On the Uniform Random Generation of Non deterministic Automata up to Isomorphism

Pierre-Cyrille Héam Jean-Luc Joly

June 24, 2015

FEMTO-ST, CNRS UMR 6174, Université de Franche-Comté, INRIA
16 route de Gray, 25030 Besançon Cedex, France

Abstract

In this paper we address the problem of the uniform random generation of non deterministic automata (NFA) up to isomorphism. First, we show how to use a Monte-Carlo approach to uniformly sample a NFA. Secondly, we show how to use the Metropolis-Hastings Algorithm to uniformly generate NFAs up to isomorphism. Using labeling techniques, we show that in practice it is possible to move into the modified Markov Chain efficiently, allowing the random generation of NFAs up to isomorphism with dozens of states. This general approach is also applied to several interesting subclasses of NFAs (up to isomorphism), such as NFAs having a unique initial states and a bounded output degree. Finally, we prove that for these interesting subclasses of NFAs, moving into the Metropolis Markov chain can be done in polynomial time. Promising experimental results constitute a practical contribution.

1 Introduction

Finite automata play a central role in the field of formal language theory and are intensively used to address algorithmic problems from model-checking to text processing. Many automata based algorithms have been developed and are still being developed, proposing new approaches and heuristics, even for basic problems like the inclusion problem[1]. Evaluating new algorithms is a challenging problem that cannot be addressed only by the theoretical computation of the worst case complexity. Several other complementary techniques can be used to measure the efficiency of an algorithm: average complexity, generic case complexity, benchmarking, evaluation on hard instances, evaluations on random instances. The first two approaches are hard theoretical problems, particularly for algorithms using heuristics and optimizations. Benchmarks, as well as known hard instances, are not always available. Nevertheless, in practice, random generation of inputs is a good way to estimate the efficiency of an algorithm. Designing uniform random generator for classes of finite automata is a challenging problem that has been addressed mostly for deterministic automata [CP05, BN07, AMR07, CN12] -the interested reader is referred to [Nic14] for a recent survey. However, the problem of uniform random generation of non deterministic automata (NFAs) is more complex, particularly for a random generation up to isomorphism: the size of the automorphism group of a n-state non deterministic automata may vary from 1 to $n!$. For most applications, the complexity of the algorithm is related to the structure of the automata, not to the names of the states: randomly generated NFAs, regardless of the number of isomorphic automata, may therefore lead to an over representation of some isomorphism classes of automata. Moreover, as discussed in the conclusion of [Nic14], the random generation of non deterministic automata has to be done on particular subclasses of automata in order to obtain a better sampler for the evaluation of algorithm (since most of the NFAs, for the uniform distribution, will accept all words).

In this paper we address the problem of the uniform generation of some classes of non deterministic automata (up to isomorphism) by using Monte-Carlo techniques. We propose this approach

[1]see `http://www.languageinclusion.org/`

for the class of n-state non deterministic automata as well as for (a priori) more interesting subclasses. Determining the most interesting subclasses of NFAs for testing practical applications is not the purpose of this paper. We would like to point out that Monte Carlo approaches are very flexible and can be applied quite easily for many classes of NFAs. More precisely:

1. We propose in Section 2 several ergodic Markov Chains whose stationary distributions are respectively uniform on the set of n-state NFAs, n-state NFAs with a fixed maximal output degree and n-state NFAs with a fixed maximal output degree for each letter. In addition, these tools can be adapted for these three classes including automata with a fixed single initial state. Moving into these Markov chains can be done in time polynomial in n.

2. The main idea of this paper is exposed in Section 3.1, where we show how to modify these Markov Chains using the Metropolis-Hastings Algorithm in order to obtain stationary distributions that are uniform for the given classes of automata but up to isomorphism. Moving into these new Markov chains requires computing the sizes of the automorphism group of the occurring NFAs.

3. The main contributions of this paper are given in Section 3. We show in Section 3.2 that, for the classes with a bounded output degree, moving into the modified Markov chains can be done in polynomial time. In Section 3.4 we explain how to use labeling techniques to do it efficiently in practice for all NFAs. Promising experiments are described in Section 3.5.

The random generation of non deterministic automata is explored in [TV05] using random graph techniques (without considering the obtained distribution relative to automata or to the isomorphism classes). In [CHPZ02], the random generation of NFAs is performed using bitstream generation. In [Nic09, NPR10] NFAs are obtained by the random generation of a regular expression and by transforming it into an equivalent automaton using Glushkov Algorithm. The use of Markov chains based techniques to randomly generate finite automata was introduced in [CF11, CF12] for acyclic automata.

1.1 Theoretical Background on NFA

For a general reference on finite automata see [HU79]. In this paper Σ is a fixed finite alphabet of cardinal $|\Sigma| \geq 2$, and m is an integer satisfying $m \geq 2$.

A *non-deterministic automaton* (NFA) on Σ is a tuple (Q, Δ, I, F) where Q is a finite set of *states*, Σ is a finite alphabet, $\Delta \subset Q \times \Sigma \times Q$ is the set of transitions, $F \subset Q$ is the set of final states and $I \subseteq Q$ is the set of initial states. For any state p and any letter a, we denote by $p \cdot a$ the set of states q such that $(p, a, q) \in \Delta$. The set of transitions Δ is *deterministic* if for every pair (p, a) in $Q \times \Sigma$ there is at most one $q \in Q$ such that $(p, a, q) \in \Delta$. Two NFAs are depicted on Fig. 2. A NFA is *complete* if for every pair (p, a) in $Q \times \Sigma$ there is at least one $q \in Q$ such that $(p, a, q) \in \Delta$. A *path* in a NFA is a sequence of transitions $(p_0, a_0, q_0)(p_1, a_1, q_1) \ldots (p_k, a_k, q_k)$ such that $q_i = p_{i+1}$. The word $a_0 \ldots a_k$ is the *label* of the path and k its *length.* If $p_0 \in I$ and $q_k \in F$ the path is successful. A word is *accepted* by a NFA if it's the label of a successful path. A NFA is *accessible* (resp. *co-accessible*) if for every state q there exists a path from an initial state to q (resp. if for every state q there exists a path from q to a final state). A NFA is *trim* if it is both accessible and co-accessible A *deterministic automaton* is a NFA where $|I| = 1$ and whose set of transitions is deterministic.

Let $\mathfrak{A}(n)$ be the class of finite automata whose set of states is $\{0, \ldots, n-1\}$. Let $\mathfrak{N}(n)$ be the subclass of $\mathfrak{A}(n)$ of trim finite automata. Let $\mathfrak{N}_m(n)$ be the class of finite automata in $\mathfrak{N}(n)$ such that, for each state p, there is at most m pairs (a, q) such that (p, a, q) is a transition. Let $\mathfrak{N}'_m(n)$ be the class of finite automata in $\mathfrak{N}_m(n)$ such that, for each state p and each letter a, there is at most m states q such that (p, a, q) is a transition. For any class $\mathfrak{X}$ of finite automata, we denote by $\mathfrak{X}^\bullet$ the subclass of $\mathfrak{X}$ of automata whose set of initial states is reduced to $\{1\}$. One has

$$\mathfrak{N}_m(n) \subseteq \mathfrak{N}'_m(n) \subseteq \mathfrak{N}(n) \subseteq \mathfrak{A}(n).$$

Two NFAs are *isomorphic* if there exists a bijection between their sets of states preserving the sets initial states, final states and transitions. More precisely, let $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$

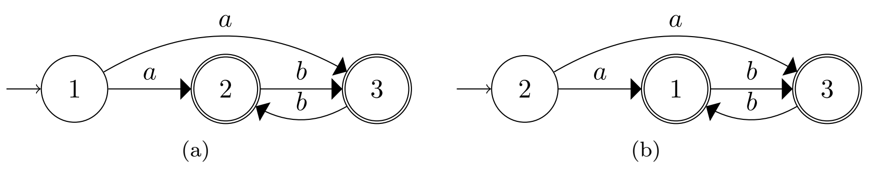

Figure 1: Two Isomorphic Automata

and let φ be a bijection from Q into a finite set $\varphi(Q)$. We denote by $\varphi(\mathcal{A})$ the automaton $(\varphi(Q), \Sigma, \Delta', \varphi(I), \varphi(F))$, with $\Delta' = \{(\varphi(p), a, \varphi(q)) \mid (p, a, q) \in \Delta\}$. Two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are isomorphic if there exists a bijection φ such that $\varphi(\mathcal{A}_1) = \mathcal{A}_2$.

Two isomorphic NFAs have the same number of states and are equal, up to the states names. The relation *is isomorphic to* is an equivalence relation. For instance, the two automata depicted on Fig. 2 are isomorphic, with $\varphi(1) = 2$, $\varphi(2) = 1$ and $\varphi(3) = 3$. An *automorphism* for a NFA is an isomorphism between this NFA and itself. Given a NFA $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$, the set of automorphisms of $\mathcal{A}$ is a finite group denoted $\mathrm{Aut}(\mathcal{A})$. For $Q' \subseteq Q$, $\mathrm{Aut}_{Q'}(\mathcal{A})$ denotes the subset of $\mathrm{Aut}(\mathcal{A})$ of automorphisms ϕ fixing each element of Q': for each $q \in Q'$, $\phi(q) = q$. Particularly $\mathrm{Aut}_{\emptyset}(\mathcal{A}) = \mathrm{Aut}(\mathcal{A})$, and $\mathrm{Aut}_Q(\mathcal{A})$ is reduce to the identity. For instance, the automorphism group of the automaton depicted on Fig. 2(a) has two elements, the identity and the isomorphism switching 2 and 3.

The size of the automorphism group of a non deterministic n-state automaton may vary from 1 to $n!$. For instance, any deterministic trim automaton whose states are all final has an automorphism group reduce to the identity. The non deterministic n-state automaton with no transition and where all states are both initial and final has for automorphism group the symmetric group.

The isomorphism problem consists in deciding whether two finite automata are isomorphic. It is investigated for deterministic automata in [Boo78]. It is naturally closed to the same problem for directed graph and the following result [Luk82] will be useful in this paper.

Theorem 1 *Let m be a fixed positive integer. The isomorphism problem for directed graphs with degree bounded by m is polynomial.*

1.2 Theoretical Background on Markov Chains

For a general reference on Markov Chains see [DLW08]. Basic probability notions will not be defined in this paper. The reader is referred for instance to [MU05].

Let Ω be a finite set. A *Markov chain* on Ω is a sequence $X_0, \ldots, X_t, \ldots$ of random variables on Ω such that $\mathbb{P}(X_{t+1} = x_{t+1} \mid X_t = x_t) = \mathbb{P}(X_{t+1} = x_{t+1} \mid X_t = x_t, \ldots, X_i = x_i, \ldots, X_0 = x_0)$, for all $x_i \in \Omega$. A Markov chain is defined by its *transition matrix* M, which is a function from $\Omega \times \Omega$ into $[0, 1]$ satisfying $M(x, y) = \mathbb{P}(X_{t+1} = y \mid X_t = x)$. The underlying graph of a Markov chain is the graph whose set of vertices is Ω and there is an edge from x to y if $M(x, y) \neq 0$. A Markov chain is *irreducible* if its underlying graph is strongly connected. It is *aperiodic* if for all node x, the gcd of the lengths of all cycles visiting x is 1. Particularly, if for each x, $M(x, x) \neq 0$, the Markov chain is aperiodic. A Markov chain is *ergodic* if it is irreducible and aperiodic. A Markov chain is symmetric if $M(x, y) = M(y, x)$ for all $x, y \in \Omega$. A distribution π on Ω is a stationary distribution for the Markov Chain if $\pi M = \pi$. It is known that an ergodic Markov chain has a unique stationary distribution [DLW08, Chapter 1]. Moreover, if the chain is symmetric, this distribution is the uniform distribution on Ω.

Given an ergodic Markov chain $X_0, \ldots, X_t, \ldots$ with stationary distribution π, it is known that, whatever is the value of X_0, the distribution of X_t converges to π when $t \to +\infty$: $\max \| M^t(x, \cdot) - \pi \|_{TV} \underset{t \to +\infty}{\to} 0$, where $\| \|_{TV}$ designates the total variation distance between two distributions [DLW08, Chapter 4]. This leads to the Monte-Carlo technique to randomly generate elements of Ω according to the distribution π by choosing arbitrarily X_0, computing $X_1, X_2, \ldots$, and returning X_t for t large enough. The convergence rate is known to be exponential, but computing the constants is a very difficult problem: choosing the step t to stop is a challenging question depending both on how close to π we want to be and on the convergence rate of $M^t(x, \cdot)$ to π. For this purpose, the

ε-mixing time of an ergodic Markov chain of matrix M and stationary distribution π is defined by $t_{\text{mix}}(\varepsilon) = \min\{t \mid \max_{x\in\Omega}\|P_t(x,\cdot) - \pi\|_{TV} \leq \varepsilon\}$. Computing mixing time bounds is a central question on Markov Chains.

The Metropolis-Hasting Algorithm is based on the Monte-Carlo technique and aims at modifying the transition matrix of the Markov chain in order to obtain a particular stationary distribution [DLW08, Chapter 3]. Suppose that M is an ergodic symmetric transition matrix of a symmetric Markov chain on Ω and ν is a distribution on Ω. The transition matrix P_ν for ν is defined by:

$$P_\nu(x,y) = \begin{cases} \min\left\{1, \frac{\nu(y)}{\nu(x)}\right\} M(x,y) & \text{if } x \neq y, \\ 1 - \sum_{z\neq x} \min\left\{1, \frac{\nu(z)}{\nu(x)}\right\} M(x,z) & \text{if } x = y. \end{cases}$$

The chain defined by P_ν is called *the Metropolis Chain for ν*. It is known [DLW08, Chapter 3] that it is an ergodic Markov chain whose stationary distribution is ν.

2 Random Generation of Non Deterministic Automata using Markov Chain

In this section, we propose families of symmetric ergodic Markov chains on $\mathfrak{A}(n)$, $\mathfrak{N}(n)$, $\mathfrak{N}_m(n)$ and $\mathfrak{N}'_m(n)$, as well as on the respective corresponding doted classes of NFAs.

Let $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ be a finite automaton. For any q in Q and any (p, a, q) in $Q \times \Sigma \times Q$, the automata $\mathsf{Ch}_{\mathsf{init}}(\mathcal{A}, q)$, $\mathsf{Ch}_{\mathsf{final}}(\mathcal{A}, q)$ and $\mathsf{Ch}_{\mathsf{trans.}}(\mathcal{A}, (p, a, q))$ are defined as follows:

- If $q \in I$, then $\mathsf{Ch}_{\mathsf{init}}(\mathcal{A}, q) = (Q, \Sigma, \Delta, I \setminus \{q\}, F)$ and $\mathsf{Ch}_{\mathsf{init}}(\mathcal{A}, q) = (Q, \Sigma, \Delta, I \cup \{q\}, F)$ otherwise.
- If $q \in F$, then $\mathsf{Ch}_{\mathsf{final}}(\mathcal{A}, q) = (Q, \Sigma, \Delta, I, F \setminus \{q\})$, and $\mathsf{Ch}_{\mathsf{final}}(\mathcal{A}, q) = (Q, \Sigma, \Delta, I, F \cup \{q\})$ otherwise.
- If $(p, a, q) \in \Delta$, then $\mathsf{Ch}_{\mathsf{trans.}}(\mathcal{A}, (p, a, q)) = (Q, \Sigma, \Delta\setminus\{(p, a, q)\}, I, F)$, and $\mathsf{Ch}_{\mathsf{trans.}}(\mathcal{A}, (p, a, q)) = (Q, \Sigma, \Delta \cup \{(p, a, q)\}, I, F)$ otherwise.

Let ρ_1, ρ_2, ρ_3 be three real numbers satisfying $0 \leq \rho_i \leq 1$ and $\rho_1 + \rho_2 + \rho_3 \leq 1$. Let $\mathfrak{X}$ be a class of automata whose set of states is Q. We define the transition matrix $S^{\mathfrak{X}}_{\rho_1,\rho_2,\rho_3}(x, y)$ on $\mathfrak{X}$ by:

- If there exists q such that $y = \mathsf{Ch}_{\mathsf{init}}(x, q)$, then $S^{\mathfrak{X}}_{\rho_1,\rho_2,\rho_3}(x, y) = \frac{\rho_1}{|Q|}$.
- If there exists q such that $y = \mathsf{Ch}_{\mathsf{final}}(x, q)$, then $S^{\mathfrak{X}}_{\rho_1,\rho_2,\rho_3}(x, y) = \frac{\rho_2}{|Q|}$.
- If there exists $(p, a, q) \in Q \times \Sigma \times Q$ such that $y = \mathsf{Ch}_{\mathsf{trans.}}(x, q)$, then $S^{\mathfrak{X}}_{\rho_1,\rho_2,\rho_3}(x, y) = \frac{\rho_3}{|\Sigma|.|Q|^2}$.
- If y is different of x and has not one of the above forms, $S^{\mathfrak{X}}_{\rho_1,\rho_2,\rho_3}(x, y) = 0$.
- $S^{\mathfrak{X}}_{\rho_1,\rho_2,\rho_3}(x, x) = 1 - \sum_{y\neq x} S^{\mathfrak{X}}_{\rho_1,\rho_2,\rho_3}(x, y)$.

Now for $\mathfrak{X} \in \{\mathfrak{N}(n), \mathfrak{N}_m(n), \mathfrak{N}'_m(n)\}$, and $0 < \rho < 1$ we define the transition matrix $S^{\mathfrak{X}^\bullet}_\rho$ on $\mathfrak{X}^\bullet$ by $S^{\mathfrak{X}^\bullet}_\rho = S^{\mathfrak{X}}_{0,\rho,1-\rho}$.

Lemma 2 *Let m, n be fixed positive integers, with $m \geq 2$. If $1 > \rho > 0$, $\rho_1 > 0$, $\rho_2 > 0$ and $\rho_3 > 0$, then $S^{\mathfrak{N}(n)}_{\rho_1,\rho_2,\rho_3}$, $S^{\mathfrak{N}_m(n)}_{\rho_1,\rho_2,\rho_3}$ and $S^{\mathfrak{N}'_m(n)}_{\rho_1,\rho_2,\rho_3}$ are irreducible, as well as $S^{\mathfrak{N}(n)^\bullet}_\rho$, $S^{\mathfrak{N}_m(n)^\bullet}_\rho$ and $S^{\mathfrak{N}'_m(n)^\bullet}_\rho$.*

PROOF. Without loss of generality, we assume that $Q = \{1, \ldots, n\}$. Let $\mathfrak{X} \in \{\mathfrak{N}(n), \mathfrak{N}_m(n), \mathfrak{N}'_m(n)\}$ and $x \in \mathfrak{X}$. We denote by $\mathcal{A}_0$ the automaton $(Q, \Sigma, \emptyset, Q, Q)$. The automaton $\mathcal{A}_0$ is trim and is in $\mathfrak{X}$. We prove there is a path in $\mathfrak{X}$ from x to $\mathcal{A}_0$. Set $x = (Q, \Sigma, \Delta, I, F)$. Since adding initial or final states to x provides automata that are still in $\mathfrak{X}$, there is a path from x to $y = (Q, \Sigma, \Delta, Q, Q)$ (using $\mathsf{Ch}_{\mathsf{init}}$ and $\mathsf{Ch}_{\mathsf{final}}$). Now, since all states are both initial and final, there is a path from y to $\mathcal{A}_0$ (by deleting all transitions). It follows there is a path in $\mathfrak{X}$ from x to $\mathcal{A}_0$. Since the graph of the Markov chain is symmetric, there is also a path from $\mathcal{A}_0$ to x. Consequently, the Markov chains are irreducible. The proof for $S^{\mathfrak{N}(n)^\bullet}_\rho$, $S^{\mathfrak{N}_m(n)^\bullet}_\rho$ and $S^{\mathfrak{N}'_m(n)^\bullet}_\rho$ are similar. $\square$

Lemma 3 *Let m, n be two fixed positive integers. If $1 > \rho > 0$, $\rho_1 > 0$, $\rho_2 > 0$ and $\rho_3 > 0$, then $S^{\mathfrak{N}(n)}_{\rho_1,\rho_2,\rho_3}$, $S^{\mathfrak{N}_m(n)}_{\rho_1,\rho_2,\rho_3}$ and $S^{\mathfrak{N}'_m(n)}_{\rho_1,\rho_2,\rho_3}$ are aperiodic, as well as $S^{\mathfrak{N}(n)^\bullet}_{\rho}$, $S^{\mathfrak{N}_m(n)^\bullet}_{\rho}$ and $S^{\mathfrak{N}'_m(n)^\bullet}_{\rho}$.*

Proof. With the notations of the proof of Lemma 2, there is a path of length n_x from any $x \in \mathfrak{X}$ to $\mathcal{A}_0$. Therefore there is a cycle of length $2n_x$ visiting x.

Now, $\mathsf{Ch}_{\mathsf{init}}(\mathcal{A}_0, 1) \notin \mathfrak{X}$ since 1 is not accessible in $\mathcal{A}_0$. It follows that $S^{\mathfrak{X}}(\mathcal{A}_0, \mathcal{A}_0) \neq 0$. Therefore, there is also a cycle of length $2n_x + 1$ visiting x. Since the gcd of $2n_x$ and $2n_x + 1$ is 1, the chain is aperiodic. The proof for $S^{\mathfrak{N}(n)^\bullet}_{\rho}$, $S^{\mathfrak{N}_m(n)^\bullet}_{\rho}$ and $S^{\mathfrak{N}'_m(n)^\bullet}_{\rho}$ are similar. □

Proposition 4 *Let m, n be two fixed positive integers with $m \geq 2$. The Markov chains with matrix $S^{\mathfrak{N}(n)}_{\rho_1,\rho_2,\rho_3}$, $S^{\mathfrak{N}_m(n)}_{\rho_1,\rho_2,\rho_3}$ and $S^{\mathfrak{N}'_m(n)}_{\rho_1,\rho_2,\rho_3}$ are ergodic and their stationary distributions are the uniform distributions.*

Proof. By lemma 3 and 2, the chain is ergodic. Since the matrix $S^{\mathfrak{N}(n)}_{\rho_1,\rho_2,\rho_3}$, $S^{\mathfrak{N}_m(n)}_{\rho_1,\rho_2,\rho_3}$ and $S^{\mathfrak{N}'_m(n)}_{\rho_1,\rho_2,\rho_3}$ are symmetric, their stationary distributions are the uniform distributions (over the respective family of automata). □

In practice, computing X_{t+1} from X_t is done in the following way: the first step consists in choosing with probabilities ρ_1, ρ_2 and ρ_3 whether we will change either an initial state, a final state or a transition. In a second step and in each case, all the possible changing operations are performed with the same probability. If the obtained automaton is in the corresponding class, X_{t+1} is set to this value. Otherwise, $X_{t+1} = X_t$. Since verifying that an automaton is in the desired class ($\mathfrak{N}(n)$, $\mathfrak{N}_m(n)$ or $\mathfrak{N}'_m(n)$), can be performed in time polynomial in n, computing X_{t+1} from X_t can be done in time polynomial in n.

We define the lazy Markov chain on $\mathfrak{A}(n)$ by $L^{\mathfrak{A}(n)}_{\rho_1,\rho_2,\rho_3}(x, y) = \frac{1}{2} S^{\mathfrak{A}(n)}_{\rho_1,\rho_2,\rho_3}(x, y)$ if $x \neq y$ and $L^{\mathfrak{A}(n)}_{\rho_1,\rho_2,\rho_3}(x, x) = \frac{1}{2} + \frac{1}{2} S^{\mathfrak{A}(n)}_{\rho_1,\rho_2,\rho_3}(x, x)$. It is known that a symmetric Markov chain and its associated lazy Markov chain have similar mixing times.

Proposition 5 *The ε-mixing time $\tau(\varepsilon)$ of $L^{\mathfrak{A}(n)}_{\rho_1,\rho_2,\rho_3}$ satisfies $\tau(\varepsilon) \leq (\frac{1}{\rho_1} + \frac{1}{\rho_2})(n \ln n + \lceil n \ln(\varepsilon^{-1}) \rceil) + \frac{2|\Sigma|^2 n^2}{\rho_3}\left(\ln(|\Sigma| n) + \lceil \ln(\varepsilon^{-1}) \rceil\right)$.*

It follows that $\tau(\varepsilon) = O(n^3)$ when $|\Sigma|$ is considered as a constant. At this stage, we are not able to compute bounds on the mixing times of the other Markov chains. Practical experiments, with various sizes of alphabets, seems to show that about 90% of the automata generated by the above lazy Markov Chain (using n^3 as mixing bound) are trim. This observation leads us to consider, for other experiments, to move n^3 steps to sample automata. Of course, this is not a proof, just an empirical estimation.

3 Random Generation of Non Deterministic Automata up to Isomorphism

In this section we show how to use the Metropolis-Hastings algorithm to uniformly generate NFAs up to isomorphism and that, for this purpose, it *suffices* to compute the sizes of the automorphism groups of involved NFAs. We prove in Section 3.2 that this computation is polynomially equivalent to testing the isomorphism problem for the involving automata. For the classes $\mathfrak{N}_m(n)$, $\mathfrak{N}_m(n)^\bullet$, $\mathfrak{N}'_m(n)$ and $\mathfrak{N}'_m(n)^\bullet$, we show that it can be done in time polynomial in n (if m is fixed). In Section 3.4 we show how to practically compute the sizes of automorphism group using labellings techniques. Finally, experimental results are given in Section 3.5.

3.1 Metropolis-Hastings Algorithm

For a class $\mathfrak{C}$ of NFAs (closed by isomorphism) and n a positive integer, let $\mathfrak{C}(n)$ be the elements of $\mathfrak{C}$ whose set of states is $\{1, \ldots, n\}$ and let γ_n be the number of isomorphism classes on $\mathfrak{C}(n)$. There are $n!$ possible bijections on $\{1, \ldots, n\}$. If $\mathcal{A} \in \mathfrak{C}(n)$, Let φ_1 and φ_2 be two bijections on $\{1, \ldots, n\}$. One has $\varphi_1(\mathcal{A}) = \varphi_2(\mathcal{A})$ iff $\varphi_2^{-1}\varphi_1(\mathcal{A}) = \mathcal{A}$, iff $\varphi_2^{-1}\varphi_1(\mathcal{A}) \in \mathrm{Aut}(\mathcal{A})$. It follows that the isomorphism classes of $\mathcal{A}$ (in $\mathfrak{C}(n)$) has $\frac{n!}{|\mathrm{Aut}(\mathcal{A})|}$ elements. This leads to the following result.

Proposition 6 *Randomly generates an element x of $\mathfrak{C}(n)$ with probability $\frac{n!}{\gamma_n|\mathrm{Aut}(x)|}$ provides a uniform random generator of the isomorphism classes of $\mathfrak{C}(n)$.*

Proof. Let H be an isomorphism class of $\mathfrak{C}(n)$; H is generated with probability

$$\sum_{x\in H}\frac{n!}{\gamma_n|\mathrm{Aut}(x)|}=\sum_{x\in H}\frac{n!}{\gamma_n|H|}=\frac{1}{\gamma_n|H|}\sum_{x\in H}1=\frac{|H|}{\gamma_n|H|}=\frac{1}{\gamma_n}.$$

□

In order to compute P_ν it is not necessary to compute γ_n, since $\frac{\nu(x)}{\nu(y)}=\frac{|\mathrm{Aut}(y)|}{|\mathrm{Aut}(x)|}$. A direct use of the Metropolis-Hastings algorithm requires to compute all the neighbors of x and the sizes of theirs automorphism groups to move from x. Since a n-state automaton has about $|\Sigma|n^2$ neighbors, it can be a quite huge computation for each move. However, practical evaluations show that in most cases the automorphism group of an automaton is quite small and, therefore, the rejection approach exposed in [CG95] is more tractable. It consists in moving from x to y using $S(x,y)$ (the non-modified chain) and to accept y with probability $\min\left\{1,\frac{\nu(y)}{\nu(x)}\right\}$. If it is not accepted, repeat the process (moving from x to y using S with probability $\min\left\{1,\frac{\nu(y)}{\nu(x)}\right\}$) until acceptance. In practice, we observe a very small number of rejects.

The problem of computing the size of the automorphism group of a NFA is investigated in the next session. Assuming it can be done in a reasonable time, an alternative solution to randomly generate NFAs up to isomorphism may be to use a rejection algorithm: randomly and uniformly generate a NFA $\mathcal{A}$ and keep it with probability $\frac{|\mathrm{Aut}(\mathcal{A})|}{n!}$. This way, each class of isomorphism is picked up with the same probability. However, as we will observe in the experiments, most of automata have a very small group of automorphisms, and the number of rejects will be intractable, even for quite small n's.

3.2 Counting Automorphisms

This section is dedicated to show how to compute $|\mathrm{Aut}(\mathcal{A})|$ by using a polynomial number of calls to the isomorphism problem. It is an adaptation of a corresponding result for directed graphs [Mat79].

Let $\mathcal{A}=(Q,\Sigma,\Delta,I,F)$ be a NFA and $Q'\subseteq Q$. Let σ be an arbitrary bijective function from Q' into $\{1,\ldots,|Q'|\}$, a_0 an arbitrary letter in Σ and $\ell=|Q|+|Q'|+2$. For each state $r\in Q\backslash Q'$ we denote by $\mathcal{A}_r^{Q'}$ the automaton $(Q_r,\Sigma,\Delta_r,I,F)$ where $Q_r=Q\cup\{(p,i)\mid p\in Q \text{ and } 1\leq i\leq\ell\}$, and $\Delta_r=\Delta\cup\{(p,a,(p,1)\mid p\in Q\}\cup\{((p,i),a_0,(p,i+1))\mid p\in Q' \text{ and } 1\leq i<|Q|+1+\sigma(p)\}\cup\{((r,i),a_0,(r,i+1))\mid 1\leq\ i\leq\ell\}\cup\{((p,i),a_0,(p,i+1))\mid p\notin Q'\cup\{r\} \text{ and } 1<i\leq|Q|+1\}$. Note that the size of $\mathcal{A}_r^{Q'}$ is polynomial in the size of $\mathcal{A}$.

The two next lemma show how to polynomially reduce the problem of counting automorphisms to the isomorphism problem.

Lemma 7 *Let $\mathcal{A}=(Q,\Sigma,\Delta,I,F)$ be a NFA and Q' a non-empty subset of Q. For every $q,q'\in Q\backslash Q'$, there exists $\phi\in\mathrm{Aut}_{Q'}(\mathcal{A})$ such that $\phi(q)=q'$ iff $\mathcal{A}_q^{Q'}$ and $\mathcal{A}_{q'}^{Q'}$ are isomorphic.*

Lemma 8 *Let $\mathcal{A}=(Q,\Sigma,\Delta,I,F)$ be a NFA and Q' a non-empty subset of Q. For every $q\in Q'$, there exists an integer d such that $|\mathrm{Aut}_{Q'\backslash\{q\}}(\mathcal{A})|=d|\mathrm{Aut}_{Q'}(\mathcal{A})|$. Moreover d can be computed with a polynomial number of isomorphism tests between automata of the form $\mathcal{A}_r^{Q'\backslash\{q\}}$.*

Lemma 8 provides a way to compute sizes of automorphism groups by testing whether two NFAs are isomorphic. Indeed, since $\mathrm{Aut}_Q(\mathcal{A})$ is reduced to the identity, and since $\mathrm{Aut}(\mathcal{A})=\mathrm{Aut}_\emptyset(\mathcal{A})$, one has, by a direct induction using Lemma 8, $\mathrm{Aut}(\mathcal{A})=d_1\ldots d_{|Q|}$, where each d_i can be computed by a polynomial number of isomorphism tests. Therefore, the problem of counting automorphism reduces to test whether two automata are isomorphic.

3.3 Isomorphism Problem for Automata with a Bounded Degree

It is proved (not explicitly) in [Boo78] that the isomorphism problem for deterministic automata is polynomially equivalent to the isomorphism problem for directed finite graphs. We prove (Theorem 9) a similar result for NFAs, by using an encoding preserving some bounds on the output degree. Therefore, combining Theorem 9 and Lemma 8, it is possible to compute the size of the automorphism group of an automaton in $\mathfrak{N}_m(n)$, $\mathfrak{N}'_m(n)$, $\mathfrak{N}_m(n)^\bullet$ and $\mathfrak{N}'_m(n)^\bullet$ in time polynomial in n (assuming that m is a constant).

Theorem 9 *Let m be a fixed integer. The isomorphism problem for automata in $\mathfrak{N}_m$ $\mathfrak{N}'_m$, $\mathfrak{N}_m(n)^\bullet$ and $\mathfrak{N}'_m(n)^\bullet$ can be solved in polynomial time.*

Note that the proof is constructive but the exponents are too huge to provide an efficient algorithm. It will be possible to work on a finer encoding but we prefer, in practice, to use labeling techniques described in the next section and that are practically very efficient on graphs (see [Gal14] for a recent survey).

3.4 Practical Computation using Labelings

For testing graph isomorphism, the most efficient currently used approach is based on labeling [Gal14] and it works practically for large graphs. Intuitively, if two n-state automata are isomorphic, then they have the same number of initial states and of final states. Rather than testing potential $n!$ possible bijections from the automata to point out an isomorphism, it suffices to test $n_1! + n_2! + n_3! + n_4!$ where n_1 is the number of states that are both initial and final, n_2 the number of final states (that are not initial), n_3 the number of initial states (that are not final), and n_4 is the number of states that are neither initial, nor final. With an optimal distribution, the number of tests falls from $n!$ to $4(n/4)!$. This idea can be generalized by the notion of labeling; the goal is to point out easily computable criteria that are stable by isomorphism to get a partition of the set of states and to reduce the search. The approach can be directly adapted for finite automata. A *labeling* is a computable function τ from $\mathfrak{N}(n) \times \{1, \ldots, n\}$ into a finite set D, such that for $\mathcal{A}_1 = (Q, \Sigma, E_1, I_1, F_1)$ and $\mathcal{A}_2 = (Q, \Sigma, E_2, I_2, F_2)$, if φ is an isomorphism from $\mathcal{A}_1$ to $\mathcal{A}_2$, then, for every $i \in \{1, \ldots, n\}$, $\tau(\mathcal{A}_1, i) = \tau(\mathcal{A}_2, \varphi(i))$. The algorithm consists in looking for functions φ preserving τ. If there exists $\alpha \in D$ such that $|\{i \mid \tau(\mathcal{A}_1, i) = \alpha\}| \neq |\{i \mid \tau(\mathcal{A}_2, i) = \alpha\}|$, then the two automata are not isomorphic. Otherwise, all possible bijections preserving the labeling are tested. In the worst case, there are $n!$ possibilities (the labeling doesn't provide any refinement), but in practice, it works very well. The algorithm is depicted in Figure 2. Note that if τ_1 and τ_2 are two labellings, then $\tau = (\tau_1, \tau_2)$ is a labeling to, allowing the combination of labeling. In our work, we use the following labellings: the labeling testing whether a state is initial, the one testing whether a state is final, the one testing whether a state is both initial and final, the one returning, for each letter a, the number of outgoing transitions labeled by a, the similar one with ongoing transitions, the one returning the minimal word (in the lexical order) from the state to a final state and the one returning the minimal word (in the lexical order) from an initial state to the given state. Using these labellings the practical computation of the sizes of automorphism groups can be done quite efficiently.

3.5 Experiments

The experiments have been done on a personal computer with processor `IntelCore i3-4150 CPU 3.50GHz x 4`, `7,7 Gio` of memory and running on a 64 bits Ubuntu 14.04 OS. The implementation is a non optimized prototype written in Python.

The first experimentation consists in measuring the time required to move into the Metropolis chains for $\mathfrak{N}(n)$ and $\mathfrak{N}_m(m)$. Results are reported in Table 1. The labellings used are those described in Section 3.4. These preliminary results show that using a 2 or 3-letter alphabet does not seem to have a significant influence. For each generation, the n^3-th elements of the walk is returned, with an arbitrary start. Moreover, bounding or not the degree does not seem to be relevant for the computation time. Note that we do not use any optimization: several computations on labellings may be reused when moving into the chain. Moreover, Python is not an efficient programming language (compared to C or Java). In practice, for directed graphs, the isomorphism problem is tractable for large graphs (see for instance [FPSV09]). Note that the number of moves (n^3) is the major factor for the increasing computation time (relatively to n): the average time for moving a single step is multiplied by about (only) 10 from $n = 20$ to $n = 90$.

```
Input: $\mathcal{A}_1, \mathcal{A}_2$ in $\mathfrak{N}(n)$, $\tau$ a labelling.
Output: 1 if $\mathcal{A}_1$ and $\mathcal{A}_2$ are isomorphic, 0 otherwise.
    $D := \{d_1, \ldots, d_k\}$ is the image of $\tau$.
    For $\alpha$ in $D$
        $D^1_\alpha := \{i \mid \tau(\mathcal{A}_1, i) = \alpha\}$
        $D^2_\alpha := \{i \mid \tau(\mathcal{A}_2, i) = \alpha\}$
        If $|D^1_\alpha| \neq |D^2_\alpha|$
            Then Return 0
        EndIf
    EndFor
    For $\varphi_{d_1}$ in the set of bijections from $D^1_{d_1}$ into $D^2_{d_1}$
      ...
        For $\varphi_{d_k}$ in the set of bijections from $D^1_{d_k}$ into $D^2_{d_k}$
            $\varphi$ is the bijection such that $\varphi_{|D^1_j} = \varphi_{d_j}$
            If $\varphi(\mathcal{A}_1) = \varphi(\mathcal{A}_2)$, Then Return 1
        EndFor
      ...
    EndFor
    Return 0
```

Figure 2: Testing isomorphism using labellings

n	10	20	50	70	90
$\lvert A\rvert = 2$	0.02	0.43	32.5	166.1	569.9
$\lvert A\rvert = 3$	0.02	0.56	47.1	248.4	848.1

n	10	20	50	70	90
$m = 2, \lvert A\rvert = 2$	0.2	0.43	32.5	166.1	566.8
$m = 2, \lvert A\rvert = 3$	0.2	0.57	47.0	246.7	847.2
$m = 3, \lvert A\rvert = 2$	0.2	0.43	33.0	167.8	561.9
$m = 3, \lvert A\rvert = 3$	0.2	0.57	47.2	248.6	851.3

Table 1: Average Time (s) to Sample a NFA in $\mathfrak{N}(n)$ (left) and in $\mathfrak{N}'_m(n)$ (right).

$\sigma = 1.5$, $n =$	5	8	11	14	17	20
s	1.5	4.3	4.7	3.8	3.1	2.7
$\sigma = 2$, $n =$	5	8	11	14	17	20
s	1.3	3.0	4.8	5.1	4.5	4.0
$\sigma = 3$, $n =$	5	8	11	14	17	20
s	2.8	4.8	4.7	3.8	3.4	3.0

$\mathfrak{N}'_2(n)^\bullet$, $n =$	5	8	11	14	17	20
s	3.7	6.1	7.9	10.0	11.5	13.9

Table 2: Average sizes of deterministic and minimal automata corresponding to automata sampling using [TV05] and in $\mathfrak{N}'_2(n)^\bullet$.

In the second experience, we also generate automata with n states by returning the n^3-th element of the walk in the Metropolis Chain. We use the algorithm to estimate the sizes of the automorphism group. By generating 1000 automata on a 2-letter alphabet, with 5, 7, 10, 15, 20 and 50 states, for each case, all the automata have a trivial automorphism group but one or two automata that have an automorphism group of size 2.

For the last experience, we propose to compare our generation for $\mathfrak{N}'_2(n)^\bullet$ with the generator proposed in [TV05] with a density of a-transitions of 2 and 3. The parameter of the algorithm is a probability p_f for final states and a density σ on a-transitions: the set of states of the automaton is $\{1, \ldots, n\}$, only 1 is the initial state, each state is final with a probability p_f and for each p and each a, (p, a, q) is a transition with a probability $\frac{\sigma}{n}$. Therefore for each state and each letter, the expected number of outgoing transitions labeled by this letter is σ. We run this algorithm with $p_f = 0.2$ and $\sigma \in \{1.5, 2, 3\}$. For each size, we compute the average size s of the corresponding minimal automata. We use a two letter alphabet and the average sizes (number of states) are obtained by sampling 1000 automata for each case. Results are reported in Table 2.

One can observe that the generator provides quite different automata. With the Markov chain approach the sizes of the related minimal automata are greater, even if there is no blow-up in both cases.

4 Conclusion

In this paper we proposed a Markov Chain approach to randomly generate non deterministic automata (up to isomorphism) for several classes of NFAs. We showed that moving into these Markov chains can be done quite quickly in practice and, in some interesting cases, in polynomial time. Experiments have been performed whithin a non optimized prototype and, following known experimental results on group isomorphism, they allow us to think that the approach can be used on much larger automata. Implementing such techniques using an efficient programming language is a challenging perspective. Moreover, the proposed approach is very flexible and can be applied to various classes of NFAs. An interesting research direction is to design particular subclasses of NFAs that look like NFAs occurring in practical applications, even if this last notion is hard to define. We think that the classes $\mathfrak{N}'_m(n)^\bullet$ and $\mathfrak{N}_m(n)^\bullet$ constitute first attempts in this direction. Theoretically -as often for Monte-Carlo approach-, computing mixing and strong stationary times are crucial and difficult questions we plan to investigate more deeply.

References

[AMR07] Marco Almeida, Nelma Moreira, and Rogério Reis. Enumeration and generation with a string automata representation. *Theor. Comput. Sci.*, 387(2):93–102, 2007.

[BN07] F. Bassino and C. Nicaud. Enumeration and random generation of accessible automata. *Theor. Comput. Sci.*, 381(1-3):86–104, 2007.

[Boo78] Kellogg S. Booth. Isomorphism testing for graphs, semigroups, and finite automata are polynomially equivalent problems. *SIAM J. Comput.*, 7(3):273–279, 1978.

[CF11] V. Carnino and S. De Felice. Random generation of deterministic acyclic automata using markov chains. In *CIAA 2011*, volume 6807 of *Lecture Notes in Computer Science*, pages 65–75, 2011.

[CF12] Vincent Carnino and Sven De Felice. Sampling different kinds of acyclic automata using markov chains. *Theor. Comput. Sci.*, 450:31–42, 2012.

[CG95] S. Chib and E. Greenberg. Understanding the metropolis-hastings al- gorithm. *American Statistician*, 49:327–335, 1995.

[CHPZ02] Jean-Marc Champarnaud, Georges Hansel, Thomas Paranthoën, and Djelloul Ziadi. Nfas bitstream-based random generation. In *Fourth International Workshop on Descriptional Complexity of Formal Systems - DCFS 2002*, pages 81–94, 2002.

[CN12] A. Carayol and C. Nicaud. Distribution of the number of accessible states in a random deterministic automaton. In *STACS 2012*, volume 14 of *LIPIcs*, pages 194–205, 2012.

[CP05] J.-M. Champarnaud and Th. Paranthoën. Random generation of dfas. *Theor. Comput. Sci.*, 330(2):221–235, 2005.

[DLW08] Yuval Peres D.A. Levin and Elizabeth L. Wilmer. *Markov Chain and Mixing Times.* American Mathematical Society, 2008. `http://pages.uoregon.edu/dlevin/MARKOV/markovmixing.pdf`.

[FPSV09] Pasquale Foggia, Gennaro Percannella, Carlo Sansone, and Mario Vento. Benchmarking graph-based clustering algorithms. *Image Vision Comput.*, 27(7):979–988, 2009.

[Gal14] Joseph A. Gallian. A dynamic survey of graph labeling. *The Electronic Journal of Combinatorics*, 17, 2014.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[Luk82] Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *J. Comput. Syst. Sci.*, 25(1):42–65, 1982.

[Mat79] Rudolf Mathon. A note on the graph isomorphism counting problem. *Inf. Process. Lett.*, 8(3):131–132, 1979.

[MU05] M. Mitzenmacher and Eli Upfal. *Probability and Computing.* Cambridge University Press, 2005.

[Nic09] Cyril Nicaud. On the average size of glushkov's automata. In *Language and Automata Theory and Applications, Third International Conference, LATA 2009,*, Lecture Notes in Computer Science, pages 626–637, 2009.

[Nic14] Cyril Nicaud. Random deterministic automata. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *MFCS'14*, volume 8634 of *Lecture Notes in Computer Science*, pages 5–23. Springer, 2014.

[NPR10] Cyril Nicaud, Carine Pivoteau, and Benoît Razet. Average analysis of glushkov automata under a bst-like model. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010*, LIPIcs, pages 388–399, 2010.

[TV05] Deian Tabakov and Moshe Y. Vardi. Experimental evaluation of classical automata constructions. In Geoff Sutcliffe and Andrei Voronkov, editors, *LPAR'05,*, volume 3835 of *Lecture Notes in Computer Science*, pages 396–411. Springer, 2005.