

Learning Automata Teams^{*}

Pedro García, Manuel Vázquez de Parga, Damián López, and José Ruiz

Departamento de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia, Valencia, Spain
{pgarcia,mvazquez,dlopez,jruiz}@dsic.upv.es

Abstract. We prove in this work that, under certain conditions, an algorithm that arbitrarily merges states in the prefix tree acceptor of the sample in a consistent way, converges to the minimum *DFA* for the target language in the limit. This fact is used to learn automata teams, which use the different automata output by this algorithm to classify the test. Experimental results show that the use of automata teams improve the best known results for this type of algorithms. We also prove that the well known Blue-Fringe EDSM algorithm, which represents the state of art in merging states algorithms, suffices a polynomial characteristic set to converge.

Keywords: *DFA* learning, Automata teams.

1 Introduction

Gold [4] proposed the identification in the limit model as a framework to study the convergence of inference algorithms. He also proved [5] that any minimum *DFA* can be reconstructed from a set of words of polynomial size in the number of states of the automaton and he proposed an algorithm to do this task in polynomial time. This algorithm uses a red-blue strategy (this denomination is taken from a later algorithm [8] that will be mentioned afterwards), that maintains two subsets in the set of states of the prefix tree acceptor of the sample:

- States belonging to the solution (the *red* set, denoted R in the sequel).
- States not in R that can be reached from a state of R using a symbol (the *blue* set, denoted B in the sequel).

In Gold's algorithm, the selection of the states of B to be promoted to R and the election of equivalent states, are both made in an arbitrary way.

The *RPNI* [9] and *Lang* [7] algorithms were both proposed in 1992. They assure the consistency of the hypothesis by merging states in lexicographical order, starting from the prefix tree acceptor of the sample. Later in that decade, the *Blue-Fringe EDSM* algorithm [8] was developed; This algorithm uses a strategy

^{*} Work partially supported by Spanish Ministerio de Educación y Ciencia under project TIN2007-60769.

named *Red-Blue* to decide the states candidates to be merged. Blue-Fringe became the state of art for *DFA* learning algorithms. The question that was left without proof if there exists a polynomial characteristic set for this algorithm.

In [6], De la Higuera et al. studied how the order of states merging could affect the convergence of *RPNI*-type algorithms. They established that if the order of merging is data-independent, both the convergence and the existence of a characteristic polynomial set are guaranteed. Otherwise, these properties do not hold for data-dependant algorithms. They also empirically showed that this later type of algorithms may behave well.

We prove in this work the existence of a polynomial characteristic set for the *Blue-Fringe EDSM* algorithm [8]¹, which realizes a data-dependent merging of states. We also propose and prove the convergence of a merging states algorithm that we denote *Generalized Red-Blue Merging (GRBM)* algorithm that shares with Gold's algorithm the red-blue strategy and state merging in an arbitrary way. This permits to learn automata teams, that is, the use of several different automata output by the algorithm for classification tasks. In the experiments, several classification criteria have been used and they obtain better recognitions rates than both the *Blue-Fringe EDSM*, which outputs *DFAs* and the *DeLeTe2* [3] which outputs a subclass of the nondeterministic automata called *Residual Finite State Automata*.

2 Definitions and Notation

Let Σ be a finite alphabet and let Σ^* be the monoid generated by Σ with concatenation as the internal operation and λ as neutral element. A *language* L over Σ is a subset of Σ^* . The elements of L are called *words*. Given $x \in \Sigma^*$, if $x = uv$ with $u, v \in \Sigma^*$, then u (resp. v) is called *prefix* (resp. *suffix*) of x . $\text{Pr}(L)$ (resp. $\text{Suf}(L)$) denotes the set of prefixes (suffixes) of L .

A *Deterministic Finite Automaton (DFA)* is a 5-tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is an alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states and $\delta : Q \times \Sigma \rightarrow Q$ is the transition function. The language accepted by an automaton \mathcal{A} is denoted $L(\mathcal{A})$.

A Moore machine is a 6-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, \Phi)$, where Σ (resp. Γ) is the input (resp. output) alphabet, δ is a partial function that maps $Q \times \Sigma$ in Q and Φ is a function that maps Q in Γ called *output function*.

Throughout this paper, the behavior of M will be given by the partial function $t_M : \Sigma^* \rightarrow \Gamma$ defined as $t_M(x) = \Phi(\delta(q_0, x))$, for every $x \in \Sigma^*$ such that $\delta(q_0, x)$ is defined.

A *DFA* $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ can be simulated by a Moore machine $M = (Q, \Sigma, \{0, 1\}, \delta, q_0, \Phi)$, where $\Phi(q) = 1$ if $q \in F$ and $\Phi(q) = 0$ otherwise. Then, the language defined by M is $L(M) = \{x \in \Sigma^* : \Phi(\delta(q_0, x)) = 1\}$.

¹ Through this paper, the strategy of maintaining three sets of states in the automaton while merging, will be denoted *Red-Blue* strategy whereas the implementation of the algorithm by Lang, using a merging score proposed by Price, will be denoted *Blue-Fringe EDSM*.

Given two disjoint finite sets of words D_+ and D_- , we define the (D_+, D_-) -*prefix tree Moore machine* ($PTM(D_+, D_-)$) as the Moore machine having $\Gamma = \{0, 1, ?\}$, $Q = Pr(D_+ \cup D_-)$, $q_0 = \lambda$ and $\delta(u, a) = ua$ if $u, ua \in Q$ and $a \in \Sigma$. For every state u , the value of the output function associated to u is 1, 0 or ? (undefined) depending whether u belongs to D_+ , to D_- or to $Q - (D_+ \cup D_-)$ respectively.

A Moore machine $M = (Q, \Sigma, \{0, 1, ?\}, \delta, q_0, \Phi)$ is *consistent* with (D_+, D_-) if $\forall x \in D_+$ we have $\Phi(x) = 1$ and $\forall x \in D_-$ we have $\Phi(x) = 0$.

Given a language L , a *characteristic set* for an inference algorithm for L is a set of words such that when they are used as input to the algorithm, a representation of the target language is obtained, and the use of further input words do not change the output. We use the model of learning called identification in the limit [4]. An algorithm identifies a class of languages H *in the limit* if and only if every language in the class has associated a finite characteristic set.

3 Gold's Algorithm

Aiming to focus our proposal and as a way to analyze the main features of most of the inference algorithms that have been proposed so far, we present in this section a version of Gold's algorithm that uses a prefix tree Moore machine of the sample as a way of representing the input data, instead of the original way, which was the so called *state characterization matrix*. The algorithm we describe (Algorithm 1) behaves exactly as the original and its main features are: 1) It does not merge states; 2) some decisions can be taken in a not specified (arbitrary) way (lines 4 and 16 of the algorithm) and 3) it converges with a polynomial characteristic sample (if there is a relation between the order in which states are considered and the way the characteristic sample is built).

The main drawback of Gold's algorithm is that if it is not supplied with enough data, the output may not be consistent with the input data. It behaves in a different way as the *RPNI*, which is a merging states algorithm whose output is always consistent with the input.

Gold's algorithm uses the function *od* (obviously distinguishable, lines 3 and 16) defined in the following way: Two states u_1 and u_2 of $PTM(D_+, D_-)$ are obviously distinguishable if there exists a word x such that $\Phi(u_1x), \Phi(u_2x) \in \{0, 1\}$ and $\Phi(u_1x) \neq \Phi(u_2x)$.

The characteristic set proposed by Gold is based in the following definition and proposition:

Definition 1. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a DFA. We say that $S \subset \Sigma^*$ is a *minimal set of test states* if for every $q \in Q$ there exists only one word $x \in S$ such that $\delta(q_0, x) = q$. Note that if S is minimal, $Card(S) = Card(Q)$.

Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be the minimum complete DFA for a language L and let S be a prefix closed minimal set of test states. Two sets $D_+(S)$ and $D_-(S)$ can be built starting from S as follows:

1. For every $u \in (S\Sigma \cap Pr(L)) \cup \{\lambda\}$ we add uv to $D_+(S)$, where v is a suffix that completes u in L ($uv \in L$). If $u \in L$ we take $v = \lambda$.

Algorithm 1. Gold($D_+ \cup D_-$)

Require: Two disjoint finite sets (D_+, D_-)
Ensure: A consistent Moore Machine

- 1: $M_0 := PTMM(D_+, D_-) = (Q_0, \Sigma, \{0, 1, ?\}, \delta_0, q_0, \Phi_0)$
- 2: $R = \{\lambda\}; B = \Sigma \cap Q_0;$
- 3: **while** there exists $s' \in B$ such that $od(s, s', M_0), \forall s \in R$ **do**
- 4: choose s'
- 5: $R = R \cup \{s'\};$
- 6: $B = (R\Sigma - R) \cap Q_0;$
- 7: **end while**
- 8: $Q = R;$
- 9: $q_0 = \lambda;$
- 10: **for** $s \in R$ **do**
- 11: $\Phi(s) = \Phi_0(s);$
- 12: **for** $a \in \Sigma$ **do**
- 13: **if** $sa \in R$ **then**
- 14: $\delta(s, a) = sa$
- 15: **else**
- 16: $\delta(s, a) = \text{any } s' \in R \text{ such that } \neg od(sa, s', M_0);$
- 17: **end if**
- 18: **end for**
- 19: **end for**
- 20: $M = (Q, \Sigma, \{0, 1, ?\}, \delta, q_0, \Phi);$
- 21: **if** M is consistent with (D_+, D_-) **then**
- 22: Return(M)
- 23: **else**
- 24: Return(M_0);
- 25: **end if**
- 26: **End**

2. For every pair (u_1, u_2) with $u_1 \in S$ and $u_2 \in S\Sigma$, if $u_1^{-1}L \neq u_2^{-1}L$ we choose $v \in \Sigma^*$ which distinguishes u_1 from u_2 , that is, v is chosen under the condition that just one of the two words u_1v or u_2v belong to L . We add u_1v and u_2v to $D_+(S)$ or to $D_-(S)$ according to their membership to L .

A rough bound for the size of $D_+(S) \cup D_-(S)$ is easily seen to be quadratic in the size of Q .

There are families of automata for which the amount of prefix closed minimal set of test states grows exponentially with the size of the automaton.

Example 1. For $n \geq 1$ let $\mathcal{A}_n = (\{1, 2, \dots, n+1\}, \{a, b\}, \delta, 1, \{n+1\})$ be the automaton defined as: $\delta(i, c) = i+1$, for $i = 1, \dots, n$, $c \in \{a, b\}$ and $\delta(n+1, a) = \delta(n+1, b) = n+1$.

For every \mathcal{A}_n there exist 2^n prefix closed minimal set of test states.

Proposition 1. [5] *Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a minimum complete automata. Let $S = \{u_0, u_1, \dots, u_n\}$ be a minimal set of test states and let $D_+(S)$ and $D_-(S)$ be the sets obtained as it is shown above. If, in Gold's algorithm, for any*

$i = 0, \dots, n$ the state u_i is considered for promotion before than any other state u such that $u^{-1}L = u_i^{-1}L$, the output is a DFA isomorphic to \mathcal{A} .

Then, if the order in which states to be promoted to B is established as in the above proposition, the characteristic set for Gold's algorithm is of polynomial size. Otherwise it requires a set of exponential size to guarantee identification. To obtain this set, let S_1, S_2, \dots, S_r all the minimal and prefix closed sets of test states. The number r is finite and roughly bounded above by $2^{|\Sigma|^n}$, where $n + 1$ is the number of states of the minimum DFA. Doing $D_+ = \bigcup_{i=1}^r D_+(S_i)$ and $D_- = \bigcup_{i=1}^r D_-(S_i)$ (obviously exponential), we have a characteristic set for Gold's algorithm with no restrictions in the *choose* sentences of the lines 4 and 16.

4 The Blue-Fringe EDSM Algorithm has a Polynomial Characteristic Set

As it has been mentioned above, De la Higuera et al. [6] proposed a general merging states inference algorithm. Aiming to avoid undesired merges of states, particularly those that take place at the first steps of the running of the algorithm, the authors proposed an algorithm that uses a function that establishes the order in which states are selected to be merged.

It is important to note that although the authors claim that the function can implement any ordering in the set of states, the structure of the algorithm makes that the only states that can possibly be merged belong to two disjoint sets: the first one contains the *consolidated* states which will belong to the set of states of the final DFA and the second set contains the states that can be reached from the first one using only one transition. These two sets are usually denominated as the set of Red (R) and Blue (B) states respectively.

The algorithm presented in [6] is used for the authors to prove that in an inference algorithm based in merging states, when the order of the merging is data independent, there is a polynomial characteristic set that makes the algorithm to converge to the target automaton. When the order of merging is data dependent, the existence of a characteristic set polynomial in size is not so clear.

The best known algorithm that implements a function to select the states to be merged is the Blue-Fringe EDSM [8]. It uses a *PTMM* as data structure to manipulate the sample and different training sets may lead to different ordering of the states. Although it has shown a very good experimental behavior, as far as we know, the existence of a characteristic sample has not been proved. In order to prove the existence of this characteristic set, we will first briefly describe the algorithm.

Blue-Fringe EDSM starts from the $PTMM(D_+, D_-) = (Q, \Sigma, \{0, 1, ?\}, \delta, q_0, \Phi)$. Initially $R = \{\lambda\}$ and $B = \Sigma \cap Q$. The algorithm compares every pair $(u, v) \in R \times B$. To make the proof easier, we assume that the states of B are visited in lexicographical order. So, every state q of B , in lexicographical order is compared with every state p of R . If q is distinguishable from every state of R

it is promoted to the set R and afterwards, the set B is recalculated. Otherwise, if no state can be promoted, the pair of states with greater score is merged (the score is assigned using the number of coincidences in the subgraphs that have p and q as initial states). The algorithm continues doing this task until B becomes empty. The merges are done in a deterministic way.

It is easily seen that Blue-Fringe EDSM converges. Let us see that it converges with a polynomial characteristic set.

Proposition 2. *The algorithm Blue-Fringe EDSM has polynomial characteristic set.*

Proof. Let $S = \{u_0, u_1, \dots, u_n\}$ be the minimal prefix-closed set of test states and such that for every i and for every $u \in \Sigma^*$ with $u^{-1}L = u_i^{-1}L$ we have that u_i is previous to u in lexicographical order, that is, S is the set of smallest (in lexicographical order) words that reach every state of the automaton. Let us see that on input of the sets $D_+(S)$ and $D_-(S)$ built as in Definition 1, the algorithm Blue-Fringe EDSM outputs the minimum DFA for L .

1. Blue-Fringe EDSM algorithm promotes to the set R all the states of S and only the states of S before any merging is done:
As the set B is traversed in canonical order, neither state that could be promoted will be considered before its equivalent state in S .
2. After the promotion step we have $R = S$ and $B = S\Sigma - S$. There are no more possible promotions. The number of states in the set B equals the number of transitions left in the subautomaton induced by R . This will be true during the whole process.
3. For definition of $D_+(S)$ and $D_-(S)$, for every state in $B = S\Sigma - S$ there is only a compatible state in $R = S$, so every state of B will be correctly merged independently from the order in which they are processed that is, data has only influence in the order merges are done.
4. Every state in B can only be merged with one in R . Once a union is made, a state in B disappears (and never comes back again). The rest of them, either have the same information they had (rooted subtree) or some of them increase it. In both cases every state in B is distinguished from every state in R except from exactly one of them. This process continues until the set B becomes empty.

$D_+(S)$ and $D_-(S)$ form a characteristic set. If we add new data to $D_+(S)$ and $D_-(S)$, the promotion process of the states of S to the set R (before any merging is done) is not altered.

From this proof it follows that not only the Blue-Fringe EDSM, but any other algorithm based in the Red-Blue strategy will converge with polynomial characteristic sample, under the condition that the promotion of states (in lexicographical order) from B to R is considered before any merging.

For better understanding of the fact that if the elements of B are traversed in lexicographical order then all the promotions are done before the first merge takes place, let us see the following example:

Example 2. Let us consider the automaton of Figure 1 (a). We have, following proposition 2, $S = \{\lambda, a, aa\}$ and thus $S\Sigma = \{a, b, aa, ab, aaa, aab\}$. From those sets we obtain $D_+(S) = \{aa, ab, bb, ba, aaa, aab\}$ and $D_-(S) = \{\lambda, a, b\}$. The prefix tree Moore machine is depicted in figure Figure 1 (b).

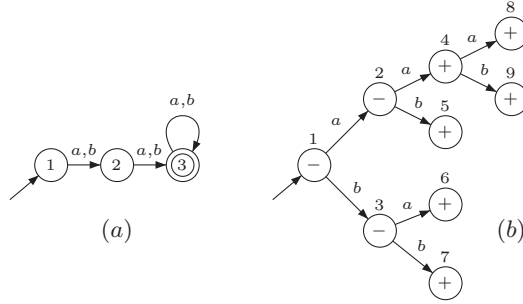


Fig. 1. (a) Starting automaton. (b) The prefix tree Moore machine for $D_+(S) = \{aa, ab, bb, ba, aaa, aab\}$ and $D_-(S) = \{\lambda, a, b\}$.

At the beginning $R = \{1\}$ and thus $B = \{2, 3\}$. As state 2 can not be merged to 1 we have $R = \{1, 2\}$ and then $B = \{3, 4, 5\}$. At this point, state 3 can be merged with 2. Finally, as state 4 can not be merged to states 1 and 2, it is added to R and thus we obtain $R = \{1, 2, 4\}$ and $B = \{3, 5, 8, 9\}$. States 5, 8 and 9 can be merged to 4 and the whole process ends. One should observe the states of S have been promoted to R before doing any merging.

Let us see that adding new data does not affect to the set R . For example, let us suppose that $D_+ = D_+(S) \cup \{baa, baaa, baaaa, baaaab, baba\}$ and $D_- = D_-(S)$. The prefix tree Moore machine is depicted in figure Figure 2, where the new states and transitions are drawn in a dashed way.

The algorithm proceeds exactly as before and thus $R = \{1, 2, 4\}$ and $B = \{3, 5, 8, 9\}$. After merging state 2 with state 3, the latter state disappears from

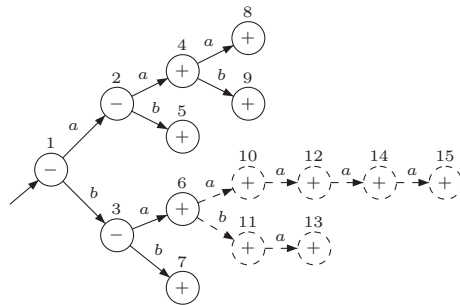


Fig. 2. The prefix tree Moore machine for $D_+ = D_+(S) \cup \{baa, baaa, baaaa, baaaab, baba\}$ and $D_- = D_-(S)$

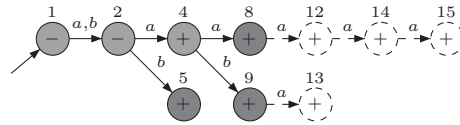


Fig. 3. Moore machine after merging states 2 and 3

B. The resulting automaton at this point is depicted in Figure 3, where states belonging to the sets R and B are depicted in different levels of grey. You should observe that state 5 has the same information it had before, whereas states 8 and 9 have increased it.

5 Generalized Red-Blue Merging Algorithm

The Generalized Red-Blue Merging Algorithm (*GRBM*) is described in Algorithm 2. It starts from the *PTM* M of the sample. First, R is initialized to the initial state of the tree. The set R contains, at every step, those states that will be part of the output hypothesis. The set B is constructed from R . It contains the sons of elements of R which do not belong to R . Next, the algorithm chooses a state of B in an arbitrary way and tries to merge it with any state of R . In case that it can not be merged with any state of R , it is added to R . The set B has to be recalculated in both cases. The algorithm continues processing the input until B becomes empty. Possible merges of the states of the sets R and B are arbitrarily done.

Merges of states in M are done in a deterministic way (merging two states may lead to future mergins to avoid non determinism) using the function *detmerge* (M, p, q). In case that this merging is not consistent it returns M .

The main difference between Gold’s algorithm and *GRBM* is that the latter merges states whereas the former does not. The other difference is that in Golds algorithm states of B which are obviously different from the set R are promoted to R while transitions are only established at the end (analyzing the equivalences between states in B and R).

Both Blue-Fringe EDSM and *GRBM* consider merging of states, but while the former tries to promote before merging, the latter arbitrarily chooses one state from B and tries to merge it with one from R . Only when merges are not possible the state is promoted to R . This fact improves the computational efficiency. The convergence of the algorithm is always guaranteed and, under certain restrictions, (the same as in Gold’s) there exists a polynomial characteristic set.

The following proposition paraphrases Proposition 3 for the new *GRBM* and establishes the conditions for the existence of a characteristic polynomial sample.

Proposition 3. *Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be an minimum automaton. Let $S = \{u_0, u_1, \dots, u_n\}$ be a minimal set of test states and let $D_+(S)$ and $D_-(S)$ be*

Algorithm 2. Generalized Red-Blue Merging($D_+ \cup D_-$)

```

1: Input: Two finite sets ( $D_+$  and  $D_-$ )
2: Output: A consistent Moore Machine
3: Method:
4:  $M := PTMM(D_+, D_-) = (Q, \Sigma, \{0, 1, ?\}, \delta, q_0, \Phi)$ 
5:  $R := \{q_0\}$ 
6:  $B := \{q \in Q : \delta(q_0, a) = q, a \in \Sigma\}$ 
7: while  $B \neq \emptyset$  do
8:   for  $q \in B$  (in arbitrary order) do
9:      $merged := false$ 
10:    for  $p \in R$  (in arbitrary order) do
11:      if  $detmerge(M, p, q) \neq M$  then
12:         $merged := true$ 
13:         $M = detmerge(M, p, q)$ 
14:        break()
15:      end if
16:    end for
17:    if  $\neg merged$  then
18:       $R := R \cup \{q\}$ 
19:    end if
20:     $B := \{q \in Q \mid \delta(p, a) = q, p \in R, a \in \Sigma\} - R$ 
21:  end for
22: end while
23: Return  $M$ 

```

the sets obtained as above. If in algorithm GRBM, for any $i = 0, \dots, n$ the state u_i (when the set B is ordered) is considered before any other u such that $u^{-1}L = u_i^{-1}L$, GRBM outputs a DFA isomorphic to \mathcal{A} .

Proof. Let $S = \{u_0, u_1, \dots, u_n\}$, $u_0 = \lambda$: It is enough to see that at every step we have $R \subseteq S$ and $B \subseteq S\Sigma - S$.

Initially, $R = \{u_0\}$ and $B = \Sigma \cap Pr(D_+ \cup D_-)$.

If the proposition holds up to a certain step, let $u \in B$ be the state to be compared with R . If it is distinguishable from any member of R (the way the set B is traversed allows us to affirm that $u = u_i$ for any i , otherwise if $v \in B$ and $v^{-1}L = u^{-1}L$, v will be processed after). When u_i is promoted to R , $R \subseteq S$ and once B is recalculated, $B \subseteq SA - S$.

If u is not distinguishable from all the states in R , as $u \in S\Sigma - S$ and any member of R is in S , There is only an element of R that can be merged with u (because of the way that $D_+(S)$ and $D_-(S)$ have been constructed). The set R does not change ($R \subseteq S$) and the update of the set B makes $B \subseteq S\Sigma - S$. When the algorithm finishes, $R = S$ and $B = \emptyset$. Besides, for definition of characteristic set, every transition of the automaton \mathcal{A} appear in the output automaton.

If one merges states in an arbitrary way, the characteristic set is exponential, so there is no guarantee that an accurate output for a given input will be obtained. The use of automata teams increases the probability of good results.

6 Experiments

In this section we experiment the behavior of the Algorithm 2 (*GRBM*) and compare it with two previous algorithms that have shown the best recognition rates: the Blue-Fringe EDSM, which outputs *DFAs* and the DeLeTe2 which outputs *NFAs*. The authors of the latter one affirm [3] that DeLeTe2 performs better than Blue-Fringe EDSM when samples are drawn from *NFAs* or regular expressions and that the opposite happens when they are drawn from *DFAs*. We will first describe the data set used in the experiments and then, the protocol and the recognition rates.

The programs used in the experiments are: the software developed by the authors of the DeLeTe2 [3] and a version of the Blue-Fringe EDSM implemented in [1] (which obtains slightly better results than the reported in [3] for the same set of experiments). Concerning the run time of the algorithms, the *GRBM*, which has $\mathcal{O}(k \times n^2)$ time complexity (where k is the number of automata in the team), is faster than both the Blue-Fringe EDSM and the DeLeTe2 algorithms, and that the Blue-Fringe EDSM algorithm is faster than the DeLeTe2.

The data set we use is the corpus of the experiments for the DeLeTe2 [3] together with the *DFAs* of [2], we eliminate the repeated automata and we then obtain 102 regular expressions (re), 120 *NFAs* and 119 *DFAs*. We then generate 500 different training samples of length (randomly) varying from 0 to 18. The percentage of positive and negative samples is not controlled. The average number of states of the automata used is 20 (in case of *NFA*'s the number of states when converted to *DFAs* is around 120), the average size of regular expressions is 8 and the size of the alphabet is 2 (see [3]). These samples are distributed in five incremental sets of size 100, 200, 300, 400 and 500. We also generate 1000 test samples which are different from the training ones. The length of the test words also vary from 0 to 18. The test set is labeled by every automaton and we thus obtain the following groups: er_100, er_200, er_300, er_400, er_500, nfa_100, nfa_200, nfa_300, nfa_400, nfa_500, dfa_100, dfa_200, dfa_300, dfa_400, dfa_500.

Different examples of run of the algorithm *GRBM* may lead to different output automata. We aim to measure the recognition rates of automata teams obtained using *GRBM*.

The protocol considered the languages of the corpus (119 languages obtained from random *DFAs*, 120 from *NFAs* and 102 from regular expressions). For each of the languages, teams of 5, 11, 21, 41 and 81 automata were inferred using training sets of increasing size (100, 200, 300, 400 and 500 samples).

Every team was used to classify the test using the following criteria: fair vote, weighted vote (inverse to the size of the automaton) and use of the smallest automaton. Aiming to obtain statistically uniform results the protocol was repeated 10 times.

The best results, as expected, were obtained using the biggest team (81 automata). Classification done using the fair vote criterium can not be compared to the other criteria. The average results obtained for these teams are shown in Table 1. They are compared with the results of algorithms Blue-Fringe EDSM and DeLeTe2.

Table 1. Comparison of the classification rates of our approach, the Blue-Fringe EDSM and the DeLeTe2 algorithms. The classification rates are established considering weighted vote (% *w.v.*), as well as those obtained by the smallest automata of the team (\downarrow *FA*). Third, fifth and seventh columns show the average of the size of the smallest automata of the GRBM team and the sizes of the automata output by Blue-Fringe EDSM and DeLeTe2 algorithms respectively.

Set	GRBM (81 FA)			Blue-fringe		DeLeTe2	
	% <i>w.v.</i>	% \downarrow <i>FA</i>	\downarrow <i>FA</i>	% rec.	<i>FA</i>	% rec.	<i>FA</i>
er_100	94.50	94.64	7.28	87.94	10.00	91.65	30.23
er_200	98.14	97.94	7.85	94.81	9.97	96.96	24.48
er_300	98.67	98.59	8.39	96.46	11.05	97.80	31.41
er_400	99.16	99.02	8.54	97.74	10.43	98.49	27.40
er_500	99.36	99.27	8.75	98.54	10.47	98.75	29.85
nfa_100	77.08	71.23	17.11	68.15	18.83	73.95	98.80
nfa_200	80.70	74.80	28.18	72.08	28.80	77.79	220.93
nfa_300	83.19	77.14	37.26	74.55	36.45	80.86	322.13
nfa_400	85.01	79.01	44.95	77.53	42.58	82.66	421.30
nfa_500	86.57	80.65	51.84	80.88	47.54	84.29	512.55
dfa_100	76.45	70.32	17.21	69.12	18.59	62.94	156.89
dfa_200	82.10	80.69	24.66	77.18	25.83	64.88	432.88
dfa_300	88.47	91.29	23.02	88.53	25.10	66.37	706.64
dfa_400	93.37	96.62	19.60	94.42	21.36	69.07	903.32
dfa_500	96.76	98.84	17.00	97.88	18.75	72.41	1027.42

Looking for uniform results, two new approaches were used in the experiments. The first one was to consider weighted vote in a way that gives more weight to the smaller automata, and thus the classification was made with a weight parameter inverse to the square of their size. The second approach aimed to select those automata with the right to vote. Several approaches were tried, and the best results were obtained when automata of size smaller than the average size were selected. Once the automata were selected, the classification were done using weighted vote inverse to the square of their size. The results are shown in Table 2 comparing them with the results of Blue-Fringe EDSM and DeLeTe2 algorithms.

Both approaches obtain better classification rates than the Blue-Fringe EDSM. Note that the number of selected automata with right to vote is near half of the size of the team. This number increases when we have much information about the target language (languages obtained from regular expressions).

Figure 4 shows the comparison of the performance of the automata teams used in the experiments with Blue-Fringe EDSM and DeLeTe2 algorithms. We have used teams of 5, 11, 21, 41 and 81 automata, although we only show the results of some of them to avoid confusion. The election of the number of automata in each team was simple (we started with 5, a small odd number and continued multiplying the number of automata times 2) It is also worth to be noted that the performance of the Blue-Fringe EDSM and DeLeTe2 algorithms is always worse than *GRBM*, except for the case of teams of 5 automata.

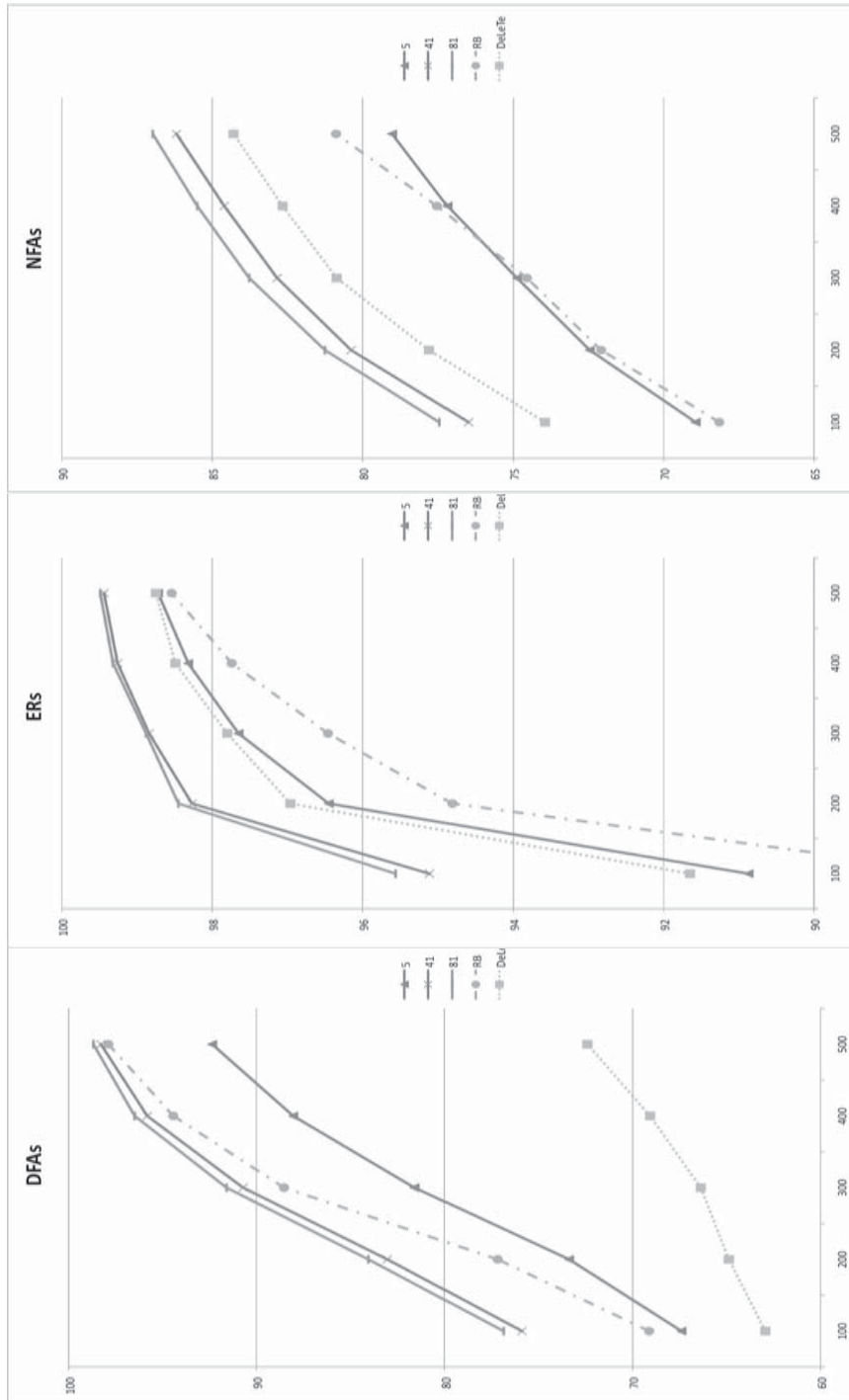


Fig. 4. Comparative behavior between our approach (GRBM), RedBlue and DeLeTe2 algorithms

Table 2. Results obtained in the second set of experiments. GBRM classification rates consider weighted vote inverse to the square of the sizes of the automata (first and second columns). Second and third columns show respectively the classification rates and the number of automata selected when only automata smaller than the average size are selected for classification purposes.

Set	GBRM (81 FA)			Blue-Fringe	DeLeTe
	no select.	sel. FA size smaller average			
	$\%w.v.^2$	$\%w.v.^2$	$\#FA$		
er_100	95.19	95.55	50.00	87.94	91.65
er_200	98.32	98.45	58.24	94.81	96.96
er_300	98.78	98.87	60.58	96.46	97.80
er_400	99.24	99.32	63.45	97.74	98.49
er_500	99.42	99.49	65.91	98.54	98.75
nfa_100	77.24	77.45	40.51	68.15	73.95
nfa_200	80.96	81.25	41.28	72.08	77.79
nfa_300	83.46	83.77	41.26	74.55	80.86
nfa_400	85.14	85.50	41.47	77.53	82.66
nfa_500	86.71	86.98	42.25	80.88	84.29
dfa_100	76.68	76.83	40.48	69.12	62.94
dfa_200	83.13	84.04	36.00	77.18	64.88
dfa_300	90.04	91.59	36.00	88.53	66.37
dfa_400	95.24	96.48	36.31	94.42	69.07
dfa_500	98.16	98.68	37.69	97.88	72.41

7 Conclusions

In this paper we propose the algorithm *GRBM* for automata inference. It uses the Red-Blue strategy to divide the states of the *PTMM* of the sample in two sets. The order in which states belonging to those sets try to be merged is arbitrarily chosen. This algorithm, under certain conditions, converges with a polynomial characteristic set. As a byproduct, the existence of a polynomial characteristic set for the previous Blue-Fringe EDSM algorithm has also been proved. Different outputs obtained by different examples of run of *GRBM* may lead to different outputs. This fact has been used for learning using automata teams. Experiments done using different classification criteria for the test sets improve the classification rates obtained by both the Blue-Fringe EDSM and the DeLeTe2 algorithms.

References

1. Álvarez, G.I.: Estudio de la Mezcla de Estados Determinista y No Determinista en el Diseño de Algoritmos para Inferencia Gramatical de Lenguajes Regulares. PhD. Thesis DSIC UPV (2008)

2. Coste, F., Fredouille, D.: Unambiguous automata inference by means of state merging methods. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) ECML 2003. LNCS (LNAI), vol. 2837, pp. 60–71. Springer, Heidelberg (2003)
3. Denis, F., Lemay, A., Terlutte, A.: Learning regular languages using RFSAs. *Theoretical Computer Science* 313(2), 267–294 (2004)
4. Gold, E.M.: Language identification in the limit. *Information and Control* 10, 447–474 (1967)
5. Gold, E.M.: Complexity of Automaton Identification from Given Data. *Information and Control* 37, 302–320 (1978)
6. de la Higuera, C., Oncina, J., Vidal, E.: Data dependant vs data independant algorithms. In: Miclet, L., de la Higuera, C. (eds.) ICGI 1996. LNCS (LNAI), vol. 1147, pp. 313–325. Springer, Heidelberg (1996)
7. Lang, K.J.: Random DFAs can be Approximately Learned from Sparse Uniform Examples. In: *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pp. 45–52 (1992)
8. Lang, K.J., Pearlmutter, D., Price, R.A.: Results of the Abbadingo one DFA Learning Competition and a new evidence-driven state merging algorithm. In: Honavar, V.G., Slutzki, G. (eds.) ICGI 1998. LNCS (LNAI), vol. 1433, pp. 1–12. Springer, Heidelberg (1998)
9. Oncina, J., García, P.: Inferring Regular Languages in Polynomial Updated Time. In: de la Blanca, P., Sanfeliú, Vidal (eds.) *Pattern Recognition and Image Analysis*. World Scientific, Singapore (1992)