Various Aspects of Automaton Synchronization

Mikhail V. Berlinkov,
Institute of Mathematics and Computer Science,
Ural Federal University (Ekaterinburg, Russia),
berlm@mail.ru

Paris, 2015

Deterministic Finite Automata and their Graphs

By deterministic finite automaton (DFA) $\mathscr{A}$ we mean $\langle Q, \Sigma \rangle$, where Q is the state set and Σ is the alphabet; each $a \in \Sigma$ is a mapping from Q to Q.

- The underlying graph of each letter $a \in \Sigma$ defined as $UG(a) = (Q, \{(p, p.a) \mid p \in Q\})$ consists of one or more connected components called clusters.
- The underlying graph of $\mathscr{A}$ is the edge union of the underlying graphs of its letters.
- Automata are usually classified by their underlying graphs. Examples: circular, one-cluster, Eulerian, etc.

Deterministic Finite Automata and their Graphs

By deterministic finite automaton (DFA) $\mathscr{A}$ we mean $\langle Q, \Sigma \rangle$, where Q is the state set and Σ is the alphabet; each $a \in \Sigma$ is a mapping from Q to Q.

- The underlying graph of each letter $a \in \Sigma$ defined as $UG(a) = (Q, \{(p, p.a) \mid p \in Q\})$ consists of one or more connected components called clusters.
- The underlying graph of $\mathscr{A}$ is the edge union of the underlying graphs of its letters.
- Automata are usually classified by their underlying graphs. Examples: circular, one-cluster, Eulerian, etc.

Deterministic Finite Automata and their Graphs

By deterministic finite automaton (DFA) $\mathscr{A}$ we mean $\langle Q, \Sigma \rangle$, where Q is the state set and Σ is the alphabet; each $a \in \Sigma$ is a mapping from Q to Q.

- The underlying graph of each letter $a \in \Sigma$ defined as $UG(a) = (Q, \{(p, p.a) \mid p \in Q\})$ consists of one or more connected components called clusters.
- The underlying graph of $\mathscr{A}$ is the edge union of the underlying graphs of its letters.
- Automata are usually classified by their underlying graphs. Examples: circular, one-cluster, Eulerian, etc.

Deterministic Finite Automata and their Graphs

By deterministic finite automaton (DFA) $\mathscr{A}$ we mean $\langle Q, \Sigma \rangle$, where Q is the state set and Σ is the alphabet; each $a \in \Sigma$ is a mapping from Q to Q.

- The underlying graph of each letter $a \in \Sigma$ defined as $UG(a) = (Q, \{(p, p.a) \mid p \in Q\})$ consists of one or more connected components called clusters.
- The underlying graph of $\mathscr{A}$ is the edge union of the underlying graphs of its letters.
- Automata are usually classified by their underlying graphs. Examples: circular, one-cluster, Eulerian, etc.

Synchronizing Automata

- The set of words Σ^* corresponds to the transformation monoid.
- A word v is reset for $\mathscr{A}$ if it is a constant mapping, that is, $q.v = p.v$ for each $p, q \in Q$. In other words, each path labeled by v leads to a particular state.
- $\mathscr{A}$ is called synchronizing if it possesses a reset word.
- The minimum length of reset words for $\mathscr{A}$ is called its reset threshold.

Applications: coding theory, data transmission, robotics, software verification, dna-computing, symbolic dynamics, etc.

Synchronizing Automata

- The set of words Σ^* corresponds to the transformation monoid.
- A word v is reset for $\mathscr{A}$ if it is a constant mapping, that is, $q.v = p.v$ for each $p, q \in Q$. In other words, each path labeled by v leads to a particular state.
- $\mathscr{A}$ is called synchronizing if it possesses a reset word.
- The minimum length of reset words for $\mathscr{A}$ is called its reset threshold.

Applications: coding theory, data transmission, robotics, software verification, dna-computing, symbolic dynamics, etc.

Synchronizing Automata

- The set of words Σ^* corresponds to the transformation monoid.
- A word v is reset for $\mathscr{A}$ if it is a constant mapping, that is, $q.v = p.v$ for each $p, q \in Q$. In other words, each path labeled by v leads to a particular state.
- $\mathscr{A}$ is called synchronizing if it possesses a reset word.
- The minimum length of reset words for $\mathscr{A}$ is called its reset threshold.

Applications: coding theory, data transmission, robotics, software verification, dna-computing, symbolic dynamics, etc.

Synchronizing Automata

- The set of words Σ^* corresponds to the transformation monoid.
- A word v is reset for $\mathscr{A}$ if it is a constant mapping, that is, $q.v = p.v$ for each $p, q \in Q$. In other words, each path labeled by v leads to a particular state.
- $\mathscr{A}$ is called synchronizing if it possesses a reset word.
- The minimum length of reset words for $\mathscr{A}$ is called its reset threshold.

Applications: coding theory, data transmission, robotics, software verification, dna-computing, symbolic dynamics, etc.

Synchronizing Automata

- The set of words Σ^* corresponds to the transformation monoid.
- A word v is reset for $\mathscr{A}$ if it is a constant mapping, that is, $q.v = p.v$ for each $p, q \in Q$. In other words, each path labeled by v leads to a particular state.
- $\mathscr{A}$ is called synchronizing if it possesses a reset word.
- The minimum length of reset words for $\mathscr{A}$ is called its reset threshold.

Applications: coding theory, data transmission, robotics, software verification, dna-computing, symbolic dynamics, etc.

Synchronizing Automata

- The set of words Σ^* corresponds to the transformation monoid.
- A word v is reset for $\mathscr{A}$ if it is a constant mapping, that is, $q.v = p.v$ for each $p, q \in Q$. In other words, each path labeled by v leads to a particular state.
- $\mathscr{A}$ is called synchronizing if it possesses a reset word.
- The minimum length of reset words for $\mathscr{A}$ is called its reset threshold.

Applications: coding theory, data transmission, robotics, software verification, dna-computing, symbolic dynamics, etc.

The History

The notion was formalized in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konečnými automatami, Matematicko-fyzikalny Časopis Slovensk. Akad. Vied 14, no.3 (1964) 208–216 [in Slovak]) though implicitly it had been around since at least 1956.

The idea of synchronization is pretty natural and of obvious importance: we aim to restore control over a device whose current state is not known.

Think of a satellite which loops around the Moon and cannot be controlled from the Earth while "behind" the Moon (Černý's original motivation).

Independently, the same notion was discovered in coding theory by Shimon Even (Test for synchronizability of finite automata and variable length codes, IEEE Trans. Inform. Theory 10 (1964) 185–189). The name synchronizing seems to have originated from Even's paper.

The History

The notion was formalized in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konečnými automatami, Matematicko-fyzikalny Časopis Slovensk. Akad. Vied 14, no.3 (1964) 208–216 [in Slovak]) though implicitly it had been around since at least 1956.

The idea of synchronization is pretty natural and of obvious importance: we aim to restore control over a device whose current state is not known.

Think of a satellite which loops around the Moon and cannot be controlled from the Earth while “behind” the Moon (Černý’s original motivation).

Independently, the same notion was discovered in coding theory by Shimon Even (Test for synchronizability of finite automata and variable length codes, IEEE Trans. Inform. Theory 10 (1964) 185–189). The name synchronizing seems to have originated from Even’s paper.

The History

The notion was formalized in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konečnými automatami, Matematicko-fyzikalny Časopis Slovensk. Akad. Vied 14, no.3 (1964) 208–216 [in Slovak]) though implicitly it had been around since at least 1956.

The idea of synchronization is pretty natural and of obvious importance: we aim to restore control over a device whose current state is not known.

Think of a satellite which loops around the Moon and cannot be controlled from the Earth while “behind” the Moon (Černý’s original motivation).

Independently, the same notion was discovered in coding theory by Shimon Even (Test for synchronizability of finite automata and variable length codes, IEEE Trans. Inform. Theory 10 (1964) 185–189). The name synchronizing seems to have originated from Even’s paper.

The History

The notion was formalized in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konečnými automatami, Matematicko-fyzikalny Časopis Slovensk. Akad. Vied 14, no.3 (1964) 208–216 [in Slovak]) though implicitly it had been around since at least 1956.

The idea of synchronization is pretty natural and of obvious importance: we aim to restore control over a device whose current state is not known.

Think of a satellite which loops around the Moon and cannot be controlled from the Earth while "behind" the Moon (Černý's original motivation).

Independently, the same notion was discovered in coding theory by Shimon Even (Test for synchronizability of finite automata and variable length codes, IEEE Trans. Inform. Theory 10 (1964) 185–189). The name synchronizing seems to have originated from Even's paper.

The History

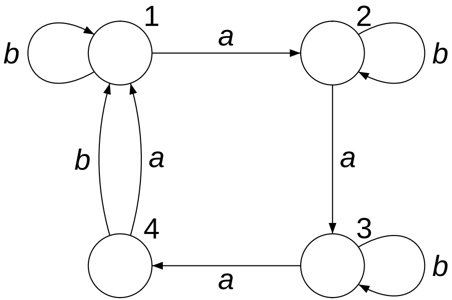
The notion was formalized in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konečnými automatami, Matematicko-fyzikalny Časopis Slovensk. Akad. Vied 14, no.3 (1964) 208–216 [in Slovak]) though implicitly it had been around since at least 1956.

The idea of synchronization is pretty natural and of obvious importance: we aim to restore control over a device whose current state is not known.

Think of a satellite which loops around the Moon and cannot be controlled from the Earth while "behind" the Moon (Černý's original motivation).

Independently, the same notion was discovered in coding theory by Shimon Even (Test for synchronizability of finite automata and variable length codes, IEEE Trans. Inform. Theory 10 (1964) 185–189). The name synchronizing seems to have originated from Even's paper.

Greedy compressing algorithm for synchronization



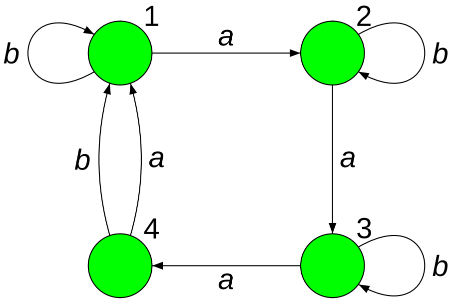
A reset word is $v = baababaaab$.

$\delta(Q, v) =$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization



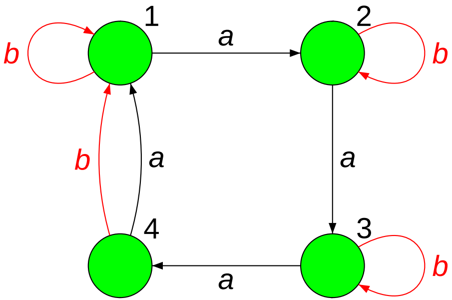
A reset word is $v = baababaaab$.

$\delta(Q, v) =$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization



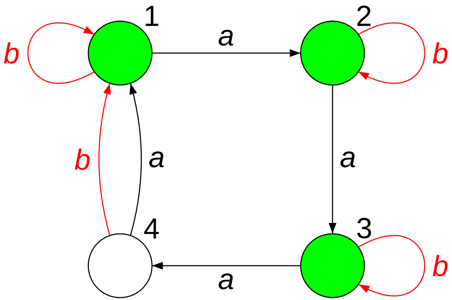
A reset word is $v =$ *baababaaab*.

$\delta(Q, v) = \{1, 2, 3, 4\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization



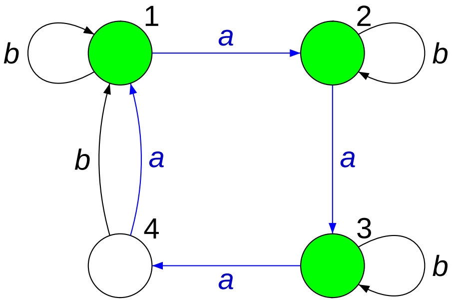
A reset word is $v = baababaaab$.

$\delta(Q, v) = \{1, 2, 3\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization

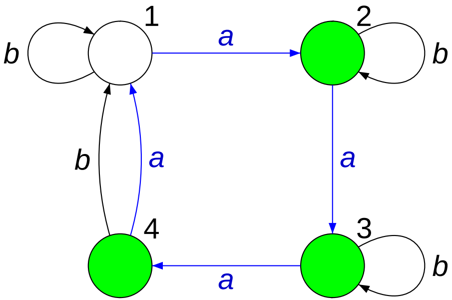


A reset word is $v = b$*aababaaab*.
$\delta(Q, v) = \{1, 2, 3\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization



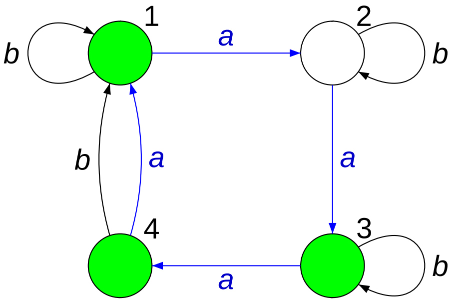
A reset word is $v = ba$ababaaab.

$\delta(Q, v) = \{2, 3, 4\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization



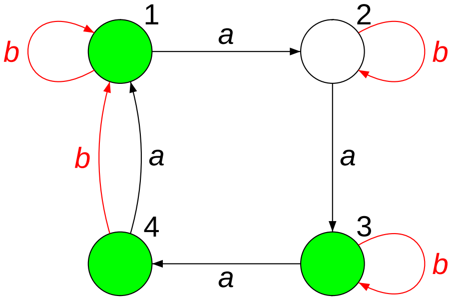
A reset word is $v = baababaaab$.

$\delta(Q, v) = \{1, 3, 4\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization

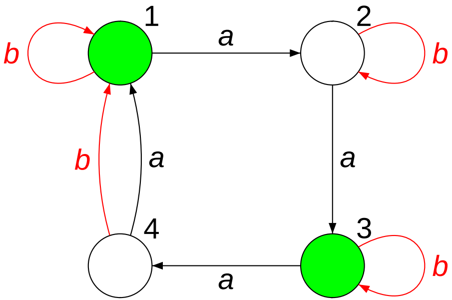


A reset word is $v =$ *baababaaab*.
$\delta(Q, v) = \{1, 3, 4\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization



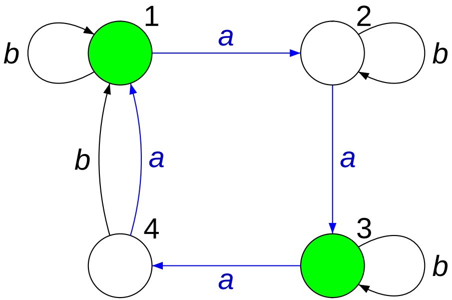
A reset word is $v = baababaaab$.

$\delta(Q, v) = \{1, 3\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization



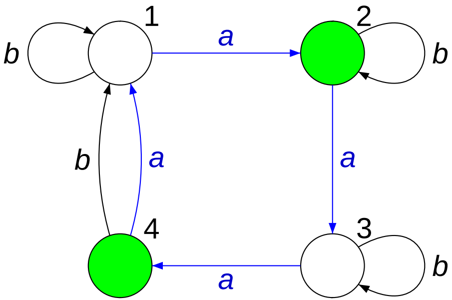
A reset word is $v = baababaaab$.

$\delta(Q, v) = \{1, 3\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization

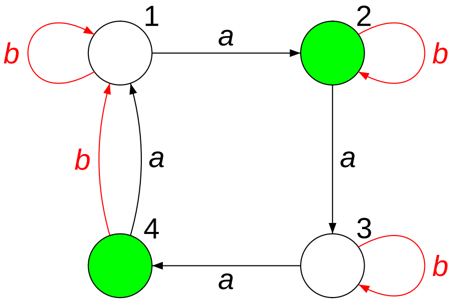


A reset word is $v = baababaaab$.
$\delta(Q, v) = \{2, 4\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization



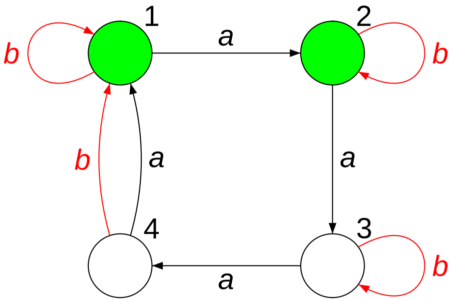
A reset word is $v =$ *baaba*baaab.

$\delta(Q, v) = \{2, 4\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization



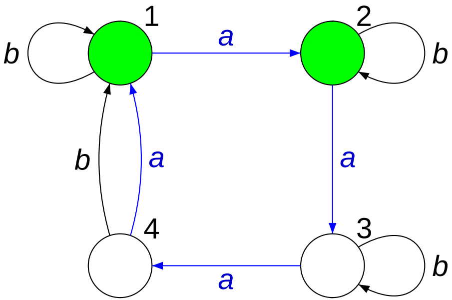
A reset word is $v = baababbaaab$.

$\delta(Q, v) = \{1, 2\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization

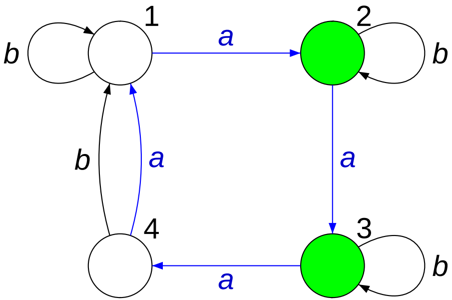


A reset word is $v =$ *baabab*aaab.
$\delta(Q, v) = \{1, 2\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization



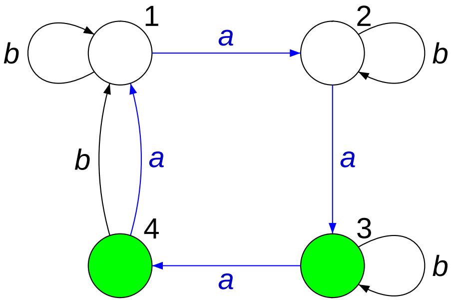
A reset word is $v = baababa\color{blue}{a}aab$.

$\delta(Q, v) = \{2, 3\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization



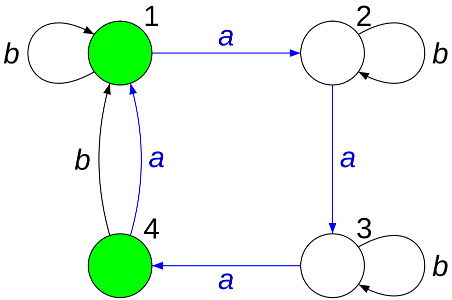
A reset word is $v = baababaaaab$.

$\delta(Q, v) = \{3, 4\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization



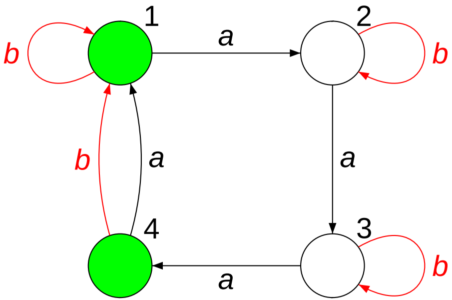
A reset word is $v = baababaaab$.

$\delta(Q, v) = \{1, 4\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization



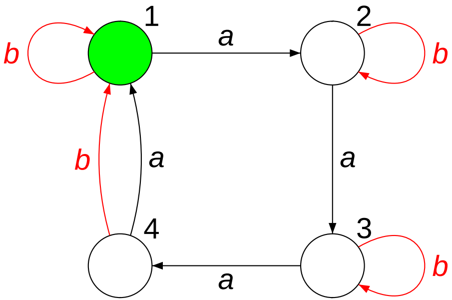
A reset word is $v = baababaaab$.

$\delta(Q, v) = \{1, 4\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization

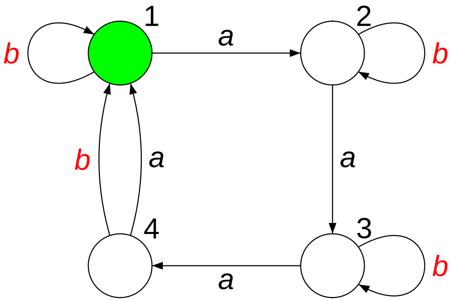


A reset word is $v = baababaaab$.
$\delta(Q, v) = \{1\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Greedy compressing algorithm for synchronization



A reset word is $v = baababaaab$.
$\delta(Q, v) = \{1\}$

The word v is reset whence $rt(\mathscr{A}) \leq |v| = 10$.

The shortest reset word for $\mathscr{A}$ is ba^3ba^3b whence $rt(\mathscr{A}) = 9 < |v|$.

Various Settings for Synchronization and Outline

Whether or not a given automaton is synchronizing?

If it is synchronizing, how hard is to synchronize it?

Various Settings for Synchronization and Outline

Whether or not a given automaton is synchronizing?

If it is synchronizing, how hard is to synchronize it?

1. Deterministic Setting
   - Černý conjecture and Markov Chains
   - Testing for Synchronization
   - Random Case
   - Expected Reset Threshold
   - Computing Reset Thresholds
2. Modifiable Setting
   - Road Coloring Problem
   - Computing Synchronizing Colorings
3. Stochastic Setting
   - Synchronization and Prediction Rates
   - Markov Chain Convergence vs Reset Threshold

The Černý conjecture

Černý, 1964

For each n there is an n-state automaton $\mathscr{C}_n$ with $rt(\mathscr{C}_n) = (n-1)^2$.

The Černý conjecture, 1964

Each n-state synchronizing automaton has a reset word of length $(n-1)^2$, i.e. $rt(\mathscr{A}) \leq (n-1)^2$.

Greedy compression algorithm yields the cubic upper bound $\Theta(n^3/2)$ for the reset threshold.

Pin, 1983 (based on a combinatorial result of Frankl, 1982)

Each n-state automaton has a reset word of length $(n^3-n)/6$.

Quadratic upper bounds on the reset threshold?

The Černý conjecture

Černý, 1964

For each n there is an n-state automaton $\mathscr{C}_n$ with $rt(\mathscr{C}_n) = (n-1)^2$.

The Černý conjecture, 1964

Each n-state synchronizing automaton has a reset word of length $(n-1)^2$, i.e. $rt(\mathscr{A}) \leq (n-1)^2$.

Greedy compression algorithm yields the cubic upper bound $\Theta(n^3/2)$ for the reset threshold.

Pin, 1983 (based on a combinatorial result of Frankl, 1982)

Each n-state automaton has a reset word of length $(n^3-n)/6$.

Quadratic upper bounds on the reset threshold?

The Černý conjecture

Černý, 1964

For each n there is an n-state automaton $\mathscr{C}_n$ with $rt(\mathscr{C}_n) = (n-1)^2$.

The Černý conjecture, 1964

Each n-state synchronizing automaton has a reset word of length $(n-1)^2$, i.e. $rt(\mathscr{A}) \leq (n-1)^2$.

Greedy compression algorithm yields the cubic upper bound $\Theta(n^3/2)$ for the reset threshold.

Pin, 1983 (based on a combinatorial result of Frankl, 1982)

Each n-state automaton has a reset word of length $(n^3-n)/6$.

Quadratic upper bounds on the reset threshold?

The Černý conjecture

Černý, 1964

For each n there is an n-state automaton $\mathscr{C}_n$ with $rt(\mathscr{C}_n) = (n-1)^2$.

The Černý conjecture, 1964

Each n-state synchronizing automaton has a reset word of length $(n-1)^2$, i.e. $rt(\mathscr{A}) \leq (n-1)^2$.

Greedy compression algorithm yields the cubic upper bound $\Theta(n^3/2)$ for the reset threshold.

Pin, 1983 (based on a combinatorial result of Frankl, 1982)

Each n-state automaton has a reset word of length $(n^3-n)/6$.

Quadratic upper bounds on the reset threshold?

Particular Cases

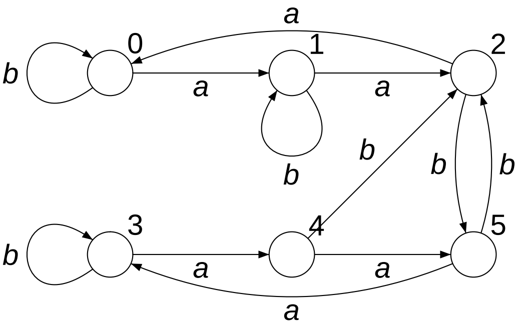
Quadratic bounds were approved for various classes:

- *Circular* automata with prime number of states [Pin, 1978];
- *Orientable* automata [Eppstein, 1990];
- *Circular* automata [Dubuc, 1998];
- *Eulerian* automata [Kari, 2003];
- *Aperiodic* automata [Trahtman, 2007];
- *Weakly-monotonic* automata [Volkov, 2009];
- *With monoids belonging to DS class* automata [Almeida, Margolis, Steinberg, Volkov, 2009];
- *One-cluster* automata [Béal M., Perrin D., 2009];
- *One-cluster* with prime number of states [Steinberg, 2011];
- *Respecting intervals of a directed graph* automata [Grech, Kisielewicz, 2012];
- ...

Linear Algebra, Group and Semigroup theories, theory of Markov chains, ...

Example from the Italian Job Movie

Kari Automaton and Greedy Extension Method

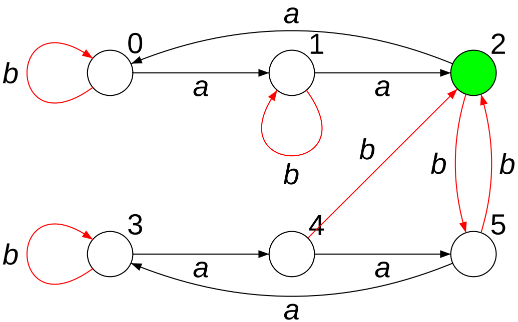


A reset word is the reverse to $v = baabbbabbaab...$

Augmenting sequence is $v_1 = b, v_2 = aabb, v_3 = babbaab, v_4 = \ldots$.

This method is optimal for the Černý series but returns a reset word of length more than $25 = (6-1)^2$ for this automaton.

Kari Automaton and Greedy Extension Method

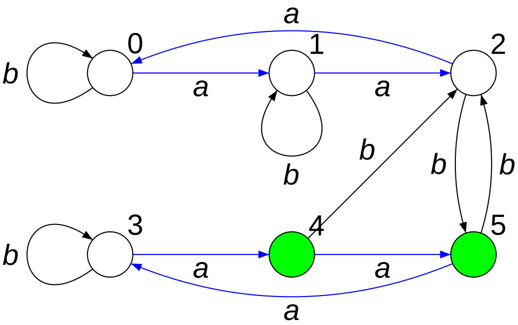


A reset word is the reverse to $v = baabbbabbaab...$

Augmenting sequence is $v_1 = b, v_2 = aabb, v_3 = babbaab, v_4 = \ldots$.

This method is optimal for the Černý series but returns a reset word of length more than $25 = (6-1)^2$ for this automaton.

Kari Automaton and Greedy Extension Method

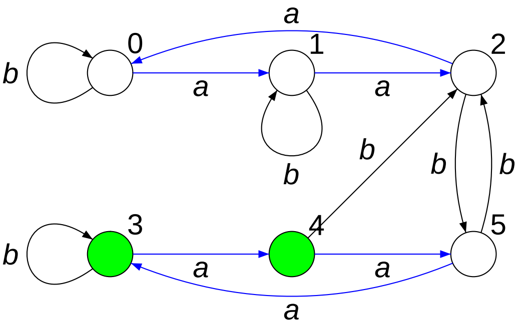


A reset word is the reverse to $v = baabbbabbaab\ldots$

Augmenting sequence is $v_1 = b, v_2 = aabb, v_3 = babbaab, v_4 = \ldots.$

This method is optimal for the Černý series but returns a reset word of length more than $25 = (6-1)^2$ for this automaton.

Kari Automaton and Greedy Extension Method

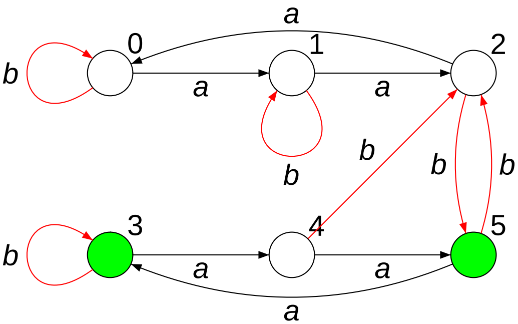


A reset word is the reverse to $v = ba$abbbabbaab...

Augmenting sequence is $v_1 = b$, $v_2 = a$abb, $v_3 = babbaab$, $v_4 = \ldots$.

This method is optimal for the Černý series but returns a reset word of length more than $25 = (6-1)^2$ for this automaton.

Kari Automaton and Greedy Extension Method

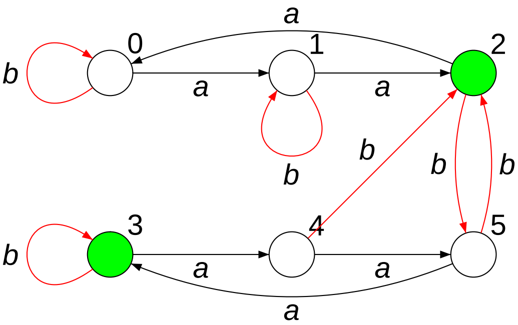


A reset word is the reverse to $v = baabbbabbaab\ldots$

Augmenting sequence is $v_1 = b$, $v_2 = aabb$, $v_3 = babbaab$, $v_4 = \ldots$.

This method is optimal for the Černý series but returns a reset word of length more than $25 = (6-1)^2$ for this automaton.

Kari Automaton and Greedy Extension Method

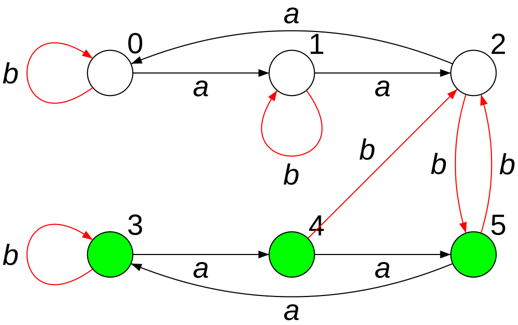


A reset word is the reverse to $v = baabbbabbaab\ldots$

Augmenting sequence is $v_1 = b$, $v_2 = aabb$, $v_3 = babbaab$, $v_4 = \ldots$.

This method is optimal for the Černý series but returns a reset word of length more than $25 = (6-1)^2$ for this automaton.

Kari Automaton and Greedy Extension Method

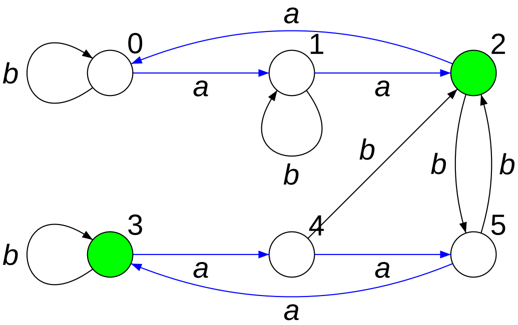


A reset word is the reverse to $v = baabb$babbaab...

Augmenting sequence is $v_1 = b$, $v_2 = aabb$, $v_3 = babbaab$, $v_4 = \ldots$.

This method is optimal for the Černý series but returns a reset word of length more than $25 = (6-1)^2$ for this automaton.

Kari Automaton and Greedy Extension Method

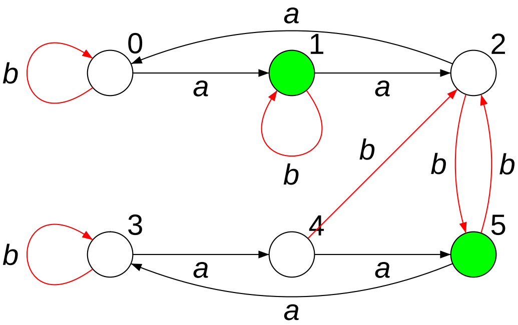


A reset word is the reverse to $v = baabbbabbaab\ldots$

Augmenting sequence is $v_1 = b$, $v_2 = aabb$, $v_3 = babbaab$, $v_4 = \ldots$.

This method is optimal for the Černý series but returns a reset word of length more than $25 = (6-1)^2$ for this automaton.

Kari Automaton and Greedy Extension Method

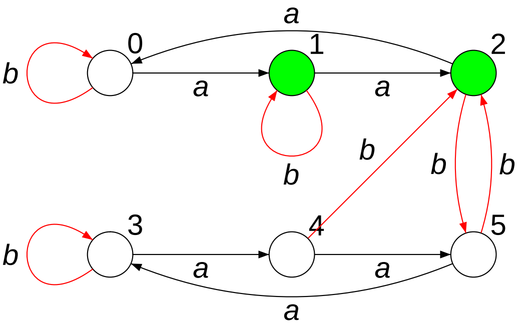


A reset word is the reverse to $v = baabbba$bbaab...

Augmenting sequence is $v_1 = b$, $v_2 = aabb$, $v_3 = ba$bbaab, $v_4 = \ldots$.

This method is optimal for the Černý series but returns a reset word of length more than $25 = (6 - 1)^2$ for this automaton.

Kari Automaton and Greedy Extension Method

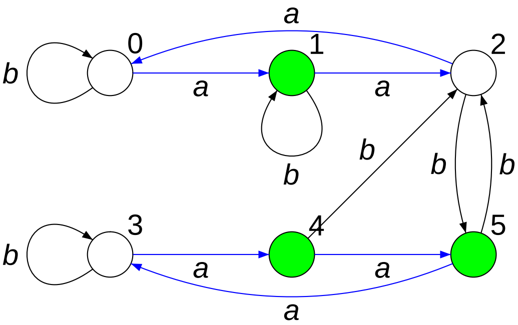


A reset word is the reverse to $v = baabbbabbaab\ldots$

Augmenting sequence is $v_1 = b$, $v_2 = aabb$, $v_3 = babbaab$, $v_4 = \ldots$.

This method is optimal for the Černý series but returns a reset word of length more than $25 = (6-1)^2$ for this automaton.

Kari Automaton and Greedy Extension Method

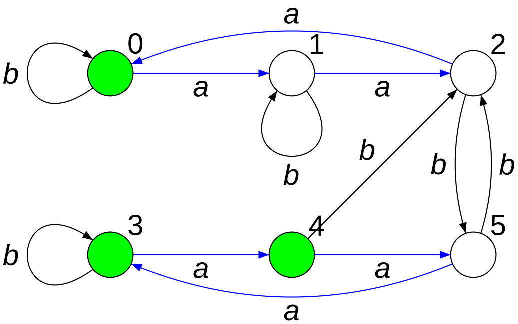


A reset word is the reverse to $v = baabbbabbaab\ldots$

Augmenting sequence is $v_1 = b$, $v_2 = aabb$, $v_3 = babbaab$, $v_4 = \ldots$.

This method is optimal for the Černý series but returns a reset word of length more than $25 = (6-1)^2$ for this automaton.

Kari Automaton and Greedy Extension Method

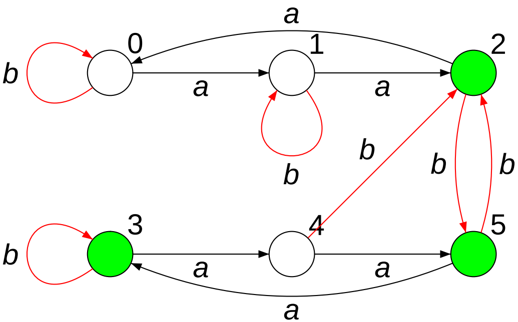


A reset word is the reverse to $v = baabbbabbaab\ldots$

Augmenting sequence is $v_1 = b$, $v_2 = aabb$, $v_3 = babbaab$, $v_4 = \ldots$.

This method is optimal for the Černý series but returns a reset word of length more than $25 = (6-1)^2$ for this automaton.

Kari Automaton and Greedy Extension Method

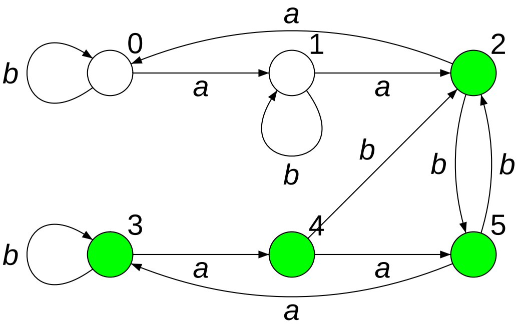


A reset word is the reverse to $v = baabbbabbaab\ldots$

Augmenting sequence is $v_1 = b$, $v_2 = aabb$, $v_3 = babbaab$, $v_4 = \ldots$.

This method is optimal for the Černý series but returns a reset word of length more than $25 = (6-1)^2$ for this automaton.

Kari Automaton and Greedy Extension Method

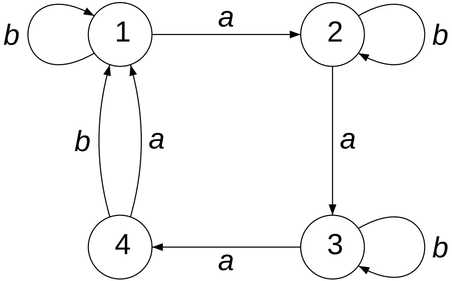


A reset word is the reverse to $v = baabbbabbaab...$

Augmenting sequence is $v_1 = b$, $v_2 = aabb$, $v_3 = babbaab, v_4 = \ldots$.

This method is optimal for the Černý series but returns a reset word of length more than $25 = (6-1)^2$ for this automaton.

Random Walk Synchronization



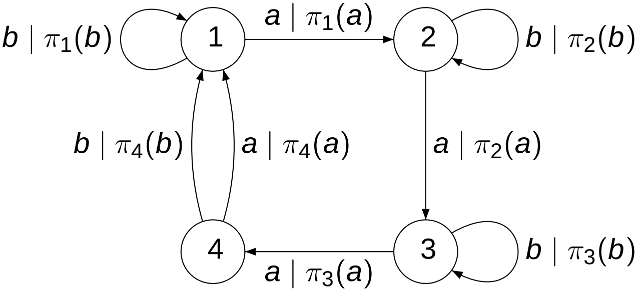
The probability of catching is

Augmenting sequence w.r.t. α is *b*.*aaa*.*ba*.*a*.*a*.*b*

The lengths of words in the augmenting sequence w.r.t. α is always at most $n - 1$ but there can be a-priori even exponential.

The method can be extended to sets of words uW where u is a "compressing words" and W is "complete" for $< Q.u >$ keeping $|uW|$ bound for augmenting words.

Random Walk Synchronization



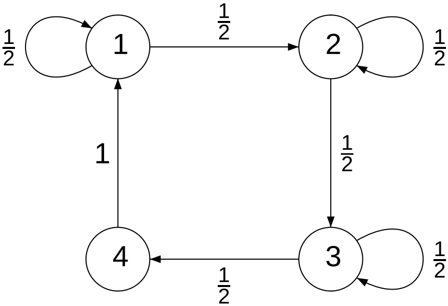
The probability of catching is

Augmenting sequence w.r.t. α is $b.aaa.ba.a.a.b$

The lengths of words in the augmenting sequence w.r.t. α is always at most $n - 1$ but there can be a-priori even exponential.

The method can be extended to sets of words uW where u is a ''compressing words" and W is "complete" for $< Q.u >$ keeping $|uW|$ bound for augmenting words.

Random Walk Synchronization



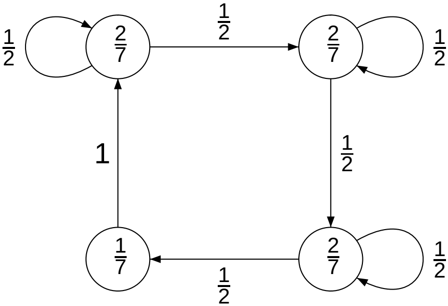
The probability of catching is

Augmenting sequence w.r.t. α is $b.aaa.ba.a.a.b$

The lengths of words in the augmenting sequence w.r.t. α is always at most $n-1$ but there can be a-priori even exponential.

The method can be extended to sets of words uW where u is a ''compressing words" and W is "complete" for $< Q.u >$ keeping $|uW|$ bound for augmenting words.

Random Walk Synchronization



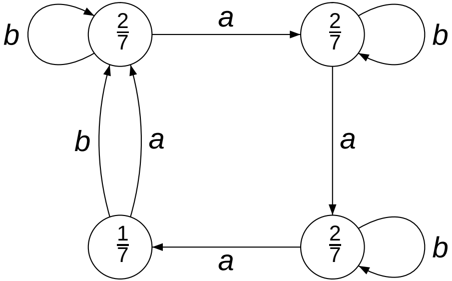
The probability of catching is

Augmenting sequence w.r.t. α is *b.aaa.ba.a.a.b*

The lengths of words in the augmenting sequence w.r.t. α is always at most $n - 1$ but there can be a-priori even exponential.

The method can be extended to sets of words uW where u is a ''compressing words" and W is "complete" for $< Q.u >$ keeping $|uW|$ bound for augmenting words.

Random Walk Synchronization



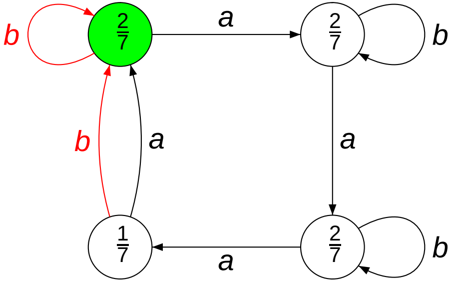
The probability of catching is

Augmenting sequence w.r.t. α is $b.aaa.ba.a.a.b$

The lengths of words in the augmenting sequence w.r.t. α is always at most $n-1$ but there can be a-priori even exponential.

The method can be extended to sets of words uW where u is a ''compressing words" and W is "complete" for $< Q.u >$ keeping $|uW|$ bound for augmenting words.

Random Walk Synchronization



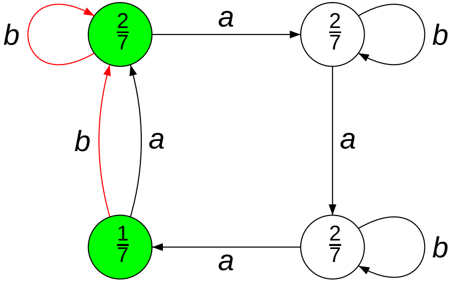
The probability of catching is $\frac{2}{7}$

Augmenting sequence w.r.t. α is *b.aaa.ba.a.a.b*

The lengths of words in the augmenting sequence w.r.t. α is always at most $n - 1$ but there can be a-priori even exponential.

The method can be extended to sets of words uW where u is a ''compressing words" and W is "complete" for $< Q.u >$ keeping $|uW|$ bound for augmenting words.

Random Walk Synchronization



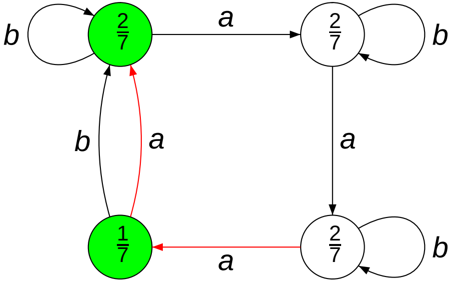
The probability of catching is $\frac{3}{7}$

Augmenting sequence w.r.t. α is *b.aaa.ba.a.a.b*

The lengths of words in the augmenting sequence w.r.t. α is always at most $n - 1$ but there can be a-priori even exponential.

The method can be extended to sets of words uW where u is a ''compressing words" and W is "complete" for $< Q.u >$ keeping $|uW|$ bound for augmenting words.

Random Walk Synchronization



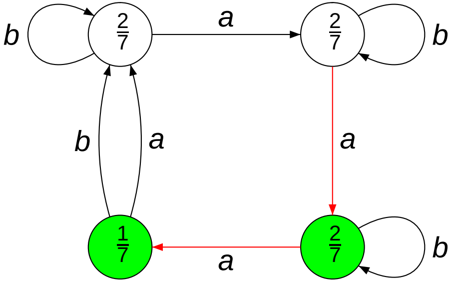
The probability of catching is $\frac{3}{7}$

Augmenting sequence w.r.t. α is *b.aaa.ba.a.a.b*

The lengths of words in the augmenting sequence w.r.t. α is always at most $n - 1$ but there can be a-priori even exponential.

The method can be extended to sets of words uW where u is a "compressing words" and W is "complete" for $< Q.u >$ keeping $|uW|$ bound for augmenting words.

Random Walk Synchronization



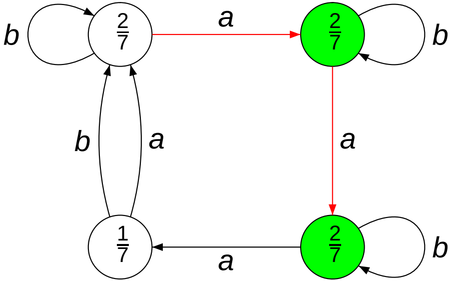
The probability of catching is $\frac{3}{7}$

Augmenting sequence w.r.t. α is $b.aaa.ba.a.a.b$

The lengths of words in the augmenting sequence w.r.t. α is always at most $n-1$ but there can be a-priori even exponential.

The method can be extended to sets of words uW where u is a ''compressing words" and W is "complete" for $< Q.u >$ keeping $|uW|$ bound for augmenting words.

Random Walk Synchronization



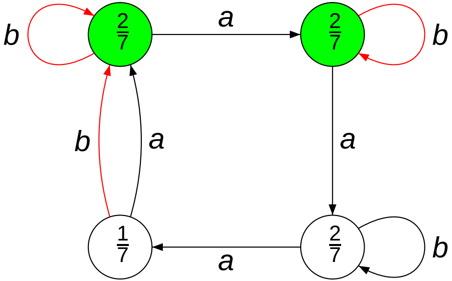
The probability of catching is $\frac{4}{7}$

Augmenting sequence w.r.t. α is *b.aaa.ba.a.a.b*

The lengths of words in the augmenting sequence w.r.t. α is always at most $n - 1$ but there can be a-priori even exponential.

The method can be extended to sets of words uW where u is a "compressing words" and W is "complete" for $< Q.u >$ keeping $|uW|$ bound for augmenting words.

Random Walk Synchronization



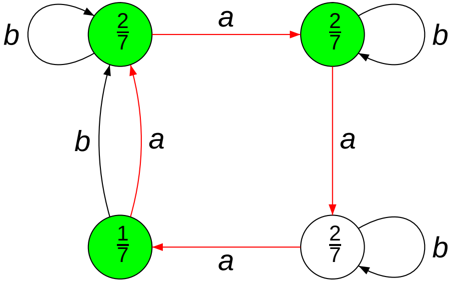
The probability of catching is $\frac{4}{7}$

Augmenting sequence w.r.t. α is *b.aaa.ba.a.a.b*

The lengths of words in the augmenting sequence w.r.t. α is always at most $n - 1$ but there can be a-priori even exponential.

The method can be extended to sets of words uW where u is a "compressing words" and W is "complete" for $< Q.u >$ keeping $|uW|$ bound for augmenting words.

Random Walk Synchronization



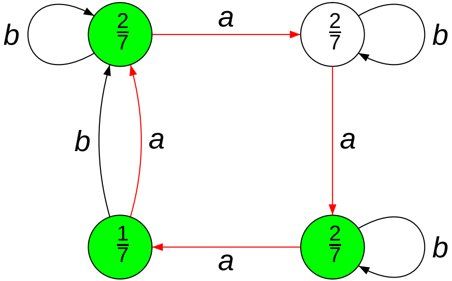
The probability of catching is $\frac{5}{7}$

Augmenting sequence w.r.t. α is *b.aaa.ba.a.a.b*

The lengths of words in the augmenting sequence w.r.t. α is always at most $n - 1$ but there can be a-priori even exponential.

The method can be extended to sets of words uW where u is a "compressing words" and W is "complete" for $< Q.u >$ keeping $|uW|$ bound for augmenting words.

Random Walk Synchronization



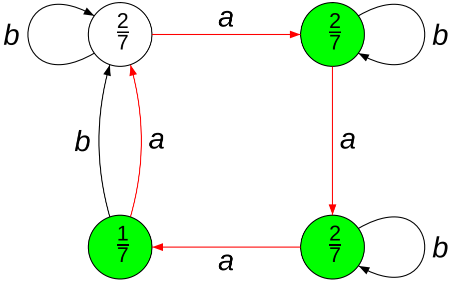
The probability of catching is $\frac{5}{7}$

Augmenting sequence w.r.t. α is $b.aaa.ba.a.a.b$

The lengths of words in the augmenting sequence w.r.t. α is always at most $n - 1$ but there can be a-priori even exponential.

The method can be extended to sets of words uW where u is a ''compressing words" and W is "complete" for $< Q.u >$ keeping $|uW|$ bound for augmenting words.

Random Walk Synchronization



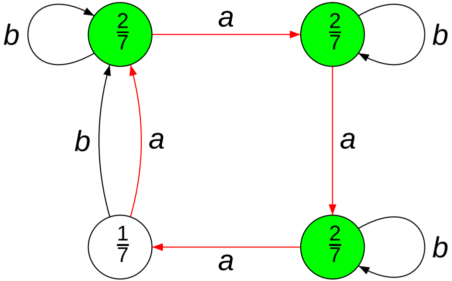
The probability of catching is $\frac{5}{7}$

Augmenting sequence w.r.t. α is $b.aaa.ba.a.a.b$

The lengths of words in the augmenting sequence w.r.t. α is always at most $n-1$ but there can be a-priori even exponential.

The method can be extended to sets of words uW where u is a "compressing words" and W is "complete" for $< Q.u >$ keeping $|uW|$ bound for augmenting words.

Random Walk Synchronization



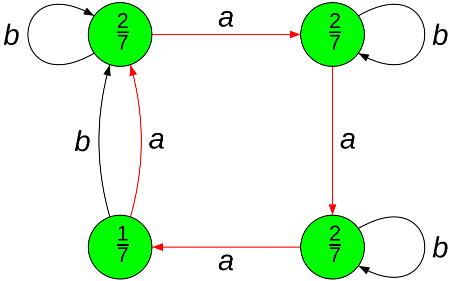
The probability of catching is $\frac{6}{7}$

Augmenting sequence w.r.t. α is $b.aaa.ba.a.a.b$

The lengths of words in the augmenting sequence w.r.t. α is always at most $n - 1$ but there can be a-priori even exponential.

The method can be extended to sets of words uW where u is a ''compressing words" and W is "complete" for $< Q.u >$ keeping $|uW|$ bound for augmenting words.

Random Walk Synchronization



The probability of catching is 1

Augmenting sequence w.r.t. α is $b.aaa.ba.a.a.b$

The lengths of words in the augmenting sequence w.r.t. α is always at most $n-1$ but there can be a-priori even exponential.

The method can be extended to sets of words uW where u is a ''compressing words" and W is "complete" for $< Q.u >$ keeping $|uW|$ bound for augmenting words.

Synchronizing Automata and Markov Chains

Let $\mathscr{A} = (Q, \Sigma)$ be a s.c. automaton.

B. IJFCS 2012

The following are equivalent

1. There is a p.d. $\pi : \Sigma^{n-1} \mapsto \mathbb{R}_+$ with the stationary distribution α of the Markov chain $\mathscr{M}(\mathscr{A}^{n-1}, \pi)$;
2. $\mathscr{A}$ is synchronizing and for each $x \notin < 1_n >$ there is a word $u \in \Sigma^{n-1}$ such that $(\alpha u, x) > (\alpha, x)$;

Corollary: Renew and generalize quadratic bounds on the r.t. for Eulerian and one-cluster case.

Berlinkov, M; Szykuła, M; 2015 (submitted to MFCS)

- $n \log^3 n$ bound for the reset threshold of Prefix Code Automata.
- The Černý conjecture for automata with a letter of rank $\sqrt[3]{6n-6}$. The previous bound is $1 + \log_2 n$.

Synchronizing Automata and Markov Chains

Let $\mathscr{A} = (Q, \Sigma)$ be a s.c. automaton.

B. IJFCS 2012

The following are equivalent

1. There is a p.d. $\pi : \Sigma^{n-1} \mapsto \mathbb{R}_+$ with the stationary distribution α of the Markov chain $\mathscr{M}(\mathscr{A}^{n-1}, \pi)$;
2. $\mathscr{A}$ is synchronizing and for each $x \notin < 1_n >$ there is a word $u \in \Sigma^{n-1}$ such that $(\alpha u, x) > (\alpha, x)$;

Corollary: Renew and generalize quadratic bounds on the r.t. for Eulerian and one-cluster case.

Berlinkov, M; Szykuła, M; 2015 (submitted to MFCS)

- $n \log^3 n$ bound for the reset threshold of Prefix Code Automata.
- The Černý conjecture for automata with a letter of rank $\sqrt[3]{6n - 6}$. The previous bound is $1 + \log_2 n$.

Synchronizing Automata and Markov Chains

Let $\mathscr{A} = (Q, \Sigma)$ be a s.c. automaton.

B. IJFCS 2012

The following are equivalent

1. There is a p.d. $\pi : \Sigma^{n-1} \mapsto \mathbb{R}_+$ with the stationary distribution α of the Markov chain $\mathscr{M}(\mathscr{A}^{n-1}, \pi)$;
2. $\mathscr{A}$ is synchronizing and for each $x \notin < 1_n >$ there is a word $u \in \Sigma^{n-1}$ such that $(\alpha u, x) > (\alpha, x)$;

Corollary: Renew and generalize quadratic bounds on the r.t. for Eulerian and one-cluster case.

Berlinkov, M; Szykuła, M; 2015 (submitted to MFCS)

- $n \log^3 n$ bound for the reset threshold of Prefix Code Automata.
- The Černý conjecture for automata with a letter of rank $\sqrt[3]{6n-6}$. The previous bound is $1 + \log_2 n$.

Synchronizing Automata and Markov Chains

Let $\mathscr{A} = (Q, \Sigma)$ be a s.c. automaton.

B. IJFCS 2012

The following are equivalent

1. There is a p.d. $\pi : \Sigma^{n-1} \mapsto \mathbb{R}_+$ with the stationary distribution α of the Markov chain $\mathscr{M}(\mathscr{A}^{n-1}, \pi)$;
2. $\mathscr{A}$ is synchronizing and for each $x \notin < 1_n >$ there is a word $u \in \Sigma^{n-1}$ such that $(\alpha u, x) > (\alpha, x)$;

Corollary: Renew and generalize quadratic bounds on the r.t. for Eulerian and one-cluster case.

Berlinkov, M; Szykuła, M; 2015 (submitted to MFCS)

- $n \log^3 n$ bound for the reset threshold of Prefix Code Automata.
- The Černý conjecture for automata with a letter of rank $\sqrt[3]{6n-6}$. The previous bound is $1 + \log_2 n$.

Testing for Synchronization

Černý, 1964

$\mathscr{A}$ is synchronizing if and only if each pair of states p, q can be synchronized, i.e. $p.v = q.v$ for some $v \in \Sigma^*$.

The criterion yields $O(n^2)$ algorithm (basically due to Eppstein) which verifies whether or not $\mathscr{A}$ is synchronizing.

Are there more effective (on average) algorithms?

Testing for Synchronization

Černý, 1964

$\mathscr{A}$ is synchronizing if and only if each pair of states p, q can be synchronized, i.e. $p.v = q.v$ for some $v \in \Sigma^*$.

The criterion yields $O(n^2)$ algorithm (basically due to Eppstein) which verifies whether or not $\mathscr{A}$ is synchronizing.

Are there more effective (on average) algorithms?

Testing for Synchronization

Černý, 1964

$\mathscr{A}$ is synchronizing if and only if each pair of states p, q can be synchronized, i.e. $p.v = q.v$ for some $v \in \Sigma^*$.

The criterion yields $O(n^2)$ algorithm (basically due to Eppstein) which verifies whether or not $\mathscr{A}$ is synchronizing.

Are there more effective (on average) algorithms?

The probability of being synchronizable

Let $\mathscr{A} = (Q, \Sigma)$ be an n-state random automaton, that is, the actions of all k letters are chosen u.a.r. and independently from the set of all n^n mappings.

The probability is $1 - \Theta(\frac{1}{n})$ for $k = 2$? [Cameron, 2011].

B. 2013 in ArXiv

The probability for automata of being synchronizable is $1 - O(\frac{1}{n^{k/2}})$ and the bound is tight for the 2-letter alphabet case.

B. 2013 in ArXiv

Given a random n-state automaton, testing for synchronization can be done in $O(n)$ expected time (and it is optimal).

1. Connected case? Supposed bound is $1 - \alpha^n$ for some $\alpha < 1$.
2. k-ary alphabet? Supposed bound is $1 - \Theta(1/n^{k-1})$.

The probability of being synchronizable

Let $\mathscr{A} = (Q, \Sigma)$ be an n-state random automaton, that is, the actions of all k letters are chosen u.a.r. and independently from the set of all n^n mappings.

The probability is $1 - \Theta(\frac{1}{n})$ for $k = 2$? [Cameron, 2011].

B. 2013 in ArXiv

The probability for automata of being synchronizable is $1 - O(\frac{1}{n^{k/2}})$ and the bound is tight for the 2-letter alphabet case.

B. 2013 in ArXiv

Given a random n-state automaton, testing for synchronization can be done in $O(n)$ expected time (and it is optimal).

1. Connected case? Supposed bound is $1 - \alpha^n$ for some $\alpha < 1$.
2. k-ary alphabet? Supposed bound is $1 - \Theta(1/n^{k-1})$.

The probability of being synchronizable

Let $\mathscr{A} = (Q, \Sigma)$ be an n-state random automaton, that is, the actions of all k letters are chosen u.a.r. and independently from the set of all n^n mappings.

The probability is $1 - \Theta(\frac{1}{n})$ for $k = 2$? [Cameron, 2011].

B. 2013 in ArXiv

The probability for automata of being synchronizable is $1 - O(\frac{1}{n^{k/2}})$ and the bound is tight for the 2-letter alphabet case.

B. 2013 in ArXiv

Given a random n-state automaton, testing for synchronization can be done in $O(n)$ expected time (and it is optimal).

1. Connected case? Supposed bound is $1 - \alpha^n$ for some $\alpha < 1$.
2. k-ary alphabet? Supposed bound is $1 - \Theta(1/n^{k-1})$.

The probability of being synchronizable

Let $\mathscr{A} = (Q, \Sigma)$ be an n-state random automaton, that is, the actions of all k letters are chosen u.a.r. and independently from the set of all n^n mappings.

The probability is $1 - \Theta(\frac{1}{n})$ for $k = 2$? [Cameron, 2011].

B. 2013 in ArXiv

The probability for automata of being synchronizable is $1 - O(\frac{1}{n^{k/2}})$ and the bound is tight for the 2-letter alphabet case.

B. 2013 in ArXiv

Given a random n-state automaton, testing for synchronization can be done in $O(n)$ expected time (and it is optimal).

1. Connected case? Supposed bound is $1 - \alpha^n$ for some $\alpha < 1$.
2. k-ary alphabet? Supposed bound is $1 - \Theta(1/n^{k-1})$.

The probability of being synchronizable

Let $\mathscr{A} = (Q, \Sigma)$ be an n-state random automaton, that is, the actions of all k letters are chosen u.a.r. and independently from the set of all n^n mappings.

The probability is $1 - \Theta(\frac{1}{n})$ for $k = 2$? [Cameron, 2011].

B. 2013 in ArXiv

The probability for automata of being synchronizable is $1 - O(\frac{1}{n^{k/2}})$ and the bound is tight for the 2-letter alphabet case.

B. 2013 in ArXiv

Given a random n-state automaton, testing for synchronization can be done in $O(n)$ expected time (and it is optimal).

1. Connected case? Supposed bound is $1 - \alpha^n$ for some $\alpha < 1$.
2. k-ary alphabet? Supposed bound is $1 - \Theta(1/n^{k-1})$.

Expected Reset Threshold

Let $\mathscr{A}$ be a random n-state synchronizing automaton.
What is the expected reset threshold of $\mathscr{A}$?

Experiments show that the expected reset threshold is in $\Omega(2.5\sqrt{n})$ [Kisielewicz, Kowalski, Szykuła 2012].

Nycaud, 2014

For each $0 < \epsilon < 1/8$ a random binary n-state automaton has a reset word of length at most $n^{1+\epsilon}$ with probability $1 - O(n^{-\frac{1}{8}+\epsilon})$.

This yields $O(n^{2.875})$ upper bound on the expected reset threshold.

Corollary; B., Szykuła, 2015 (submitted to MFCS)

The expected value of the reset threshold is at most $n^{7/4+o(1)}$.

We guess the bound can be improved to $n^{1+o(1)}$.

Expected Reset Threshold

Let $\mathscr{A}$ be a random n-state synchronizing automaton.
What is the expected reset threshold of $\mathscr{A}$?

Experiments show that the expected reset threshold is in $\Omega(2.5\sqrt{n})$ [Kisielewicz, Kowalski, Szykuła 2012].

Nycaud, 2014

For each $0 < \epsilon < 1/8$ a random binary n-state automaton has a reset word of length at most $n^{1+\epsilon}$ with probability $1 - O(n^{-\frac{1}{8}+\epsilon})$.

This yields $O(n^{2.875})$ upper bound on the expected reset threshold.

Corollary; B., Szykuła, 2015 (submitted to MFCS)

The expected value of the reset threshold is at most $n^{7/4+o(1)}$.

We guess the bound can be improved to $n^{1+o(1)}$.

Expected Reset Threshold

Let $\mathscr{A}$ be a random n-state synchronizing automaton.
What is the expected reset threshold of $\mathscr{A}$?

Experiments show that the expected reset threshold is in $\Omega(2.5\sqrt{n})$ [Kisielewicz, Kowalski, Szykuła 2012].

Nycaud, 2014

For each $0 < \epsilon < 1/8$ a random binary n-state automaton has a reset word of length at most $n^{1+\epsilon}$ with probability $1 - O(n^{-\frac{1}{8}+\epsilon})$.

This yields $O(n^{2.875})$ upper bound on the expected reset threshold.

Corollary; B., Szykuła, 2015 (submitted to MFCS)

The expected value of the reset threshold is at most $n^{7/4+o(1)}$.

We guess the bound can be improved to $n^{1+o(1)}$.

Expected Reset Threshold

Let $\mathscr{A}$ be a random n-state synchronizing automaton.
What is the expected reset threshold of $\mathscr{A}$?

Experiments show that the expected reset threshold is in $\Omega(2.5\sqrt{n})$ [Kisielewicz, Kowalski, Szykuła 2012].

Nycaud, 2014

For each $0 < \epsilon < 1/8$ a random binary n-state automaton has a reset word of length at most $n^{1+\epsilon}$ with probability $1 - O(n^{-\frac{1}{8}+\epsilon})$.

This yields $O(n^{2.875})$ upper bound on the expected reset threshold.

Corollary; B., Szykuła, 2015 (submitted to MFCS)

The expected value of the reset threshold is at most $n^{7/4+o(1)}$.

We guess the bound can be improved to $n^{1+o(1)}$.

Expected Reset Threshold

Let $\mathscr{A}$ be a random n-state synchronizing automaton.
What is the expected reset threshold of $\mathscr{A}$?

Experiments show that the expected reset threshold is in $\Omega(2.5\sqrt{n})$ [Kisielewicz, Kowalski, Szykuła 2012].

Nycaud, 2014

For each $0 < \epsilon < 1/8$ a random binary n-state automaton has a reset word of length at most $n^{1+\epsilon}$ with probability $1 - O(n^{-\frac{1}{8}+\epsilon})$.

This yields $O(n^{2.875})$ upper bound on the expected reset threshold.

Corollary; B., Szykuła, 2015 (submitted to MFCS)

The expected value of the reset threshold is at most $n^{7/4+o(1)}$.

We guess the bound can be improved to $n^{1+o(1)}$.

Expected Reset Threshold

Let $\mathscr{A}$ be a random n-state synchronizing automaton.
What is the expected reset threshold of $\mathscr{A}$?

Experiments show that the expected reset threshold is in $\Omega(2.5\sqrt{n})$ [Kisielewicz, Kowalski, Szykuła 2012].

Nycaud, 2014

For each $0 < \epsilon < 1/8$ a random binary n-state automaton has a reset word of length at most $n^{1+\epsilon}$ with probability $1 - O(n^{-\frac{1}{8}+\epsilon})$.

This yields $O(n^{2.875})$ upper bound on the expected reset threshold.

Corollary; B., Szykuła, 2015 (submitted to MFCS)

The expected value of the reset threshold is at most $n^{7/4+o(1)}$.

We guess the bound can be improved to $n^{1+o(1)}$.

Hardness of Computing a Reset Threshold

Given a k-letter n-state synchronizing automaton $\mathscr{A}$, compute its reset threshold.

Unless $P = NP$, there are no polynomial-time algorithm for the following approximation.

- exactly [Rystsov, 1980; Eppstein, 1990],
- within any constant factor for $k = 2$ [B. CSR, 2010],
- within $c \log n$ for $k \uparrow$ [Gerbush, Heeringa, 2011],
- within $0.5c \log n$ for $k = 2$ [B. 2013]
- within n^{ϵ} for $k = 2$ and certain $\epsilon > 0$ [Gawrychovski, 2014].

What is the minimum of $\epsilon \leq 1$ for which n^{ϵ}-approximation is possible?

Hardness of Computing a Reset Threshold

Given a k-letter n-state synchronizing automaton $\mathscr{A}$, compute its reset threshold.

Unless $P = NP$, there are no polynomial-time algorithm for the following approximation.

- exactly [Rystsov, 1980; Eppstein, 1990],
- within any constant factor for $k = 2$ [B. CSR, 2010],
- within $c \log n$ for $k \uparrow$ [Gerbush, Heeringa, 2011],
- within $0.5c \log n$ for $k = 2$ [B. 2013]
- within n^{ϵ} for $k = 2$ and certain $\epsilon > 0$ [Gawrychovski, 2014].

What is the minimum of $\epsilon \leq 1$ for which n^{ϵ}-approximation is possible?

Hardness of Computing a Reset Threshold

Given a k-letter n-state synchronizing automaton $\mathscr{A}$, compute its reset threshold.

Unless $P = NP$, there are no polynomial-time algorithm for the following approximation.

- exactly [Rystsov, 1980; Eppstein, 1990],
- within any constant factor for $k = 2$ [B. CSR, 2010],
- within $c \log n$ for $k \uparrow$ [Gerbush, Heeringa, 2011],
- within $0.5c \log n$ for $k = 2$ [B. 2013]
- within n^{ϵ} for $k = 2$ and certain $\epsilon > 0$ [Gawrychovski, 2014].

What is the minimum of $\epsilon \leq 1$ for which n^{ϵ}-approximation is possible?

Hardness of Computing a Reset Threshold

Given a k-letter n-state synchronizing automaton $\mathscr{A}$, compute its reset threshold.

Unless $P = NP$, there are no polynomial-time algorithm for the following approximation.

- exactly [Rystsov, 1980; Eppstein, 1990],
- within any constant factor for $k = 2$ [B. CSR, 2010],
- within $c \log n$ for $k \uparrow$ [Gerbush, Heeringa, 2011],
- within $0.5c \log n$ for $k = 2$ [B. 2013]
- within n^{ϵ} for $k = 2$ and certain $\epsilon > 0$ [Gawrychovski, 2014].

What is the minimum of $\epsilon \leq 1$ for which n^{ϵ}-approximation is possible?

Hardness of Computing a Reset Threshold

Given a k-letter n-state synchronizing automaton $\mathscr{A}$, compute its reset threshold.

Unless $P = NP$, there are no polynomial-time algorithm for the following approximation.

- exactly [Rystsov, 1980; Eppstein, 1990],
- within any constant factor for $k = 2$ [B. CSR, 2010],
- within $c \log n$ for $k \uparrow$ [Gerbush, Heeringa, 2011],
- within $0.5c \log n$ for $k = 2$ [B. 2013]
- within n^{ϵ} for $k = 2$ and certain $\epsilon > 0$ [Gawrychovski, 2014].

What is the minimum of $\epsilon \leq 1$ for which n^{ϵ}-approximation is possible?

Hardness of Computing a Reset Threshold

Given a k-letter n-state synchronizing automaton $\mathscr{A}$, compute its reset threshold.

Unless $P = NP$, there are no polynomial-time algorithm for the following approximation.

- exactly [Rystsov, 1980; Eppstein, 1990],
- within any constant factor for $k = 2$ [B. CSR, 2010],
- within $c \log n$ for $k \uparrow$ [Gerbush, Heeringa, 2011],
- within $0.5c \log n$ for $k = 2$ [B. 2013]
- within n^{ϵ} for $k = 2$ and certain $\epsilon > 0$ [Gawrychovski, 2014].

What is the minimum of $\epsilon \leq 1$ for which n^{ϵ}-approximation is possible?

Hardness of Computing a Reset Threshold

Given a k-letter n-state synchronizing automaton $\mathscr{A}$, compute its reset threshold.

Unless $P = NP$, there are no polynomial-time algorithm for the following approximation.

- exactly [Rystsov, 1980; Eppstein, 1990],
- within any constant factor for $k = 2$ [B. CSR, 2010],
- within $c \log n$ for $k \uparrow$ [Gerbush, Heeringa, 2011],
- within $0.5c \log n$ for $k = 2$ [B. 2013]
- within n^{ϵ} for $k = 2$ and certain $\epsilon > 0$ [Gawrychovski, 2014].

What is the minimum of $\epsilon \leq 1$ for which n^{ϵ}-approximation is possible?

Hardness of Computing a Reset Threshold

Given a k-letter n-state synchronizing automaton $\mathscr{A}$, compute its reset threshold.

Unless $P = NP$, there are no polynomial-time algorithm for the following approximation.

- exactly [Rystsov, 1980; Eppstein, 1990],
- within any constant factor for $k = 2$ [B. CSR, 2010],
- within $c \log n$ for $k \uparrow$ [Gerbush, Heeringa, 2011],
- within $0.5c \log n$ for $k = 2$ [B. 2013]
- within n^{ϵ} for $k = 2$ and certain $\epsilon > 0$ [Gawrychovski, 2014].

What is the minimum of $\epsilon \leq 1$ for which n^{ϵ}-approximation is possible?

Finding Reset Words of Prescribed Lengths

Given a k-letter n-state synchronizing automaton $\mathscr{A}$ such that $rt(\mathscr{A}) \leq L$, return a reset word of length at most L.

- Greedy compression algorithm for the general case;
- Particular classes for which the proof is constructive and polynomial;

B., Szykuła, 2015 (submitted)

Polynomial algorithms for (Quasi-)Eulerian, (Quasi-)One-Cluster and Prefix code automata.

Given a k-letter n-state circular synchronizing automaton, return a reset word of length at most $(n-1)^2$.

Finding Reset Words of Prescribed Lengths

Given a k-letter n-state synchronizing automaton $\mathscr{A}$ such that $rt(\mathscr{A}) \leq L$, return a reset word of length at most L.

- Greedy compression algorithm for the general case;
- Particular classes for which the proof is constructive and polynomial;

B., Szykuła, 2015 (submitted)

Polynomial algorithms for (Quasi-)Eulerian, (Quasi-)One-Cluster and Prefix code automata.

Given a k-letter n-state circular synchronizing automaton, return a reset word of length at most $(n-1)^2$.

Finding Reset Words of Prescribed Lengths

Given a k-letter n-state synchronizing automaton $\mathscr{A}$ such that $rt(\mathscr{A}) \leq L$, return a reset word of length at most L.

- Greedy compression algorithm for the general case;
- Particular classes for which the proof is constructive and polynomial;

B., Szykuła, 2015 (submitted)

Polynomial algorithms for (Quasi-)Eulerian, (Quasi-)One-Cluster and Prefix code automata.

Given a k-letter n-state circular synchronizing automaton, return a reset word of length at most $(n-1)^2$.

Finding Reset Words of Prescribed Lengths

Given a k-letter n-state synchronizing automaton $\mathscr{A}$ such that $rt(\mathscr{A}) \leq L$, return a reset word of length at most L.

- Greedy compression algorithm for the general case;
- Particular classes for which the proof is constructive and polynomial;

B., Szykuła, 2015 (submitted)

Polynomial algorithms for (Quasi-)Eulerian, (Quasi-)One-Cluster and Prefix code automata.

Given a k-letter n-state circular synchronizing automaton, return a reset word of length at most $(n-1)^2$.

Finding Reset Words of Prescribed Lengths

Given a k-letter n-state synchronizing automaton $\mathscr{A}$ such that $rt(\mathscr{A}) \leq L$, return a reset word of length at most L.

- Greedy compression algorithm for the general case;
- Particular classes for which the proof is constructive and polynomial;

B., Szykuła, 2015 (submitted)

Polynomial algorithms for (Quasi-)Eulerian, (Quasi-)One-Cluster and Prefix code automata.

Given a k-letter n-state circular synchronizing automaton, return a reset word of length at most $(n-1)^2$.

Road Coloring Problem

Let $\mathscr{A}$ be a (non-synchronizing) automaton. Is there a synchronizing automaton $\mathscr{B}$ with the same underlying graph as $\mathscr{A}$?

Road Coloring Problem [Adler, Goodwin, Weiss, 1977]

Does each strongly-connected aperiodic graph (AGW-graph) have a *synchronizing coloring*?

Trahtman, 2007

Each AGW-graph has a synchronizing coloring.

Proof sketch:

- Find a coloring s.t. one letter has a unique highest tree;
- Find a stable pair of states at the bottom of this tree.
- Consider the factor automaton with respect to the stability relation.

Road Coloring Problem

Let $\mathscr{A}$ be a (non-synchronizing) automaton. Is there a synchronizing automaton $\mathscr{B}$ with the same underlying graph as $\mathscr{A}$?

Road Coloring Problem [Adler, Goodwin, Weiss, 1977]

Does each strongly-connected aperiodic graph (AGW-graph) have a *synchronizing coloring*?

Trahtman, 2007

Each AGW-graph has a synchronizing coloring.

Proof sketch:

- Find a coloring s.t. one letter has a unique highest tree;
- Find a stable pair of states at the bottom of this tree.
- Consider the factor automaton with respect to the stability relation

Road Coloring Problem

Let $\mathscr{A}$ be a (non-synchronizing) automaton. Is there a synchronizing automaton $\mathscr{B}$ with the same underlying graph as $\mathscr{A}$?

Road Coloring Problem [Adler, Goodwin, Weiss, 1977]

Does each strongly-connected aperiodic graph (AGW-graph) have a *synchronizing coloring*?

Trahtman, 2007

Each AGW-graph has a synchronizing coloring.

Proof sketch:

- Find a coloring s.t. one letter has a unique highest tree;
- Find a stable pair of states at the bottom of this tree.
- Consider the factor automaton with respect to the stability relation.

Road Coloring Problem

Let $\mathscr{A}$ be a (non-synchronizing) automaton. Is there a synchronizing automaton $\mathscr{B}$ with the same underlying graph as $\mathscr{A}$?

Road Coloring Problem [Adler, Goodwin, Weiss, 1977]

Does each strongly-connected aperiodic graph (AGW-graph) have a *synchronizing coloring*?

Trahtman, 2007

Each AGW-graph has a synchronizing coloring.

Proof sketch:

- Find a coloring s.t. one letter has a unique highest tree;
- Find a stable pair of states at the bottom of this tree.
- Consider the factor automaton with respect to the stability relation.

Road Coloring Problem

Let $\mathscr{A}$ be a (non-synchronizing) automaton. Is there a synchronizing automaton $\mathscr{B}$ with the same underlying graph as $\mathscr{A}$?

Road Coloring Problem [Adler, Goodwin, Weiss, 1977]

Does each strongly-connected aperiodic graph (AGW-graph) have a *synchronizing coloring*?

Trahtman, 2007

Each AGW-graph has a synchronizing coloring.

Proof sketch:

- Find a coloring s.t. one letter has a unique highest tree;
- Find a stable pair of states at the bottom of this tree.
- Consider the factor automaton with respect to the stability relation.

Road Coloring Problem

Let $\mathscr{A}$ be a (non-synchronizing) automaton. Is there a synchronizing automaton $\mathscr{B}$ with the same underlying graph as $\mathscr{A}$?

Road Coloring Problem [Adler, Goodwin, Weiss, 1977]

Does each strongly-connected aperiodic graph (AGW-graph) have a *synchronizing coloring*?

Trahtman, 2007

Each AGW-graph has a synchronizing coloring.

Proof sketch:

- Find a coloring s.t. one letter has a unique highest tree;
- Find a stable pair of states at the bottom of this tree.
- Consider the factor automaton with respect to the stability relation.

Computing Synchronizing Coloring

Let $\mathscr{A}$ be an n-state automaton with AGW-graph. How complicated to find a *synchronizing coloring*?

Trahtman, 2008

Cubic time algorithm.

Béal, Perrin, 2008

Quadratic time algorithm.

How complicated to find an *optimal synchronizing coloring*?

B. 2009, Applied Discrete Mathematics J. (in Russian)

No polynomial time algorithm can *approximate* this problem within a constant factor less than 2.

Complexity of approximation within factor 2?

Computing Synchronizing Coloring

Let $\mathscr{A}$ be an n-state automaton with AGW-graph. How complicated to find a *synchronizing coloring*?

Trahtman, 2008

Cubic time algorithm.

Béal, Perrin, 2008

Quadratic time algorithm.

How complicated to find an *optimal synchronizing coloring*?

B. 2009, Applied Discrete Mathematics J. (in Russian)

No polynomial time algorithm can *approximate* this problem within a constant factor less than 2.

Complexity of approximation within factor 2?

Computing Synchronizing Coloring

Let $\mathscr{A}$ be an n-state automaton with AGW-graph. How complicated to find a *synchronizing coloring*?

Trahtman, 2008

Cubic time algorithm.

Béal, Perrin, 2008

Quadratic time algorithm.

How complicated to find an *optimal synchronizing coloring*?

B. 2009, Applied Discrete Mathematics J. (in Russian)

No polynomial time algorithm can *approximate* this problem within a constant factor less than 2.

Complexity of approximation within factor 2?

Computing Synchronizing Coloring

Let $\mathscr{A}$ be an n-state automaton with AGW-graph. How complicated to find a *synchronizing coloring*?

Trahtman, 2008

Cubic time algorithm.

Béal, Perrin, 2008

Quadratic time algorithm.

How complicated to find an *optimal synchronizing coloring*?

B. 2009, Applied Discrete Mathematics J. (in Russian)

No polynomial time algorithm can *approximate* this problem within a constant factor less than 2.

Complexity of approximation within factor 2?

Computing Synchronizing Coloring

Let $\mathscr{A}$ be an n-state automaton with AGW-graph. How complicated to find a *synchronizing coloring*?

Trahtman, 2008

Cubic time algorithm.

Béal, Perrin, 2008

Quadratic time algorithm.

How complicated to find an *optimal synchronizing coloring*?

B. 2009, Applied Discrete Mathematics J. (in Russian)

No polynomial time algorithm can *approximate* this problem within a constant factor less than 2.

Complexity of approximation within factor 2?

Computing Synchronizing Coloring

Let $\mathscr{A}$ be an n-state automaton with AGW-graph. How complicated to find a *synchronizing coloring*?

Trahtman, 2008

Cubic time algorithm.

Béal, Perrin, 2008

Quadratic time algorithm.

How complicated to find an *optimal synchronizing coloring*?

B. 2009, Applied Discrete Mathematics J. (in Russian)

No polynomial time algorithm can *approximate* this problem within a constant factor less than 2.

Complexity of approximation within factor 2?

Synchronization and Prediction Rates

Let $\mathscr{A}$ be a s.c. automaton equipped with transition probabilities defined for each state independently. If there are no pairs with equivalent probability future, $\mathscr{A}$ is called an ϵ-machine.

Travers, N.; Crutchfield, P; 2011

Let $p_j(u)$ be the probability of the most probable state if $u \in \Sigma^j$ is generated by $\mathscr{A}$. Then for some $0 < a, b < 1$

- If $\mathscr{A}$ is synchronizing then $Pr(p_j < 1) \leq O(a^L)$ - exact;
- If $\mathscr{A}$ is not synchronizing, $Pr(p_j < 1 - b^L) \to 0$ - asymptotic.

The infinum of such a and b are called synchronization rate and prediction rate constants resp.

B. 2014 (in ArXiv)

The synchronization and prediction rate constants can be approximated in polynomial time with any given precision.

Synchronization and Prediction Rates

Let $\mathscr{A}$ be a s.c. automaton equipped with transition probabilities defined for each state independently. If there are no pairs with equivalent probability future, $\mathscr{A}$ is called an ϵ-machine.

Travers, N.; Crutchfield, P; 2011

Let $p_j(u)$ be the probability of the most probable state if $u \in \Sigma^j$ is generated by $\mathscr{A}$. Then for some $0 < a, b < 1$

- If $\mathscr{A}$ is synchronizing then $Pr(p_j < 1) \leq O(a^L)$ - exact;
- If $\mathscr{A}$ is not synchronizing, $Pr(p_j < 1 - b^L) \to 0$ - asymptotic.

The infinum of such a and b are called synchronization rate and prediction rate constants resp.

B. 2014 (in ArXiv)

The synchronization and prediction rate constants can be approximated in polynomial time with any given precision.

Synchronization and Prediction Rates

Let $\mathscr{A}$ be a s.c. automaton equipped with transition probabilities defined for each state independently. If there are no pairs with equivalent probability future, $\mathscr{A}$ is called an ϵ-machine.

Travers, N.; Crutchfield, P; 2011

Let $p_j(u)$ be the probability of the most probable state if $u \in \Sigma^j$ is generated by $\mathscr{A}$. Then for some $0 < a, b < 1$

- If $\mathscr{A}$ is synchronizing then $Pr(p_j < 1) \leq O(a^L)$ - exact;
- If $\mathscr{A}$ is not synchronizing, $Pr(p_j < 1 - b^L) \to 0$ - asymptotic.

The infinum of such a and b are called synchronization rate and prediction rate constants resp.

B. 2014 (in ArXiv)

The synchronization and prediction rate constants can be approximated in polynomial time with any given precision.

Synchronization and Prediction Rates

Let $\mathscr{A}$ be a s.c. automaton equipped with transition probabilities defined for each state independently. If there are no pairs with equivalent probability future, $\mathscr{A}$ is called an ϵ-machine.

Travers, N.; Crutchfield, P; 2011

Let $p_j(u)$ be the probability of the most probable state if $u \in \Sigma^j$ is generated by $\mathscr{A}$. Then for some $0 < a, b < 1$

- If $\mathscr{A}$ is synchronizing then $Pr(p_j < 1) \leq O(a^L)$ - exact;
- If $\mathscr{A}$ is not synchronizing, $Pr(p_j < 1 - b^L) \to 0$ - asymptotic.

The infinum of such a and b are called synchronization rate and prediction rate constants resp.

B. 2014 (in ArXiv)

The synchronization and prediction rate constants can be approximated in polynomial time with any given precision.

Markov Chain Convergence vs Reset Threshold

Let $u \in \Sigma^j$ be a randomly generated word by ϵ-machine $\mathscr{A}$ and $p \in Q$ and $j \geq n - 1$. Then $rt(\mathscr{A}) \leq j$ if either

- $Pr(u \text{ is reset }) > 0$ or
- $\sum_{q \in Q} Pr(p.u \neq q.u) < 1$ or
- $Pr(q_1.u = p; q_2.u = p) \geq Pr(q_1.u = p)Pr(q_2.u = p)$.

Suppose $\mathscr{A}$ has the AGW-graph; Then

- The corresponding Markov chain $\mathscr{M}$ is *mixing*.
- Due to the RCP solution, we can define a synchronizing automaton within the probability distribution on the alphabet such that the induced Markov chain is $\mathscr{M}$ [Kouji Yano, Kenji Yasutom].

Does the condition that a graph is the AGW-graph imply faster convergence of M?

Markov Chain Convergence vs Reset Threshold

Let $u \in \Sigma^j$ be a randomly generated word by ϵ-machine $\mathscr{A}$ and $p \in Q$ and $j \geq n - 1$. Then $rt(\mathscr{A}) \leq j$ if either

- $Pr(u \text{ is reset }) > 0$ or
- $\sum_{q \in Q} Pr(p.u \neq q.u) < 1$ or
- $Pr(q_1.u = p; q_2.u = p) \geq Pr(q_1.u = p)Pr(q_2.u = p)$.

Suppose $\mathscr{A}$ has the AGW-graph; Then

- The corresponding Markov chain $\mathscr{M}$ is *mixing*.
- Due to the RCP solution, we can define a synchronizing automaton within the probability distribution on the alphabet such that the induced Markov chain is $\mathscr{M}$ [Kouji Yano, Kenji Yasutom].

Does the condition that a graph is the AGW-graph imply faster convergence of M?

Markov Chain Convergence vs Reset Threshold

Let $u \in \Sigma^j$ be a randomly generated word by ϵ-machine $\mathscr{A}$ and $p \in Q$ and $j \geq n - 1$. Then $rt(\mathscr{A}) \leq j$ if either

- $Pr(u \text{ is reset }) > 0$ or
- $\sum_{q \in Q} Pr(p.u \neq q.u) < 1$ or
- $Pr(q_1.u = p; q_2.u = p) \geq Pr(q_1.u = p)Pr(q_2.u = p)$.

Suppose $\mathscr{A}$ has the AGW-graph; Then

- The corresponding Markov chain $\mathscr{M}$ is *mixing*.
- Due to the RCP solution, we can define a synchronizing automaton within the probability distribution on the alphabet such that the induced Markov chain is $\mathscr{M}$ [Kouji Yano, Kenji Yasutom].

Does the condition that a graph is the AGW-graph imply faster convergence of M?

Markov Chain Convergence vs Reset Threshold

Let $u \in \Sigma^j$ be a randomly generated word by ϵ-machine $\mathscr{A}$ and $p \in Q$ and $j \geq n - 1$. Then $rt(\mathscr{A}) \leq j$ if either

- $Pr(u \text{ is reset }) > 0$ or
- $\sum_{q \in Q} Pr(p.u \neq q.u) < 1$ or
- $Pr(q_1.u = p; q_2.u = p) \geq Pr(q_1.u = p)Pr(q_2.u = p)$.

Suppose $\mathscr{A}$ has the AGW-graph; Then

- The corresponding Markov chain $\mathscr{M}$ is *mixing*.
- Due to the RCP solution, we can define a synchronizing automaton within the probability distribution on the alphabet such that the induced Markov chain is $\mathscr{M}$ [Kouji Yano, Kenji Yasutom].

Does the condition that a graph is the AGW-graph imply faster convergence of M?

Synchronization of Random Automata
Marne-la-Vallée (LIGM), May, 26

Toward the solution of the Černý Conjecture
Université Paris Diderot (LIAFA), May, 29

Merci!

Synchronization of Random Automata
Marne-la-Vallée (LIGM), May, 26

Toward the solution of the Černý Conjecture
Université Paris Diderot (LIAFA), May, 29

Merci!

Synchronization of Random Automata
Marne-la-Vallée (LIGM), May, 26

Toward the solution of the Černý Conjecture
Université Paris Diderot (LIAFA), May, 29

Merci!