Approximating lengths of reset words

Mikhail V. Berlinkov

Ural State University, Ekaterinburg, Russia
DAAST WIEN 2010

DAAST WIEN 2010

Supported by AuthoMathA, CMUC, ESF, TUWIEN and

Synchronizing Automata

Let $\mathscr{A}$ be a complete **deterministic finite automaton** whose input alphabet is Σ and whose state set is Q.

- Denote by $S.v$ the image of the subset $S \subseteq Q$ under the action of the word $v \in \Sigma^*$.
- A word v is called reset (or synchronizing) word for $\mathscr{A}$ iff $|Q.v| = 1$ (equivalently $q.v = p.v$ for all $q, p \in Q$).
- $\mathscr{A}$ is called synchronizing if it possesses some reset word.
- $\mathfrak{C}(\mathscr{A})$ denotes the **minimum length** of reset words for $\mathscr{A}$ and this function is usually called Cerny function and let us call its value reset length of $\mathscr{A}$.

Synchronizing Automata

Let $\mathscr{A}$ be a complete **deterministic finite automaton** whose input alphabet is Σ and whose state set is Q.

- Denote by $S.v$ the image of the subset $S \subseteq Q$ under the action of the word $v \in \Sigma^*$.
- A word v is called reset (or synchronizing) word for $\mathscr{A}$ iff $|Q.v| = 1$ (equivalently $q.v = p.v$ for all $q, p \in Q$).
- $\mathscr{A}$ is called synchronizing if it possesses some reset word.
- $\mathfrak{C}(\mathscr{A})$ denotes the **minimum length** of reset words for $\mathscr{A}$ and this function is usually called Cerny function and let us call its value reset length of $\mathscr{A}$.

Synchronizing Automata

Let $\mathscr{A}$ be a complete **deterministic finite automaton** whose input alphabet is Σ and whose state set is Q.

- Denote by $S.v$ the image of the subset $S \subseteq Q$ under the action of the word $v \in \Sigma^*$.
- A word v is called reset (or synchronizing) word for $\mathscr{A}$ iff $|Q.v| = 1$ (equivalently $q.v = p.v$ for all $q, p \in Q$).
- $\mathscr{A}$ is called synchronizing if it possesses some reset word.
- $\mathfrak{C}(\mathscr{A})$ denotes the **minimum length** of reset words for $\mathscr{A}$ and this function is usually called Cerny function and let us call its value reset length of $\mathscr{A}$.

Let $\mathscr{A}$ be a complete **deterministic finite automaton** whose input alphabet is Σ and whose state set is Q.

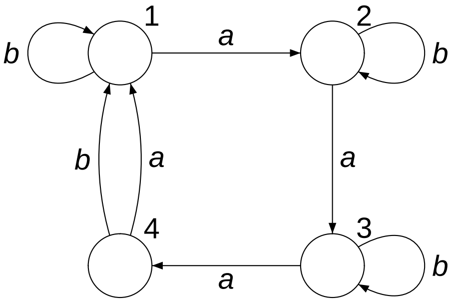
- Denote by $S.v$ the image of the subset $S \subseteq Q$ under the action of the word $v \in \Sigma^*$.
- A word v is called reset (or synchronizing) word for $\mathscr{A}$ iff $|Q.v| = 1$ (equivalently $q.v = p.v$ for all $q, p \in Q$).
- $\mathscr{A}$ is called synchronizing if it possesses some reset word.
- $\mathfrak{C}(\mathscr{A})$ denotes the **minimum length** of reset words for $\mathscr{A}$ and this function is usually called Cerny function and let us call its value reset length of $\mathscr{A}$.

Synchronizing Automata

Let $\mathscr{A}$ be a complete **deterministic finite automaton** whose input alphabet is Σ and whose state set is Q.

- Denote by $S.v$ the image of the subset $S \subseteq Q$ under the action of the word $v \in \Sigma^*$.
- A word v is called reset (or synchronizing) word for $\mathscr{A}$ iff $|Q.v| = 1$ (equivalently $q.v = p.v$ for all $q, p \in Q$).
- $\mathscr{A}$ is called synchronizing if it possesses some reset word.
- $\mathfrak{C}(\mathscr{A})$ denotes the **minimum length** of reset words for $\mathscr{A}$ and this function is usually called Cerny function and let us call its value reset length of $\mathscr{A}$.

Synchronizing Automaton $\mathscr{A}$ by "greedy" algorithm

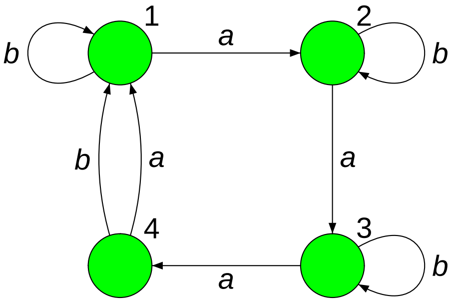


A reset word is $v =$ *baababaaab*.

$Q.v =$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.



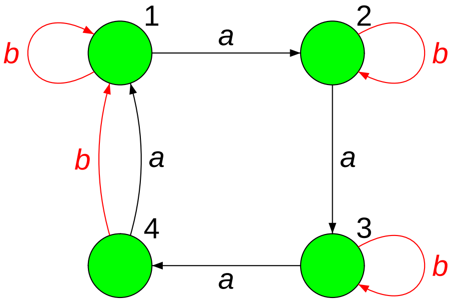
A reset word is $v =$ *baababaaab*.

$Q.v =$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.

Synchronizing Automaton $\mathscr{A}$ by "greedy" algorithm



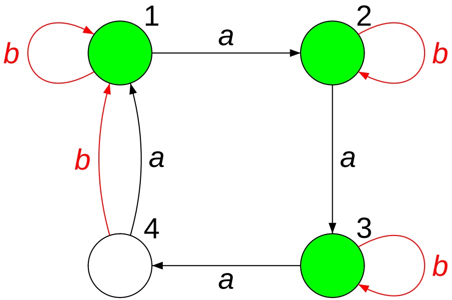
A reset word is $v =$ *baababaaab*.

$Q.v = \{1, 2, 3, 4\}$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.

Synchronizing Automaton $\mathscr{A}$ by "greedy" algorithm



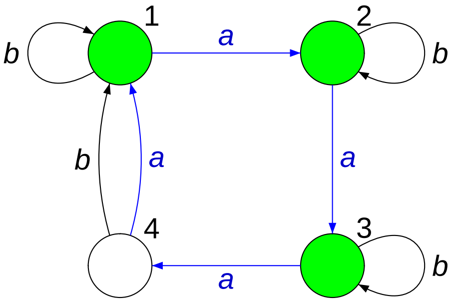
A reset word is $v = baababaaab$.

$Q.v = \{1, 2, 3\}$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.

Synchronizing Automaton $\mathscr{A}$ by "greedy" algorithm



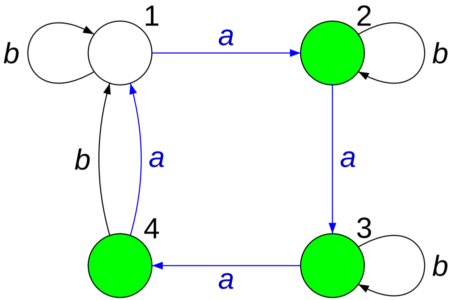
A reset word is $v =$ *baababaaab*.

$Q.v = \{1, 2, 3\}$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.

Synchronizing Automaton $\mathscr{A}$ by "greedy" algorithm



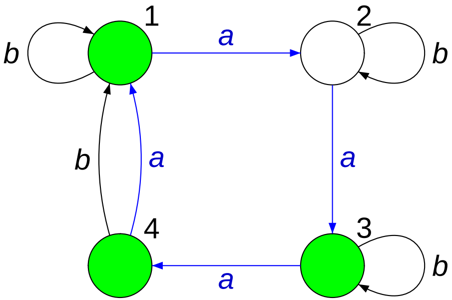
A reset word is $v = baababaaab$.

$Q.v = \{2, 3, 4\}$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.

Synchronizing Automaton $\mathscr{A}$ by "greedy" algorithm

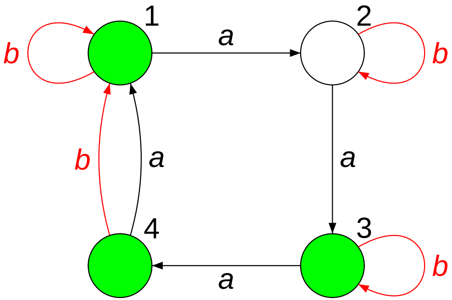


A reset word is $v = baababaaab$.

$Q.v = \{1, 3, 4\}$

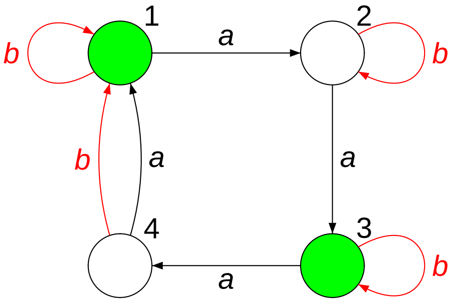
Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.



A reset word is $v = baababaaab$.
$Q.v = \{1, 3, 4\}$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.
In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.



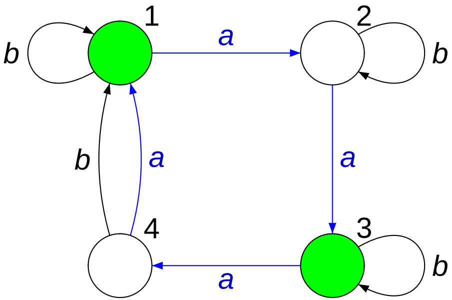
A reset word is $v = baababaaab$.

$Q.v = \{1, 3\}$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.

Synchronizing Automaton $\mathscr{A}$ by "greedy" algorithm



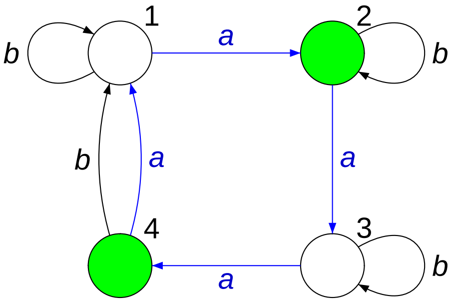
A reset word is $v = baababaaab$.

$Q.v = \{1, 3\}$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.

Synchronizing Automaton $\mathscr{A}$ by "greedy" algorithm

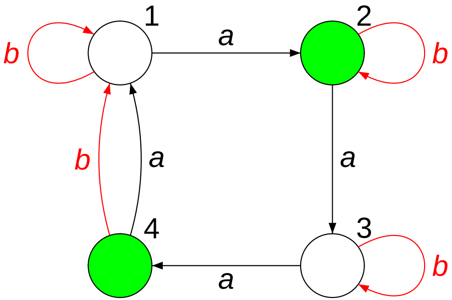


A reset word is $v = baababaaab$.

$Q.v = \{2, 4\}$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.



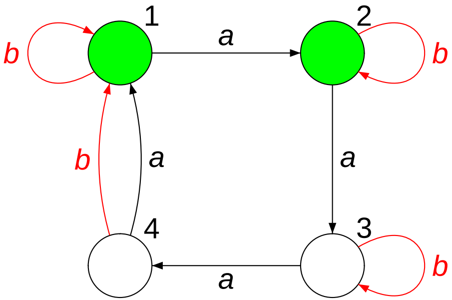
A reset word is $v = baaba$baaab.

$Q.v = \{2, 4\}$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.

Synchronizing Automaton $\mathscr{A}$ by "greedy" algorithm



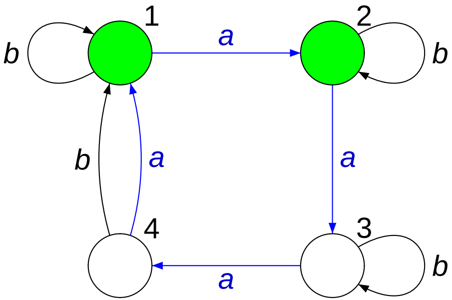
A reset word is $v = baababbaaab$.

$Q.v = \{1, 2\}$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence
$\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus
$\mathfrak{C}(\mathscr{A}) = 9 < |v|$.

Synchronizing Automaton $\mathscr{A}$ by "greedy" algorithm

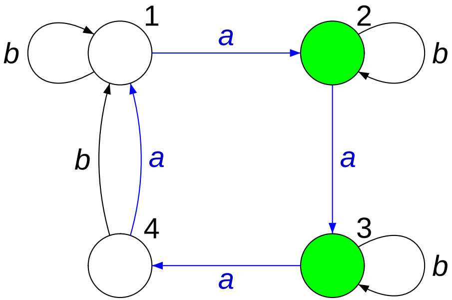


A reset word is $v =$ *baababaaab*.

$Q.v = \{1, 2\}$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.



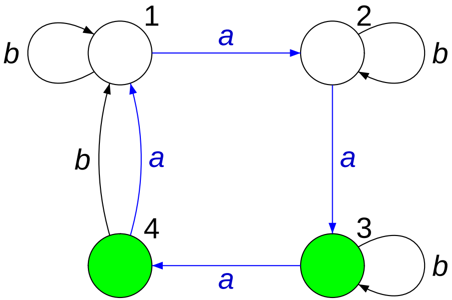
A reset word is $v = baababa\mathit{aab}$.

$Q.v = \{2, 3\}$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.

Synchronizing Automaton $\mathscr{A}$ by "greedy" algorithm



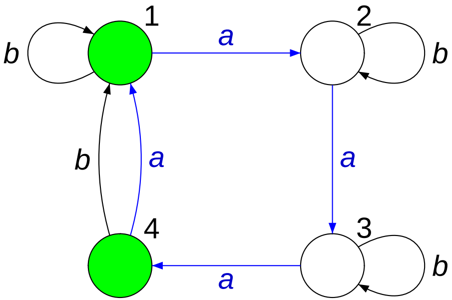
A reset word is $v = baababaa\mathbf{a}ab$.

$Q.v = \{3, 4\}$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.

Synchronizing Automaton $\mathscr{A}$ by "greedy" algorithm

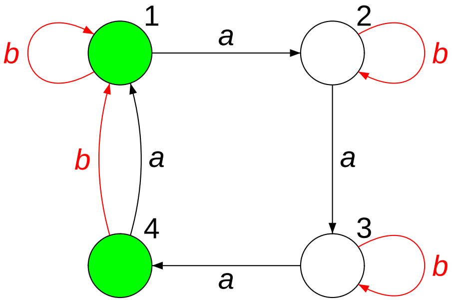


A reset word is $v = baababaaab$.

$Q.v = \{1, 4\}$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.



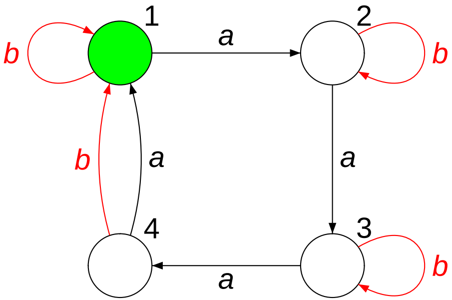
A reset word is $v =$ *baababaaab*.

$Q.v = \{1, 4\}$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.

Synchronizing Automaton $\mathscr{A}$ by "greedy" algorithm



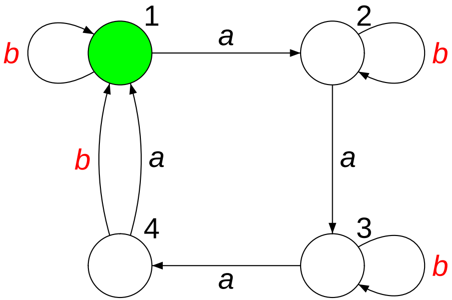
A reset word is $v =$ *baababaaab*.

$Q.v = \{1\}$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence $\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.

In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus $\mathfrak{C}(\mathscr{A}) = 9 < |v|$.

Synchronizing Automaton $\mathscr{A}$ by "greedy" algorithm



A reset word is $v = baababaaab$.
$Q.v = \{1\}$

Since $|Q.v| = 1$ the word v is a reset word for $\mathscr{A}$ whence
$\mathfrak{C}(\mathscr{A}) \leq |v| = 10$.
In fact the shortest reset word for $\mathscr{A}$ is ba^3ba^3b of length 9 and thus
$\mathfrak{C}(\mathscr{A}) = 9 < |v|$.

Main Questions And Outline of the Talk

Given an automaton $\mathscr{A}$;

- How to find some reset word for $\mathscr{A}$ if it exists?

Given a synchronizing automaton $\mathscr{A}$;

- How to find "relatively" short reset word for $\mathscr{A}$ or its length?

Main Questions And Outline of the Talk

Given an automaton $\mathscr{A}$;

- How to find some reset word for $\mathscr{A}$ if it exists?

Given a synchronizing automaton $\mathscr{A}$;

- How to find "relatively" short reset word for $\mathscr{A}$ or its length?

Main Questions And Outline of the Talk

Given an automaton $\mathscr{A}$;

- How to find some reset word for $\mathscr{A}$ if it exists?

Given a synchronizing automaton $\mathscr{A}$;

- How to find "relatively" short reset word for $\mathscr{A}$ or its length?

Search For Some Reset Word

Synchronization Criterion | Černý, 1964

An automaton $\mathscr{A}$ is synchronizing iff each pair of states p, q can be merged by some word v, i.e. $p.v = q.v$.

Find-Sync-Word | in $O(n^3)$ (Greedy algorithm)

Given An n-state automaton $\mathscr{A}$;
Return Some reset word for $\mathscr{A}$ if it exists.

Check-Sync | in $O(n^2)$

Given An n-state automaton $\mathscr{A}$;
Return Yes iff $\mathscr{A}$ is synchronizing.

Search For Some Reset Word

Synchronization Criterion | Černý, 1964

An automaton $\mathscr{A}$ is synchronizing iff each pair of states p, q can be merged by some word v, i.e. $p.v = q.v$.

Find-Sync-Word | in $O(n^3)$ (Greedy algorithm)

Given An n-state automaton $\mathscr{A}$;
Return Some reset word for $\mathscr{A}$ if it exists.

Check-Sync | in $O(n^2)$

Given An n-state automaton $\mathscr{A}$;
Return Yes iff $\mathscr{A}$ is synchronizing.

Search For Some Reset Word

Synchronization Criterion | Černý, 1964

An automaton $\mathscr{A}$ is synchronizing iff each pair of states p, q can be merged by some word v, i.e. $p.v = q.v$.

Find-Sync-Word | in $O(n^3)$ (Greedy algorithm)

Given An n-state automaton $\mathscr{A}$;
Return Some reset word for $\mathscr{A}$ if it exists.

Check-Sync | in $O(n^2)$

Given An n-state automaton $\mathscr{A}$;
Return Yes iff $\mathscr{A}$ is synchronizing.

Main Questions And Outline of the Talk

Given an automaton $\mathscr{A}$;

- How to find some reset word for $\mathscr{A}$ if it exists?
  Černý in 1964 proved synchronization criterion which allows to find reset word in $O(n^3)$ time.

Given a synchronizing automaton $\mathscr{A}$;

- How to find "relatively" short reset word for $\mathscr{A}$ or its length?
  [Unless $P = NP$], no polynomial time algorithm approximates reset length of $\mathscr{A}$ within a constant factor (CSR 2010).

Exact Decision Variants of The Problem

Check-Eq-Reset-Length | *NP* and *co-NP* hard

Given A synchronizing automaton $\mathscr{A}$ and a positive integer k;
Question: $\mathfrak{C}(\mathscr{A}) = k$?

Unless $\mathrm{NP} = \text{co-NP}$, even non-deterministic polynomial-time algorithms cannot solve the above problem.

Check-Reset-Length | *NP*-complete (Rystsov, Eppstein and others)

Given A synchronizing automaton $\mathscr{A}$ and a positive integer k;
Question: $\mathfrak{C}(\mathscr{A}) \leq k$?

For each ψ of SAT-problem with n variables and m clauses he constructed $Epp(\psi)$ such that
$\mathfrak{C}(Epp(\psi)) = n$ if ψ is satisfiable,
$\mathfrak{C}(Epp(\psi)) = n + 1$ if ψ is not satisfiable.

Exact Decision Variants of The Problem

Check-Eq-Reset-Length | *NP* and *co-NP* hard

Given A synchronizing automaton $\mathscr{A}$ and a positive integer k;
Question: $\mathfrak{C}(\mathscr{A}) = k$?

Unless $\mathrm{NP} = \text{co-NP}$, even non-deterministic polynomial-time algorithms cannot solve the above problem.

Check-Reset-Length | *NP*-complete (Rystsov, Eppstein and others)

Given A synchronizing automaton $\mathscr{A}$ and a positive integer k;
Question: $\mathfrak{C}(\mathscr{A}) \leq k$?

For each ψ of SAT-problem with n variables and m clauses he constructed $Epp(\psi)$ such that
$\mathfrak{C}(Epp(\psi)) = n$ if ψ is satisfiable,
$\mathfrak{C}(Epp(\psi)) = n + 1$ if ψ is not satisfiable.

Exact Decision Variants of The Problem

Check-Eq-Reset-Length | *NP* and *co-NP* hard

Given A synchronizing automaton $\mathscr{A}$ and a positive integer k;
Question: $\mathfrak{C}(\mathscr{A}) = k$?

Unless $\mathrm{NP} = \mathrm{co\text{-}NP}$, even non-deterministic polynomial-time algorithms cannot solve the above problem.

Check-Reset-Length | *NP*-complete (Rystsov, Eppstein and others)

Given A synchronizing automaton $\mathscr{A}$ and a positive integer k;
Question: $\mathfrak{C}(\mathscr{A}) \leq k$?

For each ψ of SAT-problem with n variables and m clauses he constructed $Epp(\psi)$ such that
$\mathfrak{C}(Epp(\psi)) = n$ if ψ is satisfiable,
$\mathfrak{C}(Epp(\psi)) = n + 1$ if ψ is not satisfiable.

Exact Decision Variants of The Problem

Check-Eq-Reset-Length | *NP* and *co-NP* hard

Given A synchronizing automaton $\mathscr{A}$ and a positive integer k;
Question: $\mathfrak{C}(\mathscr{A}) = k$?

Unless $\mathrm{NP} = \text{co-NP}$, even non-deterministic polynomial-time algorithms cannot solve the above problem.

Check-Reset-Length | *NP*-complete (Rystsov, Eppstein and others)

Given A synchronizing automaton $\mathscr{A}$ and a positive integer k;
Question: $\mathfrak{C}(\mathscr{A}) \leq k$?

For each ψ of SAT-problem with n variables and m clauses he constructed $Epp(\psi)$ such that
$\mathfrak{C}(Epp(\psi)) = n$ if ψ is satisfiable,
$\mathfrak{C}(Epp(\psi)) = n + 1$ if ψ is not satisfiable.

Examples for two instances

$\Sigma = \{a, b\}, \qquad Q = \{q_{i,j} \mid i \in [1, n+1], j \in [1, m]\} \cup \{z_0\}$

$$q_{i,j}.d = \begin{cases} z_0 & \text{if } (d = a \text{ and } x_j \in c_i) \text{ or } (d = b \text{ and } \overline{x_j} \in c_i), \\ z_0 & \text{if } j = n+1, \\ q_{i,j+1} & \text{otherwise.} \end{cases}$$

$$\psi_1 = (x_3 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3}),$$
$$\psi_2 = (x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3}).$$

It is clear ψ_1 is satisfiable for the truth assignment $\tau : x_1 = x_2 = 0,\ x_3 = 1$ while ψ_2 is not satisfiable.

The word $v(\tau) = bba$ synchronizes $Epp(\psi_1)$ and the word a^4 is a reset word of minimum length for $Epp(\psi_2)$.

Examples for two instances

$\Sigma = \{a, b\}, \qquad Q = \{q_{i,j} \mid i \in [1, n+1], j \in [1, m]\} \cup \{z_0\}$

$$q_{i,j}.d = \begin{cases} z_0 & \text{if } (d = a \text{ and } x_j \in c_i) \text{ or } (d = b \text{ and } \overline{x_j} \in c_i), \\ z_0 & \text{if } j = n+1, \\ q_{i,j+1} & \text{otherwise.} \end{cases}$$

$$\psi_1 = (x_3 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3}),$$
$$\psi_2 = (x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3}).$$

It is clear ψ_1 is satisfiable for the truth assignment $\tau : x_1 = x_2 = 0,\ x_3 = 1$ while ψ_2 is not satisfiable.

The word $v(\tau) = bba$ synchronizes $Epp(\psi_1)$ and the word a^4 is a reset word of minimum length for $Epp(\psi_2)$.

Examples for two instances

$$\Sigma = \{a, b\}, \qquad Q = \{q_{i,j} \mid i \in [1, n+1], j \in [1, m]\} \cup \{z_0\}$$

$$q_{i,j}.d = \begin{cases} z_0 & \text{if } (d = a \text{ and } x_j \in c_i) \text{ or } (d = b \text{ and } \overline{x_j} \in c_i), \\ z_0 & \text{if } j = n+1, \\ q_{i,j+1} & \text{otherwise.} \end{cases}$$

$$\psi_1 = (x_3 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3}),$$
$$\psi_2 = (x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3}).$$

It is clear ψ_1 is satisfiable for the truth assignment $\tau : x_1 = x_2 = 0,\ x_3 = 1$ while ψ_2 is not satisfiable.

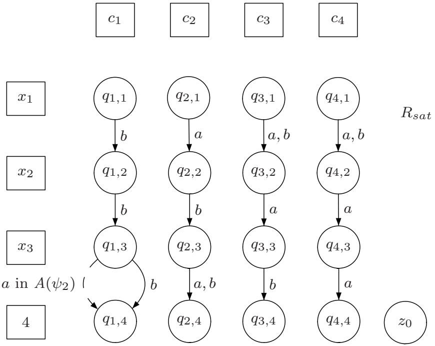
The word $v(\tau) = bba$ synchronizes $Epp(\psi_1)$ and the word a^4 is a reset word of minimum length for $Epp(\psi_2)$.

The automata $Epp(\psi_1)$ and $Epp(\psi_2)$

$\psi_1 = ({\color{red}x_3} \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3})$

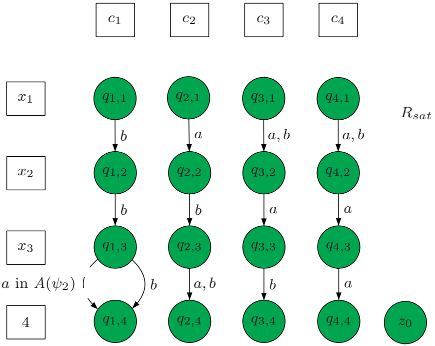
An applied prefix is $v = 1$

(First row)$.v = \{q_{1,1}, q_{2,1}, q_{3,1}, q_{4,1}, z_0\}$



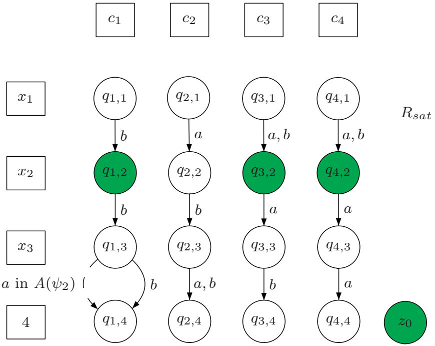
The automata $Epp(\psi_1)$ and $Epp(\psi_2)$

$\psi_1 = (x_3 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3})$
An applied prefix is $v = 1$
(First row).$v = \{q_{1,1}, q_{2,1}, q_{3,1}, q_{4,1}, z_0\}$



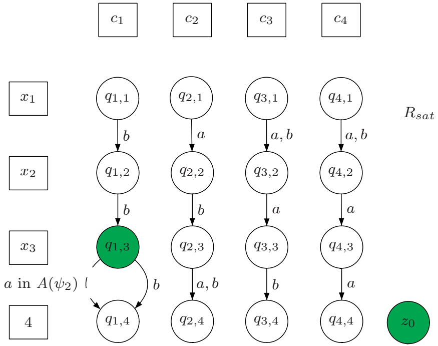
The automata $Epp(\psi_1)$ and $Epp(\psi_2)$

$\psi_1 = (x_3 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3})$
An applied prefix is $v = b$.
(First row).$v = \{q_{1,2}, q_{3,2}, q_{4,2}, z_0\}$



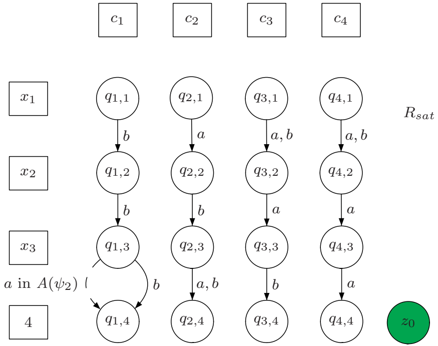
The automata $Epp(\psi_1)$ and $Epp(\psi_2)$

$\psi_1 = (x_3 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3})$
An applied prefix is $v = bb$.
(First row).$v = \{q_{1,3}, z_0\}$



The automata $Epp(\psi_1)$ and $Epp(\psi_2)$

$\psi_1 = (x_3 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3})$
An applied prefix is $v = bba$.
(First row)$.v = \{z_0\}$



Exact Search Variants of The Problem

Search-Reset-Length | $FP^{NP[log]}$-complete

Given A synchronizing automaton $\mathscr{A}$;
Return $\mathfrak{C}(\mathscr{A})$.

Search-Shortest-Reset-Word | in FP^{NP} and $FP^{NP[log]}$-hard

Given A synchronizing automaton $\mathscr{A}$;
Return Some shortest reset word for $\mathscr{A}$.

FP^{NP} and $FP^{NP[log]}$ are complexity classes of search problems that can be solved by a deterministic polynomial time algorithm equipped with an ability to use an oracle for any NP-complete problem by polynomial or logarithmic times respectively.
These results were proved by Olschëwski and Ummels in 2010

Exact Search Variants of The Problem

Search-Reset-Length | $FP^{NP[log]}$-complete

Given A synchronizing automaton $\mathscr{A}$;

Return $\mathfrak{C}(\mathscr{A})$.

Search-Shortest-Reset-Word | in FP^{NP} and $FP^{NP[log]}$-hard

Given A synchronizing automaton $\mathscr{A}$;

Return Some shortest reset word for $\mathscr{A}$.

FP^{NP} and $FP^{NP[log]}$ are complexity classes of search problems that can be solved by a deterministic polynomial time algorithm equipped with an ability to use an oracle for any *NP*-complete problem by polynomial or logarithmic times respectively.

These results were proved by Olschëwski and Ummels in 2010

Exact Search Variants of The Problem

Search-Reset-Length | $FP^{NP[log]}$-complete

Given A synchronizing automaton $\mathscr{A}$;
Return $\mathfrak{C}(\mathscr{A})$.

Search-Shortest-Reset-Word | in FP^{NP} and $FP^{NP[log]}$-hard

Given A synchronizing automaton $\mathscr{A}$;
Return Some shortest reset word for $\mathscr{A}$.

FP^{NP} and $FP^{NP[log]}$ are complexity classes of search problems that can be solved by a deterministic polynomial time algorithm equipped with an ability to use an oracle for any NP-complete problem by polynomial or logarithmic times respectively.

These results were proved by Olschëwski and Ummels in 2010

Exact Search Variants of The Problem

Search-Reset-Length | $FP^{NP[log]}$-complete

Given A synchronizing automaton $\mathscr{A}$;
Return $\mathfrak{C}(\mathscr{A})$.

Search-Shortest-Reset-Word | in FP^{NP} and $FP^{NP[log]}$-hard

Given A synchronizing automaton $\mathscr{A}$;
Return Some shortest reset word for $\mathscr{A}$.

FP^{NP} and $FP^{NP[log]}$ are complexity classes of search problems that can be solved by a deterministic polynomial time algorithm equipped with an ability to use an oracle for any NP-complete problem by polynomial or logarithmic times respectively.
These results were proved by Olschëwski and Ummels in 2010

Approximation Variant of The Problem

Key Question | Volkov 2008

Can we approximately find the **reset length** within a constant factor in a polynomial time?

An algorithm M **approximates reset length** in $\mathcal{K}$ if for an arbitrary DFA $\mathscr{A} \in \mathcal{K}$, the algorithm calculates a positive integer $M(\mathscr{A})$ such that $M(\mathscr{A}) \geq \mathfrak{C}(\mathscr{A})$.

$\sup\{\frac{M(\mathscr{A})}{\mathfrak{C}(\mathscr{A})} \mid \mathscr{A} \in \mathcal{K}\}$ is an **approximation factor** of M.

Is there a polynomial-time approximation algorithm within a constant factor for Search-Reset-Length?

Approximation Variant of The Problem

Key Question | Volkov 2008

Can we approximately find the **reset length** within a constant factor in a polynomial time?

An algorithm M **approximates reset length** in $\mathcal{K}$ if for an arbitrary DFA $\mathscr{A} \in \mathcal{K}$, the algorithm calculates a positive integer $M(\mathscr{A})$ such that $M(\mathscr{A}) \geq \mathfrak{C}(\mathscr{A})$.

$\sup\{\frac{M(\mathscr{A})}{\mathfrak{C}(\mathscr{A})} \mid \mathscr{A} \in \mathcal{K}\}$ is an **approximation factor** of M.

Is there a polynomial-time approximation algorithm within a constant factor for Search-Reset-Length?

Approximation Variant of The Problem

Key Question | Volkov 2008

Can we approximately find the **reset length** within a constant factor in a polynomial time?

An algorithm M **approximates reset length** in $\mathcal{K}$ if for an arbitrary DFA $\mathscr{A} \in \mathcal{K}$, the algorithm calculates a positive integer $M(\mathscr{A})$ such that $M(\mathscr{A}) \geq \mathfrak{C}(\mathscr{A})$.

$\sup\{\frac{M(\mathscr{A})}{\mathfrak{C}(\mathscr{A})} \mid \mathscr{A} \in \mathcal{K}\}$ is an **approximation factor** of M.

Is there a polynomial-time approximation algorithm within a constant factor for Search-Reset-Length?

Approximation Variant of The Problem

Key Question | Volkov 2008

Can we approximately find the **reset length** within a constant factor in a polynomial time?

An algorithm M **approximates reset length** in $\mathcal{K}$ if for an arbitrary DFA $\mathscr{A} \in \mathcal{K}$, the algorithm calculates a positive integer $M(\mathscr{A})$ such that $M(\mathscr{A}) \geq \mathfrak{C}(\mathscr{A})$.

$\sup\{\frac{M(\mathscr{A})}{\mathfrak{C}(\mathscr{A})} \mid \mathscr{A} \in \mathcal{K}\}$ is an **approximation factor** of M.

Is there a polynomial-time approximation algorithm within a constant factor for Search-Reset-Length?

The First Result

Theorem 1.

No polynomial-time algorithm approximates the minimum length of reset words within a constant factor.

For every ψ of SAT with n variables we construct synchronizing automaton $\mathscr{A}_r(\psi)$ for $r = 2, 3, \ldots$ such that
$\mathfrak{C}(\mathscr{A}_r(\psi)) \leq n + r$ and $c^{r-1}v(\tau)c$ is reset if ψ is satisfiable on τ,
$\mathfrak{C}(\mathscr{A}_r(\psi)) > r(n-1)$ if ψ is not satisfiable.

$\mathscr{A}_r(\psi)$ is constructed by a "substitution" $\mathscr{A}_{r-1}(\psi)$ instead every state of $\mathscr{A}_2(\psi)$ and some additional modification.

The First Result

Theorem 1.

No polynomial-time algorithm approximates the minimum length of reset words within a constant factor.

For every ψ of SAT with n variables we construct synchronizing automaton $\mathscr{A}_r(\psi)$ for $r = 2, 3, \ldots$ such that
$\mathfrak{C}(\mathscr{A}_r(\psi)) \leq n + r$ and $c^{r-1}v(\tau)c$ is reset if ψ is satisfiable on τ,
$\mathfrak{C}(\mathscr{A}_r(\psi)) > r(n-1)$ if ψ is not satisfiable.

$\mathscr{A}_r(\psi)$ is constructed by a "substitution" $\mathscr{A}_{r-1}(\psi)$ instead every state of $\mathscr{A}_2(\psi)$ and some additional modification.

Theorem 1.

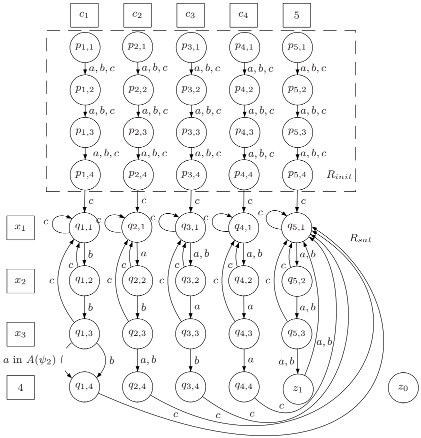
No polynomial-time algorithm approximates the minimum length of reset words within a constant factor.

For every ψ of SAT with n variables we construct synchronizing automaton $\mathscr{A}_r(\psi)$ for $r = 2, 3, \ldots$ such that
$\mathfrak{C}(\mathscr{A}_r(\psi)) \leq n + r$ and $c^{r-1}v(\tau)c$ is reset if ψ is satisfiable on τ,
$\mathfrak{C}(\mathscr{A}_r(\psi)) > r(n-1)$ if ψ is not satisfiable.

$\mathscr{A}_r(\psi)$ is constructed by a "substitution" $\mathscr{A}_{r-1}(\psi)$ instead every state of $\mathscr{A}_2(\psi)$ and some additional modification.

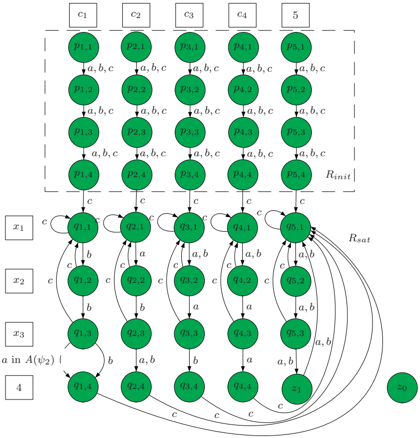
The automata $\mathscr{A}(\psi_1)$ and $\mathscr{A}(\psi_2)$

An applied prefix is



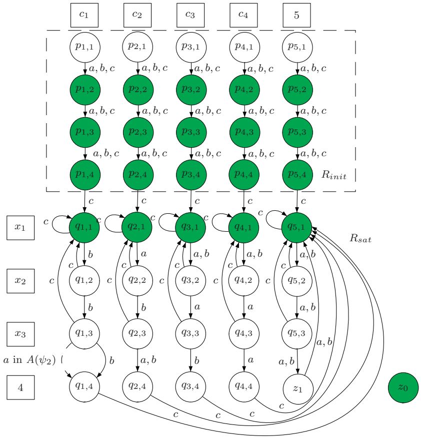
The automata $\mathscr{A}(\psi_1)$ and $\mathscr{A}(\psi_2)$

An applied prefix is



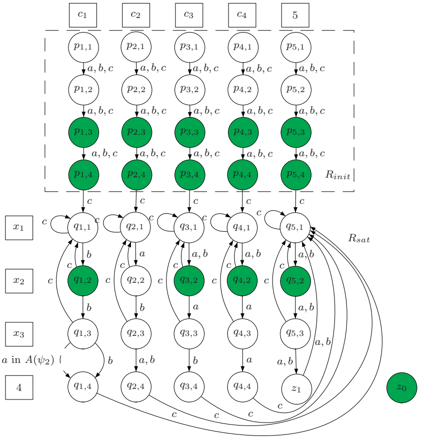
The automata $\mathscr{A}(\psi_1)$ and $\mathscr{A}(\psi_2)$

An applied prefix is c.

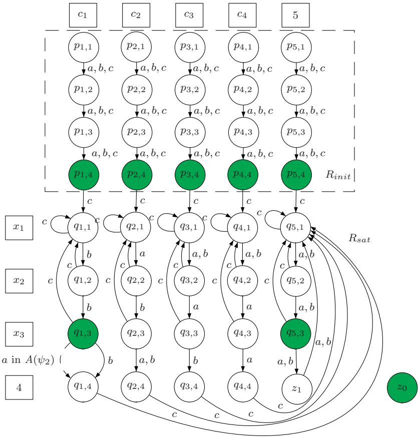


The automata $\mathscr{A}(\psi_1)$ and $\mathscr{A}(\psi_2)$

An applied prefix is *cb*.

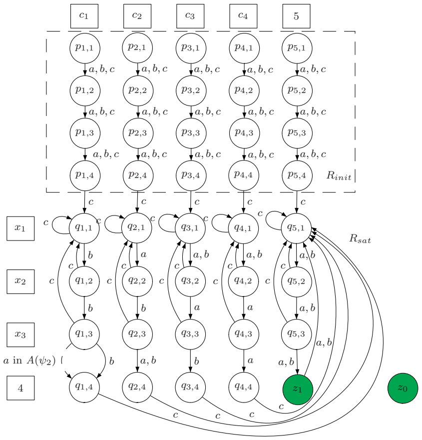


An applied prefix is *cbb*.



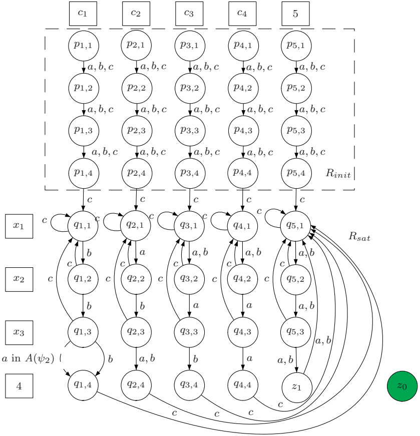
The automata $\mathscr{A}(\psi_1)$ and $\mathscr{A}(\psi_2)$

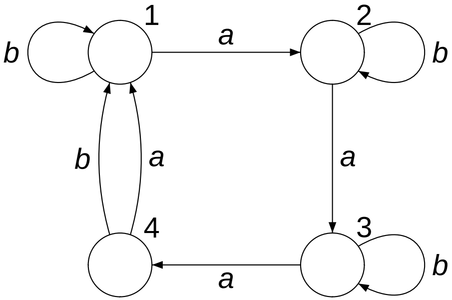
An applied prefix is $c\textcolor{red}{bba} = c\textcolor{red}{v(\tau)}$.



The automata $\mathscr{A}(\psi_1)$ and $\mathscr{A}(\psi_2)$

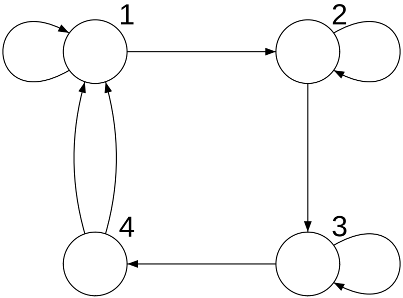
An applied prefix is $c\color{red}{bbac}\color{black} = cv(\color{red}{\tau}\color{black})c$.





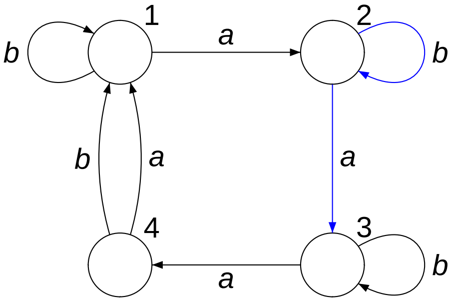
The 4-state Cerny automaton C_4 with shortest reset word ba^3ba^3b of length 9.

Relabeling of Automata and Coloring of Graphs



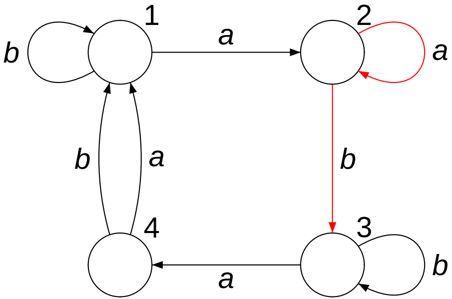
An underlying graph G_4 of the Cerny automaton C_4.

Relabeling of Automata and Coloring of Graphs



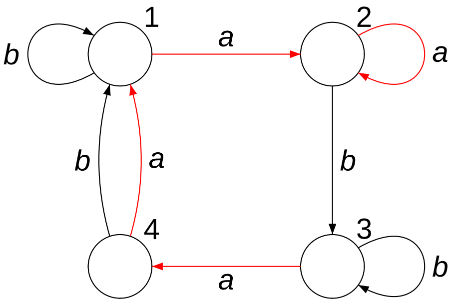
A synchronizing relabeling of C_4 by a permutation of labels on outgoing arrows (from state 2).

Relabeling of Automata and Coloring of Graphs



A synchronizing relabeling of C_4 by a permutation of labels on outgoing arrows (from state 2).

Relabeling of Automata and Coloring of Graphs



A synchronizing coloring of G_4 with shortest reset word a^3 of length 3.

An example from the real life!

Main Questions And Outline of the Talk

Given an automaton $\mathscr{A}$;

- How to find some reset word for $\mathscr{A}$ if it exists?
  Černý in 1964 proved synchronization criterion which allows to find reset word in $O(n^3)$ time.
- How to relabel $\mathscr{A}$ to make it synchronizing?

Given a synchronizing automaton $\mathscr{A}$;

- How to find "relatively" short reset word for $\mathscr{A}$ or its length?
  No polynomial time algorithm approximates reset length of $\mathscr{A}$ within a constant factor (CSR 2010).
- How to find relabeling of $\mathscr{A}$ with "relatively" short reset word or find its length?

Main Questions And Outline of the Talk

Given an automaton $\mathscr{A}$;

- How to find some reset word for $\mathscr{A}$ if it exists?
  Černý in 1964 proved synchronization criterion which allows to find reset word in $O(n^3)$ time.
- How to relabel $\mathscr{A}$ to make it synchronizing?

Given a synchronizing automaton $\mathscr{A}$;

- How to find "relatively" short reset word for $\mathscr{A}$ or its length?
  No polynomial time algorithm approximates reset length of $\mathscr{A}$ within a constant factor (CSR 2010).
- How to find relabeling of $\mathscr{A}$ with "relatively" short reset word or find its length?

Main Questions And Outline of the Talk

Given an automaton $\mathscr{A}$;

- How to find some reset word for $\mathscr{A}$ if it exists?
  Černý in 1964 proved synchronization criterion which allows to find reset word in $O(n^3)$ time.
- How to relabel $\mathscr{A}$ to make it synchronizing?

Given a synchronizing automaton $\mathscr{A}$;

- How to find "relatively" short reset word for $\mathscr{A}$ or its length?
  No polynomial time algorithm approximates reset length of $\mathscr{A}$ within a constant factor (CSR 2010).
- How to find relabeling of $\mathscr{A}$ with "relatively" short reset word or find its length?

Main Questions And Outline of the Talk

Given an automaton $\mathscr{A}$;

- How to find some reset word for $\mathscr{A}$ if it exists?
  Černý in 1964 proved synchronization criterion which allows to find reset word in $O(n^3)$ time.
- How to relabel $\mathscr{A}$ to make it synchronizing?

Given a synchronizing automaton $\mathscr{A}$;

- How to find "relatively" short reset word for $\mathscr{A}$ or its length?
  No polynomial time algorithm approximates reset length of $\mathscr{A}$ within a constant factor (CSR 2010).
- How to find relabeling of $\mathscr{A}$ with "relatively" short reset word or find its length?

Search For Some Synchronizing Coloring

Road Coloring Problem | Adler, Goodwyn, Weiss 1970,77

Does each AGW-graph (strongly connected admissible graph with g.c.d. of cycles length equals one) has a synchronizing coloring?

Particular cases [O'Brien, 1981; Fridman, 1990; Perrin and Schützenberger, 1985; Jonoska N., Suen S., 1995, Carbone A., 2001, J. Kari 2003...]

RCP Solution! | A. Trahtman 2008

Each AGW-graph has a synchronizing coloring.

This result allows to find some synchronizing coloring in $O(n^3)$-time and leads to $O(n^2)$-time algorithm invented by Beal and Perrin in 2008

Search For Some Synchronizing Coloring

Road Coloring Problem | Adler, Goodwyn, Weiss 1970,77

Does each AGW-graph (strongly connected admissible graph with g.c.d. of cycles length equals one) has a synchronizing coloring?

Particular cases [O'Brien, 1981; Fridman, 1990; Perrin and Schützenberger, 1985; Jonoska N., Suen S., 1995, Carbone A., 2001, J. Kari 2003...]

RCP Solution! | A. Trahtman 2008

Each AGW-graph has a synchronizing coloring.

This result allows to find some synchronizing coloring in $O(n^3)$-time and leads to $O(n^2)$-time algorithm invented by Beal and Perrin in 2008

Search For Some Synchronizing Coloring

Road Coloring Problem | Adler, Goodwyn, Weiss 1970,77

Does each AGW-graph (strongly connected admissible graph with g.c.d. of cycles length equals one) has a synchronizing coloring?

Particular cases [O'Brien, 1981; Fridman, 1990; Perrin and Schützenberger, 1985; Jonoska N., Suen S., 1995, Carbone A., 2001, J. Kari 2003...]

RCP Solution! | A. Trahtman 2008

Each AGW-graph has a synchronizing coloring.

This result allows to find some synchronizing coloring in $O(n^3)$-time and leads to $O(n^2)$-time algorithm invented by Beal and Perrin in 2008

Main Questions And Outline of the Talk

Given an automaton $\mathscr{A}$;

- How to find some reset word for $\mathscr{A}$ if it exists?
  Černý in 1964 proved synchronization criterion which allows to find reset word in $O(n^3)$ time.
- How to relabel $\mathscr{A}$ to make it synchronizing?
  Trahtman in 2008 proved a criterion which allows to find such relabeling in $O(n^3)$ time.

Given a synchronizing automaton $\mathscr{A}$;

- How to find "relatively" short reset word for $\mathscr{A}$ or its length?
  No polynomial time algorithm approximates reset length of $\mathscr{A}$ within a constant factor (CSR 2010).
- How to find relabeling of $\mathscr{A}$ with "relatively" short reset word or find its length?
  No polynomial time algorithm approximates optimal coloring [value] within factor 2.

Approximate Optimal Coloring Value

Let $OPT(G)$ denotes the minimal value of $\mathfrak{C}(\mathscr{A}(G))$ for possible colorings $\mathscr{A}(G)$ of AGW-graph G and let us call it optimal coloring value.

A coloring $\mathscr{B}(G)$ with $\mathfrak{C}(\mathscr{B}(G)) = OPT(G)$ is called optimal.

Opt-Coloring-Value

Given An AGW-graph G;
Return $OPT(G)$.

Key Question | Volkov 2008

Can we approximately find the optimal coloring [value] within a constant factor in a polynomial time?

Is there a polynomial-time [approximation] algorithm within a constant factor for Opt-Coloring-Value?

Approximate Optimal Coloring Value

Let $OPT(G)$ denotes the minimal value of $\mathfrak{C}(\mathscr{A}(G))$ for possible colorings $\mathscr{A}(G)$ of AGW-graph G and let us call it optimal coloring value.
A coloring $\mathscr{B}(G)$ with $\mathfrak{C}(\mathscr{B}(G)) = OPT(G)$ is called optimal.

Opt-Coloring-Value

Given An AGW-graph G;
Return $OPT(G)$.

Key Question | Volkov 2008

Can we approximately find the optimal coloring [value] within a constant factor in a polynomial time?

Is there a polynomial-time [approximation] algorithm within a constant factor for Opt-Coloring-Value?

Approximate Optimal Coloring Value

Let $OPT(G)$ denotes the minimal value of $\mathfrak{C}(\mathscr{A}(G))$ for possible colorings $\mathscr{A}(G)$ of AGW-graph G and let us call it optimal coloring value.

A coloring $\mathscr{B}(G)$ with $\mathfrak{C}(\mathscr{B}(G)) = OPT(G)$ is called optimal.

Opt-Coloring-Value

Given An AGW-graph G;

Return $OPT(G)$.

Key Question | Volkov 2008

Can we approximately find the optimal coloring [value] within a constant factor in a polynomial time?

Is there a polynomial-time [approximation] algorithm within a constant factor for Opt-Coloring-Value?

Approximate Optimal Coloring Value

Let $OPT(G)$ denotes the minimal value of $\mathfrak{C}(\mathscr{A}(G))$ for possible colorings $\mathscr{A}(G)$ of AGW-graph G and let us call it optimal coloring value.

A coloring $\mathscr{B}(G)$ with $\mathfrak{C}(\mathscr{B}(G)) = OPT(G)$ is called optimal.

Opt-Coloring-Value

Given An AGW-graph G;
Return $OPT(G)$.

Key Question | Volkov 2008

Can we approximately find the optimal coloring [value] within a constant factor in a polynomial time?

Is there a polynomial-time [approximation] algorithm within a constant factor for Opt-Coloring-Value?

The Second Result

Theorem | Roman 2010

No polynomial-time algorithm exactly finds optimal coloring value.

Theorem 2. | Izvestiya Vuzov (submitted 07.2009)

No polynomial-time algorithm approximates optimal coloring value within a constant factor less than 2.

Can we approximately find the optimal coloring value within a constant factor 2 in a polynomial time?

Proof sketch:
For each ψ of SAT with n variables we construct $G(\psi)$ such that
$OPT(G(\psi)) \leq p(m, n)$ if ψ is satisfiable, (call *GOODCASE*)
$OPT(G(\psi)) \geq (2 - 0.5\varepsilon)p(m, n)$ if ψ is not satisfiable (call *BADCASE*).

The Second Result

Theorem | Roman 2010

No polynomial-time algorithm exactly finds optimal coloring value.

Theorem 2. | Izvestiya Vuzov (submitted 07.2009)

No polynomial-time algorithm approximates optimal coloring value within a constant factor less than 2.

Can we approximately find the optimal coloring value within a constant factor 2 in a polynomial time?

Proof sketch:
For each ψ of SAT with n variables we construct $G(\psi)$ such that
$OPT(G(\psi)) \leq p(m, n)$ if ψ is satisfiable, (call *GOODCASE*)
$OPT(G(\psi)) \geq (2 - 0.5\varepsilon)p(m, n)$ if ψ is not satisfiable (call *BADCASE*).

Theorem | Roman 2010

No polynomial-time algorithm exactly finds optimal coloring value.

Theorem 2. | Izvestiya Vuzov (submitted 07.2009)

No polynomial-time algorithm approximates optimal coloring value within a constant factor less than 2.

Can we approximately find the optimal coloring value within a constant factor 2 in a polynomial time?

Proof sketch:
For each ψ of SAT with n variables we construct $G(\psi)$ such that
$OPT(G(\psi)) \leq p(m, n)$ if ψ is satisfiable, (call *GOODCASE*)
$OPT(G(\psi)) \geq (2 - 0.5\varepsilon)p(m, n)$ if ψ is not satisfiable (call *BADCASE*).

The Second Result

Theorem | Roman 2010

No polynomial-time algorithm exactly finds optimal coloring value.

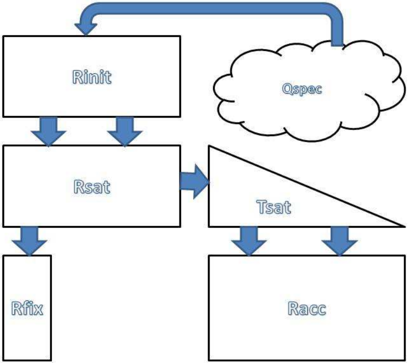
Theorem 2. | Izvestiya Vuzov (submitted 07.2009)

No polynomial-time algorithm approximates optimal coloring value within a constant factor less than 2.

Can we approximately find the optimal coloring value within a constant factor 2 in a polynomial time?

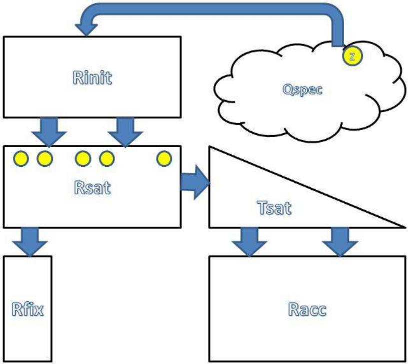
Proof sketch:
For each ψ of SAT with n variables we construct $G(\psi)$ such that
$OPT(G(\psi)) \leq p(m,n)$ if ψ is satisfiable, (call *GOODCASE*)
$OPT(G(\psi)) \geq (2-0.5\varepsilon)p(m,n)$ if ψ is not satisfiable (call *BADCASE*).

Synchronizing Coloring in GOODCASE



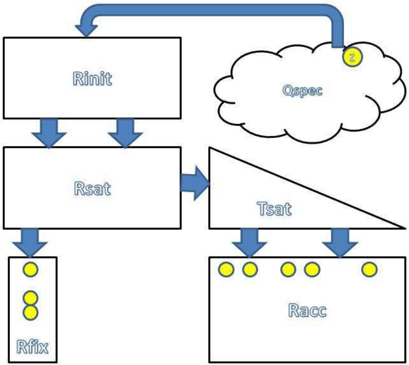
Construction of The Graph $G(\psi)$

Synchronizing Coloring in GOODCASE



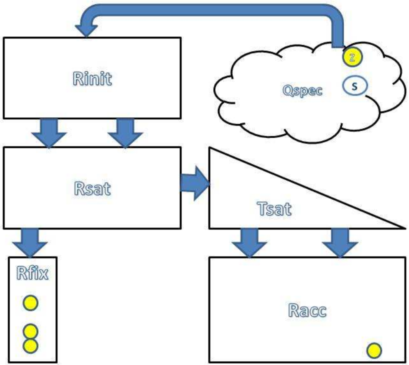
$Q.u_{init}$ equals the first row of R_{sat} and state z. The length of u_{init} is $p - p_{small}$.

Synchronizing Coloring in GOODCASE



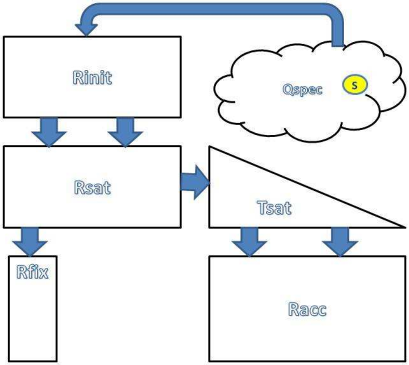
Define $u_{sat} = v(\tau)$. Then $Q.u_{init}u_{sat}$ consists of n states in the first row of R_{acc}, 3 states in R_{fix} with numbers $1, 3, 4$ and state z. The length of u_{sat} is $n + 1$.

Synchronizing Coloring in GOODCASE



$Q.u_{init}u_{sat}u'_{acc}$ consists of bottom state in R_{acc} and 3 states in R_{fix} with numbers $h-3, h-1, h$ and state z. The length of u'_{acc} is $p_{small}-(n+2)=h$.

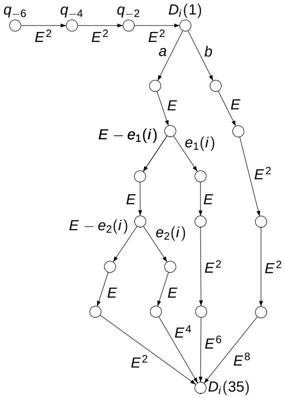
Synchronizing Coloring in GOODCASE



$Q.u_{init}u_{sat}u_{acc} = s$

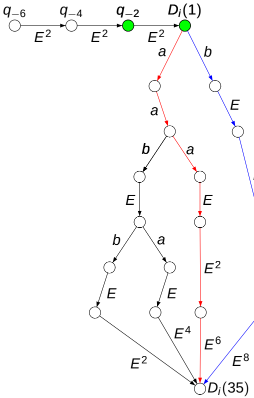
Counting States in The First Row of R_{acc}

Let $E = \{a, b\}$ and apply a word $\ldots ba^{13} \ldots$ to the green states.



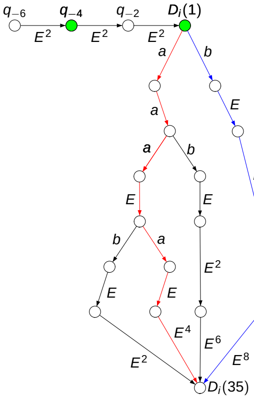
Counting States in The First Row of R_{acc}

Let $E = \{a, b\}$ and apply a word $\ldots ba^{13} \ldots$ to the green states.



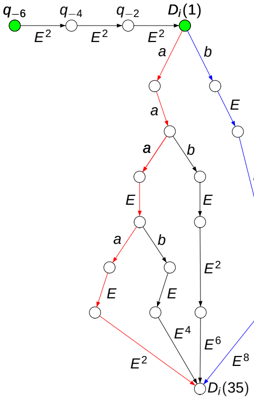
Counting States in The First Row of R_{acc}

Let $E = \{a, b\}$ and apply a word $\ldots ba^{13} \ldots$ to the green states.



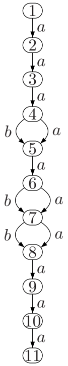
Counting States in The First Row of R_{acc}

Let $E = \{a, b\}$ and apply a word $\ldots ba^{13} \ldots$ to the green states.



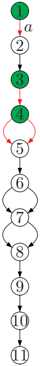
Fixing The Word $w_n(\dots)$ Using Component R_{fix}

Synchronizing Coloring of R_{fix} in GOODCASE.



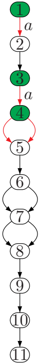
Fixing The Word $w_n(\dots)$ Using Component R_{fix}

A fixed word $w_n = 1\dots$



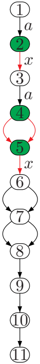
Fixing The Word $w_n(\dots)$ Using Component R_{fix}

A fixed word $w_n = a \dots$



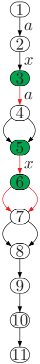
Fixing The Word $w_n(\dots)$ Using Component R_{fix}

A fixed word $w_n = ax\dots$



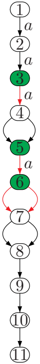
Fixing The Word $w_n(\ldots)$ Using Component R_{fix}

A fixed word $w_n = ax \ldots$



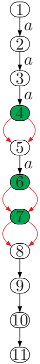
Fixing The Word $w_n(\dots)$ Using Component R_{fix}

A fixed word $w_n = aa\dots$



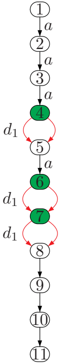
Fixing The Word $w_n(\ldots)$ Using Component R_{fix}

A fixed word $w_n = aa\ldots$



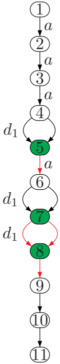
Fixing The Word $w_n(\ldots)$ Using Component R_{fix}

A fixed word $w_n = aad_1 \ldots$



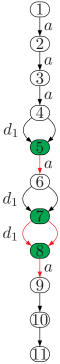
Fixing The Word $w_n(\dots)$ Using Component R_{fix}

A fixed word $w_n = aad_1 \dots$



Fixing The Word $w_n(\dots)$ Using Component R_{fix}

A fixed word $w_n = aad_1a\dots$



Approximate Optimal Coloring

Opt-Coloring

Given An AGW-graph G;
Return Optimal Coloring of G.

Key Question

Can we approximately find the optimal coloring within a constant factor in a polynomial time?

Remark 1: It doesn't follow from Theorem 2, because we should not find $\mathfrak{C}(\mathscr{A}(G))$ for quasi-optimal coloring $\mathscr{A}(G)$.

Remark 2: If we could approximate $\mathfrak{C}(\mathscr{A})$ in a polynomial time then we could make such conclusion. But it is false in view of our first result.

Approximate Optimal Coloring

Opt-Coloring

Given An AGW-graph G;
Return Optimal Coloring of G.

Key Question

Can we approximately find the optimal coloring within a constant factor in a polynomial time?

Remark 1: It doesn't follow from Theorem 2, because we should not find $\mathfrak{C}(\mathscr{A}(G))$ for quasi-optimal coloring $\mathscr{A}(G)$.

Remark 2: If we could approximate $\mathfrak{C}(\mathscr{A})$ in a polynomial time then we could make such conclusion. But it is false in view of our first result.

Approximate Optimal Coloring

Opt-Coloring

Given An AGW-graph G;
Return Optimal Coloring of G.

Key Question

Can we approximately find the optimal coloring within a constant factor in a polynomial time?

Remark 1: It doesn't follow from Theorem 2, because we should not find $\mathfrak{C}(\mathscr{A}(G))$ for quasi-optimal coloring $\mathscr{A}(G)$.

Remark 2: If we could approximate $\mathfrak{C}(\mathscr{A})$ in a polynomial time then we could make such conclusion. But it is false in view of our first result.

Approximate Optimal Coloring

Opt-Coloring

Given An AGW-graph G;
Return Optimal Coloring of G.

Key Question

Can we approximately find the optimal coloring within a constant factor in a polynomial time?

Remark 1: It doesn't follow from Theorem 2, because we should not find $\mathfrak{C}(\mathscr{A}(G))$ for quasi-optimal coloring $\mathscr{A}(G)$.

Remark 2: If we could approximate $\mathfrak{C}(\mathscr{A})$ in a polynomial time then we could make such conclusion. But it is false in view of our first result.

Corollary For Searching Optimal Coloring

Corollary 1.

No polynomial-time algorithm approximates optimal coloring within a constant factor less than 2.

It is sufficient to show how to determine in a polynomial time satisfiability of ψ by coloring of $G(\psi)$ from Theorem 2.

Proof Sketch:

Corollary For Searching Optimal Coloring

Corollary 1.

No polynomial-time algorithm approximates optimal coloring within a constant factor less than 2.

It is sufficient to show how to determine in a polynomial time satisfiability of ψ by coloring of $G(\psi)$ from Theorem 2.

Proof Sketch:

Corollary For Searching Optimal Coloring

Corollary 1.

No polynomial-time algorithm approximates optimal coloring within a constant factor less than 2.

It is sufficient to show how to determine in a polynomial time satisfiability of ψ by coloring of $G(\psi)$ from Theorem 2.
Proof Sketch:

- Suppose $R_{fix}(1).a = R_{fix}(2)$.
-
-
-

Corollary For Searching Optimal Coloring

Corollary 1.

No polynomial-time algorithm approximates optimal coloring within a constant factor less than 2.

It is sufficient to show how to determine in a polynomial time satisfiability of ψ by coloring of $G(\psi)$ from Theorem 2.
Proof Sketch:

- Suppose $R_{fix}(1).a = R_{fix}(2)$.
- For $i \in [1, n-1]$ calculate path lengths in D_i marked by degree of a and d_i as a right label from $D_i(1)$.
-
-

Corollary 1.

No polynomial-time algorithm approximates optimal coloring within a constant factor less than 2.

It is sufficient to show how to determine in a polynomial time satisfiability of ψ by coloring of $G(\psi)$ from Theorem 2.
Proof Sketch:

- Suppose $R_{fix}(1).a = R_{fix}(2)$.
- For $i \in [1, n-1]$ calculate path lengths in D_i marked by degree of a and d_i as a right label from $D_i(1)$.
- Renewal set of states S from the first row of Row_{acc} which can be merged by the word $w_n(d_1, d_2, \dots, d_n)$.
-

Corollary For Searching Optimal Coloring

Corollary 1.

No polynomial-time algorithm approximates optimal coloring within a constant factor less than 2.

It is sufficient to show how to determine in a polynomial time satisfiability of ψ by coloring of $G(\psi)$ from Theorem 2.
Proof Sketch:

- Suppose $R_{fix}(1).a = R_{fix}(2)$.
- For $i \in [1, n-1]$ calculate path lengths in D_i marked by degree of a and d_i as a right label from $D_i(1)$.
- Renewal set of states S from the first row of Row_{acc} which can be merged by the word $w_n(d_1, d_2, \ldots, d_n)$.
- Renewal variable values $b_1, b_2, \ldots, b_n$ according to S.

Corollary For Searching Optimal Coloring

Corollary 1.

No polynomial-time algorithm approximates optimal coloring within a constant factor less than 2.

It is sufficient to show how to determine in a polynomial time satisfiability of ψ by coloring of $G(\psi)$ from Theorem 2.
Proof Sketch:

- If $\psi(b_1, b_2, \ldots, b_n)$ is true then ψ is satisfiable.
-
-
-

Corollary For Searching Optimal Coloring

Corollary 1.

No polynomial-time algorithm approximates optimal coloring within a constant factor less than 2.

It is sufficient to show how to determine in a polynomial time satisfiability of ψ by coloring of $G(\psi)$ from Theorem 2.
Proof Sketch:

- If $\psi(b_1, b_2, \ldots, b_n)$ is true then ψ is satisfiable.
- In opposite case, it is sufficient to prove $\mathfrak{C}(B) \geq (2 - 0.5\varepsilon)p$.
-
-

Corollary For Searching Optimal Coloring

Corollary 1.

No polynomial-time algorithm approximates optimal coloring within a constant factor less than 2.

It is sufficient to show how to determine in a polynomial time satisfiability of ψ by coloring of $G(\psi)$ from Theorem 2.
Proof Sketch:

- If $\psi(b_1, b_2, \ldots, b_n)$ is true then ψ is satisfiable.
- In opposite case, it is sufficient to prove $\mathfrak{C}(B) \geq (2 - 0.5\varepsilon)p$.
- Renewal set of states S from the first row of Row_{acc} which can be merged by the word $w_n(d_1, d_2, \ldots, d_n)$.
-

Corollary For Searching Optimal Coloring

Corollary 1.

No polynomial-time algorithm approximates optimal coloring within a constant factor less than 2.

It is sufficient to show how to determine in a polynomial time satisfiability of ψ by coloring of $G(\psi)$ from Theorem 2.
Proof Sketch:

- If $\psi(b_1, b_2, \ldots, b_n)$ is true then ψ is satisfiable.
- In opposite case, it is sufficient to prove $\mathfrak{C}(B) \geq (2 - 0.5\varepsilon)p$.
- Renewal set of states S from the first row of Row_{acc} which can be merged by the word $w_n(d_1, d_2, \ldots, d_n)$.
- It can be done as in the BADCASE using that ψ should be true for the collection $b_1, b_2, \ldots, b_n$.

2-letter alphabet case

- We used a 3-letter alphabets in the Theorems.
- By adding letters with action of letter c, the results extend to any class of automata with bigger alphabet.
- All considered problems are trivial for 1-letter automata.

The results can be extended to the case of 2-letter alphabet.

For each 3-letter automaton $\mathscr{A}$ we can construct a 2-letter automaton $\mathscr{B}$ such that $\mathfrak{C}(\mathscr{A}) \leq \mathfrak{C}(\mathscr{B}) \leq 3\mathfrak{C}(\mathscr{A})$.

For each graph $G(\psi)$ in Theorem 2 we can construct a 2-letter graph $G^2(\psi)$ such that $2p(m,n) \leq OPT(G^2(\psi)) \leq (4-\varepsilon)p(m,n)$.

2-letter alphabet case

- We used a 3-letter alphabets in the Theorems.
- By adding letters with action of letter c, the results extend to any class of automata with bigger alphabet.
- All considered problems are trivial for 1-letter automata.

The results can be extended to the case of 2-letter alphabet.

For each 3-letter automaton $\mathscr{A}$ we can construct a 2-letter automaton $\mathscr{B}$ such that $\mathfrak{C}(\mathscr{A}) \leq \mathfrak{C}(\mathscr{B}) \leq 3\mathfrak{C}(\mathscr{A})$.

For each graph $G(\psi)$ in Theorem 2 we can construct a 2-letter graph $G^2(\psi)$ such that $2p(m,n) \leq OPT(G^2(\psi)) \leq (4-\varepsilon)p(m,n)$.

2-letter alphabet case

- We used a 3-letter alphabets in the Theorems.
- By adding letters with action of letter c, the results extend to any class of automata with bigger alphabet.
- All considered problems are trivial for 1-letter automata.

The results can be extended to the case of 2-letter alphabet.

For each 3-letter automaton $\mathscr{A}$ we can construct a 2-letter automaton $\mathscr{B}$ such that $\mathfrak{C}(\mathscr{A}) \leq \mathfrak{C}(\mathscr{B}) \leq 3\mathfrak{C}(\mathscr{A})$.

For each graph $G(\psi)$ in Theorem 2 we can construct a 2-letter graph $G^2(\psi)$ such that $2p(m,n) \leq OPT(G^2(\psi)) \leq (4-\varepsilon)p(m,n)$.

2-letter alphabet case

- We used a 3-letter alphabets in the Theorems.
- By adding letters with action of letter c, the results extend to any class of automata with bigger alphabet.
- All considered problems are trivial for 1-letter automata.

The results can be extended to the case of 2-letter alphabet.

For each 3-letter automaton $\mathscr{A}$ we can construct a 2-letter automaton $\mathscr{B}$ such that $\mathfrak{C}(\mathscr{A}) \leq \mathfrak{C}(\mathscr{B}) \leq 3\mathfrak{C}(\mathscr{A})$.

For each graph $G(\psi)$ in Theorem 2 we can construct a 2-letter graph $G^2(\psi)$ such that $2p(m,n) \leq OPT(G^2(\psi)) \leq (4-\varepsilon)p(m,n)$.

2-letter alphabet case

- We used a 3-letter alphabets in the Theorems.
- By adding letters with action of letter c, the results extend to any class of automata with bigger alphabet.
- All considered problems are trivial for 1-letter automata.

The results can be extended to the case of 2-letter alphabet.

For each 3-letter automaton $\mathscr{A}$ we can construct a 2-letter automaton $\mathscr{B}$ such that $\mathfrak{C}(\mathscr{A}) \leq \mathfrak{C}(\mathscr{B}) \leq 3\mathfrak{C}(\mathscr{A})$.

For each graph $G(\psi)$ in Theorem 2 we can construct a 2-letter graph $G^2(\psi)$ such that $2p(m,n) \leq OPT(G^2(\psi)) \leq (4-\varepsilon)p(m,n)$.

2-letter alphabet case

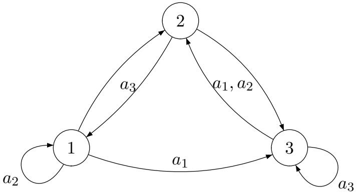
- We used a 3-letter alphabets in the Theorems.
- By adding letters with action of letter c, the results extend to any class of automata with bigger alphabet.
- All considered problems are trivial for 1-letter automata.

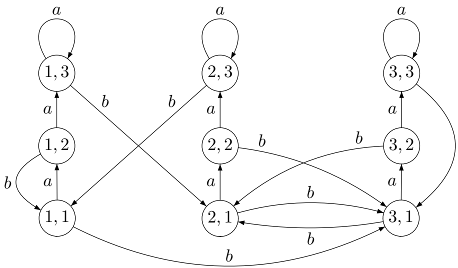
The results can be extended to the case of 2-letter alphabet.

For each 3-letter automaton $\mathscr{A}$ we can construct a 2-letter automaton $\mathscr{B}$ such that $\mathfrak{C}(\mathscr{A}) \leq \mathfrak{C}(\mathscr{B}) \leq 3\mathfrak{C}(\mathscr{A})$.

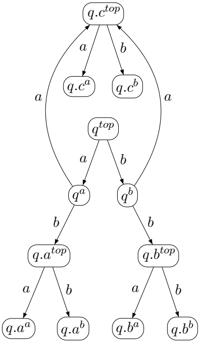
For each graph $G(\psi)$ in Theorem 2 we can construct a 2-letter graph $G^2(\psi)$ such that $2p(m,n) \leq OPT(G^2(\psi)) \leq (4-\varepsilon)p(m,n)$.

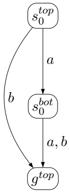
An Automata Transformation in Theorem 1.





Substitutions in Graph $G(\psi)$ in Theorem 2.





- A DFA $\mathscr{A}$ is called an automaton with 0 if it has one immoveable state called 0.
- A **partial finite automaton** (PFA) can have some undefined transitions in difference to DFA.
- A **killing** word for PFA is a word undefined for each state.

Any PFA $\mathscr{B}$ is a result of removing all incoming aroows to 0 for an appropriate DFA $\mathscr{A}$ with 0 and each killing word for $\mathscr{B}$ is reset for $\mathscr{A}$.

Corollary (The Mortality Problem)

No polynomial-time algorithm can approximate the length of the shortest killing word within a constant factor.

The Mortality Problem

- A DFA $\mathscr{A}$ is called an automaton with 0 if it has one immoveable state called 0.
- A **partial finite automaton** (PFA) can have some undefined transitions in difference to DFA.
- A **killing** word for PFA is a word undefined for each state.

Any PFA $\mathscr{B}$ is a result of removing all incoming aroows to 0 for an appropriate DFA $\mathscr{A}$ with 0 and each killing word for $\mathscr{B}$ is reset for $\mathscr{A}$.

Corollary (The Mortality Problem)

No polynomial-time algorithm can approximate the length of the shortest killing word within a constant factor.

The Mortality Problem

- A DFA $\mathscr{A}$ is called an automaton with 0 if it has one immoveable state called 0.
- A **partial finite automaton** (PFA) can have some undefined transitions in difference to DFA.
- A **killing** word for PFA is a word undefined for each state.

Any PFA $\mathscr{B}$ is a result of removing all incoming aroows to 0 for an appropriate DFA $\mathscr{A}$ with 0 and each killing word for $\mathscr{B}$ is reset for $\mathscr{A}$.

Corollary (The Mortality Problem)

No polynomial-time algorithm can approximate the length of the shortest killing word within a constant factor.

- A DFA $\mathscr{A}$ is called an automaton with 0 if it has one immoveable state called 0.
- A **partial finite automaton** (PFA) can have some undefined transitions in difference to DFA.
- A **killing** word for PFA is a word undefined for each state.

Any PFA $\mathscr{B}$ is a result of removing all incoming aroows to 0 for an appropriate DFA $\mathscr{A}$ with 0 and each killing word for $\mathscr{B}$ is reset for $\mathscr{A}$.

Corollary (The Mortality Problem)

No polynomial-time algorithm can approximate the length of the shortest killing word within a constant factor.

The Mortality Problem

- A DFA $\mathscr{A}$ is called an automaton with 0 if it has one immoveable state called 0.
- A **partial finite automaton** (PFA) can have some undefined transitions in difference to DFA.
- A **killing** word for PFA is a word undefined for each state.

Any PFA $\mathscr{B}$ is a result of removing all incoming aroows to 0 for an appropriate DFA $\mathscr{A}$ with 0 and each killing word for $\mathscr{B}$ is reset for $\mathscr{A}$.

Corollary (The Mortality Problem)

No polynomial-time algorithm can approximate the length of the shortest killing word within a constant factor.

The Mortality Problem

- A DFA $\mathscr{A}$ is called an automaton with 0 if it has one immoveable state called 0.
- A **partial finite automaton** (PFA) can have some undefined transitions in difference to DFA.
- A **killing** word for PFA is a word undefined for each state.

Any PFA $\mathscr{B}$ is a result of removing all incoming aroows to 0 for an appropriate DFA $\mathscr{A}$ with 0 and each killing word for $\mathscr{B}$ is reset for $\mathscr{A}$.

Corollary (The Mortality Problem)

No polynomial-time algorithm can approximate the length of the shortest killing word within a constant factor.

Logarithmic Approximation

- The **greedy** algorithm (Eppstein 1990) finds a reset word for n-state automata in $O(n^3)$ time.
- It finds a reset word of length at most $\frac{n^3-n}{6}$ (Pin 1983).
- All experiments with series of slowly synchronized automata generated in our scientific group show it has a logarithmic approximation factor.

Search-LogApprox-Reset-Length(d)

Given An n-state synchronizing automaton $\mathscr{A}$

Return A number between $\mathfrak{C}(\mathscr{A})$ and $d \cdot \log n \cdot \mathfrak{C}(\mathscr{A})$.

Is there a polynomial-time algorithm for the above problem?

Logarithmic Approximation

- The **greedy** algorithm (Eppstein 1990) finds a reset word for n-state automata in $O(n^3)$ time.
- It finds a reset word of length at most $\frac{n^3-n}{6}$ (Pin 1983).
- All experiments with series of slowly synchronized automata generated in our scientific group show it has a logarithmic approximation factor.

Search-LogApprox-Reset-Length(d)

Given An n-state synchronizing automaton $\mathscr{A}$
Return A number between $\mathfrak{C}(\mathscr{A})$ and $d \cdot \log n \cdot \mathfrak{C}(\mathscr{A})$.

Is there a polynomial-time algorithm for the above problem?

Logarithmic Approximation

- The **greedy** algorithm (Eppstein 1990) finds a reset word for n-state automata in $O(n^3)$ time.
- It finds a reset word of length at most $\frac{n^3-n}{6}$ (Pin 1983).
- All experiments with series of slowly synchronized automata generated in our scientific group show it has a logarithmic approximation factor.

Search-LogApprox-Reset-Length(d)

Given An n-state synchronizing automaton $\mathscr{A}$

Return A number between $\mathfrak{C}(\mathscr{A})$ and $d \cdot \log n \cdot \mathfrak{C}(\mathscr{A})$.

Is there a polynomial-time algorithm for the above problem?

Logarithmic Approximation

- The **greedy** algorithm (Eppstein 1990) finds a reset word for n-state automata in $O(n^3)$ time.
- It finds a reset word of length at most $\frac{n^3-n}{6}$ (Pin 1983).
- All experiments with series of slowly synchronized automata generated in our scientific group show it has a logarithmic approximation factor.

Search-LogApprox-Reset-Length(d)

Given An n-state synchronizing automaton $\mathscr{A}$

Return A number between $\mathfrak{C}(\mathscr{A})$ and $d \cdot \log n \cdot \mathfrak{C}(\mathscr{A})$.

Is there a polynomial-time algorithm for the above problem?

Logarithmic Approximation

- The **greedy** algorithm (Eppstein 1990) finds a reset word for n-state automata in $O(n^3)$ time.
- It finds a reset word of length at most $\frac{n^3-n}{6}$ (Pin 1983).
- All experiments with series of slowly synchronized automata generated in our scientific group show it has a logarithmic approximation factor.

Search-LogApprox-Reset-Length(d)

Given An n-state synchronizing automaton $\mathscr{A}$

Return A number between $\mathfrak{C}(\mathscr{A})$ and $d \cdot \log n \cdot \mathfrak{C}(\mathscr{A})$.

Is there a polynomial-time algorithm for the above problem?

Thank you for your attention!

Any questions?