Weighted Finite-State Transducer Algorithms An Overview

Mehryar Mohri
AT&T Labs – Research
Shannon Laboratory
180 Park Avenue, Florham Park, NJ 07932, USA
mohri@research.att.com

May 14, 2004

Abstract

Weighted finite-state transducers are used in many applications such as text, speech and image processing. This chapter gives an overview of several recent weighted transducer algorithms, including composition of weighted transducers, determinization of weighted automata, a weight pushing algorithm, and minimization of weighted automata. It briefly describes these algorithms, discusses their running time complexity and conditions of application, and shows examples illustrating their application.

1 Introduction

Weighted transducers are used in many applications such as text, speech and image processing [9, 12, 5]. They are automata in which each transition in addition to its usual input label is augmented with an *output label* from a possibly new alphabet, and carries some *weight* element of a semiring. Transducers can be used to define a mapping between two different types of information sources, e.g., word and phoneme sequences. The weights are critically needed to model the uncertainty or the variability of such information sources. Weighted transducers can be used for example to assign different pronunciations to the same word but with different ranks or probabilities. Their weights are typically derived from large data sets using various sophisticated statistical learning techniques.

Much of the theory of weighted transducers and rational power series was developed more than two decades ago [15, 7, 4]. However, many essential weighted transducer algorithms such as determinization and minimization of weighted transducers [9] are recent and arise new questions, both theoretical and algorithmic. This chapter overviews several recent weighted transducer algorithms, including composition of weighted transducers, determinization of weighted automata, a weight pushing algorithm, and minimization of weighted automata.

SEMIRING	SET	$\oplus$	$\otimes$	$\overline{0}$	$\overline{1}$
Boolean	$\{0,1\}$	$\vee$	$\wedge$	0	1
Probability	$\mathbb{R}_+$	$+$	$\times$	0	1
Log	$\mathbb{R} \cup \{-\infty, +\infty\}$	$\oplus_{\log}$	$+$	$+\infty$	0
Tropical	$\mathbb{R} \cup \{-\infty, +\infty\}$	min	$+$	$+\infty$	0

Table 1: *Semiring examples.* $\oplus_{\log}$ *is defined by:* $x \oplus_{\log} y = -\log(e^{-x} + e^{-y})$.

2 Preliminaries

This section introduces the definitions and notation used in the following.

Definition 1 ([7]) *A system* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *is a* semiring *if:* $(\mathbb{K}, \oplus, \overline{0})$ *is a commutative monoid with identity element* $\overline{0}$*;* $(\mathbb{K}, \otimes, \overline{1})$ *is a monoid with identity element* $\overline{1}$*;* $\otimes$ *distributes over* $\oplus$*; and* $\overline{0}$ *is an annihilator for* $\otimes$*: for all* $a \in \mathbb{K}, a \otimes \overline{0} = \overline{0} \otimes a = \overline{0}$.

Thus, a semiring is a ring that may lack negation. Table 1 lists some familiar semirings. A semiring is said to be *commutative* when the multiplicative operation $\otimes$ is commutative. It is said to be *left divisible* if for any $x \neq \overline{0}$, there exists $y \in \mathbb{K}$ such that $y \otimes x = \overline{1}$, that is if all elements of $\mathbb{K}$ admit a left inverse. $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is said to be *weakly left divisible* if for any x and y in $\mathbb{K}$ such that $x \oplus y \neq \overline{0}$, there exists at least one z such that $x = (x \oplus y) \otimes z$. When the $\otimes$ operation is cancellative, z is unique and we can then write: $z = (x \oplus y)^{-1}x$. When z is not unique, we can still assume that we have an algorithm to find one of the possible z and call it $(x \oplus y)^{-1}x$. Furthermore, we will assume that z can be found in a consistent way, that is: $((u \otimes x) \oplus (u \otimes y))^{-1}(u \otimes x) = (x \oplus y)^{-1}x$ for any $x, y, u \in \mathbb{K}$ such that $u \neq \overline{0}$. A semiring is *zero-sum-free* if for any x and y in $\mathbb{K}$, $x \oplus y = \overline{0}$ implies $x = y = \overline{0}$. In the following definitions, $\mathbb{K}$ is assumed to be a left semiring or a semiring.

Definition 2 *A* weighted finite-state transducer T *over a semiring* $\mathbb{K}$ *is an 8-tuple* $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ *where:* Σ *is the finite input alphabet of the transducer;* Δ *is the finite output alphabet;* Q *is a finite set of states;* $I \subseteq Q$ *the set of initial states;* $F \subseteq Q$ *the set of final states;* $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{K} \times Q$ *a finite set of transitions;* $\lambda : I \to \mathbb{K}$ *the initial weight function; and* $\rho : F \to \mathbb{K}$ *the final weight function mapping* F *to* $\mathbb{K}$.

We denote by $|T|$ the sum of the number of states and transitions of T. *Weighted automata* are defined in a similar way by simply omitting the input or output labels.

Given a transition $e \in E$, we denote by $p[e]$ its origin or previous state and $n[e]$ its destination state or next state, and $w[e]$ its weight. A *path* $\pi = e_1 \cdots e_k$ is an element of E^* with consecutive transitions: $n[e_{i-1}] = p[e_i]$, $i = 2, \ldots, k$. We extend n and p to paths by setting: $n[\pi] = n[e_k]$ and $p[\pi] = p[e_1]$. The weight function w can also be extended to paths by defining the weight of a path as the $\otimes$-product of the weights of its constituent transitions:

$w[\pi] = w[e_1] \otimes \cdots \otimes w[e_k]$. We denote by $P(q, q')$ the set of paths from q to q' and by $P(q, x, y, q')$ the set of paths from q to q' with input label $x \in \Sigma^*$ and output label y. These definitions can be extended to subsets $R, R' \subseteq Q$, by: $P(R, x, y, R') = \cup_{q \in R,\, q' \in R'} P(q, x, y, q')$. A transducer T is *regulated* if the output weight associated by T to any pair of input-output string (x, y) by:

$$[\![T]\!](x, y) = \bigoplus_{\pi \in P(I,x,y,F)} \lambda[p[\pi]] \otimes w[\pi] \otimes \rho[n[\pi]] \tag{1}$$

is well-defined and in $\mathbb{K}$. $[\![T]\!](x, y) = \overline{0}$ when $P(I, x, y, F) = \emptyset$. If for all $q \in Q$ $\bigoplus_{\pi \in P(q,\epsilon,\epsilon,q)} w[\pi] \in \mathbb{K}$, then T is regulated. In particular, when T does not have any ϵ-cycle, it is regulated. We define the *domain of* T, $Dom(T)$, as: $Dom(T) = \left\{(x, y) : [\![T]\!](x, y) \neq \overline{0}\right\}$.

3 Composition of Weighted Transducers

Composition is a fundamental algorithm used to create complex weighted transducers from simpler ones. Let $\mathbb{K}$ be a *commutative* semiring and let T_1 and T_2 be two weighted transducers defined over $\mathbb{K}$ such that the input alphabet of T_2 coincides with the output alphabet of T_1. Assume that the infinite sum $\bigoplus_z T_1(x, z) \otimes T_2(z, y)$ is well-defined and in $\mathbb{K}$ for all $(x, y) \in \Sigma^* \times \Omega^*$. This condition holds for all transducers defined over a *closed semiring* [8, 11] such as the Boolean semiring and the tropical semiring and for all acyclic transducers defined over an arbitrary semiring. Then, the result of the composition of T_1 and T_2 is a weighted transducer denoted by $T_1 \circ T_2$ and defined for all x, y by [3, 6, 15, 7]:[1]

$$[\![T_1 \circ T_2]\!](x, y) = \bigoplus_z T_1(x, z) \otimes T_2(z, y) \tag{2}$$

There exists a general and efficient composition algorithm for weighted transducers [13, 12]. States in the composition $T_1 \circ T_2$ of two weighted transducers T_1 and T_2 are identified with pairs of a state of T_1 and a state of T_2. Leaving aside transitions with ϵ inputs or outputs, the following rule specifies how to compute a transition of $T_1 \circ T_2$ from appropriate transitions of T_1 and T_2:

$$(q_1, a, b, w_1, q_2) \text{ and } (q_1', b, c, w_2, q_2') \Longrightarrow ((q_1, q_1'), a, c, w_1 \otimes w_2, (q_2, q_2')) \tag{3}$$

See [13, 12] for a detailed presentation of the algorithm including the use of a transducer filter for dealing with ϵ-multiplicity in the case of non-idempotent semirings. In the worst case, all transitions of T_1 leaving a state q_1 match all those of T_2 leaving state q_1', thus the space and time complexity of composition is quadratic: $O(|T_1||T_2|)$. However, an on-the-fly implementation of composition can be used to construct just the part of the composed transducer that is needed. Figures 1(a)-(c) illustrate the algorithm when applied to the transducers of Figures 1(a)-(b) defined over the tropical semiring.

[1] Note that we use a *matrix notation* for the definition of composition as opposed to a *functional notation*. This is a deliberate choice motivated in many cases by improved readability.

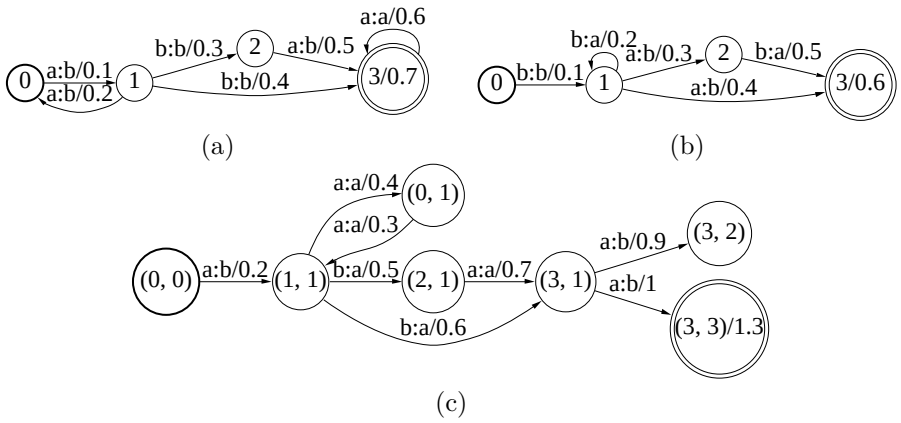


Figure 1: (a) Weighted transducer T_1 over the tropical semiring. (b) Weighted transducer T_2 over the tropical semiring. (c) Composition of T_1 and T_2.

Intersection (or Hadamard product) of weighted automata and composition of finite-state transducers are both special cases of composition of weighted transducers. Intersection corresponds to the case where input and output labels of transitions are identical and composition of unweighted transducers is obtained simply by omitting the weights.

In general, the definition of composition cannot be extended to the case of non-commutative semirings because the composite transduction cannot always be represented by a weighted finite-state transducer. Consider for example, the case of two transducers T_1 and T_2, with $Dom(T_1) = Dom(T_2) = (a, a)^*$, with $[\![T_1]\!](a, a) = x \in \mathbb{K}$ and $[\![T_2]\!](a, a) = y \in \mathbb{K}$ and let τ be the composite of the transductions corresponding to T_1 and T_2. Then, for any non-negative integer n, $\tau(a^n, a^n) = x^n \otimes y^n$ which in general is different from $(x \otimes y)^n$ if x and y do not commute. An argument similar to the classical Pumping lemma can then be used to show that τ cannot be represented by a weighted finite-state transducer.

When T_1 and T_2 are acyclic, composition can be extended to the case of non-commutative semirings. The algorithm would then consist of matching paths of T_1 and T_2 directly rather than matching their constituent transitions. The termination of the algorithm is guaranteed by the fact that the number of paths of T_1 and T_2 is finite. However, the time and space complexity of the algorithm is then exponential.

The weights of matching transitions and paths are $\otimes$-multiplied in composition. One might wonder if another operation, $\times$, can be used instead of $\otimes$, in particular when $\mathbb{K}$ is not commutative. The following proposition proves that that cannot be.

Proposition 1 *Let $(\mathbb{K}, \times, e)$ be a monoid. Assume that $\times$ is used instead of $\otimes$ in composition. Then, $\times$ coincides with $\otimes$ and $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is a commutative semiring.*

Proof. Consider two sets of consecutive transitions of two paths: $\pi_1 = (p_1, a, a, x, q_1)(q_1, b, b, y, r_1)$ and $\pi_2 = (p_2, a, a, u, q_2)(q_2, b, b, v, r_2)$. Matching these transitions using $\times$ result in the following:

$$((p_1, p_2), a, a, x \times u, (q_1, q_2)) \quad \text{and} \quad ((q_1, q_2), b, b, y \times v, (r_1, r_2)) \tag{4}$$

Since the weight of the path obtained by matching π_1 and π_2 must also correspond to the $\times$-multiplication of the weight of π_1, $x \otimes y$, and the weight of π_2, $u \otimes v$, we have:

$$(x \times u) \otimes (y \times v) = (x \otimes y) \times (u \otimes v) \tag{5}$$

This identity must hold for all $x, y, u, v \in \mathbb{K}$. Setting $u = y = e$ and $v = \overline{1}$ leads to $x = x \otimes e$ and similarly $x = e \otimes x$ for all x. Since the identity element of $\otimes$ is unique, this proves that $e = \overline{1}$.

With $u = y = \overline{1}$, identity 5 can be rewritten as: $x \otimes v = x \times v$ for all x and v, which shows that $\times$ coincides with $\otimes$. Finally, setting $x = v = \overline{1}$ gives $u \otimes y = y \times u$ for all y and u which shows that $\otimes$ is commutative. □

4 Determinization

A weighted automaton is said to be *deterministic* or *subsequential* [16] if it has a unique initial state and if no two transitions leaving any state share the same input label.

There exists a natural extension of the classical subset construction to the case of weighted automata over a weakly left divisible semiring called *determinization* [9].[2] The algorithm is generic: it works with any weakly left divisible semiring. Figures 2(a)-(b) illustrate the determinization of a weighted automaton over the tropical semiring. A state r of the output automaton that can be reached from the start state by a path π corresponds to the set of pairs $(q, x) \in Q \times \mathbb{K}$ such that q can be reached from an initial state of the original machine by a path σ with $l[\sigma] = l[\pi]$ and $\lambda[p[\sigma]] \otimes w[\sigma] = \lambda[p[\pi]] \otimes w[\pi] \otimes x$. Thus, x is the *remaining* weight at state q. The worst case complexity of determinization is exponential even in the unweighted case. However, in many practical cases such as for weighted automata used in large-vocabulary speech recognition, this blow-up does not occur. It is also important to notice that just like composition, determinization admits a natural on-the-fly implementation [9] which can be useful for saving space.

Unlike the unweighted case, determinization does not halt for some input weighted automata. In fact, some weighted automata, non *subsequentiable* automata, do not even admit equivalent subsequential machines. We say that a weighted automaton A is *determinizable* if the determinization algorithm halts for the input A. With a determinizable input, the algorithm outputs an equivalent subsequential weighted automaton [9].

[2] We assume that the weighted automata considered are all such that for any string $x \in \Sigma^*$, $w[P(I, x, Q)] \neq \overline{0}$. This condition is always satisfied with trim machines over the tropical semiring or any zero-sum-free semiring.

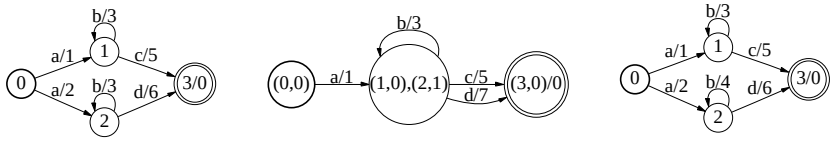


Figure 2: Determinization of weighted automata. (a) Weighted automaton over the tropical semiring A. (b) Equivalent weighted automaton B obtained by determinization of A. (c) Non-determinizable weighted automaton over the tropical semiring, states 1 and 2 are non-twin siblings.

There exists a general property, the *twins property*, first formulated for finite-state transducers by C. Choffrut [3], later generalized to weighted automata over the tropical semiring by [9], that provides a characterization of determinizable weighted automata under some general conditions.

Let A be a weighted automaton over a weakly left divisible left semiring $\mathbb{K}$. Two states q and q' of A are said to be *siblings* if there exist two strings x and y in Σ^* such that both q and q' can be reached from I by paths labeled with x and there is a cycle at q and a cycle at q' both labeled with y. When $\mathbb{K}$ is a commutative and cancellative semiring, then two sibling states are said to be *twins* iff for any string y:

$$w[P(q, y, q)] = w[P(q', y, q')] \tag{6}$$

A has *the twins property* if any two sibling states of A are twins. Figure 2(c) shows an unambiguous weighted automaton over the tropical semiring that does not have the twins property: states 1 and 2 can be reached by paths labeled with a from the initial state and admit cycles with the same label b, but the weights of these cycles (3 and 4) are different.

Theorem 1 ([9]) *Let A be a weighted automaton over the tropical semiring. If A has the twins property, then A is determinizable.*

With trim unambiguous weighted automata, the condition is also necessary.

Theorem 2 ([9]) *Let A be a trim unambiguous weighted automaton over the tropical semiring. Then the three following properties are equivalent:*

1. *A is determinizable.*
2. *A has the twins property.*
3. *A is subsequentiable.*

There exists an efficient algorithm for testing the twins property for weighted automata [2]. The test of the twins property for finite-state transducers and weighted automata over other semirings is also discussed by [2]. Note that any acyclic weighted automaton over a zero-sum-free semiring has the twins property and is determinizable.

5 Weight Pushing

The choice of the distribution of the total weight along each successful path of a weighted automaton does not affect the definition of the function realized by that automaton, but this may have a critical impact on the efficiency in many applications, e.g., natural language processing applications, when a heuristic pruning is used to visit only a subpart of the automaton. There exists an algorithm, *weight pushing*, for normalizing the distribution of the weights along the paths of a weighted automaton or more generally a weighted directed graph.

Let A be a weighted automaton over a semiring $\mathbb{K}$. Assume that $\mathbb{K}$ is zero-sum-free and weakly left divisible. For any state $q \in Q$, assume that the following sum is well-defined and in $\mathbb{K}$:

$$d[q] = \bigoplus_{\pi \in P(q,F)} (w[\pi] \otimes \rho[n[\pi]]) \tag{7}$$

$d[q]$ is the *shortest-distance* from q to F [11]. $d[q]$ is well-defined for all $q \in Q$ when $\mathbb{K}$ is a k-closed semiring [11]. The weight pushing algorithm consists of computing each shortest-distance $d[q]$ and of *reweighting* the transition weights, initial weights and final weights in the following way:

$$\forall e \in E \text{ s.t. } d[p[e]] \neq \overline{0}, w[e] \leftarrow d[p[e]]^{-1} \otimes w[e] \otimes d[n[e]] \tag{8}$$

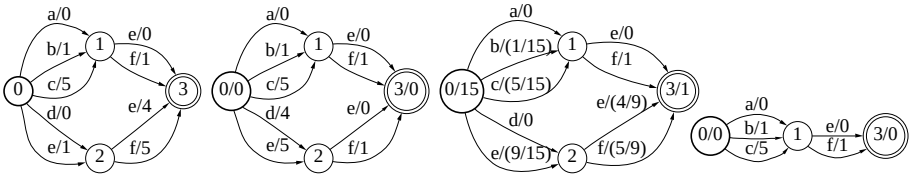
$$\forall q \in I, \lambda[q] \leftarrow \lambda[q] \otimes d[q] \tag{9}$$

$$\forall q \in F, \text{ s.t. } d[q] \neq \overline{0}, \rho[q] \leftarrow d[q]^{-1} \otimes \rho[q] \tag{10}$$

Each of these operations can be assumed to be done in constant time, thus reweighting can be done in linear time $O(T_{\otimes}|A|)$ where $T_{\otimes}$ denotes the worst cost of an $\otimes$-operation. The complexity of the computation of the shortest-distances depends on the semiring. In the case of k-closed semirings such as the tropical semiring, $d[q]$, $q \in Q$, can be computed using a generic shortest-distance algorithm [11]. The complexity of the algorithm is linear in the case of an acyclic automaton: $O(|Q| + (T_{\oplus} + T_{\otimes})|E|)$, where $T_{\oplus}$ denotes the worst cost of an $\oplus$-operation. In the case of a general weighted automaton over the tropical semiring, the complexity of the algorithm is $O(|E| + |Q| \log |Q|)$.

In the case of closed semirings such as $(\mathbb{R}_+, +, \times, 0, 1)$, a generalization of the Floyd-Warshall algorithm for computing all-pairs shortest-distances can be used. The complexity of the algorithm is $\Theta(|Q|^3(T_{\oplus} + T_{\otimes} + T_*))$ where T_* denotes the worst cost of the closure operation. The space complexity of these algorithms is $\Theta(|Q|^2)$. These complexities make it impractical to use the Floyd-Warshall algorithm for computing $d[q]$, $q \in Q$ for relatively large graphs or automata of several hundred million states or transitions. An approximate version of the shortest-distance algorithm of [11] can be used instead to compute $d[q]$ efficiently [10].

Roughly speaking, the algorithm *pushes the weights* of each path as much as possible towards the initial states. Figures 3(a)-(c) illustrate the application of the algorithm in a special case both for the tropical and probability semirings.



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

Figure 3: Weight pushing algorithm. (a) Weighted automaton A. (b) Equivalent weighted automaton B obtained by weight pushing in the tropical semiring. (c) Weighted automaton C obtained from A by weight pushing in the probability semiring. (d) Minimal weighted automaton over the tropical semiring equivalent to A.

Note that if $d[q] = \overline{0}$, then, since $\mathbb{K}$ is zero-sum-free, the weight of all paths from q to F is $\overline{0}$.

Let A be a weighted automaton over the semiring $\mathbb{K}$. Assume that $\mathbb{K}$ is closed or k-closed and that the shortest-distances $d[q]$ are all well-defined and in $\mathbb{K} - \{\overline{0}\}$. Note that in both cases we can use the distributivity over the infinite sums defining shortest distances. Let e' (π') denote the transition e (path π) after application of the weight pushing algorithm. e' (π') differs from e (resp. π) only by its weight. Let λ' denote the new initial weight function, and ρ' the new final weight function.

Proposition 2 *Let $B = (\Sigma, Q, I, F, E', \lambda', \rho')$ be the result of the weight pushing algorithm applied to the weighted automaton A, then*

1. *the weight of a successful path π is unchanged after application of weight pushing:*

$$\lambda'[p[\pi']] \otimes w[\pi'] \otimes \rho'[n[\pi']] = \lambda[p[\pi]] \otimes w[\pi] \otimes \rho[n[\pi]] \tag{11}$$

2. *the weighted automaton B is* stochastic*, i.e.*

$$\forall q \in Q, \bigoplus_{e' \in E'[q]} w[e'] = \overline{1} \tag{12}$$

Proof. Let $\pi' = e'_1 \dots e'_k$. By definition of λ' and ρ',

$$\begin{aligned} \lambda'[p[\pi']] \otimes w[\pi'] \otimes \rho'[n[\pi']] &= \lambda[p[e_1]] \otimes d[p[e_1]] \otimes d[p[e_1]]^{-1} \otimes w[e_1] \otimes d[n[e_1]] \otimes \cdots \\ &\quad \otimes d[p[e_k]]^{-1} \otimes w[e_k] \otimes d[n[e_k]] \otimes d[n[e_k]]^{-1} \otimes \rho[n[\pi]] \\ &= \lambda[p[\pi]] \otimes w[e_1] \otimes \cdots \otimes w[e_k] \otimes \rho[n[\pi]] \end{aligned}$$

which proves the first statement of the proposition. Let $q \in Q$,

$$\bigoplus_{e' \in E'[q]} w[e'] = \bigoplus_{e \in E[q]} d[q]^{-1} \otimes w[e] \otimes d[n[e]]$$

$$
\begin{aligned}
&= d[q]^{-1} \otimes \bigoplus_{e \in E[q]} w[e] \otimes d[n[e]] \\
&= d[q]^{-1} \otimes \bigoplus_{e \in E[q]} w[e] \otimes \bigoplus_{\pi \in P(n[e], F)} (w[\pi] \otimes \rho[n[\pi]]) \\
&= d[q]^{-1} \otimes \bigoplus_{e \in E[q], \pi \in P(n[e], F)} (w[e] \otimes w[\pi] \otimes \rho[n[\pi]]) \\
&= d[q]^{-1} \otimes d[q] = \overline{1}
\end{aligned}
$$

where we used the distributivity of the multiplicative operation over infinite sums in closed or k-closed semirings. This proves the second statement of the proposition. □

These two properties of weight pushing are illustrated by Figures 3(a)-(c): the total weight of a successful path is unchanged after pushing; at each state of the weighted automaton of Figure 3(b), the minimum weight of the outgoing transitions is 0, and at at each state of the weighted automaton of Figure 3(c), the weights of outgoing transitions sum to 1. Weight pushing can also be used to test the equivalence of two weighted automata [9].

6 Minimization

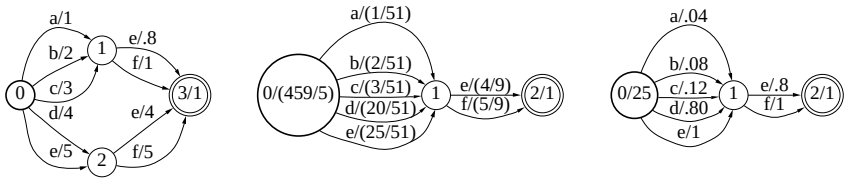
A deterministic weighted automaton is said to be *minimal* if there exists no other deterministic weighted automaton with a smaller number of states and realizing the same function. Two states of a deterministic weighted automaton are said to be *equivalent* if exactly the same set of strings with the same weights label paths from these states to a final state, the final weights being included. Thus, two equivalent states of a deterministic weighted automaton can be merged without affecting the function realized by that automaton. A weighted automaton is minimal when it admits no two distinct equivalent states after any redistribution of the weights along its paths.

There exists a general algorithm for computing a minimal deterministic automaton equivalent to a given weighted automaton [9]. The algorithm consists of first applying the weight pushing algorithm to normalize the distribution of the weights along the paths of the input automaton, and then of treating each pair (label, weight) as a single label and applying the classical (unweighted) automata minimization.

Theorem 3 ([9]) *Let A be a deterministic weighted automaton over a semiring $\mathbb{K}$. Assume that the conditions of application of the weight pushing algorithm hold, then the execution of the following steps:*

1. *weight pushing,*
2. *(unweighted) automata minimization,*

lead to a minimal weighted automaton equivalent to A.



(a) (b) (c)

Figure 4: Minimization of weighted automata. (a) Weighted automaton A' over the probability semiring. (b) Minimal weighted automaton B' equivalent to A'. (c) Minimal weighted automaton C' equivalent to A'.

The complexity of automata minimization is linear in the case of acyclic automata $O(|Q| + |E|)$ [14] and in $O(|E| \log |Q|)$ in the general case [1]. Thus, in view of the complexity results given in the previous section, in the case of the tropical semiring, the total complexity of the weighted minimization algorithm is linear in the acyclic case $O(|Q|+|E|)$ and in $O(|E| \log |Q|)$ in the general case.

Figures 3(a), 3(b), and 3(d) illustrate the application of the algorithm in the tropical semiring. The automaton of Figure 3(a) cannot be further minimized using the classical unweighted automata minimization since no two states are equivalent in that machine. After weight pushing, the automaton (Figure 3(b)) has two states (1 and 2) that can be merged by the classical unweighted automata minimization.

Figures 4(a)-(c) illustrate the minimization of an automaton defined over the probability semiring. Unlike the unweighted case, a minimal weighted automaton is not unique, but all minimal weighted automata have the same graph topology, they only differ by the way the weights are distributed along each path. The weighted automata B' and C' are both minimal and equivalent to A'. B' is obtained from A' using the algorithm described above in the probability semiring and it is thus a stochastic weighted automaton in the probability semiring.

For a deterministic weighted automaton, the first operation of the semiring can be arbitrarily chosen without affecting the definition of the function it realizes. This is because, by definition, a deterministic weighted automaton admits at most one path labeled with any given string. Thus, in the algorithm described in theorem 3, the weight pushing step can be executed in any semiring $\mathbb{K}'$ whose multiplicative operation matches that of $\mathbb{K}$. The minimal weighted automata obtained by pushing the weights in $\mathbb{K}'$ is also minimal in $\mathbb{K}$ since it can be interpreted as a (deterministic) weighted automaton over $\mathbb{K}$.

In particular, A' can be interpreted as a weighted automaton over the $(\max, \times)$-semiring $(\mathbb{R}_+, \max, \times, 0, 1)$. The application of the weighted minimization algorithm to A' in this semiring leads to the minimal weighted automaton C' of Figure 4(c). C' is also a *stochastic* weighted automaton in the sense that, at any state, the maximum weight of all outgoing transitions is one.

This fact has several interesting observations. One is related to the complexity of the algorithms. Indeed, we can choose a semiring $\mathbb{K}'$ in which the complexity of weight pushing is better than in $\mathbb{K}$. The resulting automaton is still minimal in $\mathbb{K}$ and has the additional property of being stochastic in $\mathbb{K}'$. It only differs from the weighted automaton obtained by pushing weights in $\mathbb{K}$ in the way weights are distributed along the paths. They can be obtained from each other by application of weight pushing in the appropriate semiring. In the particular case of a weighted automaton over the probability semiring, it may be preferable to use weight pushing in the $(\max, \times)$-semiring since the complexity of the algorithm is then equivalent to that of classical single-source shortest-paths algorithms. The corresponding algorithm is a special instance of the generic shortest-distance algorithm given by [11].

Another important point is that the weight pushing algorithm may not be defined in $\mathbb{K}$ because the machine is not zero-sum-free or for other reasons. But an alternative semiring $\mathbb{K}'$ can sometimes be used to minimize the input weighted automaton.

The results just presented were all related to the minimization of the number of states of a deterministic weighted automaton. The following proposition shows that minimizing the number of states coincides with minimizing the number of transitions.

Proposition 3 ([9]) *Let A be a minimal deterministic weighted automaton, then A has the minimal number of transitions.*

Proof. Let A be a deterministic weighted automaton with the minimal number of transitions. If two distinct states of A were equivalent, they could be merged, thereby strictly reducing the number of its transitions. Thus, A must be a minimal deterministic automaton. Since, minimal deterministic automata have the same topology, in particular the same number of states and transitions, this proves the proposition.

7 Conclusion

We surveyed several recent weighted finite-state transducer algorithms. These algorithms can be used in a variety of applications to create efficient and complex systems. They have been used with weighted transducers of several hundred million states and transitions to create large-vocabulary speech recognition and complex spoken-dialog systems. Other algorithms such as ϵ-removal and synchronization of weighted transducers also play a critical role in the design of such large-scale systems.

References

[1] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The design and analysis of computer algorithms.* Addison Wesley: Reading, MA, 1974.

[2] Cyril Allauzen and Mehryar Mohri. Efficient Algorithms for Testing the Twins Property. *Journal of Automata, Languages and Combinatorics*, 8(2), 2003.

[3] Jean Berstel. *Transductions and Context-Free Languages*. Teubner Studienbucher: Stuttgart, 1979.

[4] Jean Berstel and Christophe Reutenauer. *Rational Series and Their Languages*. Springer-Verlag: Berlin-New York, 1988.

[5] Karel Culik II and Jarkko Kari. Digital Images and Formal Languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 599–616. Springer, 1997.

[6] Samuel Eilenberg. *Automata, Languages and Machines*, volume A. Academic Press, 1974.

[7] Werner Kuich and Arto Salomaa. *Semirings, Automata, Languages*. Number 5 in EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, Germany, 1986.

[8] Daniel J. Lehmann. Algebraic Structures for Transitive Closures. *Theoretical Computer Science*, 4:59–76, 1977.

[9] Mehryar Mohri. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics*, 23:2, 1997.

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

[11] Mehryar Mohri. Semiring Frameworks and Algorithms for Shortest-Distance Problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.

[12] Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. Weighted Automata in Text and Speech Processing. In *Proceedings of the 12th biennial European Conference on Artificial Intelligence (ECAI-96), Workshop on Extended finite state models of language, Budapest, Hungary*. ECAI, 1996.

[13] Fernando C. N. Pereira and Michael D. Riley. Speech recognition by composition of weighted finite automata. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing*, pages 431–453. MIT Press, Cambridge, Massachusetts, 1997.

[14] Dominique Revuz. Minimisation of acyclic deterministic automata in linear time. *Theoretical Computer Science*, 92:181–189, 1992.

[15] Arto Salomaa and Matti Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Springer-Verlag: New York, 1978.

[16] Marcel Paul Schützenberger. Sur une variante des fonctions séquentielles. *Theoretical Computer Science*, 4(1):47–57, 1977.