Inapproximability of length of minimal synchronizing words

Mikhail Berlinkov

Department of mathematic and mechanics

Indo-Russian Workshop On Algebra, Combinatorics and Complexity, 2008

Outline

Outline

Synchronizing Automata

Applications

The study of synchronizing automata is one of the basic directions in the applied discrete math and computer science and has many applications, such as:

- Symbolic dynamic and coding theory
- Useful tool for testing of reactive systems (circuits and protocols) since the 60s to the 90s
- Branch of robotics, which deals with a part handling problems in industrial automata since the 80s
- DNA-computing and other...

Main Problems

- The basic problems in this field are the problems of searching and estimating length of minimal (shortest) synchronizing (reset) words for synchronizing automata.
- In particular, there is a famous Cerny's conjecture (Jan Cerny, 1964), which was formulated more than 40 years ago and hasn't been solved yet. It claims that for any synchronizing automaton with n states, the length of minimal synchronizing word is at most $(n-1)^2$.
- The best known approximate algorithm, which finds a reset word is the Greedy algorithm.

Main Problems

- The basic problems in this field are the problems of searching and estimating length of minimal (shortest) synchronizing (reset) words for synchronizing automata.
- In particular, there is a famous Cerny's conjecture (Jan Cerny, 1964), which was formulated more than 40 years ago and hasn't been solved yet. It claims that for any synchronizing automaton with n states, the length of minimal synchronizing word is at most $(n-1)^2$.
- The best known approximate algorithm, which finds a reset word is the Greedy algorithm.

Previous results

- It finds a synchronizing word of length at most $\frac{n^3-n}{6}$ (Peter Frankl, 1982). This algorithm can have an arbitrary large error in searching minimal length of reset word for a given automaton.
- There are several independent proofs of the fact, that the problem of finding the length of minimal synchronizing word is NP-complete.
- However, they don't imply, that the problem of searching the length of minimal synchronizing word with an error not greater than even 1 symbol is NP-complete.
- In this connection, existing of the approximate algorithm for finding length of minimal synchronizing word is of certain importance.

Previous results

- It finds a synchronizing word of length at most $\frac{n^3-n}{6}$ (Peter Frankl, 1982). This algorithm can have an arbitrary large error in searching minimal length of reset word for a given automaton.
- There are several independent proofs of the fact, that the problem of finding the length of minimal synchronizing word is NP-complete.
- However, they don't imply, that the problem of searching the length of minimal synchronizing word with an error not greater than even 1 symbol is NP-complete.
- In this connection, existing of the approximate algorithm for finding length of minimal synchronizing word is of certain importance.

Previous results

- It finds a synchronizing word of length at most $\frac{n^3-n}{6}$ (Peter Frankl, 1982). This algorithm can have an arbitrary large error in searching minimal length of reset word for a given automaton.
- There are several independent proofs of the fact, that the problem of finding the length of minimal synchronizing word is NP-complete.
- However, they don't imply, that the problem of searching the length of minimal synchronizing word with an error not greater than even 1 symbol is NP-complete.
- In this connection, existing of the approximate algorithm for finding length of minimal synchronizing word is of certain importance.

Paper results

Main result

The main result of this article is that there is no polynomial approximate algorithm, which finds the length of the minimal synchronizing word for a deterministic finite automaton with a finite error (assuming $P \neq NP$).

Outline

Deterministic finite automata

- ***A Deterministic finite automata*** (***DFA***) is a triple $\mathscr{A} = (Q, \Sigma, \delta)$, where Q is a finite ***set of states***, Σ is a finite ***alphabet*** and δ is a well defined ***transition function***, which maps elements from $Q \times \Sigma$ into Q.
- Action of δ can be naturally extended to the set $2^Q \times \Sigma^*$. Since the transition function is understood from the context, we'll denote $\forall S \subseteq Q, w \in \Sigma^*, S.w = \delta(S, w) = \bigcup_{q \in S} \delta(q, w)$.

synchronizing automata

- A DFA $\mathscr{A} = (Q, \Sigma, \delta)$ is called ***synchronizing***, if there exists a word $w \in \Sigma^*$, which maps all the states of the automaton to some state. Formally, $\exists q_0 \in Q : \forall q \in Q \to \delta(q, w) = q_0$. Then the word w is called ***synchronizing (reset)***.
- Let's denote $min_{synch}(A)$ the length of minimal synchronizing word for the automaton A.

Definition of error

- Let's define the error for the problem of finding minimal synchronizing words. For the synchronizing automaton A there is a ***set of possible solutions*** $F(A) = \{k \in \mathbb{N} | \exists$ reset word for the automaton $A - w : |w| \leq k\}$
- An ***optimal value*** is defined, as $OPT(A) = min\{F(A)\} = min_{s \in F(A)} s = min_{synch}(A)$ (is a length of minimal reset word).
- Assume, that M is an algorithm, which returns the feasible solution $M(A) \in F(A), i.e. min_{synch}(A) \leq M(A)$ for any given automaton A. Then M has an ***error*** $k \in \mathbb{R}$ iff $sup(\frac{M(A)}{OPT(A)}) = k$.

Base of the main result

Proposition

Any polynomial algorithm Alg, which finds the length of minimal synchronizing word, has an error at least 2 (assuming $P \neq NP$) or equivalently (in terms of NP-completeness)
The problem of approximation of minimal synchronizing words with error less than 2 is NP-complete problem.

Plan of the proof

Arguing by contradiction, suppose that there exists an algorithm Alg with an error less than 2. Perform the next steps:

- Take an arbitrary series of instances ψ of SAT. Denote the number of variables by n and number of clauses by m.
- Construct the synchronizing automaton $A(\psi)$, having polynomial size in the size of the instance, such that length of minimal reset word is equal to $n+2$, if ψ is satisfiable and not less than $2*n$ otherwise.
- Since $sup(\frac{2*n}{n+2}) = 2$ and the error of Alg is less than 2, then we conclude that we can solve the NP-complete problem SAT by using algorithm Alg in polynomial time. It gives a contradiction with assumption that $P \neq NP$.

Plan of the proof

Arguing by contradiction, suppose that there exists an algorithm Alg with an error less than 2. Perform the next steps:

- Take an arbitrary series of instances ψ of SAT. Denote the number of variables by n and number of clauses by m.
- Construct the synchronizing automaton $A(\psi)$, having polynomial size in the size of the instance, such that length of minimal reset word is equal to $n + 2$, if ψ is satisfiable and not less than $2 * n$ otherwise.
- Since $sup(\frac{2*n}{n+2}) = 2$ and the error of Alg is less than 2, then we conclude that we can solve the NP-complete problem SAT by using algorithm Alg in polynomial time. It gives a contradiction with assumption that $P \neq NP$.

Plan of the proof

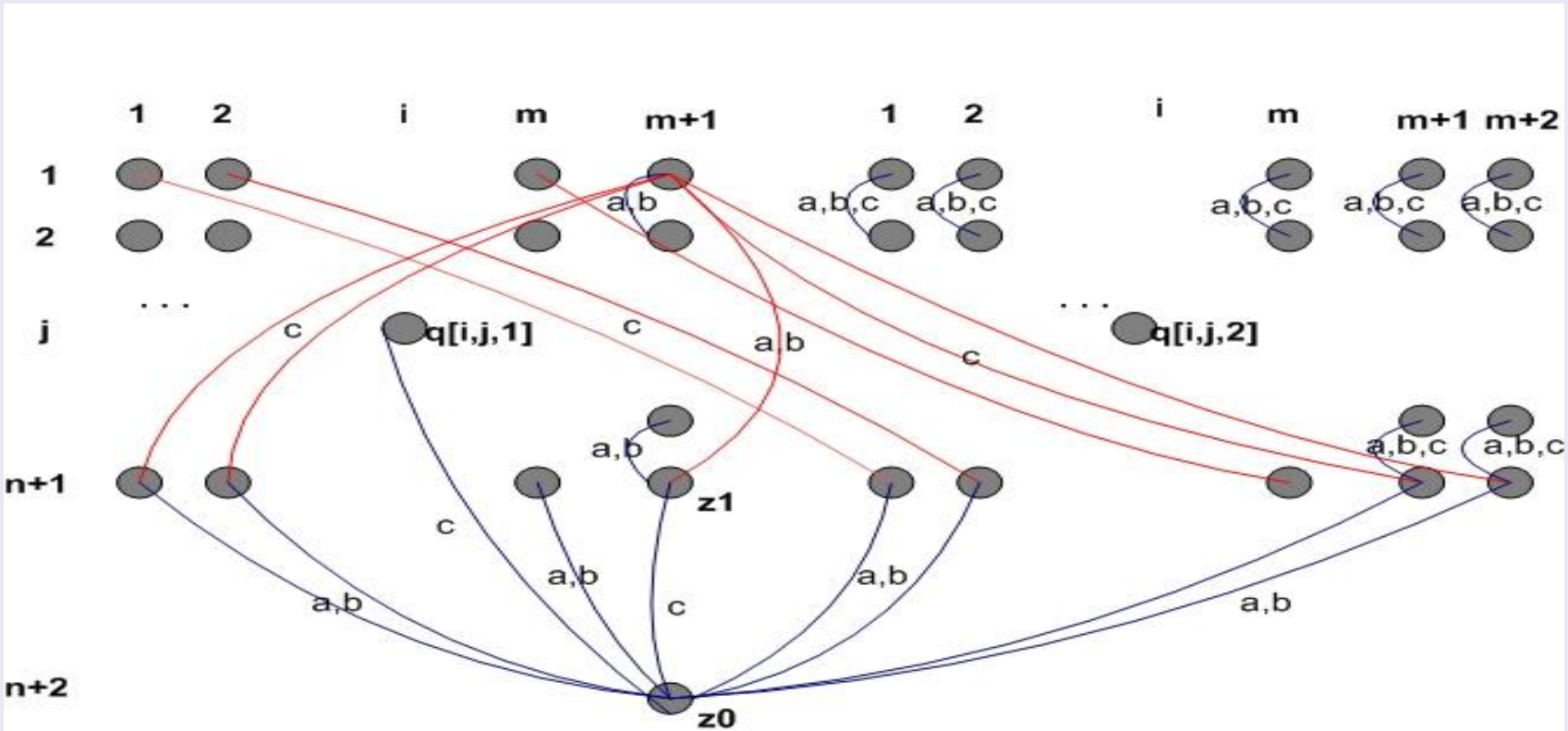
Arguing by contradiction, suppose that there exists an algorithm Alg with an error less than 2. Perform the next steps:

- Take an arbitrary series of instances ψ of SAT. Denote the number of variables by n and number of clauses by m.
- Construct the synchronizing automaton $A(\psi)$, having polynomial size in the size of the instance, such that length of minimal reset word is equal to $n + 2$, if ψ is satisfiable and not less than $2 * n$ otherwise.
- Since $sup(\frac{2*n}{n+2}) = 2$ and the error of Alg is less than 2, then we conclude that we can solve the NP-complete problem SAT by using algorithm Alg in polynomial time. It gives a contradiction with assumption that $P \neq NP$.

Plan of the proof

Construction of the automaton $A(\psi)$

On this picture blue edges stand for down direction and red edges
stand for up direction. $A = (Q = Q_1 \bigcup Q_2 \bigcup Q_3, \Sigma = \{a, b, c\}, \delta)$.



Main result

Theorem

For any $r \in \mathbb{N}$, any polynomial algorithm Alg, which finds the length of minimal synchronizing word, has an error at least r (assuming $P \neq NP$)
or equivalently (in terms of NP-completeness)
The problem of approximation of minimal synchronizing words with finite error is NP-complete problem.

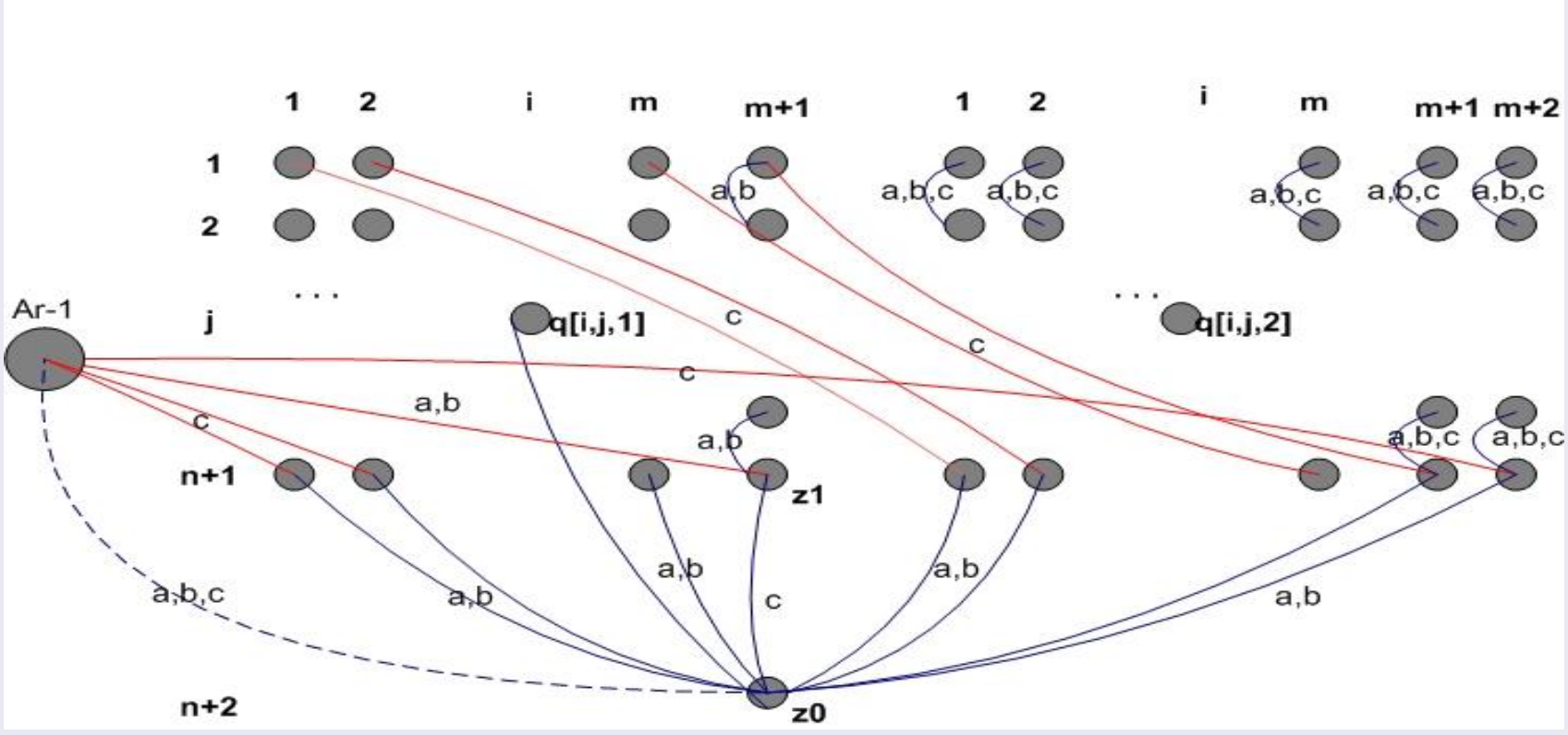
Plan of the proof

The proof of this theorem has the similar sequence of steps, as in the previous proposition, but at first uses induction, having the construction from the previous proposition as a base.

Plan of the proof

Construction of the automaton $A(\psi)$

This formula and picture show the structure of the automaton $A(\psi)$, constructed by induction. $A_r \simeq A_{r-1} \bigcup ((A_2 \setminus z_0) \times A_{r-1})$



2-letter alphabet case

In the theorem we used 3-letter alphabet. One can ask: will it hold true for a 2-letter alphabet?. Yes it does, we formulate the next corollary.

Corollary

For any $r \in \mathbb{N}$, any polynomial algorithm Alg, which finds the minimal synchronizing word for an automaton with 2-letter alphabet, has an error not less than r (assuming $P \neq NP$). Or, in other words, corresponding problem is NP-complete.

Plan for the proof

It's an easy consequence of the main theorem and automaton transformation to the 2-letter automaton, where the first letter imitate the choice of a letter in the initial automaton and the second imitate the action of this letter.

Outline

Summary

Result of the paper

This paper gives the negative answer for the question of existing of ”effective” approximation algorithm for finding minimal synchronizing word, even in 2-letter alphabet case.

Open problems

- Existing of the effective approximate algorithm for some subclasses of automata
- Conjugative problem for coloring digraph

Summary

Result of the paper

This paper gives the negative answer for the question of existing of ”effective” approximation algorithm for finding minimal synchronizing word, even in 2-letter alphabet case.

Open problems

- Existing of the effective approximate algorithm for some subclasses of automata
- Conjugative problem for coloring digraph

Thank you!