Probabilistic Automata of Bounded Ambiguity

Nathanaël Fijalkow[a,b], Cristian Riveros[c,e], James Worrell[d]

[a] *CNRS, LaBRI*
[b] *The Alan Turing Institute of data science, London*
[c] *Pontificia Universidad Católica de Chile*
[d] *University of Oxford*
[e] *Millennium Institute for Foundational Research on Data (IMFD), Chile*

Abstract

Probabilistic automata are an extension of nondeterministic finite automata in which transitions are annotated with probabilities. Despite its simplicity, this model is very expressive and many of the associated algorithmic questions are undecidable. In this work we focus on the emptiness problem, which asks whether a given probabilistic automaton accepts some word with probability greater than a given threshold. We consider a natural and well-studied structural restriction on automata, namely the degree of ambiguity, which is defined as the maximum number of accepting runs over all words. We observe that undecidability of the emptiness problem requires infinite ambiguity and so we focus on the case of finitely ambiguous probabilistic automata.

Our main contributions are to construct efficient algorithms for analysing finitely ambiguous probabilistic automata through a reduction to a multi-objective optimisation problem called the stochastic path problem. We obtain a polynomial time algorithm for approximating the value of probabilistic automata of fixed ambiguity and a quasi-polynomial time algorithm for the emptiness problem for 2-ambiguous probabilistic automata.

We complement these positive results by an inapproximability result stating that the value of finitely ambiguous probabilistic automata cannot be approximated unless $\mathbf{P} = \mathbf{NP}$.

Keywords: probabilistic automata, weighted automata, multi-objective optimisation

1. Introduction

Probabilistic automata are a natural extension of non-deterministic automata that were introduced by Rabin (1963). Such automata can also be seen as a type of weighted automata, as defined by Schützenberger (1961), over the semiring of real numbers. Syntactically, a probabilistic automaton is a non-deterministic finite automaton in which each edge is annotated by a probability. Such an automaton associates to every word a value between 0 and 1, which is the total probability that a run on the word ends in an accepting state. We call this the acceptance probability of the word.

Despite their simplicity, probabilistic automata are very expressive and have been widely studied. Unfortunately the price of this expressiveness is that almost all natural decision problems are undecidable. Consequently, various approaches based on probabilistic automata with resources, such as structure, dimension, or randomness, have been studied Chatterjee and Tracol (2012); Fijalkow et al. (2012, 2015); Chadha et al. (2017).

In this paper, we look at probabilistic automata of *bounded ambiguity*, where the ambiguity of a word relative to a given automaton is the number of accepting runs. We say that a probabilistic automaton is f-ambiguous, for a function $f : \mathbb{N} \to \mathbb{N}$, if every word of length n has at most $f(n)$ accepting runs. (Note that ambiguity is a property of the underlying nondeterministic finite automata, and is independent of the transition probabilities.) This notion has been extensively studied in automata theory; in particular, the landmark paper of Weber and Seidl (1991) gives respective structural characterisations of the classes finitely, polynomially, and exponentially ambiguous nondeterministic finite automata, from which polynomial-time algorithms are obtained for deciding membership in each of these classes.

We focus on the most natural and well-studied problem for probabilistic automata, called the *emptiness problem*: given a probabilistic automaton and a threshold, does there exist a word accepted with probability exceeding a given threshold? Since the emptiness problem is already undecidable for linearly ambiguous probabilistic automata, we focus on finitely ambiguous probabilistic automata.

We study the complexity of the emptiness problem on various classes of finitely ambiguous probabilistic automata. For each positive integer k we consider the class of *k-ambiguous probabilistic automata*, i.e., automata with at most k accepting runs on any word. More generally we fix a polynomial p and consider the class of automata whose ambiguity is at most $p(m)$, where

m is the number of states. More generally still, bearing in mind that the ambiguity can be exponential in the number of states, we have the class of all finitely ambiguous automata.

Our main results are as follows. We show that the emptiness problem for finitely ambiguous probabilistic automaton is, respectively:

- in **NEXPTIME** and **PSPACE**-hard for the class of all finitely ambiguous automata;
- **PSPACE**-complete for the class of probabilistic automata with ambiguity bounded by a fixed non-constant polynomial in the number of states.
- in **NP** for the class of k-ambiguous probabilistic automata, for every positive integer k.
- in quasi-polynomial time for the class of 2-ambiguous probabilistic automata.

A natural counterpart of the emptiness problem is the function problem of computing the value of a probabilistic automaton, that is, the supremum over all words of the acceptance probability of a word. Here we show:

- for the class of all finitely ambiguous probabilistic automata, there is no polynomial-time approximation algorithm for the value problem unless $\mathbf{P} = \mathbf{NP}$,
- for each fixed k, the value of a k-ambiguous probabilistic automaton is approximable up to any multiplicative constant in polynomial time.

The starting point to prove these results is to give an upper bound on the length of a shortest word whose probability exceeds a given threshold. More precisely, we show that for a k-ambiguous probabilistic automaton with n states there is a maximum-probability word of length at most n^k. More generally, we show that for a finitely ambiguous probabilistic automaton with n states, there is a maximum-probability word of length at most $n!$. The latter result easily leads to a **PSPACE** upper bound for the emptiness problem in the case that the ambiguity is bounded by a fixed polynomial in the number of states. Most of the remainder of the paper is devoted to the case of k-ambiguous automata for a fixed k.

We give a polynomial-time reduction from the emptiness problem for k-ambiguous probabilistic automata to a multi-objective optimisation problem, which we call the *k-stochastic path problem*. Using this reduction, we obtain a polynomial-time algorithm for approximating the value of a k-ambiguous probabilistic automata, and a quasi-polynomial time algorithm for the emptiness problem of 2-ambiguous probabilistic automata.

2. Preliminaries

Let Σ be a finite alphabet. For any word $w \in \Sigma^*$, we denote by $|w|$ its length. Given a finite set Q, a distribution is a function $\delta : Q \to [0,1]$ such that $\sum_{q \in Q} \delta(q) = 1$. The set of distributions over Q is denoted $\mathcal{D}(Q)$.

A *probabilistic automaton* is a tuple $\mathcal{P} = (Q, q_{in}, \Delta, F)$, where Q is a finite set of states, q_{in} is the initial state, $\Delta : Q \times A \to \mathcal{D}(Q)$ is the transition function, and F is the set of accepting states. Given a word $w = a_1 \cdots a_n$, a run ρ over w is a sequence of states $q_0, q_1, \ldots, q_n$. The probability of such a run is $\mathcal{P}(\rho) = \prod_{\ell \in \{1,\ldots,n\}} \Delta(q_{\ell-1}, a_\ell)(q_\ell)$. We denote by $\mathrm{Run}_{\mathcal{P}}(p \xrightarrow{w} q)$ the set of runs ρ over w starting in p and finishing in q with $\mathcal{P}(\rho) > 0$. The number $\mathcal{P}(p \xrightarrow{w} q)$ is the probability to go from p to q reading w, defined as the sum of the probabilities of its runs, namely:

$$\mathcal{P}(p \xrightarrow{w} q) = \sum_{\rho \in \mathrm{Run}_{\mathcal{P}}(p \xrightarrow{w} q)} \mathcal{P}(\rho).$$

A run ρ is accepting if it starts in q_{in}, satisfies $\mathcal{P}(\rho) > 0$, and finishes in an accepting state, *i.e.* a state in F. We denote by $\mathrm{Run}_{\mathcal{P}}(w)$ the set of accepting runs over w. The *probability* of w over $\mathcal{P}$ is defined as the sum of the probabilities of its accepting runs by:

$$\mathcal{P}(w) = \sum_{\rho \in \mathrm{Run}_{\mathcal{P}}(w)} \mathcal{P}(\rho).$$

Ambiguity. In this paper, we consider different subclasses of probabilistic automata, obtained by restrictions on *ambiguity*. More specifically, we say that:

- $\mathcal{P}$ is *unambiguous* if every word w has at most one accepting run, *i.e.* $|\mathrm{Run}_{\mathcal{P}}(w)| \leq 1$.

- $\mathcal{P}$ is *k-ambiguous* if every word w has at most k accepting runs, *i.e.* $|\operatorname{Run}_{\mathcal{P}}(w)| \leq k$.
- $\mathcal{P}$ is *finitely ambiguous*, if there exists k such that $\mathcal{P}$ is k-ambiguous.
- $\mathcal{P}$ is *polynomially ambiguous*, if there exists a polynomial P such that for every word w, we have $|\operatorname{Run}_{\mathcal{P}}(w)| \leq P(|w|)$.

If polynomial P is linear or quadratic then we say that a polynomially ambiguous automaton $\mathcal{P}$ is linearly ambiguous or quadratically ambiguous, respectively. It is proved in Weber and Seidl (1991) that it is decidable in polynomial time whether a probabilistic automaton $\mathcal{P}$ is unambiguous, finitely ambiguous, or polynomially ambiguous. Furthermore, a consequence of the results of Weber and Seidl (1991) is that an automaton which is not finitely ambiguous has ambiguity bounded below by a linear function.

Emptiness problem and value. Let $\mathcal{P}$ be a probabilistic automaton and c a threshold. Following Rabin (1963), we define the threshold language induced by $\mathcal{P}$ and c as:

$$L^{>c}(\mathcal{P}) = \{w \in \Sigma^* \mid \mathcal{P}(w) > c\}.$$

The *emptiness problem* asks, given a probabilistic automaton $\mathcal{P}$ and a threshold c, whether the language $L^{>c}(\mathcal{P})$ is non-empty, that is, whether there exists a word w such that $\mathcal{P}(w) > c$.

A related function problem is to compute the value of a probabilistic automaton $\mathcal{P}$, defined by $\operatorname{val}(\mathcal{P}) = \sup_{w\in\Sigma^*} \mathcal{P}(w)$. Note that the emptiness problem is equivalent to asking whether $\operatorname{val}(\mathcal{P}) > c$.

3. Undecidability for Linearly Ambiguous Probabilistic Automata

In this section, we discuss undecidability results for linearly ambiguous probabilistic automata, which justifies the focus of our paper on finitely ambiguous probabilistic automata.

Theorem 1 (Chadha et al. (2018); Daviaud et al. (2018))**.** *The emptiness problem is undecidable for linearly ambiguous probabilistic automata.*

Undecidability of the emptiness problem has long been known for general probabilistic automata, see Paz (1971); Bertoni (1974); Gimbert and Oualhadj (2010). However, the automata involved in the proof have exponential ambiguity.

In the conference version of this paper we explained how to adapt the proof strategy above to obtain the undecidability of the emptiness problem for quadratically ambiguous probabilistic automata, see Fijalkow et al. (2017). We left open whether the undecidability already holds for linearly ambiguous automata. Two subsequent independent papers filled the gap by Daviaud et al. (2018); Chadha et al. (2018), showing the stronger result stated in Theorem 1. We therefore focus here on the isolation problem.

Given a probabilistic automaton $\mathcal{P}$, we say that a threshold c is isolated if there exists $\varepsilon > 0$ such that for all words w, we have $|\mathcal{P}(w) - c| > \varepsilon$. Rabin (1963) proved that if a threshold c is isolated then the corresponding language $L^{\geq c}(\mathcal{P})$ is regular. The isolation problem asks to determine whether a given threshold is isolated for a given automaton. This problem was shown to be undecidable by Bertoni (1974); we refer to Fijalkow (2017) for a new presentation of this result. We can refine the result of Bertoni (1974) to obtain:

Theorem 2. *The isolation problem is undecidable for linearly ambiguous probabilistic automata.*

We start by describing the key ingredient in the undecidability proof of Bertoni (1974), which is the construction of a probabilistic automaton computing the value of a rational number given in binary with least significant digit on the left:

$$\mathrm{bin}^R(a_1 \cdots a_n) = \sum_{i=1}^{n} \frac{a_i}{2^{n-i+1}}.$$

The automaton proposed by Bertoni has exponential ambiguity. However, it is possible to construct a linearly ambiguous probabilistic automaton computing the same function but *reversing* the input:

$$\mathrm{bin}(a_1 \cdots a_n) = \sum_{i=1}^{n} \frac{a_i}{2^i}.$$

The automaton is represented in Figure 1.

Proof. We construct a reduction from a variant of the Post Correspondence's Problem, called the infinite PCP, and shown to be undecidable by Ruohonen (1985). The problem asks, given two homomorphisms $\varphi_1, \varphi_2 : \Sigma^* \to \{0,1\}^*$,

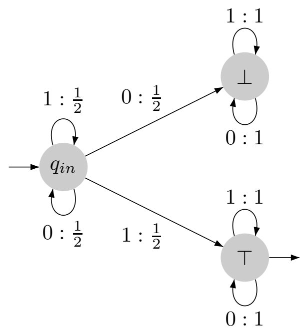

Figure 1: This probabilistic automaton computes bin.

to decide whether there exists an infinite word w in Σ^ω such that $\varphi_1(w) = \varphi_2(w)$ (where φ_1, φ_2 are extended to continuous Σ^ω maps on with respect to product topology). We first observe that equivalently, we ask whether for every $\varepsilon > 0$ there exists w in Σ^* such that $|\text{bin}(\varphi_1(w)) - \text{bin}(\varphi_2(w))| \leq \varepsilon$.

Indeed, if there exists an infinite word w such that $\varphi_1(w) = \varphi_2(w)$, then the sequences obtained by considering the images under φ_1 and φ_2 of prefixes of w have arbitrarily long common prefixes, so the difference of their binary values converges to 0. Conversely, assume that for any $\varepsilon > 0$ there exists a finite word w such that $|\text{bin}(\varphi_1(w)) - \text{bin}(\varphi_2(w))| \leq \varepsilon$, then we construct a solution to the infinite PCP using König's lemma. To this end, for each n let w_n be a finite word such that $|\text{bin}(\varphi_1(w_n)) - \text{bin}(\varphi_2(w_n))| < 2^n$, *i.e.*, such that $\varphi_1(w_n)$ and $\varphi_2(w_n)$ coincide on the first n letters. Applying König's Lemma to the infinite tree defined by the prefix closure of the set $\{w_n \mid n \geq 0\}$ (*i.e.*, each node in the tree is the prefix of some word w_n), there exists an infinite word w such that $\varphi_1(w) = \varphi_2(w)$.

We now construct the reduction from the infinite PCP to the isolation problem for linearly ambiguous probabilistic automata. Given two homomorphisms φ_1 and φ_2 we construct the linearly ambiguous probabilistic automaton $\mathcal{P}$ such that for every w in Σ^*,

$$\mathcal{P}(w) = \frac{1}{2}\left(\text{bin}(\varphi_1(w)) + 1 - \text{bin}(\varphi_2(w))\right).$$

Then for every $\varepsilon > 0$ there exists w in Σ^* such that

$$|\text{bin}(\varphi_1(w)) - \text{bin}(\varphi_2(w))| \leq \varepsilon$$

if, and only if, $|\mathcal{P}(w) - \frac{1}{2}| \leq \varepsilon$. $\square$

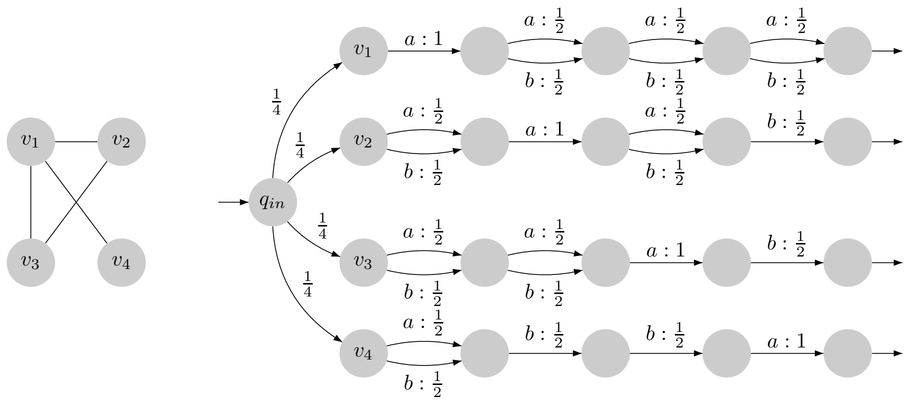

Figure 2: On the left a graph G and on the right the corresponding finitely ambiguous probabilistic automaton $\mathcal{P}_G$ such that $\text{MaxClique}(G) = 4 \cdot 2^3 \cdot \text{val}(\mathcal{P}_G)$.

An automaton is either finitely ambiguous, or at least linearly ambiguous. Bearing in mind our undecidability results for linearly ambiguous automata, we are led to focus on decidability results for finitely ambiguous automata.

4. Hardness of Approximation for Finitely Ambiguous Probabilistic Automata

In this section we show another negative result which applies to finitely ambiguous probabilistic automata. This hardness of approximation result complements the positive results obtained later in this paper, witnessing the complexity of analysing even finitely ambiguous probabilistic automata.

Theorem 3 (Lyngsø and Pedersen (2002)). *For every $\varepsilon > 0$, there is no polynomial time algorithm computing the value of finitely ambiguous probabilistic automata up to a factor $O(n^{\frac{1}{2}-\varepsilon})$ unless* $\mathbf{P} = \mathbf{NP}$.

We credit Lyngsø and Pedersen (2002) for the result although that paper uses a different framework, namely Hidden Markov models. For the sake of completeness we explain how to adapt the construction and its correctness to the case of probabilistic automata.

We construct a reduction from the size of the maximum clique, for which we know that no polynomial time approximation algorithm exists unless $\mathbf{P} = \mathbf{NP}$.

Given a graph G with n vertices, we construct a finitely ambiguous probabilistic automaton $\mathcal{P}_G$ with n^2 states such that for each m smaller than n, the automaton accepts a word with probability at least $\frac{m}{n2^{n-1}}$ if, and only if, the graph contains a clique of size at least m.

In $\mathcal{P}_G$ a word over $\{a, b\}^*$ represents a set of vertices in the graph: letter a means *in the set* and b *outside the set*. The automaton $\mathcal{P}_G$ on input w has n runs, one for each vertex, chosen each with probability $\frac{1}{n}$. Each accepting run has probability $\frac{1}{2^{n-1}}$, and the run corresponding to a vertex v is successful if, and only if, v and all its neighbours belong to the set of vertices represented by the word w. Hence a clique of size m induces a word accepted with probability $\frac{m}{n2^{n-1}}$, and conversely. In Figure 2 we illustrate this construction with a graph G with four vertices $v_1, \ldots, v_4$ and the corresponding finitely ambiguous probabilistic automaton $\mathcal{P}_G$. For example, here the word *aaab* represents the set of vertices $\{v_1, v_2, v_3\}$ which has probability $\frac{3}{4 \cdot 2^3}$ in $\mathcal{P}_G$, namely, a clique with three vertices.

Let $\text{MaxClique}(G)$ denote the size of the largest clique in G, the equivalence above reads $\text{MaxClique}(G) = n2^{n-1}\text{val}(\mathcal{P}_G)$. It follows that a $K(n)$-approximation algorithm for the value of finitely ambiguous probabilistic automata induces a $K(n^2)$-approximation algorithm for the size of the largest clique. Zuckerman (2007) proved that $\text{MaxClique}(G)$ cannot be approximated within a factor better than $O(n^{1-\varepsilon})$ for every $\varepsilon > 0$ unless $\mathbf{P} = \mathbf{NP}$, implying our result.

5. Decidability and Complexity of Finitely Ambiguous Probabilistic Automata

In this section we study threshold languages and the emptiness problem for finitely ambiguous probabilistic automata. We start by showing regularity of the threshold language $L^{>c}(\mathcal{P})$ for a finitely ambiguous probabilistic automaton $\mathcal{P}$ and threshold c. A classical result, due to Rabin (1963), shows that the threshold languages need not be regular in general. Unfortunately the proof of regularity, while constructive, is not useful for determining the complexity of the emptiness problem. However we are able to give a direct simple argument that bounds the length of witnesses for the emptiness problem. We then use these bounds to analyse the complexity of the emptiness problem.

Theorem 4. *Let $\mathcal{P}$ be a finitely ambiguous probabilistic automaton and c a threshold. Then $L^{>c}(\mathcal{P})$ is a regular language.*

Proof. Consider the set $\mathbb{N}^k$ under the pointwise order. Recall that $I \subseteq \mathbb{N}^k$ is an *ideal* if it is downward closed and directed. Every ideal I has the form

$$I = \{(n_1, \ldots, n_k) \in \mathbb{N}^k : n_{i_1} \leq a_1 \wedge \ldots \wedge n_{i_s} \leq a_s\} \tag{1}$$

for certain indices $1 \leq i_1 < \ldots < i_s \leq k$ and natural numbers $a_1, \ldots, a_s$. From the fact that $\mathbb{N}^k$ is a well-quasi-order it follows that every downward closed subset of $D \subseteq \mathbb{N}^k$ can be written as a finite union of ideals. Indeed such a decomposition can easily be computed from the finite set of minimal elements of $\mathbb{N}^k \setminus D$ as explained by Lazić and Schmitz (2015).

Let $\mathcal{P} = (Q, q_{in}, \Delta, F)$ be a finitely ambiguous probabilistic automaton with transition function $\Delta : Q \times \Sigma \to \mathcal{D}(Q)$. Say that a triple $(p, a, q) \in Q \times \Sigma \times Q$ is an *edge* if $\Delta(p, q)(q) > 0$. Suppose that $\mathcal{P}$ has s edges for some $s \in \mathbb{N}$ and fix a linear ordering on these edges. We say that $m = (m_{i,j}) \in \mathbb{N}^{s \times k}$ is *admissible* for a word $w \in \Sigma^*$ if there exist k distinct accepting runs of $\mathcal{P}$ on w such that $m_{i,j}$ is the number of times that the i-th edge is taken in the j-th accepting run.

For any ideal $I \subseteq \mathbb{N}^{s \times k}$ the set of $w \in \Sigma^*$ such that some $m \in I$ is admissible for w is a regular language. A non-deterministic automaton for this language guesses k distinct accepting runs of $\mathcal{P}$ and counts the number of times each edge is taken on each accepting run, up to a finite threshold N, where N is the largest integer appearing in the description of I in the form (1). It follows that for any downward closed subset $D \subseteq \mathbb{N}^{s \times k}$ the set of $w \in \Sigma^*$ such that some $m \in D$ is admissible for w is a regular language.

Now let $\lambda_1, \ldots, \lambda_s$ be the transition probabilities occuring in $\mathcal{P}$, listed according to the ordering on the edges. Given $k \in \mathbb{N}$, consider the set of tuples

$$S_k = \left\{ (m_{i,j}) \in \mathbb{N}^{s \times k} : \sum_{j=1}^{k} \lambda_1^{m_{1,j}} \ldots \lambda_s^{m_{s,j}} > c \right\}.$$

For any word $w \in \Sigma^*$, $w \in L^{>c}(\mathcal{P})$ if and only if there exists some k up to the (finite) degree of ambiguity of $\mathcal{P}$ and some $m \in S_k$ that is admissible for w. Since each set S_k is downwards closed, it follows that $L^{>c}(\mathcal{P})$ is regular. $\square$

The threshold language $L^{>c}(\mathcal{P})$ of a finitely ambiguous probabilistic automaton is regular, however, this does not say anything about how to decide efficiently whether $L^{>c}(\mathcal{P})$ is empty or not. Therefore, the next step is to bound the size of a witness word whenever $L^{>c}(\mathcal{P}) \neq \emptyset$. This will lead to upper bounds on the complexity of the emptiness problem restricted to k-ambiguous.

Lemma 1. *Let $\mathcal{P}$ be a k-ambiguous probabilistic automaton with n states. For every word w, there exists a word w' of length at most n^k such that $\mathcal{P}(w) \leq \mathcal{P}(w')$. This implies that the value of $\mathcal{P}$ is reached by some word of length at most n^k.*

Proof. Let $\mathcal{P} = (Q, q_{in}, \Delta, F)$ and suppose that there are exactly k' accepting runs on w for some $k' \leq k$. If w has length strictly greater than $n^{k'}$ then there exists a factorization $w = xyz$ for $x, y, z \in \Sigma^*$, with y non-empty and xz of length at most n^k, such that for each of the accepting runs on w, the infix corresponding to the factor y starts and ends in the same state. Then we have

$$\begin{aligned} \mathcal{P}(w) &= \textstyle\sum_{q\in F}\sum_{p\in Q}\mathcal{P}(q_{in} \xrightarrow{x} p)\mathcal{P}(p \xrightarrow{y} p)\mathcal{P}(p \xrightarrow{z} q) \\ &\leq \textstyle\sum_{q\in F}\sum_{p\in Q}\mathcal{P}(q_{in} \xrightarrow{x} p)\mathcal{P}(p \xrightarrow{z} q) \\ &= \mathcal{P}(xz)\,. \end{aligned}$$

□

Note that if k is fixed, then the size of a witness for $L^{>c}(\mathcal{P})$ is polynomial in the size of the automaton (i.e., n^k where n is the number of states of $\mathcal{P}$). Unfortunately, it has been shown in Weber and Seidl (1991) that the ambiguity of a finitely ambiguous automata can be exponential in the number of states and, thus, the previous lemma gives a double exponential bound for a witness of $L^{>c}(\mathcal{P})$ when k is not fixed. The next result shows that the size of a witness is at most exponential in the number of states.

Theorem 5. *Let $\mathcal{P}$ be a finitely ambiguous probabilistic automaton with n states. For every word w, there exists a word w' of length at most $n!$ such that $\mathcal{P}(w) \leq \mathcal{P}(w')$. This implies that the value of $\mathcal{P}$ is reached by some word of length at most $n!$.*

Proof. Consider a word $w = a_1 \cdots a_\ell$ of length at least $n!$. For any position i over the runs of $\mathcal{P}$ over w, denote by R_i the set of states participating in at least one accepting run over w. Furthermore, we equip R_i with the order defined by $p \leq q$ if $\mathcal{P}(q_0 \xrightarrow{a_1\cdots a_i} p) \leq \mathcal{P}(q_0 \xrightarrow{a_1\cdots a_i} q)$, *i.e.*, after reading the prefix $a_1 \cdots a_i$ of w the probability of being in state p is at most that of being in state q (with ties being resolved in a consistent way).

Since w has length at least $n!$, there exist two positions $i < j$ such that the ordered sets R_i and R_j coincide, denoted by R, and there exists a factorization $w = xyz$, with y the word in between positions i and j. Then we look at the runs of y from R to R, and make the following claims:

1. For every $p \in R$, there exists a run over y from p to a state in R.
2. For every $p \in R$, there exists at most one run over y from p to a state in R.
3. For every $p \in R$, we have $\mathcal{P}(q_0 \xrightarrow{uv} p) \leq \mathcal{P}(q_0 \xrightarrow{u} p)$.

The first claim follows from the fact that R is the set of states participating in at least one accepting run over w. For the second claim, if this were not the case, then the number of runs from R to R would increase unboundedly, contradicting that $\mathcal{P}$ is finitely ambiguous. Then it follows that for any state $p \in R$ there exists a unique run over y from p to some state in R, which is written p'.

For the last item, pick a state $p \in R$ and note that $\mathcal{P}(q_0 \xrightarrow{xy} p') = \mathcal{P}(q_0 \xrightarrow{x} p) \cdot \mathcal{P}(p \xrightarrow{y} p') \leq \mathcal{P}(q_0 \xrightarrow{x} p)$. This reduces the analysis to two cases. On one hand, $p \leq p'$ and then $\mathcal{P}(q_0 \xrightarrow{xy} p) \leq \mathcal{P}(q_0 \xrightarrow{xy} p') \leq \mathcal{P}(q_0 \xrightarrow{x} p)$. On the other hand, $p > p'$ and then there exists a state q in R such that $q \leq p$ and $p \leq q'$. This is because for any state $r \in R$ there exists a unique run over y from r to some state in R. It follows that $\mathcal{P}(q_0 \xrightarrow{xy} p) \leq \mathcal{P}(q_0 \xrightarrow{xy} q') \leq \mathcal{P}(q_0 \xrightarrow{x} q) \leq \mathcal{P}(q_0 \xrightarrow{x} p)$.

Finally, the proof of the theorem follows from the last claim (see Lemma 1). □

With the previous bounds in hand, we can study the computational complexity of the emptiness problem for various classes of finitely ambiguous probabilistic automata. For each fixed positive integer k we consider the class of k-ambiguous probabilistic automata. More generally, we can let the ambiguity of an automaton depend on the number n of states: we consider for each fixed polynomial p the class of all automata that have ambiguity at most $p(n)$. We call this the class of automata of p-bounded ambiguity. More generally still, we have the class of all finitely ambiguous probabilistic automata. (Recall that the ambiguity can be exponential in the number of states in general.)

Theorem 6.

- *For each fixed positive integer k, the emptiness problem for the class of k-ambiguous probabilistic automata is in* **NP**.
- *For each fixed polynomial p, the emptiness problem for the class of probabilistic automata with p-bounded ambiguity is in* **PSPACE**. *This problem is* **PSPACE**-*hard already in case $p(n) = n$.*
- *The emptiness problem for the class of finitely ambiguous probabilistic automata is in* **NEXPTIME** *and is* **PSPACE**-*hard.*

Proof. The algorithm for all three cases exploits Lemma 1 and Theorem 5 to guess and check a word witnessing that threshold language is non-empty.

For k-ambiguous $\mathcal{P}$ we know by Lemma 1 that a witness for checking whether $L^{>c}(\mathcal{P}) \neq \emptyset$ is of polynomial size in $\mathcal{P}$ and, therefore, we can guess a polynomial size word w and check if $\mathcal{P}(w) \geq c$, that is, the problem is in **NP**.

Similarly, for finitely ambiguous $\mathcal{P}$ we know by Theorem 5 that the witness is of size at most exponential, so we can guess and check if $L^{>c}(\mathcal{P})$ is non-empty in **NEXPTIME**.

To show that emptiness is in **PSPACE** for probabilistic automata of p-bounded ambiguity, one can guess a word w "on the fly" of size exponential and check whether $\mathcal{P}(w) \geq c$. The problem here is that the value $\mathcal{P}(w)$ (written in binary) could be of size exponential in the size of $\mathcal{P}$. To check if $\mathcal{P}(w) \geq c$ with polynomial space one can guess w, and keep a set of counters $\{c_t^i\}$ that stores how many times each transition t is used on the i-th run of $\mathcal{P}$ over w. Since w is of size at most exponential and $\mathcal{P}$ has at most $p(n)$ accepting runs, then we need polynomially many counters, each with at most polynomially many bits, namely, polynomial space to store these counters during the simulation of $\mathcal{P}$ over w. After we conclude guessing w, we can construct a polynomial-size circuit that receives $\{c_i^t\}$ and outputs $\mathcal{P}(w)$. Checking that the value of the circuit is greater or equal than a constant c correspond to decide PosSLP which can be solved in **PSPACE** as proved by Allender et al. (2009).

Next we consider a fixed polynomial $p(n) = n$, and prove **PSPACE**-hardness of emptiness for the class of probabilistic automata of $p(n)$-bounded ambiguity. The proof is by reduction from the emptiness problem of the intersection of a finite collection of deterministic finite automata: given as input a collection of deterministic finite automata, does there exist a word accepted by each of them? This problem has been shown **PSPACE**-complete in Kozen (1977). Given N deterministic automata, we construct a probabilistic automaton $\mathcal{P}$ whose first letter leads with probability $\frac{1}{N}$ to the initial state of each automaton. The probabilistic automaton $\mathcal{P}$ is N-ambiguous (note that N is at most the number of states of $\mathcal{P}$), and there exists a word w such that $\mathcal{P}(w) = 1$ if, and only if, there exists a word accepted by each of the N deterministic automata. $\square$

The aim of the last section is to give better algorithms for the k-ambiguous case: in particular, we show that the emptiness problem is in quasi polynomial time for 2-ambiguous probabilistic automata.

6. Algorithms and Approximations for Finitely Ambiguous Probabilistic Automata

This section is devoted to the construction of algorithms for both the emptiness problem and approximating the value of finitely ambiguous probabilistic automata. The first step is a reduction to a multi-objective optimisation problem that we call the stochastic path problem. We construct algorithms for this problem, relying on recent progress on the literature of multi-objective optimisation problems, and thus obtain algorithms for finitely ambiguous probabilistic automata.

6.1. The Stochastic Path Problem

The stochastic path problem is an optimisation problem on multi-weighted graphs. It is parametrised by a positive integer constant k, giving rise to the k-stochastic path problem, and the bi-stochastic path problem for $k = 2$. An instance is a triple consisting of an acyclic k-weighted graph G and two vertices s and t. A k-weighted graph is given by a set of vertices V of size n and a set of weighted edges $E \subseteq V \times (\mathbb{Q} \cap [0,1])^k \times V$. Note that the same pair of vertices (v, v') can have different edges between them and the weight of an edge is a k-tuple of rational numbers between 0 and 1.

A path π in G is a sequence of consecutive edges, and the set of feasible solutions of the problem are all paths from s to t. We denote by $(p_1(\pi), \ldots, p_k(\pi))$ the component-wise product of the weight vectors along the edges of π, namely, weights are computed multiplicatively along each component. In our applications we think of each component of a weight vector of an edge as the probability of a single event, and each component of a

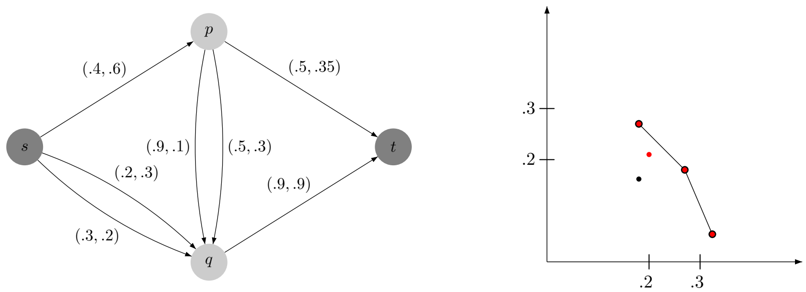

Figure 3: An instance of the bi-stochastic path problem on the left, and the values of all paths from s to t on the right. The four red dots are the Pareto curve, and the three connected red dots the convex Pareto curve.

weight vector of a path as the probability of a sequence of events. The value of the path π, denoted by $\text{val}(\pi)$, is obtained by summing each component of the weight vector of the path, namely, $\text{val}(\pi) = \sum_{i=1}^{k} p_i(\pi)$.

As a running example, on the left-hand side of Figure 3 we represent an instance of the bi-stochastic path problem. There are five paths from s to t, and their values are plotted in the right-hand side. For instance, the path s, p, q, t using the left edge from p to q has weight $(.4 \times .9 \times .9,\ .6 \times .1 \times .9) = (.324, .054)$.

The objective of the k-stochastic path problem is to maximize the *sum* of the objective functions, namely, $\text{val}(\pi)$. The value of the above path is $.324 + .054 = .378$. Finally, the decision problem associated with the k-stochastic path problem is the following:

> **The k-stochastic path problem**: given a k-weighted graph G, two vertices s and t and a threshold c in $\mathbb{Q} \cap [0, 1]$, does there exist a path π from s to t in G whose value is at least c, *i.e.* such that $\text{val}(\pi) \geq c$?

Towards finding efficient algorithms and approximations of k-ambiguous probabilistic automata, we show a polynomial time reduction from the emptiness problem of k-ambiguous probabilistic automata to the k-stochastic path problem. Intuitively, the reduction consists in constructing the powerset graph of the paths, restricting to at most k paths.

Lemma 2. *The emptiness problem of a k-ambiguous probabilistic automaton $\mathcal{P}$ reduces in polynomial time to a k-stochastic path problem $(G_{\mathcal{P}}, s, t)$. In particular, the reduction satisfies that*

1. *for any word w there exists a path π in G from s to t such that $\mathcal{P}(w) \leq val(\pi)$,*
2. *for any path π in G from s to t there exists a word w such that $val(\pi) \leq \mathcal{P}(w)$.*

Proof. Let $\mathcal{P} = (Q, q_{in}, \Delta, F)$ be a k-ambiguous probabilistic automaton with n states. The set of vertices of the k-weighted graph $G_{\mathcal{P}}$ is defined as $Q^k \times \{0, \ldots, n^k\} \times \{0,1\}^{k \times k}$ where $\{0,1\}^{k\times k}$ is the set of $k \times k$ matrices over $\{0,1\}$, plus a special source vertex s and a special target vertex t.

Intuitively, being in the vertex $((q_1, \ldots, q_k), \ell, M)$ means that we are simulating k runs which are now in the states $(q_1, \ldots, q_k)$, that the run so far has length ℓ, and the matrix M indicates which pairs of runs are different: $M(i,j) = 1$ if, and only if, the i-th run is different from the j-th run.

The set of edges is defined accordingly to the previous explanation as follows. For the source vertex, there is an edge from s to $((q_{in}, \ldots, q_{in}), 0, 0)$ with weight $(1, \ldots, 1)$, where 0 is the zero matrix. There is an edge from $((q_1, \ldots, q_k), \ell, M)$ to $((q'_1, \ldots, q'_k), \ell+1, M')$ with weight $(p_1, \ldots, p_k)$ if there exists a letter a in Σ such that for each $i \in \{1, \ldots, k\}$ we have $\Delta(q_i, a)(q'_i) = p_i$, and $M'(i,j) = 1$ if, and only if, $M(i,j) = 1$ or $q'_i \neq q'_j$. Finally, there is an edge from $((q_1, \ldots, q_k), \ell, M)$ to t with weight $(p_1, \ldots, p_k)$ where for each $i \in \{1, \ldots, k\}$ we have $p_i = 1$ if $q_i \in F$ and $M(i,j) = 1$ for every $j < i$, and $p_i = 0$ otherwise. Note that $G_{\mathcal{P}}$ is acyclic and of size polynomial in $\mathcal{P}$ given that k is fixed.

We prove the correctness of the construction. Let w be a word. Thanks to Lemma 1, we can assume without loss of generality that w has length at most n^k. Its set of accepting runs induces a path π in $G_{\mathcal{P}}$ from s to t with $\mathrm{val}(\pi) = \mathcal{P}(w)$. Conversely, a path π in $G_{\mathcal{P}}$ from s to t corresponds to a set of accepting runs for some word w with $\mathrm{val}(\pi) \leq \mathcal{P}(w)$. □

By the previous result, we can see that the emptiness problem of k-ambiguous probabilistic automata is closely related to the k-stochastic path problem. In the following we use this problem as a proxy to give approximation and efficient algorithms for the emptiness problem.

6.2. Approximating the Value in Polynomial Time

Multi-objective optimisation problems have long been studied; see Papadimitriou and Yannakakis (2000) and Diakonikolas and Yannakakis (2008) among many others. Since there is typically no single best solution, a natural notion for multi-objective optimisation problems is *Pareto curves*, which comprise sets of *dominating* solutions. To make things concrete, we illustrate the notion of Pareto curves on the k-stochastic path problem. We fix an instance (G, s, t) of the k-stochastic path problem. A Pareto curve is a set of paths $\mathcal{P}$ such that for every path π, there exists a path π' in $\mathcal{P}$ dominating π, *i.e.* such that for all i in $\{1, \dots, k\}$, we have $p_i(\pi) \leq p_i(\pi')$. In Figure 3, we can see that the Pareto curve of our running example is given by the four red dots. Here dominating means being to the right and higher, so only one path (represented by the black dot) is dominated by others. Unfortunately, the size of Pareto curves in discrete multi-objective optimisation problems is exponential in the worst case. Hence the introduction of two relaxations: convex and approximate Pareto curves.

A *convex Pareto curve* is a set of paths $\mathcal{C}$ such that for every path π, there exists a family of paths $\pi_1, \dots, \pi_m \in \mathcal{C}$ such that π is dominated by a convex combination of $\pi_1, \dots, \pi_m$ in the sense that there exist non-negative coefficients $\lambda_1, \dots, \lambda_m$ that sum to 1 such that $p_i(\pi) \leq \sum_j \lambda_j p_i(\pi_j)$ for all components i in $\{1, \dots, k\}$.

Convex Pareto curves have been studied in a general setting by Diakonikolas and Yannakakis (2008). They are in general smaller than Pareto curves, yielding efficient algorithms for convex optimisation problems.

In Figure 3, there exists a convex Pareto curve consisting of only three paths, the fourth one being dominated a convex combination of two other paths. The figure connects the three dots, showing what is sometimes called the Pareto front.

Fix $\varepsilon > 0$, an *ε-Pareto curve* is a set of paths $\mathcal{C}$ such that for every path π, there exists a path π' in $\mathcal{C}$ such that for all i in $\{1, \dots, k\}$, we have $p_i(\pi) \leq (1 + \varepsilon) \cdot p_i(\pi')$.

The notion of approximate Pareto curves is very appealing in our case for two reasons: first, knowing an approximate Pareto curve usually gives an approximately optimal solution, and second, a very general result of Papadimitriou and Yannakakis (2000) shows that in most multi-objective optimisation problems, there exists a polynomially succinct approximate Pareto curve.

The two relaxations are combined to give rise to the notion of *ε-convex Pareto curves*: in the case at hand, such a curve is a set of paths $\mathcal{C}$ such that for every path π, there exists a family of paths $\pi_1, \ldots, \pi_m \in \mathcal{C}$ and non-negative coefficients $\lambda_1, \ldots, \lambda_m$ that sum to 1 such that $p_i(\pi) \leq (1 + \varepsilon) \sum_j \lambda_j p_i(\pi_j)$ for all components i in $\{1, \ldots, k\}$.

The following result shows how to find an ε-approximation of the value of a k-ambiguous probabilistic automaton $\mathcal{P}$.

Theorem 7. *For any fixed k, there exists a polynomial time algorithm which given an instance of the k-stochastic path problem and $\varepsilon > 0$, returns an ε-convex Pareto curve in time polynomial in the instance and $\frac{1}{\varepsilon}$.*

Proof. We rely on general results of Papadimitriou and Yannakakis (2000), which give a sufficient condition for the existence of a polynomial time algorithm constructing an ε-convex Pareto curve in time polynomial in the instance and $\frac{1}{\varepsilon}$: it is enough to construct an algorithm solving the exact version in pseudo-polynomial time. Recall here that an algorithm is pseudo-polynomial if it runs in polynomial time when the numerical inputs are given in unary.

In our case, the exact k-stochastic path problem reads: given an instance (G, s, t) and a value c in $[0, 1] \cap \mathbb{Q}$, does there exist a path π in G from s to t such that $\sum_{i \in \{1,\ldots,k\}} p_i(\pi) = c$? If all transition probabilities are written using B bits, then it is enough to consider paths such that each weight uses at most $|V| \cdot B$ bits with V the set of vertices of G. Hence one can fill in a polynomially large table indexed by $(p, q, p_1, \ldots, p_k)$, which checks for the existence of a path from p to q of weights $(p_1, \ldots, p_k)$ using at most $|V| \cdot B$ bits. $\square$

Interestingly, the algorithm of Theorem 7 for the k-stochastic path problem yields a polynomial time algorithm to approximate the value of a k-ambiguous probabilistic automaton. Recall that the value of a probabilistic automaton $\mathcal{P}$ is defined by $\mathrm{val}(\mathcal{P}) = \sup_{w \in \Sigma^*} \mathcal{P}(w)$.

Theorem 8. *For a fixed k, there exists an algorithm which given a k-ambiguous probabilistic automaton and $\varepsilon > 0$, outputs an ε-approximation of the value in time polynomial in the size of the automaton and $\frac{1}{\varepsilon}$, i.e. a* *value Output such that*

$$\textit{Output} \leq \textit{val}(\mathcal{P}) \leq (1+\varepsilon) \cdot \textit{Output}.$$

Proof. Given a k-ambiguous probabilistic automaton $\mathcal{P}$, the algorithm for finding an ε-approximation of $\mathrm{val}(\mathcal{P})$ is as follows:

1. construct an instance $(G_{\mathcal{P}}, s, t)$ of the k-stochastic path problem using Lemma 2.
2. construct an ε-convex Pareto curve $\mathcal{C}$ for $(G_{\mathcal{P}}, s, t)$ thanks to Theorem 7.
3. return Output $:= \max_{\pi \in \mathcal{C}} \sum_{i \in \{1,\dots,k\}} p_i(\pi)$.

The first inequality is a direct consequence of Lemma 2 given that for every path π in $G_{\mathcal{P}}$, there exists a word w such that $\mathrm{val}(\pi) \leq \mathcal{P}(w)$, so Output $\leq$ $\mathrm{val}(\mathcal{P})$.

For the second inequality, take the word w that achieves $\mathcal{P}(w) = \mathrm{val}(\mathcal{P})$. By Lemma 2, there exists a path π such that $\mathcal{P}(w) \leq \mathrm{val}(\pi)$. Since $\mathcal{C}$ is an ε-Pareto curve, there exists a path $\pi' \in \mathcal{C}$ such that for all i in $\{1,\dots,k\}$, we have $p_i(\pi) \leq (1+\varepsilon) \cdot p_i(\pi')$. It follows that $\mathcal{P}(w) \leq (1+\varepsilon) \cdot \mathrm{val}(\pi') \leq$ $(1+\varepsilon) \cdot$ Output. □

It is interesting to compare the positive result of Theorem 8 to the negative result of Theorem 3. The key difference is in fixing the ambiguity, which allows us to go from intractable to tractable.

6.3. A Quasi-Polynomial Time Algorithm for 2-ambiguous Probabilistic Automata

The previous results show that one can ε-approximate the value of k-ambiguous probabilistic automaton in polynomial time. This is however not enough to decide the emptiness problem. In this direction, Theorem 6 shows that for any fixed k the emptiness problem of k-ambiguous probabilistic automata is in **NP**. We show that for $k = 2$ there exists a quasi-polynomial time algorithm for the emptiness problem. For this, we start by showing a quasi-polynomial time algorithm for the bi-stochastic path problem.

Theorem 9. *There exists an algorithm which given an instance of the bi-stochastic path problem, returns a convex Pareto curve in quasi-polynomial time.*

The advantage of using $k = 2$ lies in the existence of a quasi-polynomial bound on the size of convex Pareto curves. More precisely, if (G, s, t) is an instance of the bi-stochastic path problem with n vertices, then it can be shown that there exists a convex Pareto curve of size at most $n^{\log(n)}$. This result was proved in Gusfield (1980), and a matching lower bound was developed by Carstensen (1983). Note that they use a different framework, called parametric optimisation: in the parametric shortest path problem each edge has cost $c + \lambda d$, where λ is a parameter. The length of the shortest path is a piecewise linear concave function of λ, whose pieces correspond to the vertices of the convex Pareto curve for the bi-objective shortest path problem with weights (c, d). It is then easy to obtain an upper bound on the size of convex Pareto curves for the bi-stochastic path problem by reducing it to a bi-objective shortest path problem, mapping the weights (p, q) to $(-\log(p), -\log(q))$. Finally, the upper bound on the size of convex Pareto curves yields a quasi-polynomial time algorithm, by constructing them in a standard divide-and-conquer manner.

The algorithm of Theorem 9 yields a quasi-polynomial time algorithm for the emptiness problem of 2-ambiguous probabilistic automata.

Theorem 10. *There exists a quasi-polynomial time algorithm for the emptiness problem of* 2*-ambiguous probabilistic automata.*

Proof. Given a 2-ambiguous probabilistic automaton $\mathcal{P}$ and a threshold c, an algorithm for deciding the emptiness of $\mathcal{P}$ is as follows:

- construct an instance $(G_{\mathcal{P}}, s, t)$ of the bi-stochastic path problem using Lemma 2.
- construct a convex Pareto curve $\mathcal{C}$ for $(G_{\mathcal{P}}, s, t)$ thanks to Theorem 9.
- check whether Output $:= \max_{\pi \in \mathcal{C}} \sum_i p_i(\pi) > c$.

To show the correctness of this algorithm, we first prove that $\mathcal{P}(w) \leq$ Output for every word w. Let w be a word, thanks to Lemma 2, there exists a path π such that $\mathcal{P}(w) \leq p_1(\pi) + p_2(\pi)$. Since $\mathcal{C}$ is a convex Pareto curve, there exists a convex combination of paths $\pi' = \lambda'_1 \pi'_1 + \lambda'_2 \pi'_2$ in $\mathcal{C}$ such that $p_1(\pi) \leq p_1(\pi')$ and $p_2(\pi) \leq p_2(\pi')$. Now, consider all convex combinations of paths in $\mathcal{C}$; by convexity of the sum function, the maximum over this set is reached on some path π''_m, so $p_1(\pi') + p_2(\pi') \leq p_1(\pi''_m) + p_i(\pi''_m)$. It follows that $\mathcal{P}(w) \leq p_1(\pi''_m) + p_i(\pi''_m) \leq$ Output.

To conclude the proof of correctness, we show that Output $\leq \mathcal{P}(w)$ for some word w. Indeed, if π is a path such that Output $= p_1(\pi) + p_2(\pi)$, then thanks to Lemma 2 there exists a word w such that $p_1(\pi)+p_2(\pi) \leq \mathcal{P}(w)$. $\square$

We do not know whether there exist quasi-polynomial time algorithms for every $k > 2$, and leave this as an open problem.

Allender, E., Bürgisser, P., Kjeldgaard-Pedersen, J., Miltersen, P. B., 2009. On the complexity of numerical analysis. SIAM Journal on Computing 38 (5), 1987–2006.

Bertoni, A., 1974. The solution of problems relative to probabilistic automata in the frame of the formal languages theory. In: GI Jahrestagung. pp. 107–112.

Carstensen, P. J., 1983. The complexity of some problems in parametric linear and combinatorial programming. Ph.D. thesis, University of Michigan.

Chadha, R., Sistla, A. P., Viswanathan, M., 2017. Emptiness under isolation and the value problem for hierarchical probabilistic automata. In: FoSSaCS. pp. 231–247.

Chadha, R., Sistla, A. P., Viswanathan, M., 2018. Approximating probabilistic automata by regular languages. In: CSL. pp. 1–12.
URL `https://doi.org/10.4230/LIPIcs.CSL.2018.14`

Chatterjee, K., Tracol, M., 2012. Decidable problems for probabilistic automata on infinite words. In: LICS. pp. 185–194.

Daviaud, L., Jurdzinski, M., Lazic, R., Mazowiecki, F., Pérez, G. A., Worrell, J., 2018. When is containment decidable for probabilistic automata? In: ICALP. pp. 1–12.
URL `https://doi.org/10.4230/LIPIcs.ICALP.2018.121`

Diakonikolas, I., Yannakakis, M., 2008. Succinct approximate convex pareto curves. In: SODA. pp. 74–83.

Fijalkow, N., 2017. Undecidability results for probabilistic automata. SIGLOG News 4 (4), 10–17.
URL `http://doi.acm.org/10.1145/3157831.3157833`

Fijalkow, N., Gimbert, H., Kelmendi, E., Oualhadj, Y., 2015. Deciding the value 1 problem for probabilistic leaktight automata. Logical Methods in Computer Science 11 (2).

Fijalkow, N., Gimbert, H., Oualhadj, Y., 2012. Deciding the value 1 problem for probabilistic leaktight automata. In: LICS. pp. 295–304.

Fijalkow, N., Riveros, C., Worrell, J., 2017. Probabilistic automata of bounded ambiguity. In: CONCUR. pp. 19:1–19:14.
URL `https://doi.org/10.4230/LIPIcs.CONCUR.2017.19`

Gimbert, H., Oualhadj, Y., 2010. Probabilistic automata on finite words: Decidable and undecidable problems. In: ICALP (2). pp. 527–538.

Gusfield, D. M., 1980. Sensitivity analysis for combinatorial optimization. Ph.D. thesis, University of California, Berkeley.

Kozen, D., 1977. Lower bounds for natural proof systems. In: FOCS. pp. 254–266.

Lazić, R., Schmitz, S., 2015. The ideal view on Rackoffs coverability technique. In: International Workshop on Reachability Problems. Springer, pp. 76–88.

Lyngsø, R. B., Pedersen, C. N. S., 2002. The consensus string problem and the complexity of comparing hidden Markov models. Journal of Computer and System Sciences 65 (3), 545–569.

Papadimitriou, C. H., Yannakakis, M., 2000. On the approximability of trade-offs and optimal access of web sources. In: FOCS. pp. 86–92.

Paz, A., 1971. Introduction to Probabilistic Automata. Academic Press.

Rabin, M. O., 1963. Probabilistic automata. Information and Control 6 (3), 230–245.

Ruohonen, K., 1985. Reversible machines and Post's correspondence problem for biprefix morphisms. Elektronische Informationsverarbeitung und Kybernetik 21 (12), 579–595.

Schützenberger, M. P., 1961. On the definition of a family of automata. Information and Control 4 (2-3), 245–270.
URL https://doi.org/10.1016/S0019-9958(61)80020-X

Weber, A., Seidl, H., 1991. On the degree of ambiguity of finite automata. Theoretical Computer Science 88 (2), 325–349.

Zuckerman, D., 2007. Linear degree extractors and the inapproximability of max clique and chromatic number. Theory of Computing 3 (1), 103–128.