On the Computation of Some Standard Distances Between Probabilistic Automata

Corinna Cortes[1], Mehryar Mohri[1,2,⋆], and Ashish Rastogi[2]

[1] Google Research,
1440 Broadway, New York, NY 10018
[2] Courant Institute of Mathematical Sciences,
251 Mercer Street, New York, NY 10012

Abstract. The problem of the computation of a distance between two probabilistic automata arises in a variety of statistical learning problems. This paper presents an exhaustive analysis of the problem of computing the L_p distance between two automata. We give efficient exact and approximate algorithms for computing these distances for p even and prove the problem to be NP-hard for all odd values of p, thereby completing previously known hardness results. We also give an efficient algorithm for computing the Hellinger distance between unambiguous probabilistic automata. Our results include a general algorithm for the computation of the norm of an unambiguous probabilistic automaton based on a monoid morphism and efficient algorithms for the specific case of the computation of the L_p norm. Finally, we also describe an efficient algorithm for testing the equivalence of two arbitrary probabilistic automata A_1 and A_2 based on Schützenberger's standardization with a running time complexity of $O(|\Sigma| \, (|A_1| + |A_2|)^3)$, a significant improvement over the previously best algorithm reported for this problem.

1 Introduction

A probabilistic automaton is a finite automaton with transition probabilities. It represents a probability distribution over the set of all strings [14]. Probabilistic automata are used extensively in a variety of areas, including text and speech processing [11], image processing [5], and computational biology [6].

These automata are typically derived from large data sets using statistical learning algorithms. The convergence of these algorithms is often tested by measuring the distance between the probabilistic automata obtained after consecutive iterations. The computation of the distance between probabilistic automata is also needed in other learning problems such as clustering when the objects to cluster, e.g., documents, images, biosequences, are modeled as Hidden Markov Models (HMMs) or probabilistic automata.

This motivates our study of the computation of various distances between probabilistic automata. We have previously shown that the relative entropy,

[⋆] This work was partially funded by the New York State Office of Science Technology and Academic Research (NYSTAR).

O.H. Ibarra and H.-C. Yen (Eds.): CIAA 2006, LNCS 4094, pp. 137–149, 2006.

or Kullback-Leibler divergence, of unambiguous probabilistic automata can be computed efficiently [4] and that, in the general case of arbitrary probabilistic automata, the computational cost is at least $O(c^{\sqrt{n/\log n}})$, where c is a constant and n the size of the automaton [3].

Here, we present an exhaustive analysis of the problem of computing the L_p distance between two automata. We give efficient exact and approximate algorithms for computing these distances for p even and prove that the problem is NP-hard for all odd values of p using a reduction from the Max-clique problem by [15]. These latter results complete those given by [15] who showed the problem to be NP-hard for L_1 and L_∞. We also give an algorithm for computing the Hellinger distance between unambiguous probabilistic automata. In addition, we present a general algorithm for the computation of the norm of an unambiguous probabilistic automaton using a monoid morphism and give efficient algorithms for the specific case of the computation of the L_p norm.

A problem closely related to that of computing a distance between two probabilistic automata is to test for their equivalence. Our algorithm for computing the L_2 distance of two arbitrary probabilistic automata A_1 and A_2 provides in fact a polynomial-time method for testing their equivalence since A_1 and A_2 are equivalent iff their L_2 distance is null. However, we will describe a more efficient algorithm based on Schützenberger's standardization technique [17,1] with a running-time complexity of $O(|\Sigma|\,(|A_1| + |A_2|)^3)$, a significant improvement over the previously best algorithm reported for this problem whose complexity is $O(|\Sigma|\,(|A_1| + |A_2|)^4))$ [19].

The remainder of the paper is organized as follows. Section 2 introduces some basic algebraic definitions and notation related to probabilistic automata needed for the description of our algorithms. Section 3 presents several algorithms for the computation of the norm of a probabilistic automaton, including an approximate solution. The problem of the computation of the L_p distance and Hellinger distance is examined in detail in Section 4.

2 Preliminaries

Definition 1. *Let* $(\mathbb{K}, \otimes, \overline{1})$ *be a monoid. A function* $\Phi : (\mathbb{R}_+, \cdot, 1) \to (\mathbb{K}, \otimes, \overline{1})$ *is said to be a* monoid morphism *if* $\Phi(1) = \overline{1}$, $\Phi(0) = \overline{0}$, *and* $\Phi(x \cdot y) = \Phi(x) \otimes \Phi(y)$ *for all* $x, y, \in \mathbb{R}_+$.

Definition 2 ([10]). *A* semiring *is a system* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *such that:* $(\mathbb{K}, \oplus, \overline{0})$ *is a commutative monoid with* $\overline{0}$ *as the identity element for* $\oplus$*;* $(\mathbb{K}, \otimes, \overline{1})$ *is a monoid with* $\overline{1}$ *as the identity element for* $\otimes$*;* $\otimes$ *distributes over* $\oplus$*: for all* a, b, c *in* $\mathbb{K}$*:* $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$ *and* $c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)$*, and* $\overline{0}$ *is an annihilator for* $\otimes$*:* $\forall a \in \mathbb{K}, a \otimes \overline{0} = \overline{0} \otimes a = \overline{0}$.

A semiring $\mathbb{K}$ is said to be *closed* if for all $a \in \mathbb{K}$, the infinite sum $\bigoplus_{n=0}^{\infty} a^n$ is well-defined and in $\mathbb{K}$, and if associativity, commutativity, and distributivity apply to countable sums [13]. $\mathbb{K}$ is said to be k-closed if for all $a \in \mathbb{K}$, $\bigoplus_{n=0}^{k+1} a^n = \bigoplus_{n=0}^{k} a^n$. More generally, we will say that $\mathbb{K}$ is *closed* (*k-closed*) *for an automaton*

A, if the closedness (resp. k-closedness) axioms hold for all cycle weights of A. In some semirings, e.g., the probability semiring $(\mathbb{R}_+, +, \cdot, 0, 1)$, the equality $\bigoplus_{n=0}^{k+1} a^n = \bigoplus_{n=0}^{k} a^n$ may hold for the cycle weights of A only approximately, modulo $\epsilon > 0$. A is then said to be *ϵ-k-closed* for that semiring.

Definition 3 ([7,16,1]). *A* weighted automaton $A = (\Sigma, Q, I, F, E, \lambda, \rho)$ *over a semiring* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *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, $E \subseteq Q \times \Sigma \cup \{\epsilon\} \times \mathbb{K} \times Q$ 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 $\mathbb{K}$.*

Stochastic automata are probabilistic automata such that at each state the weights of the outgoing transitions and the final weight sum to one.

We denote by $|A| = |E|+|Q|$ the size of an automaton $A = (\Sigma, Q, I, F, E, \lambda, \rho)$, that is the sum of the number of states and transitions of A. Given a transition $e \in E$, we denote by $i[e]$ its input label, $p[e]$ its origin or previous state and $n[e]$ its destination state or next state, $w[e]$ its weight (weighted automata case). Given a state $q \in Q$, we denote by $E[q]$ the set of transitions leaving 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' and by $P(q, x, q')$ the set of paths from q to q' with input label $x \in \Sigma^*$. The labeling functions 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: $i[\pi] = i[e_1] \cdots i[e_k]$, $w[\pi] = w[e_1] \otimes \cdots \otimes w[e_k]$.

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

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

Definition 4. *A weighted automaton A defined over the probability semiring $(\mathbb{R}_+, +, \cdot, 0, 1)$ is said to be* probabilistic *if for any state $q \in Q$, $\sum_{\pi \in P(q,q)} w[\pi]$, the sum of the weights of all cycles at q, is well-defined and in $\mathbb{R}$, and the weights assigned to the all strings sums to one: $\sum_{x \in \Sigma^*} [\![A]\!](x) = 1$.*

A weighted automaton is said to be *unambiguous* if for any string $x \in \Sigma^*$ it admits at most one accepting path labeled with x. It is said to be *deterministic* or *subsequential* if it has a unique initial state and if no two transitions leaving the same state share the same input label.

3 Computation of the Norm of a Probabilistic Automaton

The computation of single-source shortest-distances is needed in many of the algorithms presented in this section and the following ones. We denote by $s[A]$

the $\oplus$-sum of the weights of all successful paths of a weighted automaton A when it is defined and in $\mathbb{K}$. $s[A]$ can be viewed as the *shortest-distance* from the initial states to the final states.

When the semiring $\mathbb{K}$ is closed, or when A is closed for $\mathbb{K}$, $s[A]$ can be computed exactly using a generalization of the Floyd-Warshall algorithm in time $O(|A|^3)$ and space $\Omega(|A|^2)$, assuming a constant cost for the semiring operations [13].

3.1 Case of Unambiguous Automata

In previous work, we gave a general algorithm for computing the entropy of a probabilistic automaton by relating this problem to a shortest-distance one [4]. Here, we generalize these results by considering an arbitrary monoid morphism.

Let $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ be a closed, or ϵ-k-closed semiring. Let $\Phi : (\mathbb{R}_+, \cdot, 1) \to (\mathbb{K}, \otimes, \overline{1})$ be a monoid morphism. We will say that Φ *preserves closedness*, if for all x, $0 \leq x < 1$, $\bigoplus_{n=0}^{\infty} \Phi(x^n)$ is well-defined and in $\mathbb{K}$. For a such a morphism, we can define the *Φ-norm of a probabilistic automaton* as:

$$\|A\|_{\Phi} = \bigoplus_{x \in \Sigma^*} \Phi(\llbracket A \rrbracket(x)). \tag{2}$$

Theorem 1. *Let $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ be a closed or ϵ-k-closed semiring and let $\Phi : (\mathbb{R}_+, \cdot, 1) \to (\mathbb{K}, \otimes, \overline{1})$ be a monoid morphism preserving closedness. Then, for any unambiguous probabilistic automaton A, $\|A\|_{\Phi}$ can be computed exactly in time $O(|A|^3)$.*

Proof. The automaton $\Phi(A)$ derived from A by replacing each weight x by $\Phi(x)$ is a weighted automaton over the semiring $\mathbb{K}$. Since A is unambiguous, at most one path in A, $\pi = e_1 \cdots e_k$, is labeled with any string $x \in \Sigma^*$. Since Φ is a monoid morphism, $\Phi(\llbracket A \rrbracket(x)) = \bigotimes_{j=1}^{k} \Phi(i[e_j])$, that is the weight of the path labeled with x in $\Phi(A)$. This shows that $\|A\|_{\Phi} = s(A)$ and proves the theorem. □

Theorem 1 provides an algorithm for computing the Φ-norm of unambiguous probabilistic automata for arbitrary monoid morphisms preserving closedness. We will briefly illustrate two applications of the theorem.

(a) Entropy of a Probabilistic Automaton.

Let $\mathbb{K}$ denote $(\mathbb{R} \cup \{+\infty, -\infty\}) \times (\mathbb{R} \cup \{+\infty, -\infty\})$. For pairs (x_1, y_1) and (x_2, y_2) in $\mathbb{K}$, define the following :

$$(x_1, y_1) \oplus (x_2, y_2) = (x_1 + x_2, y_1 + y_2) \tag{3}$$

$$(x_1, y_1) \otimes (x_2, y_2) = (x_1 x_2, x_1 y_2 + x_2 y_1) \tag{4}$$

Then, the system $(\mathbb{K}, \oplus, \otimes, (0, 0), (1, 0))$ defines a commutative semiring [2,8,4], called the *entropy semiring.* It can be shown [4] that the function $\Phi : (\mathbb{R}_+, +, \cdot, 0, 1) \to (\mathbb{K}, \oplus, \otimes, (0, 0), (1, 0))$ defined by: $\forall x \in \mathbb{R}_+, \Phi(x) = (x, -x \log x)$, is a monoid morphism preserving closedness. Thus, the norm-Φ of an unambiguous probabilistic automaton can be computed efficiently using a single-source shortest-distance algorithm. Its second component is exactly the entropy of A, thus this provides an efficient and simple algorithm for computing the entropy of A.

(b) Norm L_α of a Probabilistic Automaton, $\alpha \in \mathbb{R}_+$.

The function $\Phi : (\mathbb{R}_+, +, \cdot, 0, 1) \to (\mathbb{R}_+, +, \cdot, 0, 1)$ defined by $\Phi(x) = x^\alpha$ is clearly a monoid morphism. Since for $0 \leq x < 1$, $0 \leq x^\alpha < 1$, it also preserves closedness. Thus, the L_α-norm of an unambiguous probabilistic automaton A can be computed efficiently using a shortest-distance algorithm. In particular, the Bhattacharya norm, i.e., $L_{\frac{1}{2}}$-norm, of A can be computed efficiently.

3.2 General Case

In general, a probabilistic automaton may not be unambiguous. But, the L_p norm can still be computed in polynomial time for any integer $p \geq 1$.

Theorem 2. *The L_p-norm of a probabilistic automaton A can be computed exactly in time $O(|A|^{3p})$ time and $\Theta(|A|^{2p})$ space.*

Proof. Let $A^{(p)}$ denote the automaton obtained by intersecting A with itself $p-1$ times. Then, by definition of intersection, $(s[A^{(p)}])^{1/p}$ represents the L_p norm of A. The cost of intersection to create $A^{(p)}$ is in $O(|A|^p)$. □

3.3 Approximate Computation

Here we consider the specific case of the computation of the L_p norm of a probabilistic automaton. Our results can be generalized to cover more general cases, in particular in the case of unambiguous automata.

Since for any $\epsilon > 0$, a probabilistic automaton is ϵ-k-closed for the probability semiring, instead of the (generalized) Floyd-Warshall algorithm, we can use a single-source shortest-distance algorithm to compute $s[A]$ [13]. This algorithm works with any queue discipline, its space complexity is linear which is significantly more efficient than the Floyd-Warshall algorithm.

The time complexity of the algorithm depends on the queue discipline used. With a breadth-first queue discipline (as in the Bellman-Ford shortest-distance algorithm), an analysis similar to [4] can be used to show that the overall complexity of this approximate algorithm is:

$$O(|Q| + (|E| + |Q|)\frac{\log(1/\epsilon)}{\log(1/|\lambda_2|)}). \tag{5}$$

For ϵ exponentially smaller than $|\lambda_2|$ ($\epsilon = |\lambda_2|^d$), the cost in complexity is only linear: $O(|Q| + d(|E| + |Q|))$. Other queue disciplines may lead to more efficient algorithms, depending on the probabilistic automaton considered.

4 Computation of Distances Between Probabilistic Automata

There are several standard distances used to compare distributions which can be used in particular to compare probabilistic automata. Here are the definitions of some of the most commonly used ones, the *relative entropy* or *Kullback-Leibler divergence*, and the L_p *distance* between two distributions q_1, q_2 over a discrete set $\mathcal{X}$:

$$\begin{aligned} D(q_1 \,\|\, q_2) &= \sum_{x \in \mathcal{X}} q_1(x) \log \frac{q_1(x)}{q_2(x)} \\ L_p(q_1, q_2) &= \Big(\sum_{x \in \mathcal{X}} (q_1(x) - q_2(x))^p \Big)^{1/p} \\ \text{Hellinger}(q_1, q_2) &= \Big(\sum_{x \in \mathcal{X}} (\sqrt{q_1(x)} - \sqrt{q_2(x)})^2 \Big)^{1/2}. \end{aligned} \tag{6}$$

Since we have previously specifically studied the problem of the computation of the relative entropy [4,3], in what follows, we will focus on the computation of the L_p distance and the Hellinger distance.

4.1 L_{2p} Distance of Probabilistic Automata

In [15], the authors give an approximate algorithm to compute the L_2 distance between two HMMs. Their algorithm applies to the specific cases of HMMs in which each state belongs to at most one cycle.[1] This section presents a simple and general algorithm for the computation of the L_{2p} distance of two arbitrary probabilistic automata, for $p \in \mathbb{N}$.

Our algorithm computes $(L_{2p}(A_1, A_2))^{2p}$. The L_{2p} distance between A_1, A_2 can then be obtained straightforwardly by taking the $2p$th root. $(L_{2p}(A_1, A_2))^{2p}$ can be rewritten as:

$$\begin{aligned} (L_{2p}(A_1, A_2))^{2p} &= \sum_{x \in \Sigma^*} |[\![A_1]\!](x) - [\![A_2]\!](x)|^{2p} = \sum_{x \in \Sigma^*} ([\![A_1]\!](x) - [\![A_2]\!](x))^{2p} \\ &= \sum_{x \in \Sigma^*} \sum_{i=0}^{2p} \binom{2p}{i} ([\![A_1]\!](x))^i (-[\![A_2]\!](x))^{2p-i} \qquad (7) \\ &= \sum_{i=0}^{2p} \binom{2p}{i} (-1)^i \sum_{x \in \Sigma^*} ([\![A_1]\!](x))^i ([\![A_2]\!](x))^{2p-i}. \qquad (8) \end{aligned}$$

Let $T(i, 2p-i)$ denote $\sum_{x \in \Sigma^*} ([\![A_1]\!](x))^i ([\![A_2]\!](x))^{2p-i}$. Note that if A_1, A_2 are acyclic, then one can compute $T(i, 2p-i)$ exactly using a generalization of the

[1] For more general HMMs, they claim without proof that an iterative version of their method yields an approximate algorithm that works in time $O((|A_1|+|A_2|)^{6p})$, where A_1 and A_2 are the HMMs considered. The approximation does not appear explicitly in this complexity term however.

single-source shortest-distance algorithm [13] that works for arbitrary semirings, in linear time $O(|A_1| + |A_2|)$.

Next, let us consider the case of unambiguous automata A_1, A_2. If $A_i = (\Sigma, Q_i, I_i, F_i, E_i, \lambda_i, \rho_i)$, $i = 1, 2$, then the transitions in the intersection automaton $A = A_1 \cap A_2$ are defined according to the following rule:

$$(q_1, a, w_1, q_1') \in E_1 \text{ and } (q_2, a, w_2, q_2') \in E_2 \Rightarrow ((q_1, q_2), a, w_1 w_2, (q_1', q_2')) \in E.$$

Since we are dealing with unambiguous automata, we can avoid the re-computation of the intersection automaton for different is. During intersection, instead of multiplying w_1 and w_2, we can keep instead the pair (w_1, w_2). Then, we only need to intersect A_1 and A_2 once, and modify the weight of each transition in the intersection automaton for different is in the computation of $T(i, 2p - i)$ as $((q_1, q_2), a, (w_1^i (w_2)^{2p-i}), (q_1', q_2'))$. Running the shortest-distance algorithm over the intersection automaton with weights modified as described above yields $T(i, 2p - i)$. Computing the intersection automaton takes $O(|A_1||A_2|)$ time.

Thus, if we use the exact algorithm to compute the shortest-distance, then for each i, computing $T(i, 2p - i)$ costs $O(|A_1 \cap A_2|^3)$ time and $\Theta(|A_1 \cap A_2|^2)$. Therefore, the time complexity of computing the $2p$-distance between A_1, A_2 is $O((2p)|A_1 \cap A_2|^3)$ and the space complexity $\Theta(|A_1 \cap A_2|^2)$.

Theorem 3. *The L_{2p} distance of unambiguous probabilistic automata can be computed exactly in time $O(2p|A_1|^3|A_2|^3)$.*

Note that this theorem significantly improves the result of [15], which is exponential in p. Thus, for unambiguous automata, our algorithms are, to the best of our knowledge, the only polynomial time algorithms for computing the $2p$ norm exactly.

For the computation of the L_{2p}-distance of arbitrary automata, we can no longer intersect A_1, A_2 just once. Since there may be multiple paths in $A_i, i = 1, 2$ with the same label, cross terms appear in $T(i, 2p-i)$. This makes it necessary to perform $2p$ separate intersections for each i. The computational cost and space complexity of intersection to compute $T(i, 2p - i)$ is in $O(|A_1|^i|A_2|^{2p-i})$. Thus, the exact shortest-distance algorithm has complexity $O((|A_1|^i|A_2|^{2p-i})^3)$. This leads us to the following result.

Theorem 4. *The L_{2p} distance of two arbitrary probabilistic automata A_1 and A_2 can be computed in time $\sum_{i=0}^{2p} O((|A_1|^i|A_2|^{2p-i})^3) = O((|A_1| + |A_2|)^{6p})$.*

Note that our algorithm for computing the L_{2p} distance of two arbitrary probabilistic automata A_1 and A_2 clearly also provides an efficient method for testing their equivalence since A_1 and A_2 are equivalent iff their L_p distance is null. For $p = 1$, our exact algorithm can be used to test for equivalence in time $O((|A_1||A_2|)^3)$. However, the standardization algorithm of Schützenberger [17] can be used to derive a more efficient algorithm.

Theorem 5. *The equivalence of two arbitrary probabilistic automata A_1 and A_2 can be computed in time $O(|\Sigma|\,(|A_1| + |A_2|)^3)$.*

Proof. The standardization algorithm of Schützenberger [17,1] applies to any weighted automaton defined over a field. It leads to a representation of a weighted automaton with the smallest number of states. The algorithm requires the construction of bases for vectorial spaces for which spanning sets are known. Using LUP decompositions, the complexity of the standardization algorithm applied to a weighted automaton A is in $O(|\Sigma||A|^3)$.

For the purpose of equivalence, we may view a probabilistic automaton as an automaton over the field $(\mathbb{R}, +, \cdot, 0, 1)$. Since negation is allowed over this field, we can construct the automaton $A = A_1 - A_2$, which can be done in linear time, and apply standardization. A_1 and A_2 are equivalent iff A is equivalent to the null weighted machine, that is iff after standardization A has no state. Thus, this leads to an algorithm for testing the equivalence of two probabilistic automata A_1 and A_2 with overall complexity $O(|\Sigma|\,|A|^3) = O(|\Sigma|\,(|A_1| + |A_2|)^3)$. □

To our knowledge, this is the most efficient algorithm for testing the equivalence of probabilistic automata. The best algorithm previously reported in the literature was that of Wen-Guey Tzeng whose complexity is $O(|\Sigma|\,(|A_1| + |A_2|)^4))$ [19]. The alphabet factor does not appear in the expression of the complexity reported by the author most likely because the proof is restricted to a binary alphabet. The technique described by Wen-Guey Tzeng is in fact closely related to the standardization algorithm of Schützenberger [17], which the author was apparently not aware of.

4.2 L_{2p+1} and L_∞ Distance of Probabilistic Automata

It was shown by [15] that the problem of computing the L_1 or L_∞ distance of two probabilistic automata is NP-hard, even for acyclic automata. Here, we extend these results to the case of arbitrary L_{2p+1} distances, where $p \in \mathbb{N}$.

Our proof of the hardness of computing the L_{2p+1} distance between two acyclic probabilistic automata is by reduction from the Max-clique problem and is based on a technique used by [15].

Given a graph $G = (V, E)$, one can construct an acyclic weighted automaton A_G over the probability semiring of size polynomial in $|V| + |E|$ such that $[\![A]\!](x) = k$ for some string x iff G has a clique of size k. A_G is constructed as follows. It has a single initial state q_s and a single final state q_t. For each $i \in V$, it admits the following transitions:

(a) a transition from q_s to $q_{i,0}$ with label ϵ and weight 1;
(b) a transition from $q_{i,n}$ to the final state q_t with label ϵ and weight 1;
(c) a transition from $q_{i,i-1}$ to $q_{i,i}$ with label i and weight 1;
(d) a transition from $q_{i,j-1}$ to $q_{i,j}$ with label ϵ and weight 1 for each $j \neq i$; and
(e) if $(i, j) \in E$, a transition from $q_{i,j-1}$ to $q_{i,j}$ with label j and weight 1.

The size of A_G is clearly polynomial in $|V| + |E|$. Given a set $S \subseteq V$, let $[S]$ denote the ordered tuple with elements of S. For example, if $S = \{1, 2, 5, 3\}$, then $[S] = (1, 2, 3, 5)$. By construction, for any clique S, A_G contains a distinct path labeled with $[S]$ starting at the initial state and going through $q_{i,0}$ for each $i \in S$. Since all accepting paths have the same weight 1, this proves the property that $[\![A]\!](x) = k$ for some string x iff G has a clique of size k.

The automaton A_G is not probabilistic. But, an equivalent probabilistic automaton without ϵ-transitions can be computed from A_G by using the weighted ϵ-removal algorithm [12], and a weight-pushing algorithm can be used to normalize the sum of its weights to one [11]. For the sake of the simplicity of the presentation, we will continue to work with A_G. Our results can be generalized to the case of a probabilistic automaton without ϵ-transitions without difficulty.

Theorem 6. *The problem of computing the L_{2p+1} distance of two probabilistic automata is NP-hard.*

Proof. Using the notation used in [15], let a_k denote the number of strings accepted by A_G with weight exactly k. a_k is in fact exactly the number of cliques of size k in the original graph G. Thus determining the maximum k such that $a_k \neq 0$ is equivalent to determining the size of the largest clique.

For each $i \in \{0, 1, \ldots, n\}$, let C_i denote the constant weighted automaton assigning the same weight i to all subsequences of $\{1, \ldots, n\}$ and weight 0 to all other strings. By definition of the L_{2p+1} norm,

$$\forall i \geq 0, \quad [L_{2p+1}(C_i, A_G)]^{2p+1} = \sum_{j=0}^{n} a_j |i-j|^{2p+1} \tag{9}$$

This defines a system of linear equation with unknown variables a_j, $j = 0, \ldots, n$. Let $M \in \mathbb{R}^{(n+1)\times(n+1)}$ be the matrix defined by $M_{i,j} = |i-j|^{2p+1}, i \in \{0, 1, \ldots, n\}$. If M is invertible, then all a_js can be defined with respect the L_{2p+1} distance of the automata C_i and A_G, which will prove the statement of the theorem.

This matrix is a specific Toeplitz matrix, but it is not straightforward to compute its determinant [15]. Instead, we can do our reasoning in $\mathbb{Z}_3$. Indeed, in $\mathbb{Z}_3$, the coefficients of M are either 0, 1, or -1, regardless of the value of p. The determinant of M in $\mathbb{Z}_3$ is given by:

$$\det(M) = \begin{cases} -1 & \text{if } n+1 = 2 \mod 3 \\ 1 & \text{if } n+1 = 0 \mod 3 \\ 0 & \text{if } n+1 = 1 \mod 3. \end{cases} \tag{10}$$

We delay the proof of this fact to Lemma 1.

This implies that for all $n \in \mathbb{N}$ such that n is of the form $n \equiv \pm 1 \mod 3$, the matrix M of size $(n+1) \times (n+1)$ defined by $M_{i,j} = |i-j|^{2p+1}, i \in \{0, 1, \ldots, n\}$ is invertible in $\mathbb{R}$. Therefore, for $n \equiv \pm 1 \mod 3$, one can compute the matrix A and determine the size of the largest clique in the original graph G. This leaves us only with the case where $n \equiv 0 \mod 3$ in the original graph $G = (V, E)$. But, in this case, one can add a *dummy vertex* to G that is connected to all other vertices of V. Doing so increases the size of the largest clique by exactly one, and yields a graph $G' = (V', E')$ with $|V'| \equiv 1 \mod 3$. Since the size of the largest clique in G is one less than the size of the largest clique in G', the reduction is complete. Thus, the problem of determining the computing $2p+1$ distance between two probabilistic automata is NP-hard. □

We conjecture that the problem of computing the L_{2p+1} distance, or L_∞, is in fact undecidable. Note that it was shown by [15] that, in view of the hardness of approximation results for cliques of [18,9], even a polynomial approximation of the L_∞ distance within a factor of $n^{\frac{1}{4}-\epsilon}$ is impossible unless NP = P.

Lemma 1. *The determinant of M in $\mathbb{Z}_3$ is given by*

$$\det(M) = \begin{cases} -1 & \textit{if } n+1 = 2 \mod 3 \\ 1 & \textit{if } n+1 = 0 \mod 3 \\ 0 & \textit{if } n+1 = 1 \mod 3. \end{cases} \tag{11}$$

Proof. Let $M[n] \in \mathbb{R}^{n\times n}$ be the matrix defined by $M_{i,j} = |i-j|^{2p+1} \mod 3$. Note that in $\mathbb{Z}_3$, $|i-j|^{2p+1} \mod 3 = |i-j|$ for all $p \in \mathbb{N}$. Let R_i, C_j denote the ith row and the jth column of M respectively. Consider n such that $n+1 = 1 \mod 3$, i.e. let $n+1 = 3k+1$ for some $k \in \mathbb{N}$. For all $j \in \{1,\ldots,3k+1\}$,

$$M_{3k+1,j} = |3k+1-j|^{2p+1} \mod 3 = (1-j)^{2p+1} \mod 3 \tag{12}$$

$$= -|1-j|^{2p+1} \mod 3 = -M_{1,j} \tag{13}$$

Since the last row is a scalar multiple of the first row, $\det(M) = 0$ for $n+1 = 1 \mod 3$.

Next, suppose $n+1 = 2 \mod 3$. Let $n+1 = 3k+2$ for some $k \in \mathbb{N}$. In this case, we show that $\det(M[3k+2]) = -\det(M[3k])$. Given $M[3k+2]$, we perform the following row and column operations:

$$R_1 \leftarrow R_1 + R_{3k+1} \qquad C_1 \leftarrow C_1 + C_{3k+2}. \tag{14}$$

The resulting matrix has zeros everywhere in the first row and column except at $M_{1,1} = 1, M_{1,3k+2} = M_{3k+2,1} = -1$. Developing the determinant of M along R_1, it is not hard to see that $\det(M[3k+2]) = \det(M[3k+1]) - \det(M[3k])$. Since $\det(M[3k+1]) = 0$, we obtain

$$\det(M[3k+2]) = -\det(M[3k]). \tag{15}$$

Finally, suppose $n+1 = 0 \mod 3$. Let $n+1 = 3k$ for $k \in \mathbb{N}$. We show that $\det(D[3k]) = \det(D[3k-4]) - \det(D[3(k-1)])$. Given $M[3k]$, we perform the following operations

$$\begin{array}{llll} R_1 & \leftarrow R_1 + R_{3k-2} & C_1 & \leftarrow C_1 + C_{3k-2} \\ R_{3k} & \leftarrow R_{3k} + R_3 & C_{3k} & \leftarrow C_{3k} + C_3 \\ R_2 & \leftarrow R_2 + R_1 & C_2 & \leftarrow C_2 + C_1 \\ R_{3k-1} & \leftarrow R_{3k} + R_{3k-1} & C_{3k-1} & \leftarrow C_{3k} + C_{3k-1} \\ R_2 & \leftarrow R_2 + R_{3k-1} & C_2 & \leftarrow C_2 + C_{3k-1} \end{array} \tag{16}$$

The resulting matrix has zeros everywhere in the first and last row and column, except for $M_{1,3k} = 1, M_{3k,1} = 1$. Furthermore, the submatrix defined by

R_i, C_j for $i, j \in \{2, \ldots 3k-1\}$ is the same as $M[3k-2]$ except for $M_{2,2} = 1$ and $M_{2,3k-1} = M_{3k-1,2} = -1$. Developing the determinant along the following sequence of rows and columns: R_1, C_1, R_2, C_2 yields

$$\det(M[3k]) = \det(M[3k-4]) - \det(M[3(k-1)]), \tag{17}$$

and ends the proof. □

4.3 Hellinger Distance of Probabilistic Automata

The ideas presented in the previous section can be used in a straightforward manner to compute the Hellinger distance of two unambiguous probabilistic automata. The Hellinger distance Hellinger(A_1, A_2) of two probabilistic automata A_1, A_2 is given by:

$$\text{Hellinger}(A_1, A_2) = \Big(\sum_{x \in \Sigma^*} (\sqrt{[\![A_1]\!](x)} - \sqrt{[\![A_2]\!](x)})^2 \Big)^{1/2}. \tag{18}$$

Thus,

$$\begin{aligned}
[\text{Hellinger}(A_1, A_2)]^2 &= \sum_{x \in \Sigma^*} (\sqrt{[\![A_1]\!](x)} - \sqrt{[\![A_2]\!](x)})^2 && (19)\\
&= \sum_{x \in \Sigma^*} [\![A_1]\!](x) + \sum_{x \in \Sigma^*} [\![A_2]\!](x) - 2 \sum_{x \in \Sigma^*} \sqrt{[\![A_1]\!](x)[\![A_2]\!](x)} \\
&= 2(1 - \sum_{x \in \Sigma^*} \sqrt{[\![A_1]\!](x)[\![A_2]\!](x)}) && (20)
\end{aligned}$$

The problem of computing the Hellinger distance between A_1, A_2 therefore reduces to efficiently computing $\sum_{x \in \Sigma^*} \sqrt{[\![A_1]\!](x)[\![A_2]\!](x)}$. Once again, as long as A_1 and A_2 are unambiguous there is at most one accepting string with label x in $A_1 \cap A_2$. Intersecting A_1 and A_2 over the probability semiring, the weight of the transition corresponding to the intersection of the transitions $e_1 = (q_1, a, w_1, q_1')$ and $e_2 = (q_2, a, w_2, q_2')$ is given by $w_1 w_2$.

The function $\Phi : (\mathbb{R}_+, +, \cdot, 0, 1) \rightarrow (\mathbb{R}_+, +, \cdot, 0, 1)$ defined by $\Phi(x) = \sqrt{x}$ is clearly a monoid morphism. Since $0 \leq x < 1$, $0 \leq \sqrt{x} < 1$, it also preserves closedness. Since the Φ norm of the intersection automaton is precisely the quantity we are interested in, we obtain an efficient algorithm to compute the Hellinger distance. The complexity of this computation is the same as the complexity of the shortest distance algorithm on the intersection automaton $A_1 \cap A_2$. If A_1, A_2 are acyclic, then the shortest-distance computation can be done in linear time, i.e. $O(|A_1 \cap A_2|)$. For A_1, A_2 unambiguous, one could compute the Hellinger distance exactly in time that is cubic in the size of the intersection automaton and space that is quadratic using a generalization of the classical Floyd-Warshall all-pairs shortest-distance algorithm that works for arbitrary closed semirings. However, a more efficient approximate solution can be obtained using the general single-source shortest-distance algorithm [13] that uses only linear space.

5 Conclusion

We examined the problem of the computation of several standard distances between probabilistic automata. We showed that in each case, the problem can be viewed as a shortest-distance computation over an appropriate semiring. In each case, we either gave an efficient algorithm for the computation of the norm of a probabilistic automaton or the distance between two probabilistic automata, or showed the intractability of the problem.

Our algorithms can be used to compute distances between very large probabilistic automata. Some of our results could perhaps be extended to the case of finitely ambiguous probabilistic automata. Many of our results can be straightforwardly extended to the case of weighted tree automata.

References

1. Jean Berstel and Christophe Reutenauer. *Rational Series and Their Languages.* Springer-Verlag: Berlin-New York, 1988.
2. Stephen Bloom and Zoltan Ésik. *Iteration Theories.* Springer-Verlag, Berlin, 1991.
3. Corinna Cortes, Mehryar Mohri, Ashish Rastogi, and Michael Riley. Distances between Probabilistic Automata. In preparation, journal version, 2006.
4. Corinna Cortes, Mehryar Mohri, Ashish Rastogi, and Michael Riley. Efficient Computation of the Relative Entropy of Probabilistic Automata. In *Proceedings of LATIN 2006*, volume 3887 of *LNCS*, pages 323–336. Springer-Verlag, 2006.
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. R. Durbin, S.R. Eddy, A. Krogh, and G.J. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, Cambridge UK, 1998.
7. Samuel Eilenberg. *Automata, Languages and Machines*, volume A–B. Academic Press, 1974–1976.
8. Jason Eisner. Expectation Semirings: Flexible EM for Finite-State Transducers. In *Proceedings of the ESSLLI Workshop on Finite-State Methods in NLP*, 2001.
9. Lars Engebretsen and Jonas Holmerin. Clique is hard to approximate within $n^{1-o(1)}$. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP 2000)*, pages 2–12, London, UK, 2000. Springer-Verlag.
10. Werner Kuich and Arto Salomaa. *Semirings, Automata, Languages.* Number 5 in EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, Germany, 1986.
11. Mehryar Mohri. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics*, 23(2), 1997.
12. Mehryar Mohri. Generic Epsilon-Removal and Input Epsilon-Normalization Algorithms for Weighted Transducers. *International Journal of Foundations of Computer Science*, 13(1):129–143, 2002.
13. Mehryar Mohri. Semiring Frameworks and Algorithms for Shortest-Distance Problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.
14. Azaria Paz. *Introduction to probabilistic automata.* Academic Press, New York, 1971.

15. Rune B. Lyngsø and Christian N. S. Pederson. The Consensus String Problem and the Complexity of Comparing Hidden Markov Models. *Journal of Computer and System Sciences*, 65(3):545–569, 2002.
16. Arto Salomaa and Matti Soittola. *Automata-Theoretic Aspects of Formal Power Series.* Springer-Verlag, 1978.
17. Marcel-Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4, 1961.
18. J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *FOCS '96: Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, page 627, Washington, DC, USA, 1996. IEEE Computer Society.
19. Wen-Guey Tzeng. A Polynomial-Time Algorithm for the Equivalence of Probabilistic Automata. *Foundations of Computer Science (FOCS)*, pages 216–227, 1992.