Generic ϵ-Removal Algorithm for Weighted Automata

Mehryar Mohri

mohri@research.att.com
AT&T Labs – Research
180 Park Avenue, Rm E147
Florham Park, NJ 07932, USA

Abstract. We present a new generic ϵ-removal algorithm for weighted automata and transducers defined over a semiring. The algorithm can be used with any semiring covered by our framework and works with any queue discipline adopted. It can be used in particular in the case of unweighted automata and transducers and weighted automata and transducers defined over the tropical semiring. It is based on a general shortest-distance algorithm that we briefly describe. We give a full description of the algorithm including its pseudocode and its running time complexity, discuss the more efficient case of acyclic automata, an on-the-fly implementation of the algorithm and an approximation algorithm in the case of the semirings not covered by our framework. We also illustrate the use of the algorithm with several semirings.

1 Introduction

Weighted automata are efficient and convenient devices used in many applications such as text, speech and image processing [6, 2]. The automata obtained in such applications are often the result of various complex operations, some of them introducing the empty string ϵ. For example, using the classical method of Thompson, one can construct in linear time and space a non-deterministic automaton with ϵ-transitions representing a regular expression [9].

For the most efficient use of an automaton, it is preferable to *remove* the ϵ's of automata since in general they induce a delay in their use. An algorithm that constructs an automaton B with no ϵ's equivalent to an input automaton A with ϵ's is called *ϵ-removal.*

Textbooks do not present ϵ-removal of (unweighted) automata as an independent algorithm deserving a specific study. Instead, the algorithm is often mixed with other optimization algorithms such as determinization [1]. This usually makes the presentation of determinization more complex and the underlying ϵ-removal process obscure. Since ϵ-removal is not presented as an independent algorithm, it is usually not analyzed and its running time complexity not clearly determined.

We present a new generic ϵ-removal algorithm for weighted automata and transducers defined over a semiring. The algorithm can be used with any semiring covered by our framework and works with any queue discipline adopted. It can be used in particular in the case of unweighted automata and transducers and weighted automata and transducers defined over the tropical semiring. It is based on a general shortest-distance algorithm that we briefly describe. We give a full description of the algorithm including its pseudocode and its running time complexity, discuss the more efficient case of acyclic automata, an on-the-fly implementation of the algorithm and an approximation algorithm in the case of the semirings not covered by our framework. We also illustrate the use of the algorithm with several semirings.

2 Preliminaries

Weighted automata are automata in which the transitions are labeled with weights in addition to the usual alphabet symbols. For various operations to be well-defined, the weight set needs to have the algebraic structure of a semiring [5].

Definition 1. *A system* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *is a* right semiring *if:*

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. $\otimes$ *right distributes over* $\oplus$: $\forall a, b, c \in \mathbb{K}, (a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$,
4. $\overline{0}$ *is an annihilator for* $\otimes$: $\forall a \in \mathbb{K}, a \otimes \overline{0} = \overline{0} \otimes a = \overline{0}$.

Left semirings are defined in a similar way by replacing right distributivity with left distributivity. $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is a *semiring* if both left and right distributivity hold. Thus, more informally, a semiring is a ring that may lack negation. As an example, $(\mathbb{N}, +, \cdot, 0, 1)$ is a semiring defined on the set of nonnegative integers $\mathbb{N}$.

A semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is said to be *idempotent* if for any $a \in \mathbb{K}$, $a \oplus a = a$. The boolean semiring $\mathcal{B} = (\{0, 1\}, \vee, \wedge, 0, 1)$ and the tropical semiring $\mathcal{T} = (\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ are idempotent, but $(\mathbb{N}, +, \cdot, 0, 1)$ is not.

Definition 2. *A* weighted automaton $A = (\Sigma, Q, I, F, E, \lambda, \rho)$ *over the semiring* $\mathbb{K}$ *is a 7-tuple where* Σ *is the finite alphabet of the automaton,* Q *is a finite set of states,* $I \subseteq Q$ *the set of initial states,* $F \subseteq Q$ *the set of final states,* $E \subseteq Q \times \Sigma \times \mathbb{K} \times Q$ *a finite set of transitions,* $\lambda : I \to \mathbb{K}$ *the initial weight function mapping* I *to* $\mathbb{K}$*, and* $\rho : F \to \mathbb{K}$ *the final weight function mapping* F *to* $\mathbb{K}$.

Given a transition $e \in E$, we denote by $i[e]$ its input label, $w[e]$ its weight, $p[e]$ its origin or previous state and $n[e]$ its destination state or next state. Given a state $q \in Q$, we denote by $E[q]$ the set of transitions leaving q, and by $E^R[q]$ the set of transitions entering q.

A *path* $\pi = e_1 \cdots e_k$ in A 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 $p[\pi] = p[e_1]$. We denote by $P(q, q')$ the set of paths from q to q'. P can be extended to subsets $R \subseteq Q$ $R' \subseteq Q$, by:

$$P(R, R') = \bigcup_{q \in R,\, q' \in R'} P(q, q')$$

The labeling function i and the weight function w can also be extended to paths by defining the label of a path as the concatenation of the labels of its constituent transitions, and the weight of a path as the $\otimes$-product of the weights of its constituent transitions:

$$i[\pi] = i[e_1] \cdots i[e_k]$$
$$w[\pi] = w[e_1] \otimes \cdots \otimes w[e_k]$$

Given a string $x \in \Sigma^*$, we denote by $P(x)$ the set of paths from I to F labeled with x:

$$P(x) = \{\pi \in P(I, F) : i[\pi] = x\}$$

The output weight associated by A to an input string $x \in \Sigma^*$ is:

$$A \cdot x = \bigoplus_{\pi \in P(x)} \lambda(p[\pi]) \otimes w[\pi] \otimes \rho(n[\pi])$$

If $P(x) = \emptyset$, $A \cdot x$ is defined to be $\overline{0}$. Note that weighted automata over the boolean semiring are equivalent to the classical unweighted finite automata.

These definitions can be easily generalized to cover the case of weighted automata with ϵ-transitions. An ϵ-removal algorithm computes for any input weighted automaton A with ϵ-transitions an equivalent weighted automaton B with no ϵ-transition, that is such that:

$$\forall x \in \Sigma^*, A \cdot x = B \cdot x$$

The following definitions will help us define the framework for our generic ϵ-removal algorithm [7].

Definition 3. *Let $k \geq 0$ be an integer. A commutative semiring* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *is k-*closed *if:*

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

When $k = 0$, the previous expression can be rewritten as:

$$\forall a \in \mathbb{K}, \overline{1} \oplus a = \overline{1}$$

and $\mathbb{K}$ is then said to be *bounded.* Semirings such as the boolean semiring $\mathcal{B} = (\{0, 1\}, \vee, \wedge, 0, 1)$ and the tropical semiring $\mathcal{T} = (\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ are bounded.

Definition 4. *Let $k \geq 0$ be an integer, $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ a commutative semiring, and A a weighted automaton over $\mathbb{K}$. $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is* right k-closed for A *if for any cycle π of A:*

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

By definition, if $\mathbb{K}$ is k-closed, then it is k-closed for any automaton over $\mathbb{K}$.

3 Algorithm

Let $A = (\Sigma, Q, I, F, E, \lambda, \rho)$ be a weighted automaton over the semiring $\mathbb{K}$ with ϵ-transitions. We denote by A_ϵ the automaton obtained from A by removing all transitions not labeled with ϵ. We present a general ϵ-removal algorithm for weighted automata based on a generic shortest-distance algorithm which works for any semiring $\mathbb{K}$ k-closed for A_ϵ [7]. In what follows, we assume that $\mathbb{K}$ has this property. This class of semirings includes in particular the boolean semiring $(\{0,1\}, \vee, \wedge, 0, 1)$, the tropical semiring $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ and other semirings not necessarily idempotent. [1]

We present the algorithm in the case of weighted automata. The case of weighted transducers can be straightforwardly derived from the automata case by viewing a transducer as an automaton over the alphabet $\Sigma \cup \{\epsilon\} \times \Sigma \cup \{\epsilon\}$. An ϵ-transition of a transducer is then a transition labeled with (ϵ, ϵ). [2]

For p, q in Q, the ϵ-distance from p to q in the automaton A is denoted by $d[p,q]$ and defined as:

$$d[p,q] = \bigoplus_{\pi \in P(p,q),\, i[\pi]=\epsilon} w[\pi]$$

This distance is well-defined for any pair of states (p,q) of A when $\mathbb{K}$ is a semiring k-closed for A_ϵ [7]. By definition, for any $p \in Q$, $d[p,p] = \overline{1}$. $d[p,q]$ is the distance from p to q in A_ϵ.

3.1 Description and Proof

The algorithm works in two steps. The first step consists of computing for each state p of the input automaton A its ϵ-closure denoted by $C[p]$:

$$C[p] = \{(q,w) : q \in \epsilon[p],\, d[p,q] = w \in \mathbb{K} - \{\overline{0}\}\}$$

[1] The class of semirings with which our generic shortest-distance algorithm works can in fact be extended to that of right semirings *right k-closed for A_ϵ* [7], but for reasons of space we do not describe this more general framework here.

[2] We have also developed an algorithm for removing only input (or output) ϵ's of a weighted transducer without any modification to the alphabet of the transducer. That algorithm is also based on the generic shortest-distance algorithm presented in the next sections. It applies to all transducers that admit an equivalent input (or output) ϵ-free transducers. The complexity of the algorithm is exponential since in the worst case the size of the output transducer can be exponential in the input size.

where $\epsilon[p]$ represents the set of states reachable from p via a path labeled with ϵ. We describe this step in more detail later.

The second step consists of modifying the outgoing transitions of each state p by removing those labeled with ϵ and by adding to $E[p]$ non-ϵ-transitions leaving each state $q \in \epsilon[p]$ with their weights pre-$\otimes$-multiplied by $d[p, q]$. The following is the pseudocode of the second step of the algorithm.

```
ϵ-removal(A)
1  for each p ∈ Q
2      do E[p] ← {e ∈ E[p] : i[e] ≠ ϵ}
3         for each (q, w) ∈ C[p]
4             do E[p] ← E[p] ∪ {(p, a, w ⊗ w′, r) : (q, a, w′, r) ∈ E[q], a ≠ ϵ}
5                if q ∈ F
6                   then if p ∉ F
7                           then F ← F ∪ {p}
8                        ρ[p] ← ρ[p] ⊕ (w ⊗ ρ[q])
```

State p is a final state if some state $q \in \epsilon[p]$ is final and the final weight $\rho[p]$ is then:

$$\rho[p] = \bigoplus_{q \in \epsilon[p] \cap F} (d[p, q] \otimes \rho[q])$$

Theorem 1. *Let $A = (\Sigma, Q, I, F, E, \lambda, \rho)$ be a weighted automaton over the semiring $\mathbb{K}$ right k-closed for A. Then the weighted automaton B result of the ϵ-removal algorithm just described is equivalent to A.*

Proof. We show that the function defined by the weighted automaton A is not modified by the application of one step of the loop of the pseudocode above. Let $A = (\Sigma, Q, I, F, E, \lambda, \rho)$ be the automaton just before the removal of the ϵ-transitions leaving state $p \in Q$ (lines 2-8).

Let $x \in \Sigma^*$, and let $Q(p)$ denote the set of successful paths labeled with x passing through p and either ending at p or following an ϵ-transition of p. By commutativity of $\oplus$, we have:

$$A \cdot x = \bigoplus_{\pi \in P(x) - Q(p)} \lambda(p[\pi]) \otimes w[\pi] \otimes \rho(n[\pi]) \oplus \bigoplus_{\pi \in Q(p)} \lambda(p[\pi]) \otimes w[\pi] \otimes \rho(n[\pi])$$

Denote by S_1 and S_2 the first and second term of that sum. The ϵ-removal at state p does not affect the paths not in $Q(p)$, thus we can limit our attention to the second term S_2. A path in $Q(p)$ can be factored in: $\pi = \pi_1 \pi_\epsilon \pi_2$, where π_ϵ is a portion of the path from p to $n[\pi_\epsilon] = q$ labeled with ϵ. The distributivity of $\otimes$ over $\oplus$ gives us:

$$S_2 = (\bigoplus_{n[\pi_1] = p} \lambda(p[\pi_1]) \otimes w[\pi_1])) \otimes S_{22}$$

with:

$$
\begin{aligned}
S_{22} &= (\bigoplus_{p[\pi_\epsilon]=p,\, n[\pi_\epsilon]=q} w[\pi_\epsilon]) \otimes \bigoplus_{p[\pi_2]=q} w[\pi_2] \otimes \rho(n[\pi_2]) \\
&= d[p,q] \otimes \bigoplus_{p[\pi_2]=q} w[\pi_2] \otimes \rho(n[\pi_2]) \\
&= \bigoplus_{p[\pi_2]=q} (d[p,q] \otimes w[\pi_2]) \otimes \rho(n[\pi_2])
\end{aligned}
$$

For paths π such that $\pi_2 = \epsilon$:

$$
S_{22} = \bigoplus_{p[\pi_2]=q,\, q\in\epsilon[p]} (d[p,q] \otimes \rho(q))
$$

which is exactly the final weight associated to p by ϵ-removal at p. Otherwise, by definition of π_ϵ, π_2 does not start with an ϵ-transition. The second term of the sum defining S_{22} can thus be rewritten as:

$$
S_{22} = \bigoplus_{e\in E[q],\, i[e]\neq\epsilon,\, \pi_2=e\pi'_2} (d[p,q] \otimes w[e]) \otimes w[\pi'_2] \otimes \rho(n[\pi_2])
$$

The ϵ-removal step at state p follows exactly that decomposition. It adds non ϵ-transitions e leaving q with weights $(d[p,q] \otimes w[e])$ to the transitions leaving p. This ends the proof of the theorem. □

After removing ϵ's at each state p, some states may become inaccessible if they could only be reached by ϵ-transitions originally. Those states can be removed from the result in time linear in the size of the resulting machine using for example a depth-first search of the automaton.

Figures 1 (a)-(c) illustrate the use of the algorithm in the specific case of the tropical semiring. ϵ-transitions can be *removed* in at least two ways: in the way described above, or the reverse. The latter is equivalent to applying ϵ-removal to the reverse of the automaton and re-reversing the result. The two methods may lead to results of very different sizes as illustrated by the figures. This is due to two factors that are independent of the semiring:

- the number of states in the original automaton whose incoming transitions (or outgoing transitions in the reverse case) are all labeled with the empty string. As mentioned before, those states can be removed from the result. For example, state 3 of the automaton of figure 1 (a) can only be reached by ϵ-transitions and admits only outgoing transitions labeled with ϵ. Thus, that state does not appear in the result in both methods (figures 1 (b)-(c)). The incoming transitions of state 2 are all labeled with ϵ and thus it does not appear in the result of the ϵ-removal with the first method, but it does in the reverse method because the outgoing transitions of state 2 are not all labeled with ϵ;
- the total number of non ϵ-transitions of the states that can be reached from each state q in A_ϵ (the reverse of A_ϵ in the reverse case). This corresponds to the number of outgoing transitions of q in the result of ϵ-removal.

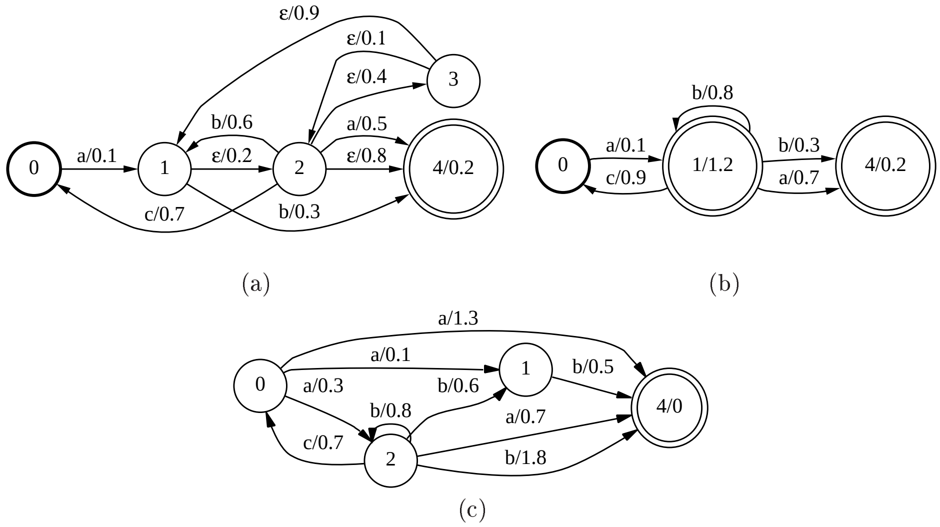


Figure 1. ϵ-removal in the tropical semiring. (a) Weighted automaton A with ϵ-transitions. (b) Weighted automaton B equivalent to A result of the ϵ-removal algorithm. (c) Weighted automaton C equivalent to A obtained by application of ϵ-removal to the reverse of A.

In practice, one can use some heuristics to reduce the number of states and transitions of the resulting machine although this will not affect the worst case complexity of the algorithm. One can for instance remove some ϵ-transitions in the reverse way when that creates less transitions and others in the way corresponding to the first method when that helps reducing the resulting size.

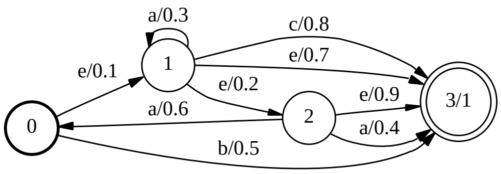
Figures 2 (a)-(b) illustrate the algorithm in the case of another semiring, the semiring of real numbers. Our general algorithm applies in this case since A_ϵ is acyclic.

3.2 Computation of ϵ-closures

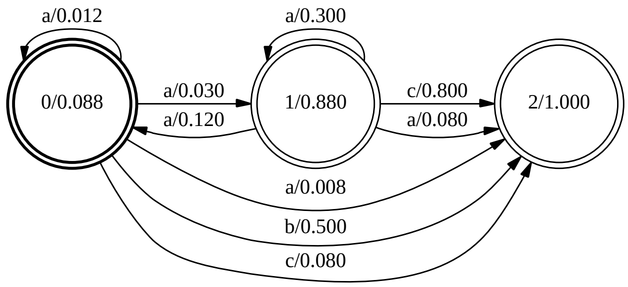
As noticed before, the computation of ϵ-closures is equivalent to that of all-pairs shortest-distances over the semiring $\mathbb{K}$ in A_ϵ. There exists a generalization of the algorithm of Floyd-Warshall [3, 11] for computing the all-pairs shortest-distances over a semiring $\mathbb{K}$ under some general conditions [7].[3] However, the running time complexity of that algorithm is cubic:

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

[3] That algorithm works in particular with the semiring $(\mathbb{R}, +, *, 0, 1)$ when the weight of each cycle of A_ϵ admits a well-defined closure.



(a)



(b)

Figure 2. ϵ-removal in the real semiring $(\mathbb{R}, +, *, 0, 1)$. (a) Weighted automaton A. A_ϵ is k-closed for $(\mathbb{R}, +, *, 0, 1)$. (b) Weighted automaton B equivalent to A output of the ϵ-removal algorithm.

where $T_\oplus$, $T_\otimes$, and T_* denote the cost of $\oplus$, $\otimes$, and closure operations in the semiring considered. The algorithm can be improved by first decomposing A_ϵ into its strongly connected components, and then computing all-pairs shortest distances in each components visited in reverse topological order. However, it is still impractical for large automata when A_ϵ has large cycles of several thousand states. The quadratic space complexity $O(|Q|^2)$ of the algorithm also makes it prohibitive for such large automata. Another problem with this generalization of the algorithm of Floyd-Warshall is that it does not exploit the sparseness of the input automaton.

There exists a generic single-source shortest-distance algorithm that works with any semiring covered by our framework [7]. The algorithm is a generalization of the classical shortest-paths algorithms to the case of the semirings of this framework.

This generalization is not trivial and does not require the semiring to be idempotent. In particular, a straightforward extension of the classical algorithms based on a relaxation technique would not produce the correct result in general [7]. The algorithm is also generic in the sense that it works with any queue discipline. The following is the pseudocode of the algorithm.

A queue S is used to maintain the set of states whose leaving transitions are to be relaxed. S is initialized to $\{s\}$ (line 4). For each state $q \in Q$, two attributes

```
GENERIC-SINGLE-SOURCE-SHORTEST-DISTANCE (B, s)
1  for each p ∈ Q
2      do d[p] ← r[p] ← 0̄
3  d[s] ← r[s] ← 1̄
4  S ← {s}
5  while S ≠ ∅
6      do q ← head(S)
7         DEQUEUE(S)
8         r ← r[q]
9         r[q] ← 0̄
10        for each e ∈ E[q]
11            do if d[n[e]] ≠ d[n[e]] ⊕ (r ⊗ w[e])
12                  then d[n[e]] ← d[n[e]] ⊕ (r ⊗ w[e])
13                       r[n[e]] ← r[n[e]] ⊕ (r ⊗ w[e])
14                       if n[e] ∉ S
15                          then ENQUEUE(S, n[e])
16 d[s] ← 1̄
```

Figure 3. Generic single-source shortest-distance algorithm.

are maintained: $d[q] \in \mathbb{K}$ 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 state $q \in Q$ and an transition $e \in E[q]$, a relaxation step on e is performed by lines 11-13 of the pseudocode, where r is the value of $r[q]$ just after the latest extraction of q from S if q has ever been extracted from S, its initialization value otherwise.

Each time through the **while** loop of lines 5-15, a state 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 transition 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 state $n[e]$ is inserted in S so that its leaving transitions be later relaxed (lines 14-15). $r[n[e]]$ is updated whenever $d[n[e]]$ is. $r[n[e]]$ stores the total weight $\oplus$-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 $\overline{1}$.

In the general case, the complexity of the algorithm depends on the semiring considered and the queue discipline chosen for S:

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

where $N(q)$ denotes the number of times state q is extracted from S, $C(E)$ the worst cost of removing a state q from the queue S, $C(I)$ that of inserting q in S, and $C(A)$ the cost of an assignment. [4]

In the case of the tropical semiring $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$, the algorithm coincides with classical single-source shortest-paths algorithms. In particular, it coincides with Bellman-Ford's algorithm when a FIFO queue discipline is used and with Dijkstra's algorithm when a shortest-first queue discipline is used. Using Fibonacci heaps [4], the complexity of Dijkstra's algorithm in the tropical semiring is:

$$O(|E| + |Q| \log |Q|)$$

The complexity of the algorithm is linear in the case of an acyclic automaton with a topological order queue discipline:

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

Note that the topological order queue discipline can be generalized to the case of non-acyclic automata A_ϵ by decomposing A_ϵ into its strongly connected components. Any queue discipline can then be used to compute the all-pairs shortest distances within each strongly connected component [7].

The all-pairs shortest-distances of A_ϵ can be computed by running $|Q|$ times the generic single-source shortest-distance algorithm. Thus, when A_ϵ is acyclic, that is when A admits no ϵ-cycle, then the all-pairs shortest distances can be computed in quadratic time:

$$O(|Q|^2 + (T_\oplus + T_\otimes)|Q| \cdot |E|)$$

When A_ϵ is acyclic, the complexity of the computation of the all-pairs shortest distances can be substantially improved if the states of A_ϵ are visited in reverse topological order and if the shortest-distance algorithm is interleaved with the actual removal of ϵ's. Indeed, one can proceed in the following way. For each state p of A_ϵ visited in reverse topological order:

1. run a single-source shortest-distance algorithm with source p to compute the distance from p to each state q reachable from p by ϵ's;
2. remove the ϵ-transitions leaving q as described in the previous section.

The reverse topological order guarantees that the ϵ-paths leaving p are reduced to the ϵ-transitions leaving p. Thus, the cost of the shortest-distance algorithm run from p only depends on the number of ϵ-transitions leaving p and the total cost of the computation of the shortest-distances is linear:

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

In the case of the tropical semiring and using Fibonacci heaps, the complexity of the first stage of the algorithm is:

$$O(|Q| \cdot |E| + |Q|^2 \log |Q|)$$

[4] This includes the potential cost of reorganization of the queue to perform this assignment.

In the worst case, in the second stage of the algorithm each state q belongs to the ϵ-closure of each state p, and the removal of ϵ's can create in the order of $|E|$ transitions at each state. Hence, the complexity of the second stage of the algorithm is:

$$O(|Q|^2 + |Q| \cdot |E|)$$

Thus, the total complexity of the algorithm in the case of an acyclic automaton A_ϵ is:

$$O(|Q|^2 + (T_\oplus + T_\otimes)|Q| \cdot |E|)$$

In the case of the tropical semiring and with a non acyclic automaton A_ϵ, the total complexity of the algorithm is:

$$O(|Q| \cdot |E| + |Q|^2 \log |Q|)$$

4 Remarks and experiments

We have fully implemented the ϵ-removal algorithm presented here and incorporated it in recent versions of the FSM library [8]. For some automata of about a thousand states with large ϵ-cycles, our implementation is up to 600 times faster than the previous implementation based on a generalization of the Floyd-Warshall algorithm.

An important feature of our algorithm is that it admits a natural on-the-fly implementation. Indeed, the outgoing transitions of state q of the output automaton can be computed directly using the ϵ-closure of q. However, with an on-the-fly implementation, a topological order cannot be used for the queue S even if A_ϵ is acyclic since this is not known ahead of the time. Thus, we have implemented both an off-line and an on-the-fly version of the algorithm.

The algorithms presented in textbooks often combine the classical powerset construction, or determinization, and ϵ-removal [1]. Each state of the resulting automaton corresponds to the ϵ-closure of a subset of states constructed as in determinization.

If one wishes to apply determinization after ϵ-removal, then the integration of ϵ-removal within determinization may often be more efficient. This is because to compute the ϵ-closure of a subset created by determinization, one can run a single shortest-distance algorithm from the states of that subset rather than a distinct one for each state of the subset. Since some states can be reached by several elements of the subset, the first method provides more sharing. This is also corroborated by experiments carried out on various automata in the context of natural language processing applications [10].

This integration of determinization and ϵ-removal can be extended to the weighted case if one replaces the classical unweighted determinization by weighted determinization [6] and if one uses the weighted ϵ-closure presented in the previous sections. It is however limited to the cases where the weighted automaton resulting from ϵ-removal is determinizable since in the weighted case this is not always guaranteed.

Notice, however, that in the integrated algorithm presented in textbooks, the computation of the ϵ-closure is limited to the use of a FIFO queue discipline [1]. Thus, it is a special case of our general algorithm which may be less efficient both in terms of complexity and in practice than using a shortest-first queue discipline as in Dijkstra's algorithm, or a topological order in the case of acyclic automata. Our experiments confirm that in the case of acyclic automata, the ϵ-removal based on a topological order queue discipline can be many times faster than the one based on a shortest-first queue discipline, which itself is faster than the one based on a FIFO queue discipline.

The algorithm we presented can be straightforwardly modified to remove transitions with a label a different from ϵ. This can be done for example by replacing ϵ by a new label and a by ϵ, applying ϵ-removal and then restoring original ϵ's.

The shortest-distance algorithm presented in the previous sections admits an approximation version where the equality of line 11 in the pseudocode of figure 3 is replaced by an approximate equality modulo some predefined constant δ [7]. This can be used to remove ϵ-transitions of a weighted automaton A over a semiring $\mathbb{K}$ such as $(\mathbb{R}, +, *, 0, 1)$, even when $\mathbb{K}$ is not k-closed for A_ϵ. Although the transition weights are then only approximations of the correct results, this may be satisfactory for many practical purposes such as speech processing applications. Furthermore, one can arbitrarily improve the quality of the approximation by reducing δ. As mentioned before, one can also use a generalization of the Floyd-Warshall algorithm to compute the result in an exact way in the case of machines A_ϵ with relatively small strongly connected components and when the weight of each cycle of A_ϵ admits a well-defined closure.

5 Conclusion

A generic algorithm for ϵ-removal of weighted automata was given. The algorithm works with any semiring covered by our framework. In particular, it can work with semirings that are not necessarily idempotent. It works with any weighted automaton or transducer over the tropical semiring or the boolean semiring. It was shown to be more efficient than the existing algorithm both in space and time complexity. Experiments confirm this improvement of efficiency in practice. The algorithm admits a natural on-the-fly implementation which can be combined with other on-the-fly algorithms such as determinization or composition of weighted automata to optimize weighted automata.

Acknowledgments

I thank Michael Riley for several discussions about this work.

References

1. A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers, Principles, Techniques and Tools.* Addison Wesley: Reading, MA, 1986.
2. K. Culik II and J. Kari. Digital images and formal languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, pages 599–616. Springer, 1997.
3. R. W. Floyd. Algorithm 97 (shortest path). *Communications of the ACM*, 18, 1968.
4. M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved optimalization problems. *Communications of the ACM*, 34, 1987.
5. W. Kuich and A. Salomaa. *Semirings, Automata, Languages.* Number 5 in EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin-New York, 1986.
6. M. Mohri. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics*, 23:2, 1997.
7. M. Mohri. General Algebraic Frameworks and Algorithms for Shortest-Distance Problems. Technical Memorandum 981210-10TM, AT&T Labs - Research, 62 pages, 1998.
8. M. Mohri, F. C. N. Pereira, and M. Riley. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231:17–32, January 2000.
9. K. Thompson. Regular expression search algorithm. *Comm. ACM*, 11, 1968.
10. G. van Noord. Treatment of epsilon moves in subset construction. *Computational Linguistics*, 26:1, 2000.
11. S. Warshall. A theorem on boolean matrices. *Journal of the ACM*, 9(1):11–12, 1962.