International Journal of Foundations of Computer Science

L_p Distance and Equivalence of Probabilistic Automata

Corinna Cortes
Google Research,
76 Ninth Avenue, New York, NY 10011.
corinna@google.com

Mehryar Mohri
Courant Institute of Mathematical Sciences and Google Research,
251 Mercer Street, New York, NY 10012.
mohri@cims.nyu.edu

Ashish Rastogi
Courant Institute of Mathematical Sciences,
251 Mercer Street, New York, NY 10012.
rastogi@cs.nyu.edu

This paper presents an exhaustive analysis of the problem of computing the L_p distance of two probabilistic automata. It gives efficient exact and approximate algorithms for computing these distances for p even and proves the problem to be NP-hard for all odd values of p, thereby completing previously known hardness results. It further proves the hardness of approximating the L_p distance of two probabilistic automata for odd values of p. Similar techniques to those used for computing the L_p distance also yield efficient algorithms for computing the Hellinger distance of two unambiguous probabilistic automata both exactly and approximately.

A problem closely related to the computation of a distance between probabilistic automata is that of testing their equivalence. This paper also describes an efficient algorithm for testing the equivalence of two arbitrary probabilistic automata A_1 and A_2 in time $O(|\Sigma|\,(|A_1| + |A_2|)^3)$, a significant improvement over the previously best reported algorithm for this problem.

1. Introduction

A probabilistic automaton is a finite automaton with transition probabilities which represents a distribution over the set of all strings defined over a finite alphabet. Probabilistic automata have been extensively studied in a variety of areas of computer science. They are used in a variety of applications, including text and speech processing [14], image processing [8], and computational biology [9].

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 standard distances between probabilistic automata.

In a companion paper [6], we give an exhaustive study of the problem of computing the relative entropy, or Kullback-Leibler divergence, of two probabilistic automata. In particular, we present an efficient algorithm for computing the relative entropy of two unambiguous probabilistic automata [5] and show that the general case is (at least) PSPACE-complete.

Here, we present a full analysis of the problem of computing the L_p distance of two probabilistic automata, extending our previous results reported in [4]. 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 [19]. These latter results complete those given by [19] who showed the problem to be NP-hard for L_1 and L_∞. We further show the hardness of approximating the L_p distance of two probabilistic automata for odd values of p.

Similar techniques to those used for computing the L_p distance can be used to compute other distances. As an example, we give efficient algorithms for computing the Hellinger distance of two unambiguous probabilistic automata both exactly and approximately.

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 zero. However, we will describe an even more efficient algorithm based on Schützenberger's standardization technique [21, 2] with a running-time complexity of $O(|\Sigma| (|A_1| + |A_2|)^3)$. This is a significant improvement over the previously best algorithm reported for this problem whose complexity is $O(|\Sigma| (|A_1| + |A_2|)^4))$ [24].

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 gives the definition of some standard distances used between distributions and some of the main inequalities relating them. Section 4 presents an exhaustive analysis of the problem of computing the L_p distance of probabilistic automata, including efficient algorithms for computing the L_{2p} distance (Section 4.1), the proof that the computation of the L_{2p+1} distance is NP-hard (Section 4.2), a hardness of approximation result (Section 4.3), and results related to the computation of the absolute value of the difference of two probabilistic automata (Section 4.4). The problem of the computation of the Hellinger distance of probabilistic automata is examined in detail in Section 5. Finally, Section 6 describes an efficient algorithm for testing the equivalence of two probabilistic automata.

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 ([13]) *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* $\mathbb{K}$,

$$(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c) \quad \text{and} \quad c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b).$$

- $\overline{0}$ *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 [16]. $\mathbb{K}$ 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 [16]. 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 ([10, 20, 2]) *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* $\mathbb{K}$.

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

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 $(\mathbb{R}_+, +, \times, 0, 1)$ is said to be* probabilistic *if for any state $q \in Q$, $\bigoplus_{\pi \in P(q,q)} w[\pi]$, the sum of the weights of all cycles at q, is well-defined and in $\mathbb{R}_+$ and*

$$\sum_{x \in \Sigma^*} [\![A]\!](x) = 1. \tag{2}$$

A probabilistic automaton A is said to be stochastic *if at each state the weights of the outgoing transitions and the final weight sum to one.*

Observe that our definition of probabilistic automata differs from that of [18] and [17]. *Probabilistic automata* as defined by these authors are weighted automata over $(\mathbb{R}_+, +, \times, 0, 1)$ such that at any state q and for any label $a \in \Sigma$, the weights of the outgoing transitions of q labeled with a sum to one. More generally, with that definition, the weights of the paths leaving state q and labeled with $x \in \Sigma^*$ sums to one. Such automata define a conditional probability distribution $\Pr[q' \mid q, x]$ over all states q' that can be reached from q by reading x.

Instead, with our definition, probabilistic automata represent distributions over Σ^*, $\Pr[x], x \in \Sigma^*$. These are the natural distributions that arise in many applications. They are inferred from large data sets using statistical learning techniques. We are interested in computing various distances between two such distributions over strings.

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.

The computation of single-source shortest-distances is needed in many of the algorithms presented in the following sections. 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 [16].

3. Distances between Distributions

There are many standard distances or discrepancies used to compare distributions which can also serve to compare probabilistic automata. Some of the most commonly used ones are: the *relative entropy* or *Kullback-Leibler divergence* D,[a] the L_p *distance*, the *Hellinger distance*, the *Jensen-Shannon distance* JS, the χ^2*-distance*, and the *triangle distance* Δ between two distributions q_1 and q_2 defined over a discrete set $\mathcal{X}$:

$$\begin{aligned}
D(q_1\|q_2) &= \sum_{x\in\mathcal{X}} q_1(x)\log\frac{q_1(x)}{q_2(x)}\\
L_p(q_1,q_2) &= \Big(\sum_{x\in\mathcal{X}} |q_1(x)-q_2(x)|^p\Big)^{1/p}\\
L_\infty(q_1,q_2) &= \max_{x\in\mathcal{X}} |q_1(x)-q_2(x)|\\
\text{Hellinger}(q_1,q_2) &= \Big(\sum_{x\in\mathcal{X}} \Big(\sqrt{q_1(x)}-\sqrt{q_2(x)}\Big)^2\Big)^{1/2}\\
\text{JS}(q_1,q_2) &= \sum_{x\in\mathcal{X}} \left(q_1(x)\log\frac{2q_1(x)}{q_1(x)+q_2(x)} + q_2(x)\log\frac{2q_2(x)}{q_1(x)+q_2(x)}\right)\\
\chi^2(q_1,q_2) &= \sum_{x\in\mathcal{X}} \frac{(q_1(x)-q_2(x))^2}{q_2(x)}\\
\Delta(q_1,q_2) &= \sum_{x\in\mathcal{X}} \frac{(q_1(x)-q_2(x))^2}{q_2(x)+q_2(x)}.
\end{aligned} \tag{3}$$

Several general inequalities relate these distances [23, 7] including the following ones (the last one holds when the set $\mathcal{X}$ is finite and of size n):

$$\begin{aligned}
&[L_1(q_1,q_2)]^2/2 \le D(q_1\|q_2)\\
&\text{Hellinger}(q_1,q_2)/2 \le \Delta(q_1,q_2)/2 \le \text{JS}(q_1,q_2)\\
&\text{JS}(q_1,q_2) \le \log(2)\Delta(q_1,q_2) \le 2\log(2)\text{Hellinger}(q_1,q_2)\\
&\frac{L_2(q_1,q_2)}{L_\infty(q_1)+L_\infty(q_2)} \le \Delta(q_1,q_2) \le L_1(q_1,q_2) \le \sqrt{n}\,L_2(q_1,q_2).
\end{aligned} \tag{4}$$

The problem of computing the relative entropy D of two probabilistic automata is examined in a previous publication [5] and exhaustively treated in a companion paper [6]. The following sections present a study of the computation of the L_p distances and the Hellinger distance of two probabilistic automata. Several of our results can be generalized straightforwardly to other distances using similar ideas.

4. L_p Distance of Probabilistic Automata

This section presents an exhaustive analysis of the problem of computing the L_p distance of 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

[a]The relative entropy is not symmetric and does not satisfy the triangle inequality.

odd values of p. These latter results complete those given by [19] who showed the problem to be NP-hard for L_1 and L_∞.

4.1. L_{2p} Distance of Probabilistic Automata

In [19], the authors give an approximate algorithm to compute the L_2 distance between two HMMs A_1 and A_2. Their algorithm applies to the specific cases of HMMs in which each state belongs to at most one cycle.[b] 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:

$$\begin{aligned}(L_{2p}(A_1, A_2))^{2p} &= \sum_{x \in \Sigma^*} |[\![A_1]\!](x) - [\![A_2]\!](x)|^{2p} = \sum_{x \in \Sigma^*} ([\![A_1]\!](x) - [\![A_2]\!](x))^{2p} \\ &= \sum_{x \in \Sigma^*} \sum_{i=0}^{2p} \binom{2p}{i} ([\![A_1]\!](x))^i (-[\![A_2]\!](x))^{2p-i} \qquad (5) \\ &= \sum_{i=0}^{2p} \binom{2p}{i} (-1)^i \sum_{x \in \Sigma^*} ([\![A_1]\!](x))^i ([\![A_2]\!](x))^{2p-i}. \qquad (6)\end{aligned}$$

In the first line, we could remove the absolute values since the exponent is even. This is crucial and is the reason why we need to treat the case of the L_{2p+1} distance separately.

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 single-source shortest-distance algorithm [16] 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 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.

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

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 5. *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 [19], 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 L_{2p} distance exactly.

For the computation of the L_{2p}-distance of arbitrary automata, we can no longer intersect A_1 and A_2 just once. Since there may be multiple paths in $A_i, i = 1, 2$ with the same label, cross terms appear in $T(i, 2p-i)$. For example if w_1 and w_2 are the weights of two paths in A_1 with labels x and the path with weight w' is the (only) path in A_2 with label x, then the contribution of string x to $T(i, 2p-i)$ is $(w_1 + w_2)^i (w')^{2p-i}$, leading to cross terms of the type $\binom{i}{j} w_1^j w_2^{i-j} w'^{2p-i}$, $j \leq i$. This makes it necessary to perform separate intersections for each i, hence a total of $2p$ intersections. 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 6. *The L_{2p} distance of two arbitrary probabilistic automata A_1 and A_2 can be computed in time $\sum_{i=0}^{2p} O((|A_1|^i |A_2|^{2p-i})^3) = O((|A_1| + |A_2|)^{6p})$.*

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

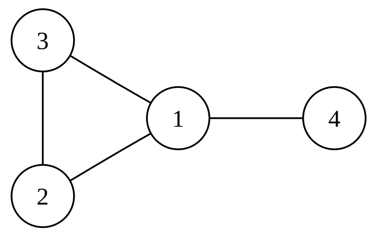
The problem of computing the L_1 or L_∞ distance of two probabilistic automata was shown to be NP-hard by [19], even for acyclic automata. Here, we extend these results to the case of arbitrary L_{2p+1} distances, where $p \in \mathbb{N}$.

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 [19].

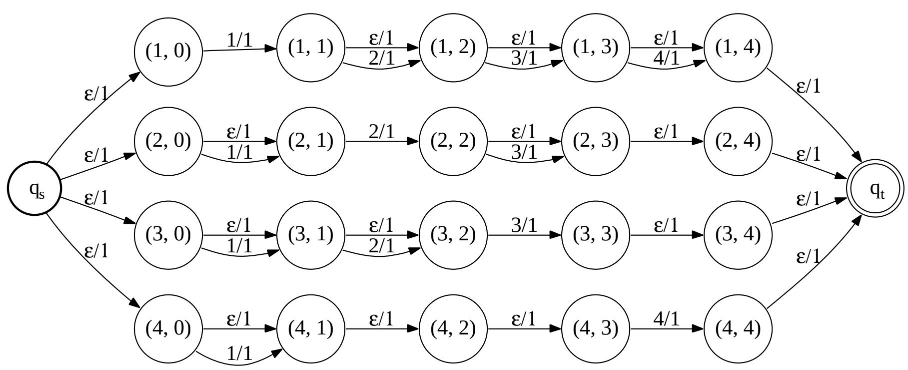
Given a graph $G = (V, E)$, one can construct an acyclic weighted automaton A_G over the probability semiring of size polynomial in $|V| + |E|$ such that $[\![A_G]\!](x) = k$ for some string x iff G has a clique of size k.

Let $n = |V|$. 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.



(a)



(b)

Fig. 1. (a) Undirected graph $G = (V, E)$. (b) The corresponding automaton A_G constructed in the reduction.

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 $S = \{3, 1, 2\}$, then $[S] = (1, 2, 3)$. 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 $i \in S$ (see Fig. 1 for an example with $[S] = (1, 2, 3)$.) 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 [15], and a weight-pushing algorithm can be used to normalize the sum of its weights to one [14]. We first establish the result with A_G and later describe how to convert A_G into a probabilistic automaton.

Theorem 7. *The problem of computing the L_{2p+1} distance of two probabilistic automata is NP-hard.*

Proof. The proof is based on a reduction using an algorithm for the computation

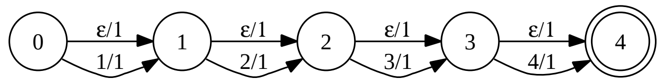


Fig. 2. The constant automaton C_4 for G assigning the weight 4 to all subsequences of the set $\{1, \ldots, 4\}$. Note that the final state has a final weight of 4.

of the L_{2p+1} distance as a subroutine to define an algorithm for solving the Max-Clique problem. Using the notation adopted by [19], let a_k denote the number of strings accepted by A_G with weight exactly k. 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 ordered subsequences of $\{1, \ldots, n\}$ and weight 0 to all other strings. Fig. 2 shows the constant automaton for $n = 4$. By definition of the L_{2p+1} distance,

$$\forall i \geq 0, \quad [L_{2p+1}(C_i, A_G)]^{2p+1} = \sum_{x \in \Sigma^*} |[\![A_G]\!](x) - [\![C_i]\!](x)|^{2p+1} \tag{8}$$

$$= \sum_{x \in \Sigma^*} |[\![A_G]\!](x) - i|^{2p+1} \tag{9}$$

$$= \sum_{j=0}^{n} a_j |i - j|^{2p+1} \tag{10}$$

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 $M_{i,j} = |i - j|^{2p+1}, i \in \{0, 1, \ldots, n\}$ and let $A \in \mathbb{R}^{n+1}$ be the column vector containing the a_js. If M is invertible, then A 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 [19]. Instead, we can do our reasoning in $\mathbb{Z}_3$. 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:

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

We delay the proof of this fact to Lemma 8.

This implies that for all $n \in \mathbb{N}$ such that n is of the form $n \equiv \pm 1 \pmod 3$, 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 $n \equiv \pm 1 \pmod 3$, one can compute the column vector A and determine the size of the largest clique in the original graph G. This leaves us only with the case where $n \equiv 0 \pmod 3$ 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 $|V'| \equiv 1 \pmod 3$. 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 [19] that, in view of the hardness of approximation results for cliques of [22, 11], even a polynomial approximation of the L_∞ distance within a factor of $n^{\frac{1}{4}-\epsilon}$ is impossible unless NP = P.

Lemma 8. *The determinant of M in $\mathbb{Z}_3$ is given by*

$$\det(M) = \begin{cases} -1 & \text{if } n+1 \equiv 2 \pmod 3 \\ 1 & \text{if } n+1 \equiv 0 \pmod 3 \\ 0 & \text{if } n+1 \equiv 1 \pmod 3. \end{cases} \tag{12}$$

Proof. Let $M[n+1] \in \mathbb{R}^{(n+1)\times(n+1)}$ be the matrix defined by $M_{i,j} \equiv |i-j| \pmod 3$. Note that $|i-j|^{2p+1} \pmod 3 \equiv |i-j| \pmod 3$ for all $p \in \mathbb{N}$. To remain consistent with the previous description, throughout this proof, we consider the matrix M of size $(n+1)\times(n+1)$.

Let R_i, C_j denote the ith row and the jth column of M respectively. We prove the lemma by showing that the following three identities in $\mathbb{Z}_3$ hold for all $k \in \mathbb{N}$, $k \geq 2$:

$$\begin{aligned} &\det(M[3k+1]) = 0 \\ &\det(M[3k+2]) = -\det(M[3k]) \\ &\det(M[3k]) \quad\;\, = \det(M[3k-4]) - \det(M[3k-3]). \end{aligned} \tag{13}$$

Case 1. $n+1 \equiv 1 \pmod 3$. Let $n+1 = 3k+1$ for some $k \in \mathbb{N}$. For all $j \in \{1, \ldots, 3k+1\}$,

$$M_{3k+1,j} \equiv |3k+1-j| \pmod 3 \equiv (1-j) \pmod 3 \equiv -|1-j| \pmod 3 = -M_{1,j}.$$

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

Case 2. $n+1 \equiv 2 \pmod 3$. 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 symmetric row and column operations:

$$R_1 \leftarrow R_1 + R_{3k+1} \qquad C_1 \leftarrow C_1 + C_{3k+1}. \tag{15}$$

Note that in Case 1, we observed that R_{3k+1} was the negation of R_1. Thus the above row operation will set all but the last entry in the first row (and by symmetry, in the first column) to zero. Let M' denote the resulting matrix. Then, $M'_{1,i} = M'_{i,1} = 0$

for $1 \leq i \leq 3k+1$ and $M'_{1,3k+2} = M'_{3k+2,1} = -1$. The entries in rows and columns 2 through $3k+1$ are unaffected. Let S be the submatrix of M' induced by rows $\{2, \ldots, 3k+2\}$ and columns $\{1, \ldots, 3k+1\}$. Fig. 3(a) illustrates the structure of the matrix M'. Developing the determinant of M' along R_1 and simplifying the powers of -1 yield:

$$\det(M) = \det(M') = (-1)^{(3k+2)+1} [(-1) \det(S)] = (-1)^{3k} \det(S). \tag{16}$$

Developing the determinant of S along the first column leads to:

$$\det(S) = (-1)^{1+(3k+1)} [(-1) \det(M[3k])] = (-1)^{3(k+1)} \det(M[3k]). \tag{17}$$

Plugging in the expression of $\det(S)$ (Eqn. 17) into Eqn. 16 leads to:

$$\det(M) = (-1)^{3(2k+1)} \det(M[3k]) = -\det(M[3k]). \tag{18}$$

Case 3. $\boldsymbol{n+1 \equiv 0 \pmod 3}$**.** Let $n+1 = 3k$ for $k \in \mathbb{N}$. We show that $\det(M[3k]) = \det(M[3k-4]) - \det(M[3k-3])$. Given $M[3k]$, we perform the following symmetric operations:

$$\begin{array}{llll} R_1 & \leftarrow R_1 + R_{3k-2} & C_1 & \leftarrow C_1 + C_{3k-2} \\ R_{3k} & \leftarrow R_{3k} + R_3 & C_{3k} & \leftarrow C_{3k} + C_3 \\ R_2 & \leftarrow R_2 + R_1 & C_2 & \leftarrow C_2 + C_1 \\ R_{3k-1} & \leftarrow R_{3k} + R_{3k-1} & C_{3k-1} & \leftarrow C_{3k} + C_{3k-1} \\ R_2 & \leftarrow R_2 + R_{3k-1} & C_2 & \leftarrow C_2 + C_{3k-1}. \end{array} \tag{19}$$

The entries of the resulting matrix are all zero in the first and last row and column, except for $M_{1,3k} = 1, M_{3k,1} = 1$ (see Fig. 3(b), 3(c) and 3(d)). Let S denote the submatrix induced by rows i and j such that for $i, j \in \{2, \ldots, 3k-1\}$. Thus S is a $(3k-2) \times (3k-2)$ matrix. For S, we have $S_{1,1} = 1, S_{1,3k-2} = -1, S_{3k-2,1} = -1$. The remainder of the entries in the first row and the first column of S are all zero. Furthermore, the submatrix of S induced by rows i and j such that $i, j \in \{3, \ldots, 3k-1\}$ is the same as $M[3k-3]$. Developing the determinant of S along the first row and simplifying the powers of -1 yield:

$$\det(S) = \det(M[3k-3]) - \det(M[3k-4]). \tag{20}$$

Developing the determinant of matrix M after the row and column operations described above along R_1 followed by R_{3k} (both these rows have only one non-zero entry, namely, $M_{1,3k} = M_{3k,1} = 1$) yields:

$$\det(M[3k]) = -\det(S) = \det(M[3k-4]) - \det(M[3k-3]), \tag{21}$$

and ends the proof. $\square$

We now comment on the fact that the automata A_G and C_i are not probabilistic. Let $L(A)$ denote the language accepted by automaton A and $\deg(v)$ denote the degree of vertex v in G. The analysis presented here is similar to that of [19], we outline it for the sake of completion.

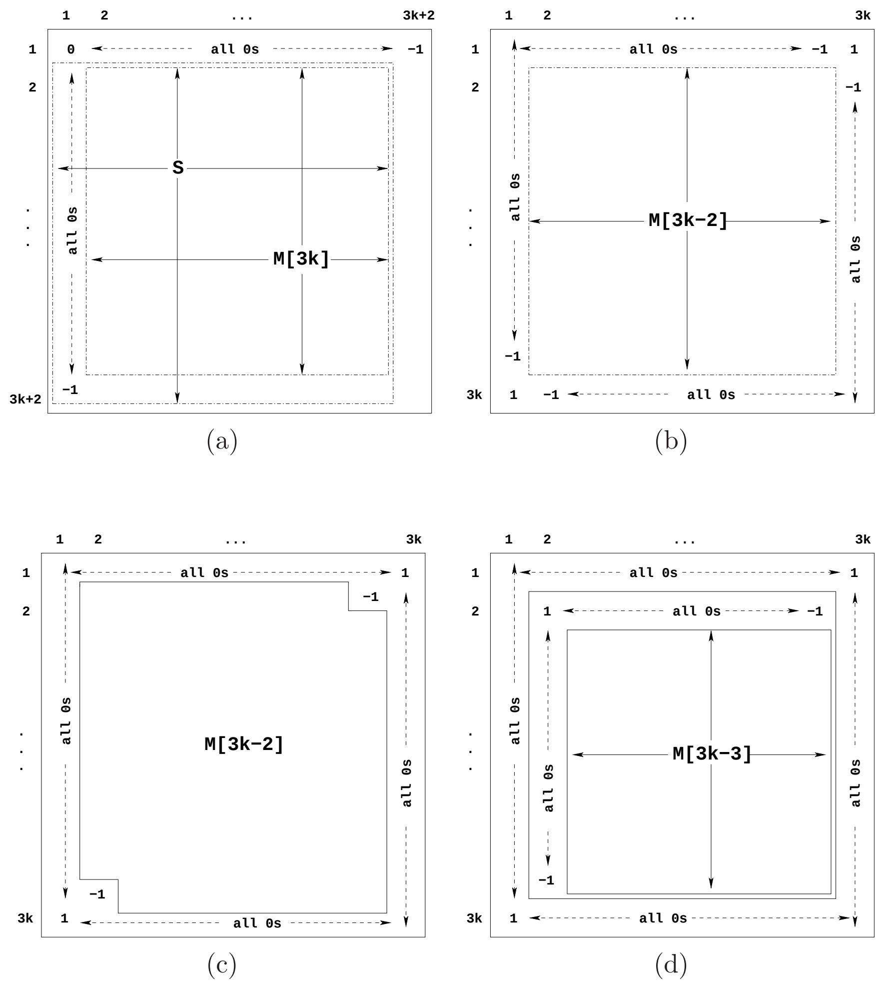


Fig. 3. (a) Case 2. The matrix M' obtained from $M[3k+2]$ after the row and column operations described in Eqn. 15. (b) Case 3. The matrix obtained from $M[3k]$ after the first four (row and column) operations described in Eqn. 19. (c) Case 3. The matrix obtained after the next four (row and column) operations. (d) Case 3. The final matrix after all row and column operations.

Lemma 9. *The sum of the weights of all accepting paths in A_G and C_i is given by*

$$\sum_{x \in L(A_G)} [\![A_G]\!](x) = \sum_{v \in V} 2^{\deg(v)} \qquad \sum_{x \in L(C_i)} [\![C_i]\!](x) = i|L(C_i)| = i2^n. \tag{22}$$

Proof. Since each transition in A_G has weight 1, the weight of every accepting path in A_G is 1. Thus, the sum of the weights of all accepting paths in A_G is exactly equal to the number of accepting paths. Let N_i denote the number of accepting paths in A_G that pass through state $q_{i,0}$. A vertex $i \in V$ has $\deg(i)$ vertices adjacent to it in G. By construction, we introduce two transitions from state $q_{i,j-1}$ to $q_{i,j}$ for each neighbor j of i, one with label j and weight 1 and another with label ϵ and weight 1, and this doubles the number of accepting paths that pass through $q_{i,0}$. Thus, $N_i = 2^{\deg(i)}$ and the number of accepting paths in A_G is equal to $\sum_{i=0}^{n} 2^{\deg(i)}$.

For automaton C_i, each string has weight i, and the language accepts 2^n strings. Thus, the sum of the weights of all accepting paths in C_i is exactly $i2^n$. □

Let Z_G denote $\sum_{v \in V} 2^{\deg(v)}$. One way to make A_G and C_i probabilistic is to assign a final weight $1/Z_G$ to A_G and $1/i2^n$ to C_i. However, this would result in a modification of matrix M as $M_{i,j}$ would then become $|i/(i2^n) - j/Z_G|^{2p+1}$ and we wish to use our proof of the invertability of M for $M_{i,j} = |i-j|^{2p+1}$ for $n+1 \not\equiv 1 \pmod 3$. This can be achieved as follows:

(1) If $Z_G \geq i2^n$, then we normalize both automata A_G and C_i by assigning them the final weight $1/Z_G$. The sum of the weights of all accepting paths in A_G is then one but that in C_i is given by $i2^n/Z_G$, which is less than one. To make C_i probabilistic (i.e. the sum of the weights of all accepting paths in C_i is exactly one), we introduce a new symbol, say \$, and add a transition in C_i from its start state to its final state with input label \$ and weight $1 - i2^n/Z_G$. Let $\widehat{A}_G, \widehat{C}_i$ denote automata A_G and C_i modified as described. It is straightforward to verify that

$$L_{2p+1}(\widehat{A}_G, \widehat{C}_i) = \sum_{j=0}^{n} \frac{a_j}{Z_G}|i-j|^{2p+1} + \left(1 - \frac{i2^n}{Z_G}\right). \tag{23}$$

(2) If $Z_G < i2^n$, then we normalize A_G and C_i by assigning them the final weight $1/i2^n$. Now the sum of the weights of all accepting paths in C_i is one but that in A_G is given by $Z_G/i2^n$. Again, we can introduce a new symbol, say \$, and add a transition in A_G from its start state to its final state with input label \$ and weight $1 - Z_G/i2^n$. For the modified automata $\widehat{A}_G$ and $\widehat{C}_i$, as before, we obtain

$$L_{2p+1}(\widehat{A}_G, \widehat{C}_i) = \sum_{j=0}^{n} \frac{a_j}{i2^n}|i-j|^{2p+1} + \left(1 - \frac{Z_G}{i2^n}\right). \tag{24}$$

By Eqn. 23 and Eqn. 24, the following holds:

$$\sum_{j=0}^{n} a_j|i-j|^{2p+1} = \begin{cases} Z_G\left(L_{2p+1}(\widehat{A}_G, \widehat{C}_i) + 1 - \frac{i2^n}{Z_G}\right) & \text{if } Z_G \geq i2^n \\ i2^n\left(L_{2p+1}(\widehat{A}_G, \widehat{C}_i) + 1 - \frac{Z_G}{i2^n}\right) & \text{if } Z_G < i2^n. \end{cases} \tag{25}$$

Since it is NP-hard to compute $\sum_{j=0}^{n} a_j|i-j|^{2p+1}$ for all i (by the previous reduction), it must be NP-hard to compute the L_{2p+1} distance between $\widehat{A}_G$ and $\widehat{C}_i$, which are both probabilistic.

4.3. *Inapproximability Result*

This section shows an inapproximability result for the computation of the L_{2p+1} distance of two probabilistic automata. Specifically, we show that given automata

A_1 and A_2, there exists an $\epsilon = f(|A_1| + |A_2|, p)$, for a specific function f, such that it is NP-hard to approximate the L_{2p+1} distance between A_1 and A_2 up to an additive factor of ϵ.

Let $X \in \mathbb{R}^{n+1}, X^0 \in \mathbb{R}^{n+1}, Y \in \mathbb{R}^{n+1}$ be the column vectors defined by

$$X_i = L_{2p+1}(\widehat{A}_G, \widehat{C}_i) \qquad X_i^0 = \begin{cases} 1 - i2^n/Z_G & \text{if } Z_G \geq i2^n \\ 1 - Z_G/i2^n & \text{if } Z_G < i2^n \end{cases} \qquad Y_i = a_i. \tag{26}$$

Further, let $D \in \mathbb{R}^{(n+1)\times(n+1)}$ be the diagonal matrix defined by

$$D_{i,i} = \begin{cases} 1/Z_G & \text{if } Z_G \geq i2^n \\ 1/i2^n & \text{if } Z_G < i2^n. \end{cases} \tag{27}$$

Eqn. 25 can be rewritten, in matrix terms as

$$X = D(MY) + X^0 \Longleftrightarrow D^{-1}(X - X^0) = MY. \tag{28}$$

Suppose that it is possible to approximate the L_{2p+1} distance between two probabilistic automata up to an additive factor of ϵ in polynomial time. Then one can compute the column vector $X' \in \mathbb{R}^{n+1}$, where X'_i is the approximation of the L_{2p+1} distance between $\widehat{A}_G$ and $\widehat{C}_i$. Thus, $|X'_i - X_i| \leq \epsilon$ for all $i \in \{0, 1, \ldots, n\}$. Let $Y' \in \mathbb{R}^{n+1}$ be the column vector such that $X' = D(MY') + X^0$. Recall that it is NP-hard to compute the column vector Y exactly:

$$\begin{aligned} X - X' &= D(MY') - D(MY) \qquad (29)\\ &= D(M(Y' - Y)) \qquad (30)\\ \Rightarrow M^{-1}(D^{-1}(X - X')) &= Y' - Y. \qquad (31) \end{aligned}$$

Since $\|X - X'\|_\infty \leq \epsilon$, it follows that

$$\begin{aligned} \|Y' - Y\|_\infty &= \|M^{-1}(D^{-1}(X - X'))\|_\infty \qquad (32)\\ &\leq \|M^{-1}\|_\infty \|D^{-1}\|_\infty \epsilon \qquad (33)\\ &\leq \frac{\|D^{-1}\|_\infty \epsilon}{\|M\|_\infty}. \qquad (34) \end{aligned}$$

Since D is a diagonal matrix, D^{-1} is defined by $D^{-1}_{i,i} = 1/D_{i,i}$. It is straightforward to verify that $\|D^{-1}\|_\infty = n2^n$ and $\|M\|_\infty = \sum_{i=1}^n i^{2p+1} = \Theta(n^{2p+2})$. Therefore,

$$\|Y' - Y\|_\infty \leq \frac{n2^n \epsilon}{cn^{2p+2}}, \tag{35}$$

for some fixed constant c (that appears in the $\Theta(\cdot)$ term). Recall that $Y_i = a_i$ is the number of strings in the automaton A_G with weight exactly i and is therefore an integer. We use the fact that each Y_i, for $i \in \{1, 2, \ldots, n\}$, is an integer and observe that in fact if $\|Y' - Y\|_\infty < 1/2$, $|Y'_i - Y_i| < 1/2$ and $Y_i - 1/2 < Y'_i < Y_i + 1/2$ for each i. Thus, the column vector Y' can be used to uniquely determine the column vector Y, which is NP-hard to compute. Therefore, it must be the case that Y' is NP-hard to compute (under the assumption that $\|Y' - Y\|_\infty < 1/2$). In order to enforce that condition, by Eqn. 35, it suffices that $\frac{n2^n\epsilon}{cn^{2p+2}} < \frac{1}{2}$. The condition on ϵ is thus

$$\epsilon < \frac{cn^{2p+2}}{2n2^n}. \tag{36}$$

Since the denominator in the bound on ϵ in Eqn. 36 is exponential while the numerator is only polynomial, we are unable to use this bound to show the hardness of approximating the L_{2p+1} distance between two automata A_1 and A_2 independently of $|A_1| + |A_2|$. Note that in the construction, $|\widehat{A}_G| = \Theta(n^2)$ and $\widehat{C}_i = \Theta(n)$ so that $|\widehat{A}_G| + |\widehat{C}_i| = \Theta(n^2)$.

Theorem 10. *Given two probabilistic automata A_1 and A_2, such that $|A_1|+|A_2| \leq s$, there exists an $\epsilon = f(s,p)$ such that it is NP-hard to approximate the L_{2p+1} distance between A_1 and A_2 within an additive factor of ϵ.*

4.4. *Absolute Value Automata*

The hardness results for the computation of the L_{2p+1} distances of probabilistic automata seem to be related to the obligatory presence of the absolute values in the definition of these distances. This brings us to examine several questions related to the absolute value.

In particular, one may ask if in general there exists a weighted automaton C over the real semiring $(\mathbb{R}, +, \cdot, 0, 1)$ representing the absolute value of the difference of two probabilistic automata A and B, that is such that

$$\forall x \in \Sigma^*,\ [\![C]\!](x) = |[\![A]\!](x) - [\![B]\!](x)|. \tag{37}$$

We could refer to C as the *absolute value automaton* and denote it by $|A - B|$. The general existence of C and even its efficient computation would not be sufficient to guarantee the efficient computability of the L_1 distance (or L_{2p+1} distance).

Indeed, by definition of C, to compute the L_1 distance of A and B, one can sum the weights of all successful paths of C. But, since the semiring $(\mathbb{R}, +, \cdot, 0, 1)$ is not closed, no general algorithm is available for computing this sum. Note that $[\![C]\!]$ takes its values in $\mathbb{R}_+$, but this does not imply that its transition weights are necessarily in $\mathbb{R}_+$, nor does it even imply the existence of an equivalent weighted automaton C' over $(\mathbb{R}_+, +, \cdot, 0, 1)$. This is because $\mathbb{R}$ is not a *Fatou extension* of $\mathbb{R}_+$: there exist indeed weighted automata over the real semiring taking their values in $\mathbb{R}_+$ that do not admit an equivalent weighted automaton over $(\mathbb{R}_+, +, \cdot, 0, 1)$ [20, 13].

The hardness of the computation of the L_1 distance guarantees however that unless P = NP, in general, there exists no absolute value weighted automaton C over $(\mathbb{R}_+, +, \cdot, 0, 1)$ that can be computed efficiently since the sum of the weights of the paths of C, i.e., the L_1 distance, could then be computed efficiently.

Note also that the general problem of determining if a weighted automaton A defined over the real semiring $(\mathbb{R}, +, \cdot, 0, 1)$ accepts no string of negative weight is undecidable [20, 13]. Since there exists an efficient algorithm for testing the equivalence of two weighted automata over the real semiring [21], this implies that in general there does not exists a computable absolute value automaton $|A|$ such that $\forall x \in \Sigma^*,\ [\![|A|]\!](x) = |[\![A]\!](x)|$.

5. Hellinger Distance

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 $\text{Hellinger}(A_1, A_2)$ of two probabilistic automata A_1 and A_2 is given by:

$$\text{Hellinger}(A_1, A_2) = \Big(\sum_{x \in \Sigma^*} \big(\sqrt{[\![A_1]\!](x)} - \sqrt{[\![A_2]\!](x)}\big)^2 \Big)^{1/2}. \tag{38}$$

Thus,

$$\begin{aligned}
[\text{Hellinger}(A_1, A_2)]^2 &= \sum_{x \in \Sigma^*} \big(\sqrt{[\![A_1]\!](x)} - \sqrt{[\![A_2]\!](x)}\big)^2 \qquad (39) \\
&= \sum_{x \in \Sigma^*} [\![A_1]\!](x) + \sum_{x \in \Sigma^*} [\![A_2]\!](x) - 2 \sum_{x \in \Sigma^*} \sqrt{[\![A_1]\!](x)[\![A_2]\!](x)} \\
&= 2\big(1 - \sum_{x \in \Sigma^*} \sqrt{[\![A_1]\!](x)[\![A_2]\!](x)}\big).
\end{aligned}$$

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 $\Phi : (\mathbb{R}_+, +, \cdot, 0, 1) \to (\mathbb{R}_+, +, \cdot, 0, 1)$ 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 [4, 6]. 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 and 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 [16] that uses only linear space.

6. Equivalence of Probabilistic Automata

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 zero. 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 [21] can be used to derive a more efficient algorithm.

Theorem 11. *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 [21, 2] 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 zero 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. Note that the same algorithm can be used to test the equivalence of probabilistic automata as defined by [18]. The best algorithm previously reported in the literature was that of Wen-Guey Tzeng whose complexity is $O(|\Sigma|\,(|A_1|+|A_2|)^4)$ [24]. 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 [21], which the author was apparently not aware of.

There is also a claim by [1] that the equivalence of weighted automata with transition weights in $\mathbb{Z}$ can be tested in cubic time. However, the paper does not include a full proof of the correctness of the algorithm and the complexity. Instead it relies on several claims made by others in private communications or results appearing in a Siberian journal not accessible to us. It also seems to be specifically using the property of the coefficients being integers. The algorithm we are describing does not require transition weights to be integers and applies to all probabilistic automata and other weighted automata with real-valued weights.

7. Conclusion

We presented an exhaustive analysis of the problem of computing the L_p distance of probabilistic automata. We gave efficient exact and approximate algorithms for the computation of the L_{2p} and showed the intractability of the problem for L_{2p+1}

distances. As shown for the specific case of the Hellinger distance, our algorithms can be straightforwardly generalized to other distances. 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.

Note that the hard cases of computing the L_p-norm of a probabilistic automaton do not coincide with those of computing the L_p distance. As shown elsewhere [6], the L_p-norm of any probabilistic automaton can be computed in polynomial time, for all finite values of p. The problem of computing the L_∞-norm is however NP-hard [19].[c] As shown here, computing the L_p-distance of probabilistic automata is polynomial for all even values of p, and is NP-hard for all odd values and for $p = \infty$.

Acknowledgments

The research of Mehryar Mohri and Ashish Rastogi was partially supported by the New York State Office of Science Technology and Academic Research (NYSTAR). This project was also sponsored in part by the Department of the Army Award Number W81XWH-04-1-0307. The U.S. Army Medical Research Acquisition Activity, 820 Chandler Street, Fort Detrick MD 21702-5014 is the awarding and administering acquisition office. The content of this material does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

References

[1] Kostyantyn Archangelsky. Efficient Algorithm for Checking Multiplicity Equivalence for the Finite $Z - \Sigma^*$-Automata. In *Developments in Language Theory (DLT 2002)*, pages 283–289, 2002.

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

[3] Francisco Casacuberta and Colin De La Higuera. Computational complexity of problems on probabilistic grammars and transducers. In *Proceedings of the 5th International Colloquium on Grammatical Inference (ICGI 2000)*, pages 15–24, London, UK, 2000. Springer-Verlag.

[4] Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. On the Computation of Some Standard Distances between Probabilistic Automata. In *Proceedings of the 11th International Conference on Implementation and Application of Automata (CIAA 2006)*, volume 4094 of *Lecture Notes in Computer Science*, pages 137–149, Taipei, Taiwan, August 2006. Springer-Verlag, Heidelberg, Germany.

[5] 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.

[c] Note that this implies that the harder problem of determining the most likely sequence of a probabilistic automaton is also NP-hard, a result that was proven earlier by [12] and [3].

[6] Corinna Cortes, Mehryar Mohri, Ashish Rastogi, and Michael Riley. On the computation of the Relative Entropy of Probabilistic Automata. *International Journal of Foundations of Computer Science*, to appear, 2007.

[7] Imre Csiszar and Janos Korner. *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Akademiai Kiado, 1997.

[8] 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.

[9] 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.

[10] Samuel Eilenberg. *Automata, Languages and Machines*, volume A–B. Academic Press, 1974–1976.

[11] 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.

[12] Joshua Goodman. *Parsing inside-out*. PhD thesis, Harvard University, May 1998.

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

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

[15] 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.

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

[17] Azaria Paz. *Introduction to probabilistic automata*. Academic Press, New York, 1971.

[18] Michael O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.

[19] 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.

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

[21] Marcel-Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4, 1961.

[22] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS 1996)*, page 627, Washington, DC, USA, 1996. IEEE Computer Society.

[23] Flemming Topsøe. Some inequalities for information divergence and related measures of discrimination. *IEEE Trans. Inform. Theory*, 46:1602–1609, 2000.

[24] Wen-Guey Tzeng. A Polynomial-Time Algorithm for the Equivalence of Probabilistic Automata. *Foundations of Computer Science (FOCS 1992)*, pages 216–227, 1992.