Inverse problems for finite automata: a solution based on Genetic Algorithms

B. Leblanc[1], E. Lutton[1] and J.-P. Allouche[2]

[1] *INRIA - Rocquencourt, B.P. 105, F-78153 LE CHESNAY Cedex, France*
Tel: +33 (0)1 39 63 55 23 - Fax: +33 (0)1 39 63 59 95
e-mail: Benoit.Leblanc@inria.fr, Evelyne.Lutton@inria.fr
`http://www-rocq.inria.fr/fractales/`
[2] *CNRS, LRI, Bât. 490, Université Paris-Sud, F-91405 Orsay Cedex, France*
Tel: 33 (0)1 69 15 64 54
e-mail: Jean-Paul.Allouche@lri.fr

Abstract. The use of heuristics such as Genetic Algorithm optimisation methods is appealing in a large range of inverse problems. The problem presented here deals with the mathematical analysis of sequences generated by finite automata. There is no known general exact method for solving the associated inverse problem. GA optimisation techniques can provide useful results, even in the very particular area of mathematical analysis. This paper presents the results we have obtained on the inverse problem for fixed point automata. Software implementation has been developed with the help of "ALGON", our home-made Genetic Algorithm software.

1 Introduction

A *finite automaton* is defined as a symbolic substitution σ acting on strings of symbols. More precisely σ is a map from a finite set of symbols S to S^*, the set of strings of symbols in S. The elements of S^* are called *words* and the images by σ of elements of S are called *words of the automaton.* The map σ is extended to S^* by concatenation (the image of a word is obtained by concatenating the images of its symbols), making σ a morphism of the free monoid S^*.

A sequence of words can be produced by successive applications of σ to an initial word s_0. If we denote by $s_n = s_{n_1}s_{n_2}...s_{n_p}$ the word at step n, the word obtained at step $n+1$ is then:

$$s_{n+1} = \sigma(s_n) = \sigma(s_{n_1})\sigma(s_{n_2})\ldots\sigma(s_{n_p}).$$

Note that the words $(s_n)_{n\in\mathbb{N}}$ are concatenations of the words of the automaton (in Example 1 this fact is highlighted by the alternation of bold and standard fonts).

Example 1. $S = \{1, 2, 3\}$

$$\sigma \begin{cases} 1 \to 211 \\ 2 \to 13 \\ 3 \to 123 \end{cases}$$

Iteration	Word
0	1
1	211
2	**13**211**211**
3	**211**123**13**211**211**13...

Of course, it is clear that the sequence of words $(s_n)_{n\in\mathbb{N}}$ is determined by the automaton σ and the initial word s_0.

An interesting property of such words concerns the frequency of occurrences of symbols of S. Let σ be an automaton acting on $S = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$, let s_0 be an initial word. The number of occurrences of any of the symbols observed in the word s_k (for any k) can be computed. Let us denote by

$$O_k = \begin{pmatrix} o_k^1 \\ o_k^2 \\ \ldots \\ o_k^m \end{pmatrix}$$

the occurrence vector of s_k (o_k^i being the number of occurrences of the symbol α_i observed in s_k). Then $O_{k+1} = A * O_k$ with $A = (a_{ij})_{(i,j)\in\{1,\ldots,m\}^2}$ the "growth" matrix, a_{ij} being the number of symbols α_i in the word $\sigma(\alpha_j)$.

We thus obtain $O_k = A^k * O_0$ with O_0 the occurrence vector of s_0. For Example 1, we have:

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

Let us also define the square (and in the same way any power) of an automaton: $\forall i \in \{1, \ldots, m\}$, $\sigma^2(\alpha_i) = \sigma(\sigma(\alpha_i))$. The associated matrix is then A^2. For more information about substitutions, see [4].

2 The inverse problem for finite automata

2.1 Motivations and formulation

To know whether a given sequence is generated by a finite automaton and to know explicitly one such automaton, can be useful in many situations. We list but three of them.

- In combinatorics on words: the third author, J. Currie and J. Shallit proved recently that the lexicographically least overlap-free[3] sequence on a two-letter alphabet, that begins with a given word (if it exists), must end with a

[3] An *overlap* is a string of the form $axaxa$ where a is a letter and x a (finite) word. A (finite or infinite) word is called *overlap-free* if it does not contain any overlap.

tail of the Thue-Morse sequence $10010110\cdots$, hence is the pointwise image (i.e. image under a morphism that sends each letter to a letter) of a fixed point of a morphism of constant length [8]. It is not known whether the least square-free sequence on a three-letter alphabet has the same property.

– In number theory: let $(u_n)_{n\in\mathbb{N}}$ be a sequence with values in the finite field $\mathbb{F}_q$. Then the formal power series $\sum u_n X^n$ is algebraic over the field of rational functions $\mathbb{F}_q(X)$ if and only if the sequence $(u_n)_{n\in\mathbb{N}}$ is the pointwise image of a fixed point of a morphism of length q over a finite alphabet, [10, 11]. For example the transcendence of values of Carlitz functions can be proved by showing the non-automaticity of the corresponding formal power series (see for example [9, 7, 12]). A hint that a sequence is not a fixed point of a morphism (it is more complicated for the pointwise image of a fixed point) is that, when solving the inverse problem with longer and longer prefixes of the sequence, the automaton we obtain keeps growing. That means almost certainly that there is no automaton that generates the infinite sequence.
– In physics: quasi-crystals are the 3-D analogue of the Penrose tiling. A one-dimensional description of this tiling involves the Fibonacci sequence, i.e., the fixed point of the morphism $0 \to 01$, $1 \to 0$.

Another occurrence of this "inverse" question for finite automata, in music, is recalled below.

Suppose we have a string of symbols and we want to know whether it has been produced by iterating an automaton. Of course if the string is finite, there always exists a trivial solution: the substitution sending the first letter of the string to the string itself. But we are interested in non-trivial solutions if any. Apart from the mathematical aspects of this "inverse" problem, it might be interesting to note that this work began as a composer, T. Johnson, produced a sequence of notes using a finite automaton, kept only the sequence of notes, and wanted to remember the automaton he used. (For the use of finite automata in a piece of T. Johnson, see [1].)

If the word s_0 happens to be a prefix of the word $s_1 = \sigma(s_0)$, it is not hard to see that the sequence of words $(s_n)_{n\in\mathbb{N}}$ converges to an infinite word. And this infinite word is clearly a *fixed point* of the substitution σ (extended to infinite words by concatenation). Now suppose an infinite word is given, is this word the fixed point of a substitution? Or is it the pointwise image of a fixed point of a substitution? No general answer to these questions is known either: a theoretical answer to the second question is known if the substitution has constant length [3] (i.e., if all words of the automaton have the same length), and also if the substitution is *primitive* (i.e., is such that there exists an m with the property that σ^m of any symbol contains at least one occurrence of each symbol of the set S) as proved recently by Durand [5]. Looking at finite prefixes of the given infinite sequence that have well-chosen lengths, we see that we can first restrict to finite words.

Hence, ideally, we would like to solve in the general case (i.e., not only for fixed point automata) what we called the inverse problem, that is:

Given a finite word s, find an automaton σ and an initial word s_0 such that $\sigma^n(s_0) = s$ for some n.

Of course, in its generality, this problem is extremely complex and can have many solutions or no non-trivial solution at all[4].

It can be reformulated as an optimisation problem on the search space of all possible σ, n and s_0 that minimizes a distance between $\sigma^n(s_0)$ and s. In order to reduce the complexity of this problem one also has to give some restrictions to the search space. We suppose in the following that the length of the words of σ is limited to l_{max} and that s_0 is a single symbol.

2.2 A bruteforce GA implementation

The first GA implementation that comes to mind is to perform a search over the space of all automata having word lengths smaller than or equal to l_{max}. Each individual of the GA thus represents an automaton with the following characteristics:

- m chromosomes for an individual: one per word of the automaton;
- variable length chromosomes: their lengths may vary between 1 and l_{max};
- an m-ary coding: the allele set is S.

These particular characteristics imply of course some modified GA operators, that are implemented in ALGON [6].

Thus, setting $l_{max} = 4$, the automaton of Example 1 would have the coding shown in Figure 1.

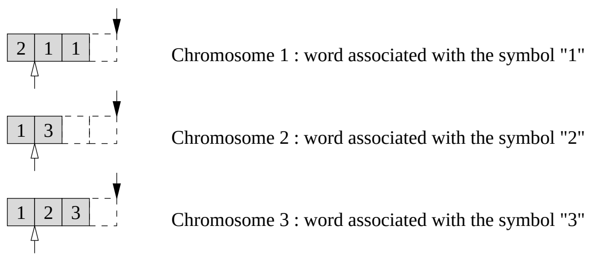
2 1 1
Chromosome 1 : word associated with the symbol "1"
1 3
Chromosome 2 : word associated with the symbol "2"
1 2 3
Chromosome 3 : word associated with the symbol "3"

Fig. 1. Direct coding of example 1 with $l_{max} = 4$.

The fitness function that must reflect the "resemblance to the target", is based on a comparison between words (Hamming distance, frequencies of occurrences of symbols, of couples of symbols, etc. . .).

[4] In fact, for a given solution couple (σ, s_0) and for any divisor p of n, the couple (σ^p, s_0) is also a solution.

This approach did not lead to interesting results, due to the size of the search space with respect to m and l_{max}: $|S| = \left(\sum_{i=1}^{l_{max}} m^i\right)^m$. The size would not be a problem if the resulting fitness landscape was smooth enough[5], but in our case one can easily check that a single change in the genetic code leads to important changes in the observed words.

Though this direct approach is not appropriate, its benefit is to highlight the difficulty of solving this problem. In fact, it is obvious that the coding should use more efficiently the information contained in the target word.

3 The fixed point hypothesis

If we restrict the inverse problem to the search of automata with fixed points, the complexity of the problem is reduced.

3.1 Definition and properties

A finite automaton has a fixed point if there exists a symbol α such that the first letter of $\sigma(\alpha)$ is α itself.

The sequence of words s_n produced by such an automaton starting with the initial symbol $s_0 = \alpha$ converge to a fixed point of σ: the beginning of the word at iteration $n + 1$ is exactly the word at iteration n. In fact each iteration adds symbols to the end of the previous word.

Example 2. $S = \{1, 2, 3\}$

$$\sigma \begin{cases} 1 \rightarrow 21 \\ 2 \rightarrow \mathbf{2}31 \\ 3 \rightarrow 13 \end{cases}$$

Iteration	Word
0	**2**
1	**231**
2	**231**13**21**
3	**231**13**21**21**13**231*21* ...

The inverse problem for a fixed point automaton is then much easier to solve than in the general case. Indeed, the information contained in the target word can be efficiently exploited, taking advantage of the fact that a fixed point is a succession of words of the automaton as well as the succession of symbols which generated them. Of course, it is necessary to know the lengths of the words in order to identify the connection between the two successions. A simple assumption on the lengths of words of the automaton permits then to identify it with a mechanism of simultaneous identification and reconstruction.

Checking an hypothesis is then a direct process of “reconstruction-comparison”. As previously outlined, the first symbol of the fixed point is associated to the first word, its size is given by assumption, so it can be identified. The second

[5] With a few secondary optima.

Incorrect hypothesis (2,2,2).	
$\sigma(2) = 23$	**2**_3_ 11321211323121
$\sigma(3) = 11$	2**3** _11_ 321211323121
$\sigma(1) = 32$	23**1**1 _32_ 1211323121
$\sigma(1) = 12$	231**1**32 _12_ 11323121
Contradiction on “1”.	

Correct hypothesis (2,3,2).	
$\sigma(2) = 231$	**2**_31_ 1321211323121
$\sigma(3) = 13$	2**3**1 _13_ 21211323121
$\sigma(1) = 21$	23**1**13 _21_ 211323121
$\sigma(1) = 21$	231**1**321 _21_ 1323121
$\sigma(3) = 13$	2311**3**2121 _13_ 23121
$\sigma(2) = 231$	23113**2**12113 _231_ 21
$\sigma(1) = 21$	231132**1**2113231 _21_

Fig. 2. Hypothesis propagation.

word, associated to the second letter, start ritght after the first, and knowing its size by hypothesis it can be identified too, and so on all along the fixed point. If the hypothesis in not correct, then the case will arise when the same symbol will be associated to two different words, discarding it. If it is correct the whole fixed point will be “reconstructed” without such contradiction.

Let us consider Example 2 and take $s = \sigma^3(2)$ as the target word. Assume each word of σ has two symbols:

- The initial symbol s_0 is simply the first symbol of s, i.e., $s_0 = 2$.
- The word associated with this symbol is a prefix of the target word, and it is assumed to be composed of two symbols, then we directly identify $\sigma(2) = 23$.
- The identification process is continued until a contradiction appears or the end of the target word is reached, as shown in Figure 2: in the incorrect hypothesis case we get $\sigma(1) = 32$ at step 2 and $\sigma(1) = 12$ at step 3 then the hypothesis is infirmed. Conversely, in the correct case, for the same symbol, the same word is always recognized, so the whole fixed point word is “reconstructed”.

3.2 A GA to search the space of word lengths

Coding the individuals :

An individual of the GA population may just represent an assumption on the words lengths, the corresponding automaton being reachable trough the identification mechanism previously exposed.

If we set an upper limit l_{max} for the possible lengths of the words, the genetic coding is the following:

- A set of alleles of cardinality l_{max}.
- A single chromosome per individual containing as many genes as elements in the symbol set S. The gene k codes the length of the word associated with the symbol α_k. The coding of a right assumption for Example 2 is:

$$|2|3|2| \;\rightarrow\; \sigma \begin{cases} 1 \rightarrow ?? \\ 2 \rightarrow ??? \\ 3 \rightarrow ?? \end{cases}$$

Compared to the “brute-force” implementation, a substantial improvement is the reduction of the search space which size is now $|S| = (l_{max})^m$.

Fitness **function** :

The evaluation of an individual relies on the validation process of the assumption it encodes. If the assumption appears to be valid, it is assigned a maximal fitness value. Note that any power of an automaton that is a solution to the problem is also a solution. Hence there is a potentially infinite number of solutions, as soon as one solution is found. But practically the number of solutions is limited by the length of the target word and the l_{max} value. The minimal solutions (in terms of lengths) are obviously the most interesting ones.

Invalid assumptions are given an intermediate value, and it is also desirable to differentiate these non-valid assumptions in order to drive the search towards a solution.

If a contradiction arises, two cases are considered:

- The contradiction arises before the identification of all the words of the automaton:
$$f(i) = \epsilon * \text{Number of identified words}$$
with ε a very small positive value.
- If a contradiction arises after the identification of all words of the automaton:
$$f(i) = \left(\frac{\text{Length of the “checked” sequence}}{\text{Length of the target sequence}}\right)$$
The “checked” word denotes the part of the target word checked before the contradiction occurs.

The maximum of f is then 1, corresponding to the case where the target word has been entirely checked. In order to give a best fitness value to any assumption leading to a complete identification of the words of the automaton than to any other that doesnt't, the number ε simply has to fulfill the following condition:
$$\varepsilon < \left(\frac{m+1}{\text{Length of the target word}}\right).$$

3.3 Results and discussions

We present here results obtained with ALGON [6], on two target words which are prefixes of fixed points of two different automata using 6 symbols. The maximal lengths of words being $l_{max} = 6$, the size of the search space is then: $|S| = 6^6 = 45656$

The general parameters of the GA are:

- A population of 100 individuals. Each individual being unique.
- A mutation probability $p_m = 0.125$.
- One point crossover with probability $p_c = 0.85$.

– An elitist population replacement with a ratio $r_s = 0.4$ of surviving individuals, i.e., 60 new individuals are created at each generation replacing the 60 worst individuals of the previous population.
– Selection performed with *Stochastic Universal Sampling* (see [2]).

Automaton 1:

$$\sigma \begin{cases} 1 \rightarrow 61 \\ 2 \rightarrow \mathbf{2}34 \\ 3 \rightarrow 52 \\ 4 \rightarrow 234 \\ 5 \rightarrow 6551 \\ 6 \rightarrow 433 \end{cases} \tag{1}$$

The target word s is then obtained by iterating 5 times the automaton starting from the initial seed "2", that is a word of length 196. We present in table 1 some statistics obtained over 20 runs. The following quantities are computed:

N_1 : Number of generations to obtain an assumption leading to a complete automaton (before a contradiction arises in the identification process).
N_2 : Number of generations to find a solution.
$N_3 = N_2 - N_1$: number of generations to find a solution when at least one individual has lead to a complete automaton.

The results are summarized in table 3.3.

Table 1. Results for automaton 1.

	Mean	Std.
N_1	4.2	2.38
N_2	18.95	24
N_3	14.75	23.3

About 1000 fitness evaluations are necessary to find a solution, which is to be compared to the search space size.

Automaton 2:

$$\sigma \begin{cases} 1 \rightarrow 11116 \\ 2 \rightarrow \mathbf{2}4 \\ 3 \rightarrow 5 \\ 4 \rightarrow 35 \\ 5 \rightarrow 23341 \\ 6 \rightarrow 666 \end{cases} \tag{2}$$

The target word s is again obtained by iterating 5 times the automaton starting from the initial seed "2". It has been designed to slow down the automaton identification process: the symbol “6” first appears quite far in s (26th position), so a contradiction has a greater chance to arise before all words have been identified.

Table 2. Results for automaton 2.

	Mean	Std.
N_1	8.65	2.38
N_2	13.2	7.76
N_3	4.55	3.76

We can see that, for this apparently more tricky automaton, the performances of the GA are better. But it can certainly be explained by the fact that the frequency of the hypothesis leading to a complete automaton identification (before a contradiction arises) is lower than previously, but other points of the search space seem to lead quite easily to those interesting regions.

4 Conclusion and further works

The results we obtained on fixed points automata suggest a coding of the general problem based on a set of possible words observed in the target word to be analysed. Such an approach, by considerably reducing the search space of possible automata, allows to obtain interesting results in the general case. This will be studied in a forthcoming paper.

References

1. J.-P. Allouche, T. Johnson (1995): *Finite automata and morphisms in assisted musical composition.* Journal of New Music Research **24**, 97–108.
2. J. E. Baker (1987): *Reducing bias and inefficiency in the selection algorithm.* Genetic Algorithms and their application: Proceedings of the Second International Conference on Genetic Algorithms, p. 14-21.
3. A. Cobham (1972): *Uniform tag sequences.* Math. Systems Theory **6**, 164–192.
4. S. Eilenberg (1974): *Automata, Languages, and Machines.* Vol. A, Academic Press.
5. F. Durand (1997): *A characterization of substitutive sequences using return words.* Disc. Math., to appear.
6. B. Leblanc, E. Lutton (1997): *ALGON: A Genetic Algorithm software package,* `http://www-rocq.inria.fr/fractales/`
7. J.-P. Allouche (1996): *Transcendence of the Carlitz-Goss Gamma function at rational arguments.* J. Number Theory **60**, 318–328.
8. J.-P. Allouche, J. Currie, J. Shallit (1997): *Extremal infinite overlap-free binary words.* Preprint.

9. V. Berthé (1994): *Automates et valeurs de transcendance du logarithme de Carlitz.* Acta Arith. **66**, 369–390.
10. G. Christol (1979): *Ensembles presque périodiques k-reconnaissables.* Theoret. Comput. Sci. **9**, 141–145.
11. G. Christol, T. Kamae, M. Mendès France, G. Rauzy (1980): *Suites algébriques, automates et substitutions.* Bull. Soc. Math. France **108**, 401–419.
12. M. Mendès France, J.-y. Yao (1997): *Transcendence and the Carlitz-Goss gamma function.* J. Number Theory **63**, 396–402.