Generating Small Automata and the Černý Conjecture

Andrzej Kisielewicz* and Marek Szykuła**

Department of Mathematics and Computer Science, University of Wrocław

Abstract. We present a new efficient algorithm to generate all nonisomorphic automata with given numbers of states and input letters. The generation procedure may be restricted effectively to strongly connected automata. This is used to verify the Černý conjecture for all binary automata with $n \leq 11$ states, which improves the results in the literature. We compute also the distributions of the length of the shortest reset word for binary automata with $n \leq 10$ states, which completes the results reported by other authors.

Keywords: Černý conjecture, synchronizing word, nonisomorphic automata.

We consider deterministic finite automata $A = \langle Q, \Sigma, \delta \rangle$, where Q is the set of the states, Σ is the input alphabet, and $\delta : Q \times \Sigma \to Q$ is the (complete) transition function. The cardinality $n = |Q|$ is the *size* of A, and if $k = |\Sigma|$ then A is called *k-ary*.

If there exists a w such that the image of Q by w consists of a single state, then w is called a *reset* (or *synchronizing*) word for A, and A itself is called *synchronizing*. The length of a shortest reset word of A is called its *reset length*.

The Černý conjecture states that every synchronizing automaton A with n states has a reset word of length $\leq (n-1)^2$. This conjecture was formulated by Černý in 1964, and is considered the longest-standing open problem in combinatorial theory of finite automata. So far, the conjecture has been proved only for a few special classes of automata and a cubic upper bound has been established (see Volkov [19] for an excellent survey). It is known (and not difficult to prove) that to verify the conjecture it is enough to consider only *strongly connected* automata, that is, those whose underlying digraph is strongly connected.

Trahtman [17,18] reports that, using a computer program, he has verified the Černý conjecture for all strongly connected k-ary automata of size n with $k = 2$ and $n \leq 10$, $k \leq 4$ and $n \leq 7$, and $k = 3$ and $n = 8$. Unfortunately, no method of generating such automata is described and no details of computations are given. There are 10^{20} binary automata of size $n = 10$, and it is out of reach of the present computer technology to generate all of them, so some methods to generate only

* Supported in part by Polish MNiSZW grant N N201 543038.

** Supported in part by NCN grant number 2011/01/D/ST6/07164, 2011–2014, and by a scholarship co-financed by an ESF project *Human Capital*.

S. Konstantinidis (Ed.): CIAA 2013, LNCS 7982, pp. 340–348, 2013.

strongly connected automata (or a restricted class containing all the strongly connected automata) must be used. Such a method is described in [1], the authors restrict themselves to the class of *initially-connected* automata (with each state reachable from a single start state), using a special string representation for such automata and parallel programming. With these tools, they are able to verify the Černý conjecture only for binary automata with $n \leq 9$ states. (For 9 states, there are about 700 billions initially-connected automata with 2 input letters.)

The theoretical part of Trahtman's work [17,18] is devoted mainly to the problem of efficiently finding the shortest (or a short) reset word. A number of good algorithms are known at present for solving this problem (see [7] and references given therein). We found however that the main problem arising in verifying the Černý conjecture for small automata is to overcome somehow the huge number of automata involved rather than to compute the reset length fast. Ideally, one would like to consider only all nonisomorphic strongly connected automata for such verification, but no efficient method to generate only automata from this class is known. There are formulas enumerating the number of nonisomorphic automata (see [6] and [5,11,13]), and methods to enumerate nonisomorphic strongly connected automata ([8,14]) Unfortunately, the ways they approach the problem do not seem useful in the task of efficient generation of the objects.

In this paper we present a new algorithm to generate efficiently all nonisomorphic automata with given numbers of states and letters, and to compute the reset length for them. The method can be extended to generate only specific classes of automata without much additional cost. In particular, a version of the algorithm generates all nonisomorphic strongly connected automata. While the algorithm still produces isomorphic copies (and some not strongly connected automata for the second version), it greatly reduces the number of considered automata as well as the overall computation cost. Also we are able to speed-up computation of reset length making use of the specific properties of the generating method.

Our method allows us us to verify and extend the known computational results. In particular, we prove that the Černý conjecture is true for all binary automata with $n \leq 11$ states. We obtain complete distributions of the reset length for all automata of size $n \leq 10$. For $n = 11$ a new gap in the distribution is observed, leading to a new conjecture concerning reset lengths.

1 Generating Automata

The algorithm is recursive. Given $n > 1$, we use known lists of all nonisomorphic automata of size n and arity 1 (which are equivalent to certain digraphs). For $k \geq 2$, having two lists of all nonisomorphic automata of size n and of arity $k-1$ and 1, respectively, our algorithm generates a list of automata of size n and arity k. To this aim, for each pair of automata, A from the first list, and B from the second list, a special procedure, called Permutation procedure, is applied. It (1) takes as an input the pair of automata A and B, from the first and the second list, respectively, (2) generates all automata isomorphic to B (by permutations of the states of B), and (3) matches each resulting automaton with A. In this

way, we obtain all the automata of arity k whose restriction to the first $k-1$ letters is isomorphic to A, while the restriction to the last letter is isomorphic to B. Matching all the pairs A, B, we obtain all nonisomorphic k-ary automata of size n. Yet, many of these automata may appear in isomorphic copies.

Using more specific ideas we design a few variant of the algorithm with different task. They generate all nonisomorphic automata of a given size either without isomorphic pairs or (for lower computational cost) with the number of such pairs relatively small. We also show how the generation process can be restricted effectively to strongly connected automata. The latter is used to verify the Černý conjecture for automata of a given size. Because of the space limit, in this paper, we describe only the theoretical aspects of the procedure, called Permutation procedure, which is designed to skip efficiently permutations of B leading to isomorphic copies. Other variants and the details of the algorithm will be given in the extended version of the paper.

1.1 Permutation Procedure

We say that two automata $A = \langle Q_A, \Sigma_A, \delta_A \rangle$ and $B = \langle Q_B, \Sigma_B, \delta_B \rangle$, are *isomorphic*, if there exist two bijections $\phi : Q_A \to Q_B$ and $\psi : \Sigma_A \to \Sigma_B$ such that for all $q \in Q_A$ and $a \in \Sigma_A$

$$\phi(\delta_A(q, a)) = \delta_B(\phi(q), \psi(a)). \tag{1}$$

In other words, isomorphic automata are equal up to renaming the states and the letters. In particular, two isomorphic automata have the same reset lengths, and the classes of shortest reset words differ only up to renaming the letters (given by ψ).

We note that various authors use various terminology here. For example, Harrison [6] calls such automata *equivalent with respect to input permutations*, and reserves the term "isomorphic automata" for the situation when the bijection ψ in (1) is the identity. If $\Sigma_A = \Sigma_B$, and A and B are isomorphic with ψ being the identity, we will say that A and B over the same alphabet are *strongly isomorphic*. Then the bijection ϕ itself is called a *strong isomorphism* or simply *isomorphism* (meaning that it forms an isomorphism itself with the second bijection being the identity). In the case, when $A = B$, ϕ is called an *automorphism*.

We consider now an automorphism that fixes the states in a given set. For an automaton $A = \langle Q, \Sigma, \delta \rangle$, and a subset S of Q, we say that the states $u, v \in Q$, $u, v \notin S$ are *conjugate under* S, and write $u \overset{S}{\simeq} v$, if there exists a (strong) automorphism $\phi : Q \to Q$ such that

$$\begin{aligned} &\phi(w) = w \text{ for each } w \in S, \\ &\phi(u) = v. \end{aligned} \tag{2}$$

Figure 1 shows an example of an automaton over a one-letter alphabet with some states conjugate under $S = \emptyset$ and $S = \{5\}$. Namely, we have $1 \overset{S}{\simeq} 2$ and $3 \overset{S}{\simeq} 4$ for $S \subseteq \{5\}$. Note that there are no two different conjugate states under any other S in this automaton.

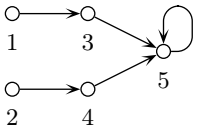

Fig. 1. An unary automaton with nontrivial automorphisms and conjugate states

The following facts are routine to prove.

Lemma 1. *For each $S \subseteq Q$, the conjugation under S is an equivalence relation. Moreover, if $R \subseteq S$, then the relation $\overset{R}{\simeq} \supseteq \overset{S}{\simeq}$.*

Checking whether two states are S-conjugate may be done by computing the corresponding group of automorphisms (fixing the states from S) or by generating and checking permutations with suitably prescribed images for S and u. During the generation of permutations, natural conditions for a permutation to be an automorphism, such as equality of indegrees, may be taken into account. Although it has an exponential cost in the worst case, our experiments show that for most of automata of small size, it works pretty fast.

In our procedure, to be able to skip superfluous permutations effectively we do some preprocessing. We assume that the set of states of the automata on both the input lists is $Q = \{1, 2, \ldots, n\}$. Before running the procedure, the following structures are created for each pair A and B of automata (with $k-1$ and 1 letter alphabet, respectively):

1. The structure **PrevB**. For each of the 2^n subsets $S \subseteq Q$ and for each $j \in Q \setminus S$, **PrevB**$[S][j]$ contains *true*, if and only if there exists some state $h \in Q \setminus S$ $(1 \leq h < j)$ for which $h \overset{S}{\simeq} j$ in B.
2. The structure **PrevA**. For each i $(1 \leq i \leq n)$ the entry **PrevA**$[i]$ contains the largest index h $(1 \leq h < i)$ such that $i \overset{S_h}{\simeq} h$ in A with $S_h = \{1, \ldots, h-1\}$. It is possible that the index does not exist.

The first structure requires computing automorphisms of B for as many as $2^n\binom{n}{2}$ conditions (in the worst case) fixing a set S and unordered pair $\{i, j\}$ with $i, j \notin S$. For each automaton B (which is of arity 1) we compute it only once and then we process all the pairs with B. The second structure requires computing automorphisms only for $\binom{n}{2}$ pairs of states (determining the set of fixed elements). For small n, this preprocessing can be done quickly and takes only a negligible amount of time compared with processing the resulting automata.

Let $A = \langle Q, \Sigma_A, \delta_A \rangle$ and $B = \langle Q, \Sigma_B, \delta_B \rangle$ be two automata with $Q = \{1, 2, \ldots, n\}$, $|\Sigma_A| = k - 1$ for some $k > 1$, and $\Sigma_B = \{b\}$ for some $b \notin \Sigma_A$. Let π be a permutation of Q. Then, by $U(A, B, \pi)$ we denote the automaton $\langle Q, \Sigma_A \cup \Sigma_B, \delta \rangle$, where δ is an extension of δ_A given by

$$\delta(q, b) = \pi^{-1}(\delta_B(\pi(q), b)). \tag{3}$$

We call it the (disjoint) *union of A and B under permutation π*. The condition $b \notin \Sigma_A$ is purely technical, so we may assume it always without further mention. Note that this construction may be viewed as identifying each state q in B with the state $\pi^{-1}(q)$ in A. An example is given in Figure 2 (loops are omitted).

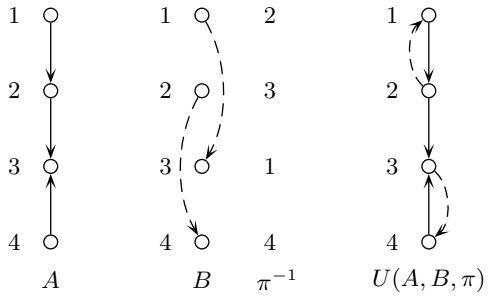

Fig. 2. The union $U(A, B, \pi)$ with $\pi = (1, 3, 2)$

The main part of our algorithm is the PermutB procedure presented as Algorithm 1. It takes as the input two automata A and B on $Q = \{1, 2, \ldots, n\}$, with alphabets of arity $k - 1$ and 1, respectively. It starts from the empty (partial) permutation $\pi_0 = \emptyset$ and extends it in a recursive manner. A partial permutation π_i is an injective function from $\{1, \ldots, i\}$ to $\{1, \ldots, n\}$. For each complete permutation π_n the automaton $U(A, B, \pi_n)$ is generated. The permutations are generated in the lexicographical order subject to two restrictions (reducing the number of isomorphic automata):

1. Let $h = \mathbf{PrevA}[i{+}1]$. If such h exists and $\pi_i(h) > j$ then matching $(i{+}1) \to j$ is skipped, since (as we prove below) a suitable isomorphic automaton has been generated earlier. This results in starting the corresponding "for loop" from $m = \pi_i(h) + 1$.
2. For each j, if $\mathbf{PrevB}[S_B][j]$ is true then matching $(i + 1) \to j$ is skipped. Again, we will prove that a suitable isomorphic automata have been already generated.

In the theorem below we use the notation $A|_\Gamma$ for the automaton obtained from $A = \langle Q, \Sigma_A, \delta_A \rangle$ by restricting its alphabet to a subset Γ of Σ.

Theorem 1. *Let $A = \langle Q, \Sigma_A, \delta_A \rangle$ and $B = \langle Q, \Sigma_B, \delta_B \rangle$ be two automata with disjoint alphabets Σ_A and Σ_B (where $|\Sigma_B| = 1$) and the same set of states $Q = \{1, 2, \ldots, n\}$. Then, for each automaton C over the alphabet $\Sigma_A \cup \Sigma_B$ such that $C|_{\Sigma_A} \cong A$ and $C|_{\Sigma_B} \cong B$,* PermutB$(0, \emptyset, \emptyset)$ *generates at least one isomorphic copy of C.*

The proof will be given in the extended version of the paper.

Algorithm 1. Permutation Procedure

Require: A, B – the input automata.
Require: **PrevA**, **PrevB** – preprocessed structures.
procedure PERMUTB(i,π_i,S)
  if $i = n$ **then**
    Report automaton $U(A, B, \pi_n)$ – the union of A and B under π_n
  else
    $m \leftarrow \pi_i(\mathbf{PrevA}[\mathrm{i}+1]) + 1$, or $m \leftarrow 1$ if **PrevA**[i + 1] does not exist.
    for $j = m, \ldots, n$ **do**
      if $j \notin S$ **and not PrevB**[S][j] **then**
        Extend π_i to $\pi_{i+1}(x)$ putting $\pi_{i+1}(i+1) = j$.
        PERMUTB($i+1$,π_{i+1},$S \cup \{j\}$)
      end if
    end for
  end if
end procedure

2 Some Experimental Results

The problem of computing the reset length of an automaton is computationally hard (see [12], and [3] for approximating hardness). In spite of this the exponential algorithms used so far can work efficiently enough. Yet, they can vary in efficiency for different automata (see [7,9,15,17,19]).

To compute the reset length for each of the generated automata we use the standard BFS algorithm in the power automaton with storing visited subsets of states in an array (see [15,19,9]), and with preprocessing transitions (computing the images of subsets) allowing faster computations for a huge number of automata. We have found this the fastest method for considered small n values when using bit-vector encoding for sets, allowing to represent them as integers. It can also report that an automaton is not synchronizing, without separately using the standard synchronization checking algorithm on the pair automaton ([4,17]). Further technical improvements applied are described in the extended version.

We have computed the exact numbers of all nonisomorphic binary automata and those strongly connected and/or synchronizing for sizes $n \leq 10$. Also complete distributions of the reset length in this range are computed. Our results confirm all the results reported in [1] and particular facts formulated in [17]. For $n = 11$ we have computed a partial distribution proving, in particular, that all binary DFA of size 11 satisfy the Černý conjecture. We plan also to perform similar computations for $k > 2$.

2.1 The Number of Nonisomorphic Automata

The results up to 10 states are shown in Table 1. The total number of DFA is known due to the formula in [6], and we have obtained computationally exactly the same numbers. We have computed also the numbers of synchronizing DFA,

strongly connected, and the number of synchronizing strongly connected DFA. The numbers of nonisomorphic strongly connected DFA on 2 labeled letters (up to strong isomorphism) have been considered in [10] (up to $n \leq 6$). They are about 2 times larger than those with unlabeled (for example, there are 658,885 such DFA for $n = 6$).

We can see that the fraction of synchronizing DFA to all DFA grows, and we may conjecture that it tends to 1 as it has been conjectured for the labeled model (P. Cameron and [16]). This growth is more rapid in strongly connected DFA; the corresponding fraction here is about 0.999 for $n = 10$.

Table 1. The exact numbers of nonisomorphic binary DFA of size n in the classes of all, synchronizing, strongly connected, and strongly connected synchronizing DFA. In the last column there is the fraction of the number of synchronizing DFA to all DFA.

n	Total	Synchronizing	Strongly connected	S. c. and synchronizing	Synch./Total
2	7	4	4	2	0.57
3	74	51	29	21	0.69
4	1,474	1,115	460	395	0.76
5	41,876	34,265	10,701	10,180	0.82
6	1,540,696	1,318,699	329,794	322,095	0.86
7	68,343,112	60,477,844	12,310,961	12,194,323	0.88
8	3,540,691,525	3,210,707,626	538,586,627	536,197,356	0.91
9	209,612,916,303	193,589,241,468	26,959,384,899	26,904,958,363	0.92
10	13,957,423,192,794	13,070,085,476,528	1,518,185,815,760	1,516,697,994,964	0.94

Let us compare our method of generating all strongly connected DFA with that of [1,2] by generating of all ICDFA (initially connected DFA). There are about 7×10^{11} and 4.4×10^{13} of ICDFA with $n = 9$ and $n = 10$ states, respectively. In our method we have generated only about 3×10^{10} and 1.7×10^{12} DFA in these cases. In fact there are about 2.7×10^{10} and 1.5×10^{12} nonisomorphic strongly connected DFA, so in our method the relative number of extra generated DFA is really low. This is confirmed by statistics we have made.

2.2 The Distribution of Reset Lengths for $n = 10$.

Since generating automata for each pair in Algorithm 1 can be computed independently we performed paralleled computations on a small computer grid. Our computations have been done on 16 computers with Intel(R) Core(TM) i7-2600 CPU 3.40GHz 4 cores and 16GB of RAM. Computing the complete distribution for all DFA with $n = 10$ states took above 800 days of total CPU time

Table 2. The exact numbers $N(\ell)$ of all and $N_{\mathrm{sc}}(\ell)$ of strongly connected nonisomorphic binary automata of size 10 with the shortest reset word of length $\ell \geq 56$

ℓ	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81
$N(\ell)$	607	369	168	49	18	10	8	9	106	21	3	0	0	0	0	0	2	1	1	0	0	0	0	0	0	1
$N_{\mathrm{sc}}(\ell)$	343	160	58	38	18	10	8	9	18	10	3	0	0	0	0	0	2	1	1	0	0	0	0	0	0	1

($\sim$ 13 days of paralleled computations). Restricting to the class of strongly connected DFA reduced this time to about 80 days of CPU ($\sim$ 2 days of paralleled computations).

2.3 The Distribution of Reset Lengths for $n = 11$.

In order to verify the Černý conjecture for all binary DFA of size $n = 11$ it is sufficient to restrict the tested class of DFA to strongly connected. We have not obtained the complete distribution of reset lengths because of the huge number of DFA. In this case, we were performing the isomorphism test only for DFA with long reset length. We have also excluded the automata with a single synchronizing letter. The number of remaining strongly connected DFA we have to check was 79,246,008,127,339. The total CPU time of this experiment was above 4 years ($\sim$ 25 days of parallel computations). Note that for $n = 11$ there are about 3×10^{15} of ICDFA, so we really needed a different method than that used in [1].

Table 3. The exact numbers $N_{\mathrm{sc}}(\ell)$ of strongly connected nonisomorphic binary automata of size 11 with the reset length $\ell \geq 76$

ℓ	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
$N_{\mathrm{sc}}(\ell)$	3	2	0	0	9	22	12	2	1	0	0	0	0	0	3	2	1	0	0	0	0	0	0	0	1
Classes			–	–			$\mathscr{D}_n$ $\ldots$	$\mathscr{H}_n$ $\dot{\mathscr{H}}_n$	$\mathscr{G}_n$	–	–	–	–	–	$\mathscr{E}_n$ $\mathscr{D}''_n$ $\mathscr{B}_n$	$\mathscr{W}_n$ $\mathscr{F}_n$	$\mathscr{D}'_n$	–	–	–	–	–	–	–	$\mathscr{C}_n$

Table 3 presents the obtained exact numbers of all nonisomorphic binary DFA of size $n = 11$ with large reset lengths. Also some slowly synchronizing DFA classes are presented in the table. The notation here follows [1] (this topic is discussed in more detail in the extended version of the paper). The most interesting observation is a gap between $\ell = 77$ and 80: there exist no binary automaton of size $n = 11$ with the reset length equal to 78 or 79. First, Trahtman [17] noted that the reset length $(n-1)^2$ corresponding to the class of the Černý automata is separated from the second large reset length by a gap (in the classes of considered DFA of small size). Then the authors of [1] observed that there is a second gap in the distribution for $n = 9$. They called the DFA between the two gaps *slowly synchronizing*. There is no other gap for $n \leq 10$.

We suppose that this kind of irregularity in the upper part of the reset length distributions occur also for larger numbers of states and that more gaps for larger number of states appear. We state the following:

Gap Conjecture. *For any natural number $g \geq 1$, there exists a big enough natural number n such that there are at least g gaps in the distribution of the reset length of all binary automata of size n.*

References

1. Ananichev, D., Gusev, V., Volkov, M.: Slowly synchronizing automata and digraphs. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 55–65. Springer, Heidelberg (2010)
2. Ananichev, D., Gusev, V., Volkov, M.: Primitive digraphs with large exponents and slowly synchronizing automata. In: Zapiski Nauchnyh Seminarov POMI (Kombinatorika i Teorija Grafov. IV), vol. 402, pp. 9–39 (2012) (In Russian)
3. Berlinkov, M.: Approximating the minimum length of synchronizing words is hard. In: Ablayev, F., Mayr, E.W. (eds.) CSR 2010. LNCS, vol. 6072, pp. 37–47. Springer, Heidelberg (2010)
4. Eppstein, D.: Reset sequences for monotonic automata. SIAM Journal on Computing 19, 500–510 (1990)
5. Harary, F., Palmer, E.M.: Graphical Enumeration. Academic Press (1973)
6. Harrison, M.: A census of finite automata. Canadian Journal of Mathematics 17, 100–113 (1965)
7. Kisielewicz, A., Kowalski, J., Szykuła, M.: A Fast Algorithm Finding the Shortest Reset Words. In: Du, D.-Z., Zhang, G. (eds.) COCOON 2013. LNCS, vol. 7936, pp. 182–196. Springer, Heidelberg (2013)
8. Koršunov, A.D.: On the number of non-isomorphic strongly connected finite automata. Journal of Information Processing and Cybernetics 22(9), 459–462 (1986)
9. Kudałcik, R., Roman, A., Wagner, H.: Effective synchronizing algorithms. Expert Systems with Applications 39(14), 11746–11757 (2012)
10. Liskovets, V.A.: Enumeration of non-isomorphic strongly connected automata. Vesci Akad. Navuk BSSR Ser. Fiz.-Téhn. Navuk 3, 26–30 (1971)
11. Liskovets, V.A.: Exact enumeration of acyclic deterministic automata. Discrete Applied Mathematics 154(3), 537–551 (2006)
12. Olschewski, J., Ummels, M.: The complexity of finding reset words in finite automata. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 568–579. Springer, Heidelberg (2010)
13. Read, R.C.: A note on the number of functional digraphs. Mathematische Annalen 143, 109–110 (1961)
14. Robinson, R.W.: Counting strongly connected finite automata. In: Graph Theory with Applications to Algorithms and Computer Science, pp. 671–685 (1985)
15. Sandberg, S.: Homing and synchronizing sequence. In: Broy, M., Jonsson, B., Katoen, J.-P., Leucker, M., Pretschner, A. (eds.) Model-Based Testing of Reactive Systems. LNCS, vol. 3472, pp. 5–33. Springer, Heidelberg (2005)
16. Skvortsov, E., Tipikin, E.: Experimental study of the shortest reset word of random automata. In: Bouchou-Markhoff, B., Caron, P., Champarnaud, J.-M., Maurel, D. (eds.) CIAA 2011. LNCS, vol. 6807, pp. 290–298. Springer, Heidelberg (2011)
17. Trahtman, A.N.: An efficient algorithm finds noticeable trends and examples concerning the Černy conjecture. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 789–800. Springer, Heidelberg (2006)
18. Trahtman, A.N.: Modifying the upper bound on the length of minimal synchronizing word. In: Owe, O., Steffen, M., Telle, J.A. (eds.) FCT 2011. LNCS, vol. 6914, pp. 173–180. Springer, Heidelberg (2011)
19. Volkov, M.V.: Synchronizing automata and the Černý conjecture. In: Martín-Vide, C., Otto, F., Fernau, H. (eds.) LATA 2008. LNCS, vol. 5196, pp. 11–27. Springer, Heidelberg (2008)