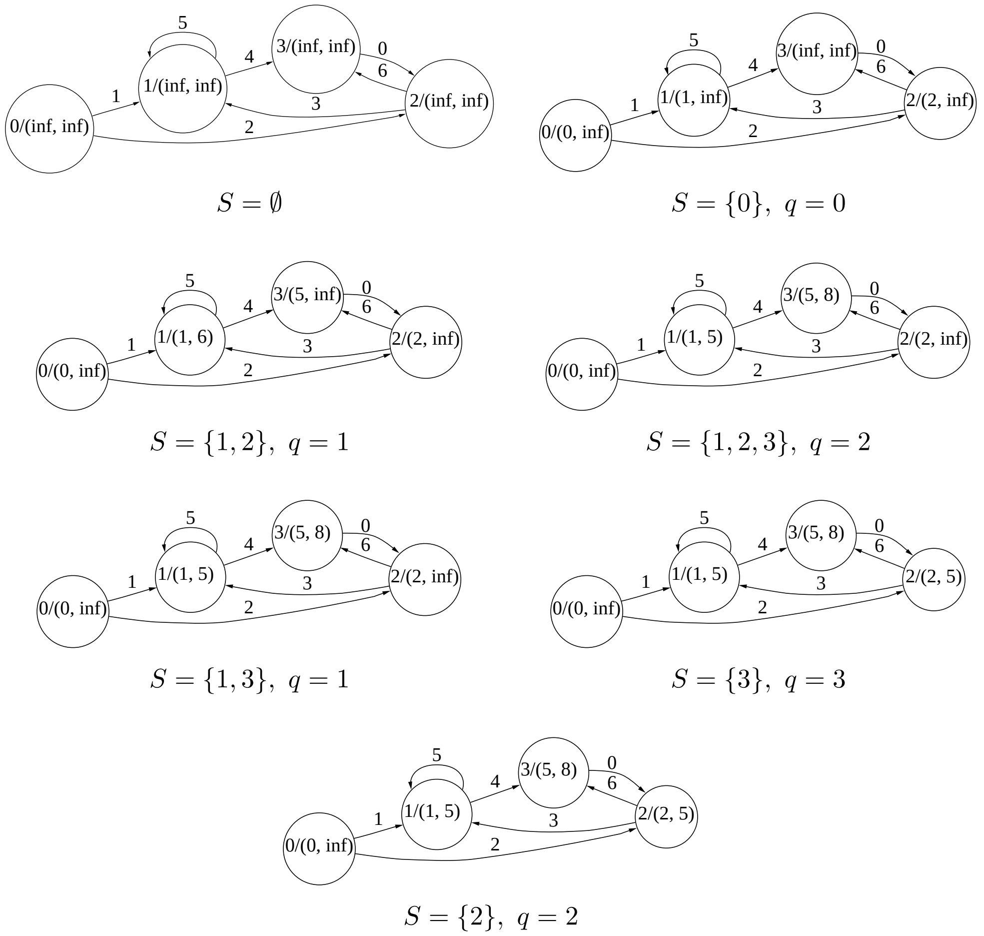

Figure 5: Consecutive steps of the execution of the GENERIC-SINGLE-SOURCE-SHORTEST-DISTANCE algorithm used to compute 2-shortest distances

Complexity

We gave the general expression of the complexity of the GENERIC-SINGLE-SOURCE-SHORTEST-DISTANCE algorithm in a previous section. The $\oplus_k$ operation can be performed in $O(k)$ using a 2-merge sort. The $\otimes_k$ can also be performed efficiently because it is only used in a relaxation step (computation of $r' \otimes_k w_k[e]$), and because $w_k[e]$

has a specific form. Indeed, since $w_k[e] = (w[e], \infty, \ldots, \infty)$, with $r' = (r_1, \ldots, r_k)$,

$$r' \otimes_k w_k[e] = \min_k((r_i + w[e])_i).$$

Thus, k additions are sufficient to compute the result, and $T_{\otimes_k} = O(k)$. The number of times a vertex q is inserted in the queue S depends on the queue discipline chosen.

We define a new queue discipline that coincides with the shortest-first order used in Dijkstra's algorithm when $k = 1$. For each vertex q, we maintain an attribute $X[q]$ which gives the number of times q has been extracted from S, and define $\mu(a, l)$ for $a = (a_1, \ldots, a_k) \in \mathbb{T}_k$ and any non-negative integer l by

$$\mu(a, l) = a_{l+1} \text{ if } (l + 1 \leq k),\ a_k \text{ otherwise.}$$

Our queue discipline is based on the following pre-order over the vertices Q:

$$q \leq_X q' \Longleftrightarrow \mu(d[q], X[q]) \leq \mu(d[q'], X[q']).$$

In many applications such as routing problems, one is only interested in determining the k-shortest distances or paths from s to a fixed destination or final vertex t. The efficiency of the algorithm can then be improved by defining the queue discipline as:

$$q \leq_X q' \Longleftrightarrow \mu(d[q], X[q]) + f[q] \leq \mu(d[q'], X[q']) + f[q']$$

where $f[q]$ and $f[q']$ denote the shortest distance from q and q' to t. f can be precomputed using a single-source shortest-distance algorithm with source t.

Using the array X, the comparison of two vertices q and q' for this pre-order can be done in constant time. Let N_k be the maximum number of insertions of a vertex in the queue S using the order $\leq_X$.[13] In view of the general formula of the complexity of the Generic-Single-Source-Shortest-Distance algorithm, if we use classical binary heaps to implement the queue discipline $\leq_X$ we defined, the complexity of the algorithm for finding the k-shortest distances from s to each vertex q is $O(|Q|+(k+k+\log(|Q|))|E|N_k + (\log|Q| + \log|Q|)N_k|Q|)$. Hence $O(N_k(|Q| + |E|)\log|Q| + kN_k|E|)$. If we use Fibonacci heaps, the complexity of the algorithm is

$$O(N_k|Q|\log|Q| + kN_k|E|).$$

When the graph is acyclic, the Generic-Topological-Single-Source-Shortest-Distance algorithm can be used. The complexity of the algorithm is then

$$O(|Q| + k|E|).$$

The Generic-Single-Source-Shortest-Distance algorithm can also be used to determine the k shortest paths from s to each vertex $q \in Q$. Since in the worst case the length of the ith shortest path can be $i \times |Q|$, the total length of the k shortest paths is in $O(k^2|Q|)$. Thus, using Fibonacci heaps, the complexity of the computation of the k shortest paths is $O(N_k|Q|\log|Q| + k(k|Q| + N_k|E|))$.

Figure 5 illustrates the execution of the Generic-Single-Source-Shortest-Distance algorithm based on the queue discipline $\leq_X$ for $k = 2$. Each step corresponds to the extraction from the queue S of a vertex q. The tentative shortest

[13] We will leave the proof of the fact that $N_k = k$ to a separate study.

distance pairs are indicated for each vertex at each step of the execution of the algorithm.

In some applications such as speech recognition where weighted automata are used, one wishes to determine the k shortest paths to each state labelled with distinct strings. To determine these paths, one can use on-the-fly weighted determinization of automata followed by a k-shortest distance algorithm [27, 31].

The complexity of the k-shortest-distance algorithm we presented can be substantially improved with a more careful choice of the data structures specific to this problem and with a finer analysis of the algorithm. Our main objective here was to illustrate the application of the Generic-Single-Source-Shortest-Distance algorithm to the k-shortest-distance problem.

6.5. k-Distinct-Shortest-Distance Algorithm

Let $G = (Q, E, w)$ be a weighted graph over non-negative real-valued numbers, and let s be a fixed source vertex of G. For $q \in Q$, we denote by $\delta'_k(s, q) \in \mathbb{T}'_k$ the ordered set of the k distinct shortest distances from s to q. $\delta'_k(s, q)$ is defined by

$$\forall q \in Q,\ \delta'_k(s, q) = \min'_k(w[\pi] : \pi \in P(q)).$$

The problem of determining $\delta'_k(s, q)$, $q \in Q$, can be viewed as a single-source shortest-distance problem with the graph $G_k = (Q, E, w_k)$ weighted over $\mathbb{T}'_k$. Indeed, we have

$$\forall q \in Q,\ \delta'_k(s, q) = \bigoplus_{\pi \in P(q)}{}'_k\ w_k[\pi].$$

Thus, the generic single-source shortest-distance algorithm can be used to solve k-distinct-shortest-distance problems.

Theorem 4 *Let $G = (Q, E, w)$ be a weighted directed graph over non-negative real-valued numbers, and let s be a fixed source vertex. Then if we run the* Generic-Single-Source-Shortest-Distance *algorithm on the graph $G_k = (Q, E, w_k)$ weighted over $\mathcal{T}'_k$, the algorithm terminates, and at termination, for any vertex $q \in Q$, $d[q] = \delta'_k(s, q)$, the ordered list of the k distinct shortest distances from s to q.*

Proof. By Proposition 3, $\mathcal{T}'_k$ is a k-closed semiring. The semiring is covered by our general framework, thus, by Theorem 1, the Generic-Single-Source-Shortest-Distance algorithm computes the k distinct shortest distances from s to each vertex $q \in Q$. □

Complexity

The complexity of the Generic-Single-Source-Shortest-Distance algorithm for computing the k-distinct-shortest distances can be determined in a way similar to what was presented for the case of k-shortest distances. Using the queue discipline $\leq_X$ defined in the previous section and classical binary heaps the complexity is

$$O(N_k(|Q| + |E|) \log |Q| + kN_k|E|).$$

With Fibonacci heaps the complexity is

$$O(N_k|Q|\log|Q| + kN_k|E|).$$

When the graph is acyclic, the GENERIC-TOPOLOGICAL-SINGLE-SOURCE-SHORTEST-DISTANCE algorithm can be used and the complexity of the algorithm is

$$O(|Q| + k|E|).$$

7. Conclusion

We presented new generic algorithms for single-source shortest-distance problems based on the structure of semirings, gave the general expression of their complexity in terms of the costs of elementary operations depending on the choice of the queue discipline and the cost of the semiring operations, and illustrated their use in various problems such as the k-shortest-distance problems. Single-source shortest distance algorithms can be used in a variety of other applications with various other semirings.

Our approach consists of defining general algebraic frameworks for shortest-distance problems and of devising *generic* algorithms, algorithms that work with any algebra falling within our general frameworks, for solving these problems. It follows a general principle of *separation of algorithms and algebras* that seems to us as important as the classical software engineering principle of separation of programs and data.

A general algebraic framework helps to bridge the gap between theoretical computer science and software design. It increases considerably the reusability of code avoiding one to reinvent the wheel: a single generic algorithm can be used to solve a variety of different problems. Furthermore, an efficient implementation of the algebraic operations can make the algorithm practical for each problem. The separation of algorithms and algebra also often helps in anticipating on the computational and algorithmic needs, and leads to a better understanding of the deep mechanisms of some algorithms.

Acknowledgements

I thank Corinna Cortes, Michael Riley, and the reviewers, for their comments which helped improve an earlier draft of this paper.

References

[1] L. ACETO, Z. ÉSIK, A. INGÓLFSDÓTTIR, Equational Theories of Tropical Semirings. Technical Report RS-01-21, BRICS, IESD, June 2001. 52 pages.

[2] A. V. AHO, J. E. HOPCROFT, J. D. ULLMAN, *The Design and Analysis of Computer Algorithms.* Addison Wesley, Reading, MA, 1974.

[3] R. C. BACKHOUSE, B. CARRÉ, Regular Algebra Applied to Path-Finding Problems. *Journal of the Institute of Mathematics and Its Applications* **15** (1975), 161–186.

[4] R. Bellman, On a Routing Problem. *Quarterly of Applied Mathematics* **16** (1958).

[5] J. Berstel, *Transductions and Context-Free Languages.* Teubner Studienbücher, Stuttgart, 1979.

[6] J. Berstel, C. Reutenauer, *Rational Series and Their Languages.* Springer-Verlag, Berlin-New York, 1988.

[7] B. Carré, An Algebra for Network Routing Problems. *Journal of the Institute of Mathematics and Its Applications* **7** (1971), 273–294.

[8] T. H. Cormen, C. E. Leiserson, R. E. Rivest, *Introduction to Algorithms.* The MIT Press, Cambridge, MA, 1992.

[9] E. W. Dijkstra, A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik* **1** (1959).

[10] S. Eilenberg, *Automata, Languages and Machines*, vol. A. Academic Press, 1974.

[11] D. Eppstein, Finding the k Shortest Paths. In: *Proc. 35th Symp. Foundations of Computer Science.* Inst. of Electrical & Electronics Engineers, November 1994, 154–165.

[12] D. Eppstein, K Shortest Paths and Other “K Best” Problems. `http://www1.ics.uci.edu/~eppstein/bibs/kpath.bib`, 2001.

[13] Z. Ésik, W. Kuich, Locally Closed Semirings. *Monatshefte für Mathematik*, to appear (2002).

[14] Z. Ésik, W. Kuich, Rationally Additive Semirings. *Journal of Universal Computer Science* **8** (2002), 173–183.

[15] E. Fink, A survey of sequential and systolic algorithms for the algebraic path problem. Technical Report, Department of Computer Science, University of Waterloo, 1992.

[16] R. W. Floyd, Algorithm 97 (SHORTEST PATH). *Communications of the ACM* **18** (1968).

[17] L. R. Ford, D. R. Fulkerson, Maximal Flow through a Network. *Canadian Journal of Mathematics* **8** (1956), 399–404.

[18] L. R. Ford, D. R. Fulkerson, Constructing Maximal Dynamic Flow from Static Flows. *The Journal of the Operations Research Society of America* **6** (1958), 419–433.

[19] L. R. Ford, D. R. Fulkerson, Flows in Network. Technical Report, Princeton University Press, 1962.

[20] M. Gondran, M. Minoux, *Graphs and Algorithms.* John Wiley and Sons, New York, 1984.

[21] S. C. Kleene, Representation of events in nerve nets and finite automata. *Automata Studies, Annals of Mathematics Studies*, vol. 34. Princeton University Press, 1956, 3–42.

[22] D. E. KNUTH, *Fundamental Algorithms, The Art of Computer Programming*, vol. 1. Addison-Wesley, Reading, MA, 1968.

[23] W. KUICH, A. SALOMAA, *Semirings, Automata, Languages.* No. 5 in EATCS Monographs on Theoretical Computer Science, Springer-Verlag, Berlin-New York, 1986.

[24] E. L. LAWLER, *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart, and Winston, 1976.

[25] D. J. LEHMANN, Algebraic Structures for Transitive Closures. *Theoretical Computer Science* **4** (1977), 59–76.

[26] H. LEUNG, On the Topological Structure of a Finitely Generated Semigroup of Matrices. *Semigroup Forum* **37** (1988), 273–287.

[27] M. MOHRI, Finite-State Transducers in Language and Speech Processing. *Computational Linguistics* **23** (1997) 2.

[28] M. MOHRI, General Algebraic Frameworks and Algorithms for Shortest-Distance Problems. Technical Memorandum 981210-10TM, AT&T Labs – Research, 62 pages, 1998.

[29] M. MOHRI, Minimization Algorithms for Sequential Transducers. *Theoretical Computer Science* **234** (2000), 177–201.

[30] M. MOHRI, F. C. N. PEREIRA, M. RILEY, The Design Principles of a Weighted Finite-State Transducer Library. *Theoretical Computer Science* **231** (2000), 17–32.

[31] M. MOHRI, M. RILEY, An Efficient Algorithm for the N-Best-Strings Problem. In: *Proceedings of the International Conference on Spoken Language Processing* (ICSLP 2002), Denver, Colorado, September 2002.

[32] J.-E. PIN, Tropical Semirings. Technical Report LITP 95/40, Université Paris 7, 1995.

[33] G. ROTE, A Systolic Array Algorithm for the Algebraic Path Problem (Shortest Paths; Matrix Inversion). *Computing* **34** (1985), 191–219.

[34] G. ROTE, Path Problems in Graphs. *Computing Supplementum* **7** (1990), 155–189.

[35] B. ROY, Transitivité et Connexité. *C. R. Académie des Sciences, Paris*, **249** (1959), 216–218.

[36] A. SALOMAA, M. SOITTOLA, *Automata-Theoretic Aspects of Formal Power Series.* Springer-Verlag, New York, 1978.

[37] D. R. SCHIER, Iterative Methods for Determining the k Shortest Paths in a Network. *Networks* **6** (1976), 205–229.

[38] I. SIMON, The Nondeterministic Complexity of Finite Automata. Technical Report RT-MAP-8073, Instituto de Matemática e Estatística da Universidade de São Paulo, 1987.

[39] R. E. Tarjan, Depth First Search and Linear Graph Algorithms. *SIAM Journal of Computing* **1** (1972) 2, 146–160.

[40] R. E. Tarjan, A Unified Approach to Paths Problems. *Journal of the Association of Computing Machinery* **28** (1981).

[41] S. Warshall, A Theorem on Boolean Matrices. *Journal of the ACM* **9** (1962) 1, 11–12.

[42] M. Yoeli, A Note on a Generalization of Boolean Matrix Theory. *American Mathematical Monthly* **68** (1961), 552–557.

[43] U. T. Zimmermann, *Linear and Combinatorial Optimization in Ordered Algebraic Structures*. Annals of Discrete Mathematics **10**, North Holland, Amsterdam, 1981.

(Received: October 15, 2001; revised: August 8, 2002)