JMLR: Workshop and Conference Proceedings 21:237–242, 2012 The 11th ICGI

Induction of Non-Deterministic Finite Automata on Supercomputers

Wojciech Wieczorek WOJCIECH.WIECZOREK@US.EDU.PL
Institute of Computer Science, University of Silesia, Poland

Editors: Jeffrey Heinz, Colin de la Higuera and Tim Oates

Abstract

The problem of inducing automata of minimal size consistent with finite sets of examples and counter-examples is the combinatorial optimization task of grammatical inference and is known to be computationally hard. Both an exact and a heuristic method of finding a non-deterministic finite automaton (NFA) with a given number of states such that all examples are accepted and all counter-examples are rejected by the automaton will be evaluated. The methods are based on a translation of NFA identification into integer nonlinear programming.

Keywords: Automata induction, Integer programming, Grammatical inference

1. Introduction

Selecting a minimal automaton consistent with examples and counter-examples is known to be a very hard problem. Specifically, Gold (1978) proved that given a finite alphabet Σ, two finite sets of strings $\mathcal{S}_+$ and $\mathcal{S}_-$ built from symbols taken from Σ, and an integer k, then determining whether there is a k-state DFA (deterministic finite automaton) that recognizes $\mathcal{L}$ such that every string from $\mathcal{S}_+$ is also in $\mathcal{L}$ and no string from $\mathcal{S}_-$ is in $\mathcal{L}$, is a task which is NP-complete. Furthermore, it is known that even finding a DFA with a number of states polynomial in the number of states of the minimal solution is NP-complete (Pitt and Warmuth, 1993). As regards non-determinism, it is well known that NFA or regular expression minimization is computationally hard: it is PSPACE-complete (Meyer and Stockmeyer, 1972). Moreover, Jiang and Ravikumar (1993) showed that the minimization problem for NFAs or regular expressions remains PSPACE-complete even when specifying the regular language by a DFA. Angluin (1969) showed that there is no polynomial time algorithm for finding a shortest compatible regular expression for arbitrary given data (if P $\neq$ NP). Thus the problem of inferring an NFA with a minimal size that matches a labeled set of input strings is probably of exponential complexity. In order to obtain an NFA from a complete sample as input, one can use an algorithm developed by García et al. (2008).

The purpose of the present proposal is threefold. The first objective is to formulate finding a k-state NFA as a nonlinear integer programming problem (INLP), so as to take advantage of massively parallel computers for its solution. The second objective is to determine the limitations of these proposed algorithms in terms of the input data size and time spent computing. The third objective is to report the results of some experiments performed on some input data investigated by other researchers. The last objective includes

checking these algorithms' ability to solve a grammatical inference task. The present paper's content is organized into four sections. Section 2 translates the task into an INLP problem. Section 3 gives the experimental results referred to. Concluding comments are contained in Section 4.

2. Translation of NFA Identification into an INLP

Below, the problem of NFA induction will be formulated, then it will be re-formulated as an INLP. Let Σ be an alphabet, let $\mathcal{S}_+$ (examples) and $\mathcal{S}_-$ (counter-examples) be two finite sets of words over Σ, and let k be an integer. The goal of NFA induction is to determine a k-state NFA $A = (\mathcal{Q}, \Sigma, \delta, s, \mathcal{F})$, as defined in Hopcroft et al. (2001), such that $L(A)$ contains $\mathcal{S}_+$ and is disjoint with $\mathcal{S}_-$.

Let $\mathcal{S} = \mathcal{S}_+ \cup \mathcal{S}_-$ ($\mathcal{S}_+ \cap \mathcal{S}_- = \emptyset$), and let $P(\mathcal{S})$ be the set of all prefixes excluding the empty word of all words of $\mathcal{S}$. The integer variables will be $x_{pq} \in \{0,1\}$, $p \in P(\mathcal{S})$, $q \in \mathcal{Q}$; $y_{aqr} \in \{0,1\}$, $a \in \Sigma$, $q, r \in \mathcal{Q}$; and $z_q \in \{0,1\}$, $q \in \mathcal{Q}$. The value of x_{pq} is 1 if $q \in \delta(s,p)$ holds in an automaton A, $x_{pq} = 0$ otherwise. The value of y_{aqr} is 1 if $r \in \delta(q,a)$, $y_{aqr} = 0$ otherwise. Finally, we let $z_q = 1$ if $q \in \mathcal{F}$ and zero if not. Let us now see how to describe the constraints of the relationship between an automaton A and a set $\mathcal{S}$ in terms of nonlinear equations and inequalities.

1. Naturally, according to the presence of the empty word we require that

$$\begin{aligned} z_s &= 1 \qquad \lambda \in \mathcal{S}_+ \\ z_s &= 0 \qquad \lambda \in \mathcal{S}_- \end{aligned}$$

   One of the above equations is needed only if $\lambda \in \mathcal{S}$. In the opposite case, the variable should not be settled in advance.

2. Every example has to be accepted by the automaton, but no counter-example should be. This can be written as

$$\begin{aligned} \sum_{q \in \mathcal{Q}} x_{pq} z_q &\geq 1 \qquad p \in \mathcal{S}_+ - \{\lambda\} \\ \sum_{q \in \mathcal{Q}} x_{pq} z_q &= 0 \qquad p \in \mathcal{S}_- - \{\lambda\} \end{aligned}$$

3. For $P(\mathcal{S}) \ni p = a \in \Sigma$ we can have x_{pq} equal to 1 only in cases in which $q \in \delta(s,a)$; thus

$$x_{pq} - y_{psq} = 0 \qquad p \in \{a \in \Sigma \mid a \in P(\mathcal{S})\},\ q \in \mathcal{Q}$$

4. Finally, we want to express the fact that whenever $x_{pq} = 1$ for $p = wa$, $w \in \Sigma^+$, $a \in \Sigma$, we have $q \in \delta(r,a)$ for at least one state r such that $x_{wr} = 1$. And vice versa, if a word w is spelled out by a path from s to a state r and there is a transition $r \xrightarrow{a} q$, then $x_{pq} = 1$ has to be fulfilled. We can guarantee this by requiring

$$\begin{aligned} -x_{pq} + \sum_{r \in \mathcal{Q}} x_{wr} y_{arq} &\geq 0 \qquad p \in \{wa \in P(\mathcal{S}) \mid w \in \Sigma^+ \wedge a \in \Sigma\} \\ x_{pq} - x_{wr} y_{arq} &\geq 0 \qquad q, r \in \mathcal{Q} \end{aligned}$$

Consequently any instance of NFA induction with c symbols in an alphabet, $d = \sum_{w \in \mathcal{S}} |w|$, and k states can be expressed as an INLP with $O(ck^2 + dk)$ variables and $O(ck + dk^2)$ equations and inequalities.

3. Experimental Results

At first, let us answer the question of whether NFA induction is a harder problem than DFA induction. The search space for the automata induction problem can be assessed by the number of automata with a fixed number of states. It has been shown that the number of pairwise non-isomorphic minimal k-state DFAs over a c-letter alphabet is of order $k2^{k-1}k^{(c-1)k}$, while the number of NFAs on k states over a c-letter alphabet such that every state is reachable from the start state is of order 2^{ck^2} (Domaratzki et al., 2002). Thus, switching from determinism to non-determinism increases the search space enormously. On the other hand, for $c, k \geq 2$, there are at least 2^{k-2} distinct languages $\mathcal{L} \subseteq \Sigma^*$ such that: (a) $\mathcal{L}$ can be accepted by an NFA with k states; and (b) the minimal DFA accepting $\mathcal{L}$ has 2^k states (Domaratzki et al., 2002). It is difficult to resist the conclusion that—despite its hardness—NFA induction is extremely important and deserves exhaustive research. The choice of a computer cluster as a place for performing these complex computations is fully justified as well.

Since we are dealing with solving a system of nonlinear equations and inequalities which requires in the worst case exponential time in the number of variables, there ought to be opportunities for using an exact algorithm, parallel backtrack search (Quinn, 2004), and a heuristic algorithm, parallel tabu search (Dréo et al., 2006). In all experiments, the algorithms were implemented in C++. The programs ran under the Linux operating system on the following computer clusters: Zeus, Galera, and Reef (Poland). As regards parallel programming, the MVAPICH (on Galera) and Open MPI (on Zeus and Reef) MPI (message passing interface) libraries were used.

In order to compare this proposed automata induction approach with other patterns from the literature, experiments were performed on the Tomita (1982) language set. These programs were run on examples and counter-examples used by Angeline (1997), Luke et al. (1999), and Unold (2009). For each target language attempts to construct a k-state NFA for $k = 1, 2, \ldots$ continued until one was obtained. Then, the correctness of the induced automaton was checked. The computational results are presented in Table 1. The column captioned N$^{\text{o}}$ contains the number of the language. In the second column the number of states in the NFA is given. In the next two columns the cardinalities of the constraints (after reformulating them as bit-wise expressions) and variables are given. The number of processors, p, is given in the fifth (and seventh) column along with the corresponding CPU time, τ, (in seconds) given in the sixth (and eighth) column. Next, we have the kind of algorithm applied (backtrack search or tabu search). The programs were run several times until automata equivalent to the prototype regular expressions were obtained. In the case of a successful tabu search, the number of executed iterations in the last run is given in the last but one column. Entries for 'Run' and 'Iteration' relate to the second pair of columns p, τ.

As far as the generalization context of these experiments is concerned, the results are very good in comparison with those of Angeline (1997); Luke et al. (1999); Unold (2009).

Table 1: Problem characteristics and CPU time of computations

Nº	k	m	n	p	τ	p	τ	**Alg.**	**Run**	**Iteration**	**Cluster**
1	1	59	24	4	<1	8	<1	BS	1	–	Reef
2	2	214	74	4	<1	8	<1	BS	1	–	Galera
3	3	877	231	16	91	32	74	TS	1	no solut.	Zeus
3	4	1437	316	–	–	80	1757	TS	15	240795	Zeus
4	3	878	234	60	14	120	6	BS	1	–	Zeus
5	3	742	198	60	55	120	28	BS	1	no solut.	Reef
5	4	1214	272	–	–	64	1688	TS	22	243813	Zeus
6	3	622	168	40	2	80	1	BS	1	–	Zeus
7	3	853	228	120	110	240	68	BS	1	no solut.	Zeus
7	4	1401	312	–	–	72	1433	TS	17	327497	Zeus

With the present proposed algorithms it was possible to find an expected automaton (i.e., equivalent to a model regular expression) in all cases. Others had not discovered, in up to 50 attempts: the 5th language (Luke et al., 1999); and the 7th language (priv. comm.) (Unold, 2009). Angeline (1997), in a single run per language consisting of five sub-runs, had not discovered: the 2nd, 3rd, 4th, 5th, and 7th languages. Unold (2009) reported also that the evidence-driven approach, EDSM (de la Higuera, 2010), had failed in four out of the seven data sets. The new proposals have also achieved satisfactory results in respect of the efficiency of the parallel computation. It can be seen from the table that the algorithms have the ability to decrease the time spent computing as the number of processors increases. Of course, while choosing p, the total amount of time spent by all processors performing communications and redundant computations has to be taken into account so as to fairly predict the overall execution time.

4. Conclusions

This paper treated the induction of NFAs based on finite languages. That is constituted by the following task: given two disjoint finite sets $\mathcal{S}_+, \mathcal{S}_- \subset \Sigma^*$ of words and an integer $k > 0$, build a k-state NFA that accepts the language $\mathcal{S}_+$ and does not accept any word from the set $\mathcal{S}_-$. For deterministic automata this problem has many theoretical results and practical methods. In order to address the non-deterministic case, this problem was re-formulated as an INLP. On this foundation, two further algorithms were designed: an exact one and a heuristic one, which not only are able to generate concise automata, but also might do this efficiently on today's parallel platforms. The experiments conducted showed that the new proposed algorithms work satisfactorily for standard benchmarks. From additional experiments, it transpired that there are a number of languages for which the new approach outperforms conventional methods such as RPNI (Oncina and García, 1992) or EDSM, especially when the learning sample is severely limited and inevitably is not structurally complete.

Acknowledgments

This research was supported in part by PL-Grid Infrastructure, and by Polish National Science Centre, Grant No. DEC-2011/03/B/ST6/01588.

References

P. Angeline. An alternative to indexed memory for evolving programs with explicit state representations. In *Proceedings of the Second Annual Conference on Genetic Programming*, pages 423–430. Morgan Kaufmann, 1997.

D. Angluin. *An application of the theory of computational complexity to the study of inductive inference*. PhD thesis, University of California, 1969.

C. de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, 2010.

M. Domaratzki, D. Kisman, and J. Shallit. On the number of distinct languages accepted by finite automata with n states. *Journal of Automata, Languages and Combinatorics*, 7:469–486, 2002.

J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard. *Meta-heuristics for Hard Optimization*. Springer, 2006.

Pedro García, Manuel Vázquez de Parga, Gloria I. Álvarez, and José Ruiz. Universal automata and nfa learning. *Theoretical Computer Science*, 407:192–202, 2008.

E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.

J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, second edition, 2001.

T. Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM Journal on Computing*, 22:1117–1141, 1993.

S. Luke, S. Hamahashi, and H. Kitano. "Genetic" programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1098–1105. Morgan Kaufmann, 1999.

A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th Annual Symposium on Switching and Automata Theory*, pages 125–129, 1972.

J. Oncina and P. García. Identifying regular languages in polynomial time. In H. Bunke, editor, *Advances in Structural and Syntactic Pattern Recognition*, volume 5 of *Machine Perception and Artificial Intelligence*, pages 99–108. World Scientific, 1992.

L. Pitt and M. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the ACM*, 40:95–142, 1993.

M. J. Quinn. *Parallel Programming in C with MPI and OpenMP*. McGraw-Hill, 2004.

M. Tomita. Dynamic construction of finite automata from examples using hill-climbing. In *Proceedings of the Fourth Annual Conference of the Cognitive Science Society*, pages 105–108, 1982.

O. Unold. Regular language induction with grammar-based classified system. In S. Soomro, editor, *Engineering the Computer Science and IT*, pages 13–22. InTech, 2009.