On two Algorithmic Problems about Synchronizing Automata

M. V. Berlinkov

Institute of Mathematics and Computer Science
Ural Federal University, 620000 Ekaterinburg, Russia
berlm@mail.ru

Abstract. We consider two basic problems arising in the theory of synchronizing automata: deciding, whether or not a given $n$-state automaton is synchronizing and the problem of approximating the reset threshold for a given synchronizing automaton.

For the first problem of deciding whether or not a given $n$-state automaton is synchronizing we present an algorithm based on [5] with linear in $n$ expected time, while the best known algorithm is quadratic on each instance.

For the second problem, we prove that unless $P =NP$, no polynomial time algorithm approximates the reset threshold for a given $n$-state 2-letter automata within performance ratio less than $0.5c\ln(n)$ where $c$ is a specific constant from [1]. This improves the previous result of the author [4] about non-approximability within any constant factor and also gives the positive answer to the corresponding conjecture from [8].

1 Preliminaries

A complete deterministic finite automata (DFA) $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is called synchronizing if there exists a word $w \in \Sigma^*$ such that $\delta(q, w) = \delta(p, w)$ for all $q, p \in Q$. Each word $w$ with this property is said to be a reset word for $\mathcal{A}$. The minimum length of reset words for $\mathcal{A}$ is called the reset threshold of $\mathcal{A}$ and is denoted by $rt(\mathcal{A})$. The reset threshold can be thought of as a natural complexity measure for synchronizing automata, and its estimation in terms of the state number has been widely studied since the pioneering paper by Černý [6].

Complete DFA serve as transparent and natural models of discrete reactive systems in many applications (coding theory, robotics, testing of reactive systems) and also reveal interesting connections with symbolic dynamics and other parts of mathematics. Synchronizing automata

* Supported by the Russian Foundation for Basic Research, grant 13-01-00852, and by the Presidential Program for young researchers, grant MK-266.2012.1.
serve as models of error-resistant systems, because applying a reset word reestablish the control on the current system state.

Thus, the primary question here is whether or not a given automaton \( \mathcal{A} \) is synchronizing. The following synchronization criterion has been presented in the pioneering paper about synchronizing automata.

**Proposition 1 (Černý [6]).** An automaton \( \mathcal{A} \) is synchronizing if and only if for each pair of states \( p, q \in Q \) there is a word \( v \) which merges these states, i.e. \( p.v = q.v \).

This criterion can be verified by running *Breadth First Search* (BFS) from the diagonal set \( \{ (q, q) \mid q \in Q \} \) by reverse arrows in the square automaton \( \mathcal{A}^2 = (Q \times Q, \Sigma, \delta^2) \) where \( \delta^2((p, q), a) = (\delta(p, a), \delta(q, a)) \) for all \( p, q \in Q, a \in \Sigma \). Let us call this algorithm \textit{IsSynch}. Since \( \mathcal{A}^2 \) has \( |\Sigma|^n \) arrows, this algorithm is quadratic in time and space. Notice that this algorithm is quadratic in \( n \) for each automaton whence it is expected time for random automata is also quadratic. In Section 2 we present an algorithm based on the proof from [5] having linear expected time. Moreover, we show this algorithm has optimal expected time up to a constant factor.

If \( \mathcal{A} \) is synchronizing, the next natural problem is to calculate its reset threshold. It is known that a precise calculation of the reset threshold is computationally hard. Olschewski and Ummels [10] proved that the problem of deciding, given a synchronizing automaton \( \mathcal{A} \) and a positive integer \( \ell \), whether or not \( rt(\mathcal{A}) = \ell \) is complete for the complexity class \( \text{DP} \) (Difference polynomial time). This class consists of languages of the form \( L_1 \cap L_2 \), where \( L_1 \) is a language from \( \text{NP} \) and \( L_2 \) is a language in \( \text{co-NP} \), and strictly contains \( \text{NP} \), unless \( \text{NP} = \text{co-NP} \). Thus, if \( \text{NP} \neq \text{co-NP} \), even non-deterministic algorithms cannot decide in polynomial time whether or not the reset threshold of a given synchronizing automaton is equal to a given integer. Olschewski and Ummels [10] also proved that the problem of computing the reset threshold (as opposed to deciding whether it is equal to a given integer) is complete for the functional analogue \( \text{FP}^{\text{NP}[\log]} \) of the class \( \text{P}^{\text{NP}[\log]} \) consisting of all problems solvable by a deterministic polynomial time Turing machine that has an access to an oracle for an \( \text{NP} \)-complete problem, with the number of queries being logarithmic in the size of the input. The class \( \text{DP} \) is contained in \( \text{P}^{\text{NP}[\log]} \) (in fact, for every problem in \( \text{DP} \) only two oracle queries suffice) and the inclusion is believed to be strict.

There are some polynomial time algorithms that, given a synchronizing automaton, find a reset word for it, see, e.g., [7]. These algorithms can
be used as approximation algorithms for calculating the reset threshold, and it is quite natural to ask how good such a polynomial approximation can be. The quality of an approximation algorithm is measured by its performance ratio, which for our problem can be defined as follows. Let $K$ be a class of synchronizing automata. We say that an algorithm $M$ approximates the reset threshold in $K$ if, for an arbitrary DFA $\mathcal{A} \in K$, the algorithm calculates an integer $M(\mathcal{A})$ such that $M(\mathcal{A}) \geq rt(\mathcal{A})$. The performance ratio of $M$ at $\mathcal{A}$ is $R_M(\mathcal{A}) = \frac{M(\mathcal{A})}{rt(\mathcal{A})}$. The author [3] proved that, unless $P = NP$, for no constant $r$, a polynomial time algorithm can approximate the reset threshold in the class of all synchronizing automata with 2 input letters with performance ratio less than $r$.

When no polynomial time approximation within a constant factor is possible, the next natural question is whether or not one can approximate within a logarithmic factor. Gerbush and Heeringa [8] conjectured that if $P \neq NP$, then there exists $\alpha > 0$ such that no polynomial time algorithm approximating the reset threshold in the class of all synchronizing automata with a fixed number $k > 1$ of input letters achieves the performance ratio $\alpha \log |Q|$ at all DFAs $(Q, \Sigma)$. Using a reduction from the problem Set-Cover and a powerful non-approximation result from [1], Gerbush and Heeringa proved a weaker form of this conjecture when the number of input letters is allowed to grow with the state number.

Here in Section 3 we prove the conjecture from [8] in its full generality, for each fixed size $k$ of the input alphabet. Though we depart from the same reduction from Set-Cover as in [8], we use not only the result from [1], but also some ingredients from its proof, along with an encoding of letters in states. We conjecture that the obtained bound is already tight in the sense that there exists a polynomial time algorithm that achieves the performance ratio $\beta \log |Q|$ at all synchronizing automata $(Q, \Sigma)$ with $|\Sigma| = k > 1$ for some constant $\beta > \alpha$.

2 The probability of being synchronizable and related algorithm

Let $Q$ stand for $\{1, 2, \ldots n\}$ and $k > 1$ for the alphabet size. Denote by $\Sigma_n$ the probability space of all maps from $Q$ to $Q$ with the uniform probability distribution. Denote by $\Omega_n^k$ the probability space of all $k$-letter $n$-state automata where all letters $c \in \Sigma$ are chosen uniformly at random and independently from $\Sigma_n$. 
**Theorem 1.** There is a deterministic algorithm for deciding whether or not a given automaton is synchronizing having linear in $n$ expected time with respect to $\Omega_k^2 n$. Moreover, for this problem the proposed algorithm is optimal by expected time up to a constant factor.

*Proof.* We first design an algorithm for $k = 2$ and then explain how to generalize it for each $k > 1$. The proposed algorithm is based on the following

**Theorem 2 (Theorem 1 from [5]).** The probability of being synchronizable for 2-letter random automaton with $n$ states equals $1 - \Theta(\frac{1}{n})$.

Let $\mathcal{A} = (Q, \{a, b\})$ be a random automaton from $\Omega_2^2 n$. The idea of the algorithm is to subsequently check that all required properties in the proof of Theorem 2 really holds true for $\mathcal{A}$; if so, we return ‘Yes’; otherwise, we just run aforementioned quadratic algorithm $IsSynch$ for $\mathcal{A}$. Since the probability of the opposite case is $O(\frac{1}{n})$ by Theorem 2, the overall expected time is linear in $n$ if all required properties can be checked in linear time.

In what follows, by $wlp$ we mean ‘with probability $O(\frac{1}{n})$’ and by $whp$ we mean ‘with probability $1 - O(\frac{1}{n})$’.

First by calling Tarjan’s linear algorithm [12] we find minimal strongly connected components (MSCC) and, if there are several MSCC, we return ‘No’ because $\mathcal{A}$ is not synchronizing in this case. Otherwise, there is a unique MSCC $\mathcal{B}$. We may assume that its size is at least $n/4e^2$ due to the following lemma (otherwise we run $IsSynch$ wlp).

**Lemma 1 (Lemma 1 from [5]).** The number of states in any subautomaton of $\mathcal{A}$ is at least $n/4e^2$ whp.

Let us redefine $\mathcal{A}$ to this MSCC $\mathcal{B}$ because $\mathcal{A}$ is synchronizing if and only if $\mathcal{B}$ is synchronizing.

In order to continue, we need to do some remarks about graphs related to letters. Fix a letter $x \in \Sigma$ and remove all edges of $\mathcal{A}$ except those labeled $x$. The remaining graph is called the underlying graph of $x$ and is denoted by $UG(x)$. Every connected component of the underlying graph of a letter consists of a unique cycle (that can degenerate to a loop) and possibly some trees rooted on this cycle, see Fig. 1. We call the connected components of the underlying graph of a letter clusters.

Each state $q$ is located in some tree $T_q$ of some cluster $C_q$ of $UG(x)$. For both letters $x \in \{a, b\}$ we want to calculate the cluster structure of their underlying graphs. That is, for some enumeration of clusters and trees, for each $q \in Q$ we want to get the index $tree_x(q)$ of $T_q$, the index
cluster\(_x(q)\) of \(C_q\) with respect to the chosen enumerations, and also the level \(lvl\_x(q)\) of \(q\) in \(T_q\).

As a secondary information we evaluate the number of clusters for each letter, the cluster size \(CL\_x(i)\) and the cycle length \(cl\_x(i)\) for all clusters \(i\) and the unique highest tree for some letter if it exists.

**Lemma 2.** The cluster structure of each letter \(x \in \Sigma\) can be calculated in linear in \(n\) time.

**Proof.** At each new step we choose an unobserved state \(p \in Q\), set \(cluster\_x(p) = p\) and walk by the unique path

\[ p = p_0, p_1 = p_0.x, \ldots, p_m = p_{m-1}.x \]

in the underlying graph until we meet a state \(p_m\) such that \(p_m = p_k = p_k.x^{m-k}\) for some \(k < m\). Then we set \(lvl\_x(p_i) = 0, tree\_x(p_i) = p_i\) for \(k \leq i \leq m\) because these are cycle states. After that, for each of these cycle state \(q\) we run BFS in the tree \(T_q\) rooted in \(q\) by reverse arrows, and at \(j\)-th step we set

\[ lvl\_x(s) = j, tree\_x(s) = q, cluster\_x(s) = cluster\_x(p). \]

We process a full cluster by this procedure. Since we observe each state only in one procedure and at most twice, the algorithm is linear. Clearly we can simultaneously evaluate the number of clusters and the unique highest tree if it exists.

Next, we may assume that the number of clusters does not exceed \(5 \ln n\) for both letters due to the following

**Lemma 3 (Lemma 3 from [5]).** The underlying graph of a random map from \(\Sigma_n\) has at most \(5 \ln n\) cycles with probability \(1 - o(\frac{1}{n})\).
Similarly, using the cluster structure, one can check in linear time that there is a letter (say $a$) whose underlying graph has a unique highest tree of size at least $32 \ln n$. The opposite case appears w.p. due to the following statements from [5].

**Corollary 1 (Corollary 1 from [5]).** Whp the underlying graph of one letter (say $a$) has a unique highest tree $T$. Let $H$ be the subset of vertices in $T$ of levels greater than levels of the vertices in all other trees. Then $H$ is random for the letter $b$ and contains at least $32 \ln n$ vertices.

**Lemma 4 (Lemma 2 from [5]).** The subset $H$ of top-level vertices of the underlying graph of any letter (satisfying Corollary 1) intersects with any subautomaton whp.

By Corollary 1 the original automaton whp has a letter with a unique highest tree and the corresponding set $H$ of size at least $32 \ln n$ and, by Lemma 4 $H$ intersects with any subautomaton whence with $\mathcal{A}$ also.

If we found such a letter (say $a$) and the corresponding tree $T$ then one can find a pair $(p, q)$ in linear time such that $p$ is the first level ancestor with the minimum index of some top vertex of $T$ and $q$ is the unique state on the cycle (of the same cluster as $p$) such that $p.a = q.a$. If $\mathcal{A}$ is synchronizing, all pairs are stable, otherwise it is stable by [5, Theorem 2]. Since $(p, q)$ is completely defined by $a$, it is random for $b$.

Next, we try to extend $(p, q)$ to sets $P_a, P_b$ of $n^{0.4}$ stable pairs each, random for $b$ and $a$ respectively according to [5, Lemma 9,10 and Theorem 3]. The maximum number of pairs that we could observe during this procedure appears in [5, Theorem 3] and is bounded by $O(n^{0.4})$ whence this step can be done in linear time. If we fail to extend at some step we again run $IsSynch$ w.p. due to the statement of the correspondent proposition.

Denote $S = P_a$ and consider the underlying graph of $a$. Note that each pair from $S$ is either in a one cluster or connects some two clusters of $a$, i.e. we can consider the graph $G = (V, E(S))$ where $V$ is the set of $a$-clusters with size at least $n^{0.45}$ and $\{C_1, C_2\} \in E(S)$ if and only if there is a pair $\{p_1, p_2\} \in S$ such that $p_1 \in C_1$ and $p_2 \in C_2$. Since $|V| \leq 5 \ln n$ and $|S| \leq n^{0.4} + 1$, we can verify that $G$ is connected by running BFS in linear time. We may assume that $G$ is connected due to the following

**Lemma 5 (Lemma 4 from [5]).** Whp there is a subset of connected by $S$ clusters with common size at least $n - n^{0.45}$ states.

Then we calculate the greatest common divisor $d$ of the cycle lengths of the clusters corresponding to $V$. Using Euclidian algorithm it can be
done in $O(\ln^2 n)$ time. If $d > 1$ we additionally must verify the following property from Lemma 5: there are some \( \{ x_i \mid i \in \{1, 2, \ldots, |V|\} \} \) such that \( 0 \leq x_i \leq cl(i) \) and for all pairs \( \{ p, q \} \in S \)

\[
d \mid (lvl(p) - lvl(q)) - (x_{\text{cluster}(p)} - x_{\text{cluster}(q)}).
\]

If this property holds true, we run \( \text{IsSynch} \) wlp due to the proof of Lemma 6 (Lemma 5 from [5]). Whp there is a stable cluster among clusters connected by \( S \).

Let us show how this property can be checked in linear time. We subsequently consider all pairs \( \{ p, q \} \in S \). If the cluster of \( p \) (or \( q \)) has not yet been observed then we can set \( x_{\text{cluster}(p)} = 0 \) because we are interested in validating condition (1) for all pairs. Otherwise we just verify condition (1) for \( p, q, d \). Since there are at most \( n^{0.4} + 1 \) such pairs, it can be done in linear time.

Thus we may assume that all clusters of \( UG(a) \) of size at least \( n^{0.45} \) are contained in a one synchronizing class \( S_a \), i.e. each pair from \( S_a \) can be merged. Moreover, since \( S_a \) can be defined by the letter \( a \), this class is random for \( b \). Analogously, we can do the same for \( b \) and obtain the corresponding set \( S_b \) with the same properties.

Denote by \( T_a \) and \( T_b \) the complements for \( S_a \) and \( S_b \) in \( Q \) respectively. Equivalently, they can be defined as the set of clusters of \( a \) (or \( b \)) of size less than \( n^{0.45} \). It remains to show that we can implement the procedure from the proof of [5, Lemma 6] in linear time. We have to verify that there are no two (probably equal) clusters \( C_p, C_q \) such that for some \( x \in \{0, 1, \ldots, d - 1\} \) all pairs \( p_1 \in C_p, p_2 \in C_q \) such that \( d \mid lvl(p_1) - lvl(p_2) + x \) have a nonempty intersection with \( T_b \) or pairs \( \{ p_1.b, p_2.b \}, \{ p_1.b^2, p_2.b^2 \} \) have nonempty intersections with \( T_a \). Since \( |C_p|, |C_q| < n^{0.45} \), it can be done in linear time. If there are such clusters, we run \( \text{IsSynch} \) wlp due to the proof of [5, Lemma 6]. Otherwise, we return ‘Yes’. The correctness of the algorithm finally follows from [5, Lemma 6], and this completes the description of the algorithm for 2-letter alphabet case.

For the general case when \( A = (Q, \{ a_1, a_2, \ldots, a_k \}) \) for \( k \geq 2 \) we can just run described algorithm for the automaton \( A_2 = (Q, \{ a_1, a_2 \}) \); If the result is ‘No’, we run \( \text{IsSynch} \) algorithm wlp due to Theorem 2.

It remains to prove that any algorithm for deciding whether or not a given \( n \)-state \( k \)-letter automaton is synchronizing has to do at least linear expected number of steps. By Theorem 2 \( A \) is synchronizing whp. Hence, whp it is necessary to confirm that \( A \) is also weakly connected, and this requires to check that each state has at least one incoming arrow. \( \square \)
3 Set-Cover as Finding Reset Threshold

Let us follow [1] and [8] in this section. Given a universe $\mathcal{U} = \{u_1, \ldots, u_n\}$ and a family of its subsets, $\mathcal{S} = \{S_1, \ldots, S_m\} \subseteq \mathcal{P}(\mathcal{U})$, $\bigcup_{j \in \mathcal{S}} S_j = \mathcal{U}$, Set-Cover is the problem of finding a minimal sub-family $\mathcal{C} \subseteq \mathcal{S}$ that covers the whole universe, $\bigcup_{j \in \mathcal{C}} S_j = \mathcal{U}$. Denote the size of the minimal sub-family by $OPT(\mathcal{U}, \mathcal{S})$. Set-Cover is a classic NP-hard combinatorial optimization problem, and it is known that it can be approximated in polynomial time to within $\ln(n) - \ln(\ln(n)) + \Theta(1)$ (see [9,11]).

The following transparent reduction from Set-Cover is presented in [8]. Given a Set-Cover instance $(\mathcal{U}, \mathcal{S})$, define the automaton $A(\mathcal{U}, \mathcal{S}) = (\mathcal{U} \cup \{\hat{q}\}, \Sigma = \{a_1, \ldots, a_m\})$ where the transition function is defined as follows.

$$\delta(u, a_i) = \begin{cases} \hat{q}, & u \in S_i \\
q, & u \notin S_i. \end{cases} \quad (2)$$

Remark 1. Let $A = (Q, \Sigma, \delta)$ be the automaton defined by $(\mathcal{U}, \mathcal{S})$ as above. Then

$$rt(A) = OPT(\mathcal{U}, \mathcal{S}), \quad |Q| = |\mathcal{U}| + 1, \quad |\Sigma| = |\mathcal{S}|.$$

The following sophisticated result has been obtained in [1].

Theorem 3 (Alon, Moshkovitz, Safra, 2006). Unless $P = NP$, no polynomial time algorithm approximates Set-Cover within performance ratio less than $c_{sc} \ln n$ where $n$ is the size of the universe and $c_{sc} > 0.2267$ is a specific constant.

Thus, the following theorem from [8] immediately follows from Remark 1 and Theorem 3.

Theorem 4 (Gerbush, Heeringa, 2011). Unless $P = NP$, no polynomial time algorithm approximates reset threshold within performance ratio less than $c_{sc} \ln n$, where $n$ is the number of states.

However, the alphabet size of automata in Theorem 4 grows together with the number of states. Let us recall that the case of fixed size alphabet, especially 2-letter alphabet case is of main importance.

Here we bypass this obstacle by encoding binary representation of letters in states and using some properties from the proof of Theorem 3.
Lemma 7. For every $m$-letter synchronizing automaton $\mathcal{A} = (Q, \Sigma, \delta)$, there is a 2-letter synchronizing automaton $B = B(\mathcal{A}) = (Q', \{0,1\}, \delta')$ such that

$$rt(\mathcal{A})[\log_2 m + 1] \leq rt(B) \leq [\log_2 m + 1](1 + rt(\mathcal{A})),\quad \delta \text{ has at most } 2m|Q| \text{ states and can be constructed in polynomial time of } m \text{ and } |Q|.$$ 

Proof. Let $\Sigma = \{a_1, \ldots, a_m\}$, $\lambda$ is the empty word, and for simplicity assume that $m$ is a power of 2, i.e. $m = 2^k$ (otherwise we can add at most $m - 1$ letters with identical action without impact on the bounds). Then we may assign to each binary string $w \in \{0,1\}^k$ its number $\ell(w)$ in the lexicographical order from 1 to $m$. We let $Q' = Q \times \{0,1\}^k$ and define the transition function $\delta' : Q' \times \{0,1\}^k \rightarrow Q'$ as follows:

$$\delta'((q, w), x) = \begin{cases} (q, wx), & |w| < k \\ (q, a_{\ell(w)}), & |w| = k, x = 1 \\ (q, w), & |w| = k, x = 0. \end{cases}$$

Let $u = a_{j_1}a_{j_2} \ldots a_{j_t}$ be a reset word for $\mathcal{A}$. Then the word

$$1^{k+1}\ell^{-1}(j_1)1 \ldots \ell^{-1}(j_t)1$$

is reset for $B$ and its length equals $(k+1)(t+1)$. The upper bound follows.

In order to prove the lower bound it is enough to consider the shortest binary word $w$ which merges the subset $(Q, \lambda)$ in $B$. Since $w$ is chosen shortest, $w = w_1w_21 \ldots w_r1$ where $|w_j| = k$ for each $j \in \{1, \ldots r\}$. Then the word $a_{\ell(w_1)}a_{\ell(w_2)} \ldots a_{\ell(w_r)}$ is reset for $B$ and the lower bound follows. $\square$

Now, suppose that for some constant $d > 0$, there is a polynomial time algorithm $f_2$ such that

$$rt(B) \leq f_2(B) \leq d \ln (n)rt(B)$$

for every 2-letter $n$-state synchronizing automaton $B$. Then Lemma 7 implies that there is also a polynomial time algorithm $f$ such that

$$rt(\mathcal{A}) \leq f(\mathcal{A}) \leq d \ln (2nm)(1 + rt(\mathcal{A}))$$

for every $m$-letter $n$-state synchronizing automaton $\mathcal{A}$. Indeed, such algorithm first constructs $B(\mathcal{A})$ with at most $2nm$ states as in Lemma 7, and then applies $f_2$ for $B(\mathcal{A})$:

$$rt(\mathcal{A}) \leq f(\mathcal{A}) \leq f_2(B(\mathcal{A})) \leq d \ln (2nm)(rt(\mathcal{A}) + 1).$$
Combining this with Theorem 3 and Remark 1 we immediately get the following corollary.

**Corollary 2.** Let $m(n)$ be an upper bound on the cardinality of the set family $S$ as a function of the size of the universe $U$ from the reduction to Set-Cover from $[1]$. Then, unless $P = NP$, no polynomial time algorithm approximates reset threshold within performance ratio $\frac{d \log m(n)}{\log m(n)+1} \ln(n)$ for any $d < c_{sc}$ in the class of all 2-letter synchronizing automata.

Thus it suffices to find a lower bound on the size of the universe $U$ and an upper bound on the size of the family of subsets $S$ in the reduction to Set-Cover presented in $[1]$.

Due to the space limit, we will use some notation from $[1]$ without reproducing all definitions. First, the universe $\mathcal{U}$ is defined as $[D] \times \Phi \times B$ where $D = \lfloor \frac{|\Phi|}{\eta |X|} \rfloor$, $\eta$ is a constant. Hence the rough lower bound for the size of the universe $\mathcal{U}$ is $|\Phi|$.

The size of the family of subsets $S$ is equal to $D |X||\Phi| + |\Phi||F|^d$ where $F$ is a field of cardinality at most $2^{\log_2 1 - \beta} |X| \leq |X|$ and $d \geq 2$ is a positive integer which can be taken equal 3. Hence the upper upper bound for $|S|$ is $\Theta(1) |\Phi| |X|^d$.

Hence we get that

$$\log_{|\mathcal{U}|} |S| \leq \frac{d + \log_{|X|} |\Phi|}{\log_{|X|} |\Phi|}.$$ 

Note that $|\Phi|$ is only restricted to be some polynomial of $|X|$, i.e. it can be chosen to be $|X|^r$ for arbitrary large constant $r$.

As a conclusion we get the following lemma, which gives a nice property of Set-Cover itself.

**Lemma 8.** Given any $\gamma > 0$, unless $P = NP$, no polynomial time algorithm approximates the Set-Cover with performance ratio $d \ln n$ for any $d < c_{sc}$ in the class of all Set-Cover instances $(\mathcal{U}, S)$ satisfying $\log_{|\mathcal{U}|} |S| \leq 1 + \gamma$.

Combining this with Corollary 2 gives us the main result of this section.

**Theorem 5.** Unless $P = NP$, no polynomial time algorithm approximates the reset threshold within performance ratio less than $0.5c_{sc} \ln n$ in the class of all $n$-state synchronizing automata with 2 input letters.

Let us notice that the same bound holds true for any fixed non-singleton alphabet. Experiments show that the polynomial Greedy algorithm (basically due to Černý $[2]$) approximates the reset threshold with
logarithmic performance ratio. Hence, the result presented in Theorem 5 might be exact up to a constant factor.

References

1. Alon, N., Moshkovitz, D., Safra, S.: Algorithmic construction of sets for k-restrictions. ACM Trans. Algorithms, 2(2):153–177, 2006.
2. Araujo, J., Bentz, W., Cameron P.: Groups Synchronizing a Transformation of Non-Uniform Kernel. arXiv:1205.0682
3. Berlinkov, M.: Approximating the minimum length of synchronizing words is hard. Lect. Notes Comp. Sci. 6072:37–47, 2010.
4. Berlinkov, M.: Approximating the Minimum Length of Synchronizing Words Is Hard. Theory of Computing Systems, 2013, DOI doi:10.1007/s00224-013-9511-y.
5. Berlinkov, M.: On the probability of being synchronizable, 2013, arXiv:1304.5774 (submitted to ICALP 2014).
6. Černý, J.: Poznámka k homogénnym eksperimentom s konečnými automatami. Matematicko-fyzikálny Časopis Slovensk. Akad. Vied. 14(3):208–216, 1964 (in Slovak).
7. Eppstein, D.: Reset sequences for monotonic automata. SIAM J. Comput. 19:500–510, 1990.
8. Gerbush, M., Heeringa, B.: Approximating minimum reset sequences. Lect. Notes Comp. Sci. 6482:154–162, 2011.
9. Lovász, L.: On the ratio of optimal integral and fractional covers. Discrete Mathematics, 13:383–390, 1975.
10. Olschewski, J., Ummels, M.: The complexity of finding reset words in finite automata. Lect. Notes Comp. Sci. 6281:568–579, 2010.
11. Slavik, P.: A tight analysis of the greedy algorithm for set cover. In Proc. 28th ACM Symp. on Theory of Computing, pp. 435–441, 1996.
12. Tarjan, R.: Depth-first search and linear graph algorithms, SIAM Journal on Computing 1 (2): 146160, doi:10.1137/0201010, 1972