Strong inapproximability of the shortest reset word

Pawel Gawrychowski ∗
Max-Planck-Institut für Informatik

Damian Straszak †
University of Wroclaw

Abstract

The famous Černý conjecture, a 50-year-old mathematical problem, states that every \( n \)-state synchronizing automaton has a reset word of length at most \((n - 1)^2\). We consider the question of finding short reset words. It is known that the exact version of the problem, i.e., finding the shortest reset word, is both \( \text{NP}-\text{hard} \) and \( \text{coNP}-\text{hard} \), and actually complete for a class known as \( \text{DP} \). It is also known that approximating the length of the shortest reset word within a factor of \( \mathcal{O}(\log n) \) is \( \text{NP}-\text{hard} \) [9], even if the alphabet is binary [4]. We significantly improve on these results by showing that, unless \( P = \text{NP} \), it is not possible to approximate the length of the shortest reset word within a factor of \( n^{0.0095} \) in polynomial time.

1 Introduction

Let \( A = (Q, \Sigma, \delta) \) be a deterministic finite automaton. We say that \( w \in \Sigma^* \) resets (or synchronizes) \( A \) if \( |\delta(Q, w)| = 1 \). If at least one such \( w \) exists, \( A \) is called synchronizing. In 1964 Černý conjectured that every synchronizing \( n \)-state automaton admits a reset word of length \((n - 1)^2\). The problem remains open as of today. It is known that an \( \frac{(n-1)^2}{6} \) bound holds [16] and that there are automata requiring words of length \((n - 1)^2\). It is known that the conjecture holds for various special classes of automata [1, 17, 18]. Quite a few generalization and similar questions have been considered, the most famous one being the road coloring problem, finally solved by Trahtman [19].

Computational problems related to synchronizing automata were also thoroughly studied. It is known that finding the shortest reset word is both \( \text{NP}-\text{hard} \) and \( \text{coNP}-\text{hard} \) [7]. Moreover, it was shown to be \( \text{DP}-\text{complete} \) by Olschewski and Ummels [15]. The fastest known algorithm for finding shortest reset words by Kisielewicz et al [13] works in reasonable time for automata with roughly one hundred states.

In this paper, rather than looking at the exact version, we consider the problem of finding short reset words for automata, or to put it differently, the question of approximating the length of the shortest reset word. For a given \( n \)-state synchronizing automaton, we want to find a reset word which is at most \( \alpha \) times longer than the shortest one, where \( \alpha \) can be either a constant or a function of \( n \). There is a simple polynomial time algorithm achieving \( \mathcal{O}(n) \)-approximation, that is \( \alpha = \mathcal{O}(n) \). In contrast, it was shown by Berlinkov [5] that \( \mathcal{O}(1) \)-approximation is not possible in polynomial time, unless \( P = \text{NP} \). An even stronger result of such type was later obtained by Gerbush and Heeringa [9], who proved that no polynomial time \( \mathcal{O}(\log n) \)-approximation is possible. This was later improved on by Berlinkov [4], who reduced the alphabet size to binary. In the same

∗gawry@mpi-inf.mpg.de
†damian.straszak@gmail.com
paper, he also conjectured an existence of a polynomial time $O(\log n)$-approximation algorithm. We refute this conjecture by showing that, assuming $P \neq NP$, no $n^{0.0095}$-approximation is possible.

## 2 Preliminaries

The main object of our study is a DFA (deterministic finite automaton or in short automaton) $A = (Q, \Sigma, \delta)$, where $Q$ is some nonempty finite set of states, $\Sigma$ is a nonempty, finite alphabet and $\delta$ is the transition function $\delta : Q \times \Sigma \rightarrow Q$. In the usual definition of a DFA one includes additionally a starting state and a set of accepting states. We omit them in our definition, since they are irrelevant in our setting. Equivalently, we can treat a DFA as a collection of a starting state and a set of accepting states. We omit them in our definition, since they are irrelevant in our setting. As shown in Lemma 7 of [4], given an automaton $A$, we can efficiently construct an automaton $B$ on $2|\Sigma|n$ states over the binary alphabet, such that $\text{Syn}(A)t \leq \text{Syn}(B) \leq t(1 + \text{Syn}(A))$, where $t = \lceil \log_2 |\Sigma| \rceil + 1$. Then, if we can approximate $\text{Syn}(B)$ within a factor of $\frac{1}{2}n^{\varepsilon_0}$, we can compute in polynomial time $x$ such that $\text{Syn}(B) \leq x \leq \text{Syn}(B)\frac{1}{2}n^{\varepsilon_0}$. Then $\text{Syn}(A)t \leq x$ and $x \leq t(1 + \text{Syn}(A))\frac{1}{2}n^{\varepsilon_0} \leq t\text{Syn}(A)n^{\varepsilon_0}$, as $n \leq 1$. Therefore, $\text{Syn}(A) \leq \frac{x}{t} \leq \text{Syn}(A)n^{\varepsilon_0}$, so $\frac{x}{t}$ approximates $\text{Syn}(A)$ within a factor of $n^{\varepsilon_0}$.

### Synchronizing Automata

We say that an automaton $A = (Q, \Sigma, \delta)$ is synchronizing if there exists a word $w$ for which $|\delta(Q, w)| = 1$. Such $w$ is then called a synchronizing (or reset) word. By $\text{Syn}(A)$ we denote the length of the shortest word synchronizing $A$. Now the problem we want to study is as follows.

| SYNAPPX($\Sigma, \alpha$) |
|-------------------------|
| Given a synchronizing automaton $A$ on $n$ states over an alphabet $\Sigma$, find a synchronizing word for $A$ of length at most $\alpha \cdot \text{Syn}(A)$. Here both $\alpha$ and $|\Sigma|$ can be a function of $n$. |

Note that one can easily check if a given automaton is synchronizing in polynomial time. We are interested in polynomial time solutions to the SYNAPPX problem, with $\alpha$ as small as possible. In the general size, the size of the alphabet can be arbitrary (it might even be a function of $n$), but in all our lowerbounds $|\Sigma| = 3$ will be enough. Our main result will be that, for all $|\Sigma| \geq 3$, SYNAPPX($\Sigma, n^{\varepsilon_0}$) cannot be solved in polynomial time, for some positive constant $\varepsilon_0$, unless $P = NP$.

### Reducing size of the alphabet

Our lowerbound requires $|\Sigma| \geq 3$. Nevertheless, it is possible to reduce the size of the alphabet to just 2, as shown by Berlinkov [4]. The idea is to simply encode every letter in binary, but one must be careful as to be able to bound the length of the shortest synchronizing word of the resulting automaton.

**Lemma 2.1 (Lemma 7 of [4])** If SYNAPPX($\{0, 1\}, \frac{1}{7}n^{\varepsilon_0}$) can be solved in polynomial time, then so can be SYNAPPX($\Sigma, n^{\varepsilon_0}$) for any $\Sigma$.

**Proof:** As shown in Lemma 7 of [4], given an automaton $A$ on $n$ states over an alphabet $\Sigma$, one can efficiently construct an automaton $B$ on $2|\Sigma|n$ states over the binary alphabet, such that $\text{Syn}(A)t \leq \text{Syn}(B) \leq t(1 + \text{Syn}(A))$, where $t = \lceil \log_2 |\Sigma| \rceil + 1$. Then, if we can approximate $\text{Syn}(B)$ within a factor of $\frac{1}{2}n^{\varepsilon_0}$, we can compute in polynomial time $x$ such that $\text{Syn}(B) \leq x \leq \text{Syn}(B)\frac{1}{2}n^{\varepsilon_0}$. Then $\text{Syn}(A)t \leq x$ and $x \leq t(1 + \text{Syn}(A))\frac{1}{2}n^{\varepsilon_0} \leq t\text{Syn}(A)n^{\varepsilon_0}$, as $n \leq 1$. Therefore, $\text{Syn}(A) \leq \frac{x}{t} \leq \text{Syn}(A)n^{\varepsilon_0}$, so $\frac{x}{t}$ approximates $\text{Syn}(A)$ within a factor of $n^{\varepsilon_0}$. \(\square\)
Therefore, if we can prove that, for some fixed $\Sigma$, SYNAPPX$(\Sigma, n^{\varepsilon_0})$ is $NP$-hard, then so is SYNAPPX$((0, 1), n^{\varepsilon_1})$ for any $\varepsilon_1 < \varepsilon_0$.

$\lceil \frac{n}{2} \rceil$-approximation. It is known [12] that for any fixed $k$ the problem SYNAPPX$(\Sigma, \frac{n}{2})$ can be solved in $O(n^{k+1})$ time (we assume that $\Sigma$ is of constant size). The basic idea is that, for a given automaton $A = (Q, \Sigma, \delta)$, we construct a graph $G$ with the vertex set $V = \{ S \subseteq Q : |S| \leq k + 1 \}$ and a directed edge $S \rightarrow \delta(S, a)$ labeled with $a$ for every $S \in V$ and $a \in \Sigma$. Then for a given $S \in V$ the shortest word synchronizing $S$ to a single state corresponds to the shortest path connecting $S$ with some singleton set $\{ q \} \in V$. Each such word is of length at most $\text{Syn}(A)$. The algorithm works in $\lceil \frac{n}{2} \rceil$ phases. We start with the full set of states to reset $R := Q$ and with an empty word $w := \varepsilon$, and in each phase we will decrease the size of $R$ by $k$, while assuring that $\delta(Q, w) = R$. In a single phase we take any subset $S$ of $R$ of size $k + 1$ (if possible) and find the shortest word $w'$ resetting $S$ to a single state (note that $|w'| \leq \text{Syn}(A)$). We set $w := ww'$, $R := \delta(R, w')$ and continue. One can easily see that at the end we obtain a synchronizing word $w$ of length at most $\lceil \frac{n}{2} \rceil \cdot \text{Syn}(A)$.

Černý Conjecture. Setting $k = 1$ in the above reasoning we obtain an upper bound for $\text{Syn}(A)$. This follows from the fact that the graph $G$ has $O(n^2)$ vertices and consequently every word $w$ has length $O(n^2)$. At the end we have $\text{Syn}(A) \leq |w| = (n - 1) \cdot O(n^2) = O(n^3)$. In contrast, the famous Černý conjecture states that for every synchronizing automaton $A$ it holds $\text{Syn}(A) \leq (n - 1)^2$. Interestingly, the best bound known up to now is $\frac{(n-1)^3}{6}$ [11], which is also cubic. Any $o(n^3)$ upper bound would be a very interesting result for this problem.

3 Simple Hardness Result

In this section we are going to obtain a simple introductory hardness result on approximating the shortest reset word. Namely, we will show that, for any fixed positive constant $\varepsilon$, it is $NP$-hard to find a synchronizing word $w$ such that $|w| \leq (2 - \varepsilon)\text{Syn}(A)$ for a given $n$-state synchronizing automaton. Even though the final goal is to prove a much stronger result, the basic construction presented in this section will serve as a core idea for all the further proofs. A similar construction was also given by Berlinkov [5], who showed inapproximability within a factor of $2 - \varepsilon$, and then used it as a base step for proving inapproximability within any constant factor. Let us now formally state the theorem.

**Theorem 3.1** For every $\Sigma$ such that $|\Sigma| \geq 3$ and for every fixed constant $\varepsilon > 0$, SYNAPPX$(\Sigma, 2 - \varepsilon)$ is not solvable in polynomial time, unless $P = NP$.

**Construction Idea.** Our construction is based on the $NP$-completeness of the 3-SAT problem. Fix $\Sigma = \{0, 1, 2\}$ and $\varepsilon > 0$. We will show that if there exists an algorithm solving SYNAPPX$(\Sigma, 2 - \varepsilon)$ in polynomial time, then we can decide satisfiability of 3-CNF formulas in polynomial time. This will stem from the following reduction. For a given 3-CNF formula $\phi$ with $N$ variables and $M$ clauses we can build in polynomial time a synchronizing automaton $A_\phi$, such that:

1. if $\phi$ is satisfiable then $\text{Syn}(A_\phi) = N + O(1)$,
2. if $\phi$ is not satisfiable then $\text{Syn}(A_\phi) \geq 2N$. 


This will imply Theorem 3.1 since applying an \((2 - \varepsilon)\)-approximation algorithm to \(A_\phi\) would allow us to find out whether \(\phi\) is satisfiable or not.

**Gadget Design.** Let \(\phi = C_1 \land C_2 \land ... \land C_M\) be a 3-CNF formula with \(N\) variables \(x_1, x_2, ..., x_N\) and \(M\) clauses. We want to build an automaton \(A_\phi = (\{0, 1, 2\}, Q, \delta)\) with properties as described above. \(A_\phi\) consists of \(M\) gadgets, one for each clause in \(\phi\) and a single sink state \(s\). Formally, \(Q = \{s\} \cup \bigcup_{i=1}^{M} Q_i\). All letters leave the state \(s\) intact, that is \(\delta(s, 0) = \delta(s, 1) = \delta(s, 2) = s\). We describe now the gadget for clause \(C_i\). Its state set \(Q_i\) consists of \(2(M + 1)\) states denoted \(q^1_{ij}\) and \(q^0_{ij}\) for \(j = 0, 1, 2, ..., M\). Since \(i\) is fixed, for notational convenience we will drop the subscript \(i\) in the state names, so \(q^0_{ij}\) becomes \(q^0_j\) and so on. We want to define the action of the letters 0 and 1 on \(Q_i\). We first give a formal description of the transitions\(^1\) and then list some of its interesting properties. See Fig. 1 for an example.

For any \(j \in \{0, 1, ..., N - 1\}\) and \(c \in \{0, 1\}\):

- \(\delta(q^1_j, c) = q^1_{j+1}\),
- if \(C_i\) does not depend on \(x_{j+1}\) then \(\delta(q^0_j, c) = q^0_{j+1}\),
- if \(C_i\) depends on \(x_{j+1}\) positively (i.e. \(x_{j+1} = 1\) makes \(C_i\) true) then \(\delta(q^0_j, 1) = q^1_{j+1}\) and \(\delta(q^1_j, 0) = q^0_{j+1}\),
- if \(C_i\) depends on \(x_{j+1}\) negatively then \(\delta(q^0_j, 0) = q^1_{j+1}\) and \(\delta(q^1_j, 1) = q^0_{j+1}\),
- \(\delta(q^1_N, c) = s\), \(\delta(q^0_N, c) = q^0_N\).

Note that a word \(v \in \{0, 1\}^N\) can be identified with an assignment of boolean values to variables by simply setting \(x_i := v_i \in \{0, 1\}\). After such an identification we can consider the value of \(C_i\) under assignment \(v\). The state \(q^0_0\) can be regarded as the starting state of our gadget. We can imagine this gadget as a smaller automaton which reads an assignment and determines whether it is satisfying for \(C_i\). More precisely:

1. applying a binary word \(w\) of length \(|w| = j \leq N\) to \(q^0_0\) lands us in state \(q^S_j\) for some \(S \in \{0, 1\}\), i.e. \(\delta(q^0_0, w) \in \{q^S_j, q^1_j\}\),
2. if \(v \in \{0, 1\}^N\) is a satisfying assignment for \(C_i\), then \(\delta(q^0_0, v) = q^1_N\), else \(\delta(q^0_0, v) = q^0_N\).

\(^1\)To make this definition correct, we assume \(C_i\) contains at most one (positive or negative) occurrence of every variable \(x_j\).
Suppose we apply an assignment \( v \in \{0,1\}^N \) which is satisfying for \( C_i \) to the state \( q_{0,i}^0 \), then we land in \( q_{N}^0 \), so now applying either 0 or 1 will push us to the sink \( s \), and there is no escape from \( s \), so the gadget gets synchronized. On the other hand, if we perform the same procedure starting with a non-satisfying assignment, then we will certainly end up in \( q_{0,i}^0 \). To summarize, we have the following.

**Observation 3.2** Let \( w = vc \) be a binary word with \( |v| = N \) and \( c \in \{0,1\} \), then \( \delta(q_{0,i}^0, w) = s \) if and only if \( v \) satisfies \( C_i \). Consequently, let \( w = v_1c_1v_2c_2...v_kc_k \) be a binary word, with all \( v_i \)'s of length \( N \) and all \( c_i \)'s in \( \{0,1\} \), then

\[
\delta(q_{0,i}^0, w) = \begin{cases} 
q_{0,i}^0 & \text{if } C_i \text{ is satisfied by any of } v_1, v_2, ..., v_k, \\
q_{N}^0 & \text{otherwise}.
\end{cases}
\]

**Properties of \( A_\phi \).** Before we define the action of the letter 2 let us observe some properties of the automaton. Let \( P = \{q_{0,i}^0 : 1 \leq i \leq N\} \). We consider now running all the clause gadgets simultaneously. The following proposition follows easily from Observation 3.2.

**Proposition 3.3** If \( v \in \{0,1\}^N \) is a satisfying assignment for \( \phi \) then \( \delta(P,v0) = \{s\} \), so \( v0 \) is a word of length \( N + 1 \) synchronizing \( P \) to \( s \). If \( \phi \) is not satisfiable then for every word \( w \in \{0,1\}^* \) synchronizing \( P \) it holds that \( |w| \geq 2N + 2 \).

**Proof:** The first part directly follows from Observation 3.2 so we focus on proving the second part. Let \( w \) be the shortest word synchronizing \( P \), thus \( \delta(P,w) = \{s\} \). Then one can see \( w \) is of length \( k(N + 1) \) for some \( k \geq 1 \), because a gadget gets synchronized only after integral number of length-(\( N + 1 \)) whole loops around the gadget. If \( k = 1 \) then from Observation 3.2 \( \phi \) is satisfied under an assignment encoded by the prefix of \( w \) of length \( N \). Hence \( k \geq 2 \) and the result follows.

Finally, we need to define the action of letter 2 in such a way that Proposition 3.3 still holds and in case of satisfiable \( \phi \) we can synchronize the whole automaton (not only the set \( P \)) in \( N + O(1) \) steps. To achieve this, we just let 2 send all states (except \( s \)) into \( P \). More precisely, \( \delta(s,2) = s \) and for each \( 1 \leq i \leq M \), \( S \in \{0,1\} \) and \( 1 \leq j \leq N \) we define \( \delta(q_{ij}^0,2) = q_{0,i}^0 \). In other words, the letter 2 resets all states within the \( i \)th gadget to the starting state \( q_{0,i}^0 \).

**Proposition 3.4** Let \( \phi \) be a 3-CNF formula, and \( A_\phi \) be the automaton built for \( \phi \).

1. \( A_\phi \) is synchronizing.
2. If \( \phi \) is satisfiable then \( \text{Syn}(A_\phi) \leq N + 2 \).
3. If \( \phi \) is not satisfiable then \( \text{Syn}(A_\phi) \geq 2N + 2 \).

**Proof:** The first part is clear. For the second part suppose \( \phi \) is satisfied under an assignment \( v \), then the word \( 2v0 \) (of length \( N + 2 \)) synchronizes \( A_\phi \) to \( s \). It remains to prove the third part. We will show that if \( \phi \) is not satisfiable then the set \( P \) cannot be synchronized in less then \( 2N + 2 \) steps. We know by Proposition 3.3 that this is the case when we only use letters 0, 1. We argue that the use of symbol 2 never helps us in synchronizing \( P \). Assume \( w \in \{0,1,2\}^* \) synchronizes \( P \), we will show that there exists \( w' \in \{0,1\}^* \) of length at most \( |w| \) which synchronizes \( P \). Suppose \( w = w_02w_1 \), where \( w_0 \in \{0,1\}^* \). Let \( r := |w_0| \mod (N + 1) \) and \( w'_0 \) be the prefix of \( w_0 \) of length \( (|w_0| - r) \). We can assume that \( w_0 \) does not synchronize \( P \). Fix any \( i \) such that \( \delta(q_{0,i}^0, w_0) \neq s \). One
can easily see that $\delta(q^0_{0r}, w_0) \in \{q^0_{0r}, q^1_{0r}\}$ and $\delta(q^0_{1r}, w_0) = q^0_{1r}$. It follows that $\delta(P, w_02) = \delta(P, w_0')$ and further $\delta(P, w_0'w_1) = \delta(P, w_02w_1) = \{s\}$. The word $w_0'w_1$ synchronizes $P$, is shorter than $w$ and has less occurrences of the symbol 2. By repeating this procedure we get the claimed word $w' \in \{0,1\}^*$. 

Theorem 3.1 follows easily from the above proposition.

## 4 Getting Stronger Hardness

One may wonder whether it is possible to obtain a stronger hardness result using the same kind of construction as presented in Section 3. We will prove the following theorem.

**Theorem 4.1** For every $\Sigma$ such that $|\Sigma| \geq 3$ and for every $\alpha > 1$, SYNAPPX$(\Sigma, \alpha)$ is not solvable in polynomial time, unless $P = NP$.

The above theorem was already proved by Berlinkov [5], who used a recursively defined reduction from the SAT problem, which returns an automaton with some specific synchronizing properties, similar to our $A_\phi$. His construction is elementary, but rather tricky and nontrivial to analyze. We will give a simpler proof using the PCP theorem as a blackbox.

Recall that the construction from Section 3 guarantees that if $\phi$ is satisfiable, then every gadget gets synchronized in $N + 1$ steps, and otherwise, not all the gadgets can be synchronized in one run (a run meaning applying one binary word of length $N + 1$), so another run of length $N + 1$ is required. This gives us a lower bound of $2N + 2$ for the length of a synchronizing word when $\phi$ is not satisfiable. Our goal is to improve it. To proceed we need the following definition.

**Definition 4.2** Let $\phi = C_1 \land C_2 \land ... \land C_M$ be a CNF-formula over $N$ variables.

1. Define $\text{Cov}(\phi)$ to be the minimum $k \in \mathbb{N}$, such that there exist $k$ assignments $v_1, v_2, ..., v_k \in \{0,1\}^N$ with the property that every $C_i$ is satisfied by some $v_\ell$ with $\ell \in \{1,2,\ldots,k\}$. $\text{Cov}(\phi)$ is called the covering number of the formula.

2. $\text{Val}(\phi) \in [0,1]$ to be the maximum fraction of clauses possible to satisfy with a single assignment.

So, in particular $\phi$ is satisfiable iff $\text{Cov}(\phi) = 1$, and similarly $\phi$ is satisfiable iff $\text{Val}(\phi) = 1$. It is easy to see that for any 3-CNF formula $\phi$, by Observation 3.2 we have that $\text{Syn}(A_\phi) \geq \text{Cov}(\phi) \cdot N$. So we are now interested in formulas with a high covering number. One promising property of this quantity is that $\text{Cov}(\phi) \geq \lceil 1/\text{Val}(\phi) \rceil$, so we can focus on formulas having a low value of $\text{Val}(\phi)$. However, here we encounter a major obstacle: by a simple probabilistic argument, $\text{Val}(\phi) \geq \frac{7}{8}$ for any 3-CNF formula $\phi$, which blocks the road to beating $(2 - \varepsilon)$ by a simple adjustment of our construction. We will look at a certain generalization of 3-CNF formulas instead.

**Definition 4.3** A $q$CSP instance $\phi$ is a collection of boolean functions $\phi_1, \phi_2, \ldots, \phi_M : \{0,1\}^N \rightarrow \{0,1\}$ such that each $\phi_i$ depends on at most $q$ of its input locations. That is, for every $i \in \{1,2,\ldots,M\}$ there exist $j_1, j_2, \ldots, j_q$ and a function $f : \{0,1\}^q \rightarrow \{0,1\}$ such that $\phi_i(v) = f(v_{j_1}, \ldots, v_{j_q})$ for every $v \in \{0,1\}^N$. An assignment $v \in \{0,1\}^N$ satisfies the constraint $\phi_i$ if $\phi_i(v) = 1$. We define $\text{Cov}(\phi)$ and $\text{Val}(\phi)$ in an analogous way as for 3-CNF formulas.

\(^2\)CSP stands for Constraint Satisfaction Problem, each $\phi_i$ in the definition is called a constraint.
One special case of a qCSP instance (with $q = 3$) are 3-CNF formulas, there each constraint is simply a disjunction of 3 literals. As seen previously, for each 3-CNF formula $\phi$ we have $\text{Val}(\phi) \geq \frac{7}{8}$. The situation for general CSP instances is much better for us since $\text{Val}(\phi)$ can be arbitrarily small. Moreover we have the following powerful theorem.

**Theorem 4.4** For every $\varepsilon > 0$ there exists $q \in \mathbb{N}$ such that distinguishing between qCSP instances $\phi$ with $\text{Val}(\phi) = 1$ and with $\text{Val}(\phi) < \varepsilon$ is $\text{NP}$-hard. More precisely, there exists a polynomial time reduction $f$ taking a 3-CNF formula $\phi$ and returning a qCSP instance $f(\phi)$ such that:

- if $\phi$ is satisfiable then $\text{Val}(f(\phi)) = 1$,
- if $\phi$ is not satisfiable then $\text{Val}(f(\phi)) < \varepsilon$.

The above theorem is also known as the PCP theorem in the ”hardness of approximation” version. The first proof was given by Arora et al. in 1998 [3]. Later in 2005 a simpler proof was presented by Dinur [6]. We omit here a broad discussion of this theorem. In the next section we state PCP theorem in the original version and then deduce from it a stronger version of Theorem 4.4. A weakening of the reasoning given there yields Theorem 4.4.

**Modifying automata construction.** The construction presented in Section 3 transforms in polynomial time a 3-CNF formula $\phi$ into an automaton $A_\phi$ satisfying certain synchronization properties. We want to generalize this construction so that it works well for qCSP instances. Let us make this formal by stating the following theorem.

**Theorem 4.5** There exists a procedure which takes a qCSP $\phi$ with $M$ constraints and $N$ input locations and returns a synchronizing automaton $B_\phi = (\{0, 1, 2\}, Q, \delta)$ satisfying:

- $|Q| = O(NM2^q)$,
- if $\text{Val}(\phi) = 1$ then $\text{Syn}(B_\phi) = N + O(1)$,
- $\text{Syn}(B_\phi) \geq \frac{N}{\text{Val}(\phi)}$.

Furthermore, the procedure runs in polynomial time with respect to $NM2^q$.

**Proof:** The automaton $B_\phi$ is built analogously to $A_\phi$. It has one sink state $s$ and $M$ gadgets corresponding to the constraints. We will modify a bit the construction of gadgets.

 Previously, a gadget for clause $C_i$ could verify whether given assignment $v \in \{0, 1\}^N$ satisfies $C_i$, meaning that $\delta(q_{0i0}^j, v) = q_{1N}^j$ iff $v$ satisfies $C_i$. We want to achieve a similar aim for a constraint $\phi_i$ instead of a clause. For clauses, we need only two states $\{q_{0i0}^j, q_{1i}^j\}$ for each layer $j \in \{1, 2, \ldots, N\}$. This is so because when the gadget keeps reading an assignment we need to keep only one bit of information (which stands for ”have we already satisfied any literal?”). For constraints the situation is more complicated, but we can build the gadget in such a way that it gathers information about all the already met important input places. Namely, we want to ensure that applying an assignment $v \in \{0, 1\}^N$ to an initial state $q_{0i0}^0$ lands us in the state $q_{WN}^w$ where $w \in \{0, 1\}^q$ contains the values from the input locations $\phi_i$ depend on. Note that the knowledge of $w$ is sufficient to check if $v$ satisfies $\phi_i$. So depending on $w$ we add transitions from $q_{WN}^w$ to either $s$ or back to $q_{0i0}^0$. We do not write down the details of the structure of the gadget, instead we provide an example in Fig. 2.
Figure 2: Gadget constructed for a constraint depending on two input locations. The constraint is 

\[
\text{xor of the third and the fifth input bit.}
\]

general (if we omit the transitions from the last, \(N\)-th layer) the gadget is a tree with \(2^N\) leaves. (There are \(q\) levels on which the tree branches.) The bound \(|Q| = O(\text{NM}^2)\) is now clear.

We define the action of the letter 2 similarly to the case of \(A_\phi\), meaning that it resets every 
gadget to the initial state: 

\[
\delta(q_w^{i0}, 2) = q_w^{i0} \quad \text{and} \quad \delta(s, 2) = s.
\]

If \(\text{Val}(\phi) = 1\) and \(v \in \{0, 1\}^N\) is a satisfying assignment, then the word \(2v0\) of length \(N + 2\) synchronizes \(B_\phi\). It remains to prove the \(\text{Syn}(B_\phi) \geq N\text{Val}(\phi)\). Define \(P = \{q_w^{i0} : i = 1, 2, \ldots, N\}\) and let \(w\) be the shortest word resetting \(P\) to \(s\). As previously, one can show that \(w\) does not 
contain any occurrence of symbol 2, see the proof of Proposition 3.4. Furthermore, the length of 
\(w\) is \(k(N + 1)\) for some \(k \geq 1\). Hence we can represent \(w\) as \(w = v_1c_1v_2c_2\ldots v_kc_k\), where each \(v_\ell\) has 
length \(N\) and each \(c_\ell\) is a single letter. Analyzing the structure of the gadgets one can see, that 
every constraint of \(\phi\) is satisfied by at least one of \(v_1, v_2, \ldots, v_k\), analogously to Observation 3.2. So \(\text{Cov}(\phi) \leq k\) and therefore \(\text{Syn}(B_\phi) \geq k(N + 1) \geq \text{Cov}(\phi) \cdot (N + 1) \geq \frac{N}{\text{Val}(\phi)}\).

**Proof of Theorem 4.1.** Theorem 4.1 now follows as a corollary. Suppose there is a polynomial 
time algorithm solving \(\text{SYNAPPX}\{0, 1, 2\}, \alpha\) for some \(\alpha > 1\). We will show how to solve 3-SAT in 
polynomial time. Take a 3-CNF formula \(\phi\). Use Theorem 4.4 with \(\varepsilon = \frac{1}{N\text{Val}(\phi)}\) to obtain the desired 
polynomial time reduction \(f\). Observe that \(q\) from the Theorem 4.4 is a fixed constant, so the 
automaton \(B_{f(\phi)}\) from Theorem 4.5 is of polynomial size. To test the satisfiability of \(\phi\), simply 
construct \(B_{f(\phi)}\) and find the \(\alpha\)-approximate shortest reset word \(w\) for it. If \(|w| \leq \alpha(\text{N} + \text{O}(1))\) then 
\(\phi\) is satisfiable, otherwise \(|w| \geq \text{Syn}(B_{f(\phi)}) \geq (\alpha + 1)N\) and \(\phi\) is not satisfiable. So we can decide 
satisfiability in polynomial time.

5 Final Preparations

This section is devoted to providing machinery and proving some introductory theorems necessary 
for the main result. Our main goal is to obtain appropriate \(NP\)-hard version of gap \(q\text{CSP}.\) For 
this we need a strong formulation of PCP theorem proved by Guruswami et al. [10]. We also need 
a method of reusing random bits when repeating a random experiment introduced by Impagliazzo
and Zuckerman [11]. Because we want to explicitly compute the parameters of the resulting gap qCSP, we briefly recap how the method works in this section. The idea is that, instead of generating fresh random bits for each repetitions of the experiment, we use a random walk in an expander to construct a pseudorandom sequence of bits, which is then used in subsequent repetitions. Because of the properties of an expander, which we summarize below, this is enough to significantly decrease the error probability.

5.1 Expanders and error reduction

A very good treatment of expanders can be found in [2]. For completeness we include all essential definitions, but we refer to the book for the proofs.

Definition 5.1 \((\lambda(G))\) Let \(A_G\) be the random walk matrix of \(G\) (that is, \(A_G\) is the adjacency matrix of \(G\) with each entry scaled by \(\frac{1}{d}\)). Let \(\lambda_1, \lambda_2, \ldots, \lambda_n \in [-1,1]\) be the eigenvalues of \(A_G\), sorted so that \(|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|\). We define \(\lambda(G)\) to be \(|\lambda_2|\).

Definition 5.2 \(((n,d,\lambda))-\text{expander}\) If \(G\) is an \(n\)-vertex \(d\)-regular multigraph with \(\lambda(G) \leq \lambda < 1\), then we say that \(G\) is an \((n,d,\lambda))-\text{graph}\).

It turns out that constructing a family of \((n,d,\lambda))-\text{graphs}\) for some fixed \(\lambda > 0\) is a pretty simple task, since a random \(d\)-regular graph is an expander with high confidence. However, a true challenge is to construct expanders explicitly and without any use of random bits. A beautiful example of such an construction was given by Margulis [14] and its analysis was later improved and simplified first by Gabber and Galil [8], and then by Jimbo and Maruoka [12], to yield the following.

Theorem 5.3 (Margulis, Galil and Gabber, Jimbo and Maruoka) Let \(G_{n^2}\) be the 8-regular graph on vertex set \(\mathbb{Z}_n^2\), with edges defined as follows: \((x,y)\) has neighbors \((x \pm 2y, y), (x \pm (2y + 1), y), (x, y \pm 2x), (x, y \pm (2x + 1))\) (addition is performed modulo \(n\)). Then \(G_{n^2}\) is an \((n^2,8,\frac{5\sqrt{2}}{8})\)-graph.

The following theorem can be now used to reduce the error probability.

Theorem 5.4 (Expander walks, 21.12 in [2]) Let \(G\) be an \((n,d,\lambda))-\text{graph}\) and let \(B \subseteq [n]\) satisfying \(|B| \leq \beta n\) for some \(\beta \in (0,1)\). Let \(X_1, X_2, \ldots, X_k\) be random variables denoting a \((k-1)\)-step random walk in \(G\), meaning that \(X_1\) is chosen uniformly in \([n]\) and \(X_{i+1}\) is a uniform random neighbor of \(X_i\). Then \(P(X_1 \in B \land X_2 \in B \land \ldots \land X_k \in B) \leq ((1-\lambda)\sqrt{\beta} + \lambda)^{k-1}\).

5.2 PCP theorem and hardness of approximation

We will obtain a certain gap version of qCSP, which will be then used as the key ingredient in our main result. We start with a strong formulation of the non-adaptive PCP theorem. It was first presented and proved by Guruswami et al. [10].

Theorem 5.5 \(NP = naPCP_{1,\frac{1}{4} + \varepsilon}(O(\log n), 6)\).

The theorem says that there exists a non-adaptive PCP verifier \(V\) for 3-SAT, which given a 3-CNF formula \(\phi\) over \(n\) variables and a proof of polynomial length, performs a polynomial time computation and returns a YES/NO answer, such that:
• $V$ uses $O(\log n)$ random bits,
• $V$ accesses only 6 bits from the proof, and their locations depend only on $\phi$ and on the random bits,
• if $\phi$ is satisfiable, then there exists a proof, such that $V$ answers YES with probability 1,
• if $\phi$ is not satisfiable, then for every proof, $V$ answers YES with probability at most $\frac{1}{4} + \varepsilon$,

where $\varepsilon$ is any fixed positive constant. Let us now formulate and prove hardness of a specific gap qCSP. This is an improved version of Theorem 4.4.

**Theorem 5.6** For every $c \in \mathbb{N}$ there exists a polynomial time reduction $f$ taking an $n$-variable 3-CNF formula $\phi$ and returning a qCSP instance $f(\phi)$ satisfying:

- $q \leq 6c \log n$,
- $f(\phi)$ has $O(n^r)$ variables and $O(n^{r + 3c})$ constraints,
- if $\phi$ is satisfiable, then $\text{Val}(f(\phi)) = 1$,
- if $\phi$ is not satisfiable, then $\text{Val}(f(\phi)) < n^{-\alpha}$,

where $\alpha = 0.08629$ and $r \in \mathbb{N}$.

The main idea is to start with the verifier $V$ from Theorem 5.5 and reduce its error probability. However, we want to keep the number of used random bits small, which can be done by using a random walk in an expander as explained in Theorem 5.4. Using the new verifier we can produce qCSP instances with the desired properties. Let us now dive into the details.

**Proof of Theorem 4.4.** Take a very small $\varepsilon > 0$ and consider the verifier $V$ from Theorem 5.5. Suppose it consumes at most $r \log n$ random bits, thus we may assume that the length of the proof is $O(n^r)$. We want to define another verifier $V'$, which makes more queries to the proof and needs more random bits, but its failure probability will be at most $n^{-\Omega(1)}$. Running $V$ independently $\Omega(\log n)$ times and checking if there was at least one reject is not acceptable, as it increases the number of random bits used to $\Omega(\log n)$ bits, which is too much. For this reason, instead of making $\Omega(\log n)$ fully independent runs, we save some random bits using expanders. Let $k = c \log n$ be the number of repetitions. Construct an $(O(2^r \log n), 8, \lambda)$-graph with $\lambda = \frac{5\sqrt{2}}{8}$ from Theorem 5.3, select a random starting vertex $X_1$ there, and choose a random walk of length $k - 1$ starting from $X_1$ obtaining vertices $X_2, X_3, \ldots, X_k$. Now run $V$ $k$ times using $X_i$ as the required stream of $r \log n$ bits for the $i$-th run. Answer YES if all runs returned YES.

Suppose the input formula $\phi$ is not satisfiable, then $V$ answers YES with probability at most $\frac{1}{4} + \varepsilon$. This means that the fraction of all vertices of $G$ corresponding to streams of random bits causing $V$ to return YES is at most $\frac{1}{4} + \varepsilon$. Now applying Theorem 5.4 to $G$ and $\beta = \frac{1}{4} + \varepsilon$, we obtain that the failure probability for $V'$ is at most $((1 - \lambda)\sqrt{\frac{1}{4} + \varepsilon + \lambda})^{c \log n}$, which can be bounded by $n^{-\alpha}$ with $\alpha < -\log((1 - \lambda)\sqrt{\frac{1}{4} + \varepsilon + \lambda}) \approx 0.0862902$ for $\varepsilon$ small enough. So taking $\alpha = 0.08629$ makes our bound true. Let us conclude the properties of our new verifier $V'$ on the input formula $\phi$:
• the length of the proof is still $O(n^r)$,
• $V'$ reads $6c \log n$ bits from the proof,
• $V'$ uses at most $r \log n + 3c \log n$ random bits, because choosing $X_1$ consumes $r \log n$ bits, and then we need to select $c \log n$ times a random neighbor in an 8-regular graph, one such step requires exactly 3 bits,
• if $\phi$ is satisfiable then there exists a proof, such that $V'$ answers YES with probability 1,
• if $\phi$ is not satisfiable then for every proof, $V'$ answers YES with probability at most $n^{-c_0}$.

Having such a verifier it is easy to construct the desired reduction. Take an $n$-variable 3-CNF formula $\phi$. $V'$ takes $\phi$ together with a proof of length $O(n^r)$. $f(\phi)$ has $O(n^r)$ variables corresponding to the locations in the proof. There are $2^{r \log n + 3c \log n} = n^{r+3c}$ constraints, one for every possible stream of used random bits. For a fixed random stream $V'$ is deterministic, meaning that it chooses $q = 6c \log n$ locations in the proof and accepts or not depending on the content of these locations there. This naturally defines a constraint depending on $q$ variables. Moreover, using $V'$ we can build this constraint in polynomial time (the size of the constraint is $O(2^q) = O(n^{6c})$). The number of constraints is polynomial as well, so the whole construction takes polynomial time. It is now easy to see that the probability of accepting $\phi$ by $V'$ corresponds to $\text{Val}(f(\phi))$. Hence $f(\phi)$ satisfies all the claimed requirements.

By the same reasoning, but with $k = c$, we can obtain Theorem 4.4. In the simpler version we can simply do $c$ independent repetitions without using expanders.

6 Main Result

In this section we present our main result, which is the following theorem.

**Theorem 6.1** For $\Sigma$ such that $|\Sigma| \geq 2$ and for some fixed constant $\varepsilon_0 > 0$, $\text{SYNAppx}(\Sigma, n^{\varepsilon_0})$ is not solvable in polynomial time, unless $P = NP$.

Our proof of the theorem will yield $\varepsilon_0 \approx 0.0095$. This can be improved significantly by using better quality expanders and working with a different version of PCP theorem. However, our main goal was to show Theorem 6.1 for some $\varepsilon_0$. We concentrate mainly on making the reasoning clear rather than obtaining a bigger constant.

We will show the theorem for $\Sigma = \{0, 1, 2\}$ as in the previous sections. This is sufficient, because the general version, with $|\Sigma| = 2$, follows from it by alphabet binarization as explained in Lemma 2.1.

**Proof of Theorem 6.1** The reasoning will be analogous to the one from the proof of Theorem 4.1, we only need to be a bit more careful, since now we are dealing with functions of $n$ instead of just constants.

Fix $c \in \mathbb{N}$, we will specify its value later. Let $\phi$ be an $n$-variable 3CNF formula, we transform it into a qCSP instance $f(\phi)$ using Theorem 5.6. Let $N$ and $M$ be the number of variables and the number of constraints of $f(\phi)$, respectively. Then $N = O(n^r)$, $M = O(n^{r+3c})$ and $q = 6c \log n$. We further transform $f(\phi)$ into an automaton $B_{f(\phi)}$ using Theorem 4.5 which satisfies the following conditions:
• the number of states of $B_f(\phi)$ is $s = \mathcal{O}(NM2^q) = \mathcal{O}(n^{2r+9c})$,
• if $\phi$ is satisfiable then $\text{Syn}(B_f(\phi)) = N + \mathcal{O}(1)$,
• if $\phi$ is not satisfiable then $\text{Syn}(B_f(\phi)) \geq \frac{N}{\text{Val}(f(\phi))} > Nn^{c\alpha}$.

The transformation $\phi \mapsto B_f(\phi)$ is performed in polynomial time. Note that if we could approximate the shortest reset word within a factor of $n^{c\alpha}$ then it would be possible to decide whether $\phi$ is satisfiable or not. Thus such an approximation is not possible. It remains to express $n^{c\alpha}$ as a function of the number of states of $B_f(\phi)$ instead of $n$. We see that $n = \Omega(s^{1/(2r+9c)})$, so $n^{c\alpha} = \Omega(s^{\frac{c\alpha}{2r+9c}})$. We don’t know any specific bound on $r$ (in fact $r$ is a very big number), so we can take a very big $c$ ($c \to \infty$) and then $\frac{c\alpha}{2r+9c} \approx \frac{\alpha}{9} = 0.0095878$. Thus $\text{SYNAppx}(\Sigma, n^{0.0095})$ is not solvable in polynomial time, unless $P = NP$. □

7 Conclusions and Open Problems

We have shown that approximating $\text{Syn}(A)$ within a factor of $n^{0.0095}$ is $NP$-hard. This considerably improves upon previous results: constant hardness [5] and $O(\log n)$ hardness [4, 9]. Assuming $P \neq NP$, our result gives a negative answer to a conjecture of Berlinkov, who asked for an $O(\log n)$ approximation algorithm [4]. As the best known polynomial time algorithm approximates the shortest reset sequence within a factor of $O(n)$, it is natural to ask whether $o(n)$-approximation is possible for this problem. At the same time another question appears: how far the inapproximability bound can be pushed further. It seems that even a stronger version of the PCP theorem and better quality expanders will not suffice to break the $O(n^{1/2})$ barrier. However, we strongly believe that this barrier can in fact be broken, and we conjecture that, for every $\varepsilon > 0$, $\text{SYNAppx}(\Sigma, n^{1-\varepsilon})$ is $NP$-hard.

References

[1] D. S. Ananichev and M. V. Volkov. Synchronizing generalized monotonic automata. *Theor. Comput. Sci.*, 330(1):3–13, Jan. 2005.

[2] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[3] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[4] M. Berlinkov. On two algorithmic problems about synchronizing automata. In A. Shur and M. Volkov, editors, *Developments in Language Theory*, volume 8633 of *Lecture Notes in Computer Science*, pages 61–67. Springer International Publishing, 2014.

[5] M. V. Berlinkov. Approximating the minimum length of synchronizing words is hard. *Theor. Comp. Sys.*, 54(2):211–223, Feb. 2014.

[6] I. Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3), 2007.
[7] D. Eppstein. Reset sequences for monotonic automata. *SIAM J. Comput.*, pages 500–510, 1990.

[8] O. Gabber and Z. Galil. Explicit constructions of linear-sized superconcentrators. *J. Comput. Syst. Sci.*, 22(3):407–420, 1981.

[9] M. Gerbush and B. Heeringa. Approximating minimum reset sequences. In *Implementation and Application of Automata*, volume 6482 of *Lecture Notes in Computer Science*, pages 154–162. Springer Berlin Heidelberg, 2011.

[10] V. Guruswami, D. Lewin, M. Sudan, and L. Trevisan. A tight characterization of NP with 3 query PCPs. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, FOCS ’98, pages 8–17, Washington, DC, USA, 1998. IEEE Computer Society.

[11] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, SFCS ’89, pages 248–253, Washington, DC, USA, 1989. IEEE Computer Society.

[12] S. Jimbo and A. Maruoka. Expanders obtained from affine transformations. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC ’85, pages 88–97. ACM, 1985.

[13] A. Kisielewicz, J. Kowalski, and M. Szykula. A fast algorithm finding the shortest reset words. In *COCOON*, pages 182–196, 2013.

[14] G. A. Margulis. Explicit constructions of concentrators. *Probl. Peredachi Inf.*, 9(4):71–80, 1973.

[15] J. Olschewski and M. Ummels. The complexity of finding reset words in finite automata. In *Proceedings of the 35th International Conference on Mathematical Foundations of Computer Science*, MFCS’10, pages 568–579. Springer-Verlag, Berlin, Heidelberg, 2010.

[16] J. Pin. On two combinatorial problems arising from automata theory. In *Combinatorial Mathematics Proceedings of the International Colloquium on Graph Theory and Combinatorics*, volume 75 of *North-Holland Mathematics Studies*, pages 535 – 548. North-Holland, 1983.

[17] I. Rystsov. Reset words for commutative and solvable automata. *Theoretical Computer Science*, 172(1–2):273 – 279, 1997.

[18] B. Steinberg. The Cerny conjecture for one-cluster automata with prime length cycle. *CoRR*, abs/1005.1835, 2010.

[19] A. Trakhtman. The road coloring and Cerny conjecture. In *Stringology*, pages 1–12, 2008.