A Comparison of Selected Variable Ordering Methods for NFA Induction*

Tomasz Jastrząb[1][0000−0002−7854−9058]

Institute of Informatics, Silesian University of Technology, Gliwice, Poland
Tomasz.Jastrzab@polsl.pl

Abstract. In the paper, we study one of the fundamental problems of grammatical inference, namely the induction of nondeterministic finite automata (NFA). We consider the induction of NFA consistent with the given sets of examples and counterexamples. We transform the induction problem into a constraint satisfaction problem and propose two variable ordering methods to solve it. We evaluate experimentally the proposed variable ordering methods and compare them with a state-of-the-art method. Additionally, through the experiments, we assess the impact of sample sets sizes on the performance of the induction algorithm using the respective variable ordering methods.

Keywords: Grammatical inference · Nondeterministic finite automata · Variable ordering · Constraint satisfaction

1 Introduction

In the paper, we deal with automata, which are important for numerous practical applications, such as compiler design, bioinformatics [?], and grammatical inference [?]. These automata are *finite*, *nondeterministic* and *minimal* in terms of the number of states. They are given by $A = (Q, \Sigma, \delta, q_0, Q_F)$, where Q is the finite set of states of the automaton, Σ is the input alphabet, $\delta : Q \times \Sigma \to 2^Q$ is the transition function, $q_0 \in Q$ is the initial state and $Q_F \subseteq Q$ is the set of final states [?]. The automata are also *consistent* with the given sample $S = (S_+, S_-)$, in which S_+ denotes the examples, and S_- denotes the counterexamples. The automaton A is said to be consistent with the sample S if for each word $w \in S_+$ there exists a sequence of transitions between state q_0 and at least one state $q \in Q_F$ and for each word $w \in S_-$ the condition does not hold.

In order to find a minimal nondeterministic finite automaton (NFA) consistent with the given sample S, we transform the induction problem into a constraint satisfaction problem (CSP) as discussed in [?,?,?]. Using the CSP formulation we ask whether for the given sample S and a given positive integer k there exists a k-state automaton consistent with S. By taking $k = 1, 2, \ldots$, we ensure that we find the *minimal* NFA for the given sample.

* The preliminary version of this paper was presented at ICGI 2018 [?]. In the current paper, we substantially extended the algorithmic descriptions and made the experimental part much more comprehensive.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_73

The main contributions of the paper are as follows. Firstly, we discuss two variable ordering methods devised specifically to solve the given CSP formulation of the problem. And secondly, we perform comprehensive experiments on input samples built of amino acid sequences from WaltzDB database [?]. To assess how the sizes of sets S_+ and S_- affect the performance of the induction algorithms, we consider samples for which $|S_+| \ll |S_-|$, $|S_+| = |S_-|$, and $|S_+| \gg |S_-|$ hold.

The paper is organized into 6 sections. In Section ?? we present a brief literature review. In Section ?? we recall the formulation of minimal NFA induction problem as a CSP. In Section ?? we present the parallel induction algorithm and the two proposed variable ordering methods. We present the results of conducted experiments in Section ??. Finally, Section ?? contains the conclusions.

2 State of the Art

The problem of finding a minimal NFA consistent with the given sample is hard. In particular, it cannot be done from polynomial time and data [?]. It was also shown that inducing a minimal NFA given a deterministic finite automaton (DFA) is PSPACE-complete [?].

The existing NFA induction methods focus around state merging algorithms, designed initially for DFAs. The merging is performed over a prefix tree acceptor constructed for the sample S. The algorithms remove redundant states keeping the consistency with the sample. Among the existing algorithms there are *DeLeTe2* [?], NRPNI [?], and the state merging methods proposed in [?,?]. Some algorithms transform the induction problem to CSPs [?,?,?,?].

The variable ordering methods are either *static* – defined before the algorithm begins, or *dynamic* – changing as the algorithm proceeds. Examples of static methods include *deg* [?] and *ddeg* heuristics based on the initial and current number of constraints involving the variable (degree). Dynamic methods include domain-size based method *dom* [?], as well as *dom-deg* [?], *dom-ddeg* [?] or *dom-futdeg* [?] methods, which follow the *dom* heuristic but in case of ties use the initial, current or future degree of the variables, respectively.

3 Problem Formulation

Since we assume the fixed number of states k, to find the NFA we only need to find the transition function δ and the set of final states Q_F. As the result, the following statements about the given CSP always hold [?,?,?]:

1. Let $l = |\Sigma|$ be the number of symbols in the alphabet. Then we need exactly $n = k^2l + k$ binary variables x and y.
2. For each pair of states $p, q \in Q$ and each symbol $a \in \Sigma$, if $q \in \delta(p, a)$, then $x_i = 1$, for some $i \in [1, k^2l]$. Likewise, $x_i = 0$ means that $q \notin \delta(p, a)$ holds.
3. For each state $p \in Q_F$, $y_i = 1$ holds for some $i \in [1, k]$. Otherwise, $y_i = 0$.
4. Let i, j, $i \neq j$ be the indices of x variables. Let $a, b \in \Sigma$, and $p, q, r \in Q$, $p \neq r, q \neq r$. Then the x variables are ordered as follows:

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_73

   (a) If $a < b$ in lexicographical order, then $i < j$ holds.
   (b) Given the transitions $p \xrightarrow{a} r$ and $q \xrightarrow{a} r$, if p precedes q in Q, then $i < j$.
   (c) Given the transitions $r \xrightarrow{a} p$ and $r \xrightarrow{a} q$, if p precedes q in Q, then $i < j$.
5. If p precedes q in Q, then $i < j$, where i, j are the y variables indices.

Example 1. Let the set of states be $Q = \{q_0, q_1\}$, and so $k = 2$, and let $\Sigma = \{a, b\}$. Then variables y_1 and y_2 correspond to the possible final states q_0 and q_1. Since $a < b$ lexicographically, and $q_0 < q_1$ in set Q, then the following relations hold: variable x_1 corresponds to the transition $q_0 \xrightarrow{a} q_0$, variable x_2 corresponds to the transition $q_0 \xrightarrow{a} q_1$, ..., variable x_8 corresponds to the transition $q_1 \xrightarrow{b} q_1$.

Given the above considerations, a *path* is a product of x variables followed by a y variable. Using Example **??**, the product $x_1x_6x_8y_1$ corresponds to the transitions $q_0 \xrightarrow{a} q_0$, $q_0 \xrightarrow{b} q_1$, and $q_1 \xrightarrow{b} q_1$, i.e., a word abb ending in state q_1.

Let us note that for the given number of states k and for each word w, there are exactly $k^{|w|}$ paths on which this word can be spelled out. There are also $k^{|w|}$ products of length $|w| + 1$ corresponding to these paths. In the sequel, we use P^w_{mi} to denote the m-th product of x variables corresponding to word w, and ending in variable y_i. Hence an NFA is consistent with the sample S when:

1. for all words $w \in S_+ \setminus \{\lambda\}$ it holds that (examples acceptance):

$$\sum_{i=1..k} \sum_{m=1..k^{|w|-1}} P^w_{mi} y_i = 1, \tag{1}$$

2. for all words $w \in S_- \setminus \{\lambda\}$ it holds that (counterexamples rejection):

$$\sum_{i=1..k} \sum_{m=1..k^{|w|-1}} P^w_{mi} y_i = 0. \tag{2}$$

Note that the summation is performed according to the rules of Boolean algebra. Note that (**??**) is satisfied iff there exists a path for word w that ends in a final state. Note also that (**??**) is satisfied iff either there is no path for word w, or the path ends in a non-final state. Finally, if $\lambda \in (S_+ \cup S_-)$ holds, then we set:

$$y_1 = \begin{cases} 1, \text{for } \lambda \in S_+, \\ 0, \text{for } \lambda \in S_-. \end{cases} \tag{3}$$

The above condition allows to reduce the solution space to be searched [**?**].

4 Algorithms and Methods

Hereinafter, let C be the set of constraints, $c_+ \in C$ be a constraint given by (**??**), and $c_- \in C$ be a constraint given by (**??**). Let $|c|$ be the number of *active* (non-zero) products P in (**??**) or (**??**), and $d(c, x)$ be the number of active products in c containing variable x. The variable ordering methods pertain to the x variables.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_73

```
   ▷ input: i, v – index and value of the most recently set variable, initially empty
   ▷ global: f – flag showing that a solution was found, initially false
 1: procedure InduceNFA(i, v)
 2:     if i ≠ ∅ and v ≠ ∅ then f, p ← Evaluate(i, v)
 3:     if f = true then print solution; return                    ▷ solution found
 4:     if p = false then return          ▷ contradiction found – backtracking needed
 5:     i, v ← Reorder()
 6:     InduceNFA(i, v)
 7:     if f = false then InduceNFA(i, !v)              ▷ assigns opposite value to x_i
 8:     if f = false then unset x_i
 9: end procedure
```

Fig. 1. The induction algorithm

```
   ▷ input: i, v – index and value of the most recently set variable
 1: procedure Evaluate(i, v)
 2:     if v = 0 then
 3:         for all c+ : c+ is not satisifed and d(c+, x_i) > 0 do
 4:             |c+| ← |c+| − d(c+, x_i); ∀x_j : x_i, x_j ∈ P ∧ P ∈ c+, update d(c+, x_j)
 5:         end for
 6:         if ∃c+ : |c+| = 0 then p ← false else p ← true
 7:     else
 8:         if ∃c− : d(c−, x_i) > 0 ∧ ∃P ∈ c− : P = 1 then p ← false; return f, p
 9:         ∀c+ : d(c+, x_i) > 0 ∧ ∃P ∈ c+ : P = 1, mark c+ as satisfied
10:         if all c+ are satisfied then f ← true; ∀x_j : x_j = ∅, x_j ← 0 else p ← true
11:     end if
12:     return f, p
13: end procedure
```

Fig. 2. The Evaluate procedure for *min-max-ex* variable ordering method

The y variables are set first to produce independent instances of the CSP, solved in parallel [?,?]. It also applies to the *deg* method used in the experiments.

The induction procedure InduceNFA, executed for each independent instance of the CSP, is given in Fig. ??. The procedure consists of the evaluation phase (represented by the Evaluate procedure in line ??), and the ordering phase (represented by the Reorder procedure in line ??). They both differ depending on the selected variable ordering method.

The *min-max-ex* Method. The Evaluate procedure (Fig. ??) implements a 'fail-fast' behavior, by checking first the constraints that may result in a contradiction (lines ?? and ??). It aims at explicitly satisfying constraints $c_+ \in C$ (line ??), setting the other x_i variables to zeros if all c_+ are satisfied (line ??). It updates $|c_+|$ and $d(c_+, x_j)$ using the techniques described in [?] (lines ??–??).

The Reorder procedure (Fig. ??) selects a constraint $c'_+ \in C$, for which $|c'_+|$ is minimal (line ??) and the index of the most frequent variable $x_i \in c'_+$

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_73

```
1: procedure REORDER
2:     c'_+ ← argmin_{c_+ ∈ C} |c_+|          ▷ considers only not-yet-satisifed c_+
3:     i ← argmax_{x_i ∈ c'_+} d(c'_+, x_i)   ▷ considers only variables x_i : x_i = ∅
4:     return i, 0
5: end procedure
```

Fig. 3. The REORDER procedure for *min-max-ex* variable ordering method

(line ??). It sets v to zero (line ??). The ordering makes the evaluation shorter, by choosing constraint with fewer products and enforcing the 'fail-fast' behavior.

The *min-max-cex* Method. The second variable ordering method, called *min-max-cex*, differs from *min-max-ex* in that, that in the EVALUATE procedure, for $v = 1$ it only checks for a contradiction in any $c_- \in C$. For $v = 0$ it checks for a contradiction in any $c_+ \in C$, it updates the $|c_-|$ and $d(c_-, x_j)$ values as in lines ??–??, and checks for satisfied constraints c_-. If all c_- are satisfied it sets all unset x_j variables to 1. In the REORDER procedure, in line ?? we look for a constraint $c_- \in C$, such that $|c_-|$ is minimal, and we propose that $v \leftarrow 1$ in line ??.

The *deg* Method. In the EVALUATE procedure the *deg* method first checks for any contradictions, either in c_+ or in c_- (for $v = 0$ or $v = 1$, respectively), and then it checks for satisfied constraints. We require that all constraints are explicitly satisfied. Since the method is static, the order of variables is established before the induction begins. So the REORDER procedure merely returns the next variable according to the already known order. It sets $v \leftarrow 0$ in line ?? of Fig. ??.

Example 2. Due to space limitations, we provide an example trace of the induction algorithm at https://github.com/tjastrzab/iccs.

5 Experimental Evaluation

We generated 450 samples based on the sets of amino acid sequences [?]. The number of sequences in set S_+ (resp. S_-) was 5 (resp. 45), 25 (resp. 25), and 45 (resp. 5), for the samples, for which $|S_+| \ll |S_-|$, $|S_+| = |S_-|$, and $|S_+| \gg |S_-|$ hold. The algorithms were implemented in Java and run on an Intel Xeon E5-2640 2.60GHz processor (16 cores, 8 GB RAM). The time limit was 3 hours. We considered two scenarios. In *Exp. 1*, we sought the first consistent k-state automaton. In *Exp. 2*, the configuration of the final states was also given to see how the algorithms perform when searching for a specific NFA. All minimal automata had 2 states.

The distribution of the run times, given by the minimum, maximum and average values, is shown in Tab. ??. Note that in all cases, the average times for the best- and worst-performing methods, differ by an order of magnitude or more.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_73

Moreover, the balanced samples are the hardest to solve, since the average times are the longest for both *Exp. 1* and *Exp. 2*. The *min-max-ex* method prevails when $|S_+| \ll |S_-|$ holds, while *min-max-cex* is the fastest for the cases in which $|S_+| \gg |S_-|$ is true. This is expected since the time spent in the REORDER procedure is then much shorter (as the ordering is based on S_+ or S_- only). The *min-max-cex* method is also the fastest for the hardest sample type. This proves that the new methods are competitive with respect to the *deg* method.

Table 1. Minimum, maximum and average run times (in seconds)

Sample type	*min-max-ex*		*deg*		*min-max-cex*	
	Exp. 1	*Exp. 2*	*Exp. 1*	*Exp. 2*	*Exp. 1*	*Exp. 2*
Minimum and maximum run times						
$\|S_+\| \ll \|S_-\|$	0.0, 0.2	0.0, 10.8	0.0, 32.9	0.0, 5752.7	0.0, 5795.5	0.0, 5678.8
$\|S_+\| = \|S_-\|$	0.0, 8083.2	0.0, 8058.2	0.2, 4681.9	0.3, 4612.5	0.1, 25.5	0.1, 118.8
$\|S_+\| \gg \|S_-\|$	0.0, 7327.1	0.0, 9085.7	0.0, 2652.3	0.0, 8618.0	0.0, 1.0	0.0, 1.8
Average run times						
$\|S_+\| \ll \|S_-\|$	0.0	0.1	0.5	58.7	79.2	229.9
$\|S_+\| = \|S_-\|$	460.6	647.7	176.4	279.9	4.6	7.6
$\|S_+\| \gg \|S_-\|$	129.6	471.8	49.6	269.4	0.2	0.2

To explain the differences in the run times, we counted the number of calls of the INDUCENFA procedure, i.e., the number of visited solution tree nodes. We observed that the number of calls differed by two to four orders of magnitude.

Finally, to observe the relation between the run time performance and the automaton size, we counted the number of transitions in each NFA. We observed that *min-max-ex* and *deg* methods produce NFAs of similar size, while the *min-max-cex* method generates on average 17–30 transitions more. It is so because *min-max-ex* and *deg* methods produce mostly the transitions necessary to accept the examples. The *min-max-cex* method satisfies all counterexamples and makes the remaining variables equal to one, which generates more transitions.

6 Conclusions

In the paper we have investigated the problem of nondeterministic finite automata induction. The experiments have shown that the proposed variable ordering methods perform better than the state-of-the-art one, especially in case of imbalanced sizes of sets S_+ and S_-. The result is important since it is not uncommon that, for instance, we know just a few factors causing a disease (set S_+) and much more factors that are not responsible for this particular disease (set S_-). Hence, being able to classify these factors efficiently and correctly using the induced NFAs, can be of help in some bioinformatics tasks.

Acknowledgment The research was supported by National Science Centre Poland (NCN), project registration no. 2016/21/B/ST6/02158.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_73