Generic complexity of two algorithmic problems about semigroups

Alexander Rybalov

Sobolev Institute of Mathematics, Omsk

3 june 2021

Generic-case approach

Kapovich, Myasnikov, Schupp and Shpilrain in 2003 developed generic approach to algorithmic problems, which considers an algorithmic problem on "most"of the inputs (i.e., on a generic set) instead of the entire domain and ignores it on the rest of inputs (a negligible set). It turned out, that many famous undecidable problems are easily decidable on most inputs.

Generic sets

Let I be the set of all inputs and I_n be the set of all inputs of size n (sphere of radius n). For a subset $S \subseteq I$ define the following sequence

$$\rho_n(S) = \frac{|S \cap I_n|}{|I_n|}, \ n = 1, 2, 3, \ldots$$

The asymptotic density of set S is the following limit (if it exists)

$$\rho(S) = \lim_{n \to \infty} \rho_n(S).$$

S is called **generic** if $\rho(S) = 1$ and **negligible** if $\rho(S) = 0$.

Generic algorithms

Algorithm $\mathcal{A} : I \to J$ is called **generic** if the set

$$BH(\mathcal{A}) = \{x \in I : \mathcal{A}(x) \uparrow\}$$

is negligible.

Generic algorithms

Algorithm $\mathcal{A} : I \to J$ is called **generic** if the set

$$BH(\mathcal{A}) = \{x \in I : \mathcal{A}(x) \uparrow\}$$

is negligible.

Generic algorithm $\mathcal{A} : I \to J$ **computes** a function $f : I \to J$ if

$$\forall x \in I \ \mathcal{A}(x) \downarrow \Rightarrow f(x) = \mathcal{A}(x).$$

Generic algorithms

Algorithm $\mathcal{A} : I \to J$ is called **generic** if the set

$$BH(\mathcal{A}) = \{x \in I : \mathcal{A}(x) \uparrow\}$$

is negligible.

Generic algorithm $\mathcal{A} : I \to J$ **computes** a function $f : I \to J$ if

$$\forall x \in I \ \mathcal{A}(x) \downarrow \Rightarrow f(x) = \mathcal{A}(x).$$

Generic algorithm $\mathcal{A}$ is polynomial, if

$$\forall x \in I \ \mathcal{A}(x) \downarrow \Rightarrow t_{\mathcal{A}}(x) < p(size(x)).$$

Effective generic algorithms

Algorithm $\mathcal{A} : I \to J \cup \{?\}$ is called **effective generic** if

1. $\forall x \in I \ \mathcal{A}(x) \downarrow$,
2. set $\{x \in I : \mathcal{A}(x) = ?\}$ is negligible.

Effective generic algorithms

Algorithm $\mathcal{A} : I \to J \cup \{?\}$ is called **effective generic** if

1. $\forall x \in I \ \mathcal{A}(x) \downarrow$,
2. set $\{x \in I : \mathcal{A}(x) = ?\}$ is negligible.

Effective generic algorithm $\mathcal{A} : I \to J$ **computes** a function $f : I \to J$, if

$$\forall x \in I \ \mathcal{A}(x) \neq ? \ \Rightarrow f(x) = \mathcal{A}(x).$$

Effective generic algorithms

Algorithm $\mathcal{A} : I \to J \cup \{?\}$ is called **effective generic** if

1. $\forall x \in I \ \mathcal{A}(x) \downarrow$,
2. set $\{x \in I : \mathcal{A}(x) = ?\}$ is negligible.

Effective generic algorithm $\mathcal{A} : I \to J$ **computes** a function $f : I \to J$, if

$$\forall x \in I \ \mathcal{A}(x) \neq ? \ \Rightarrow f(x) = \mathcal{A}(x).$$

Effective generic computability $\Rightarrow$ Generic computability.

Effective generic algorithms

Algorithm $\mathcal{A} : I \to J \cup \{?\}$ is called **effective generic** if

1. $\forall x \in I \ \mathcal{A}(x) \downarrow$,
2. set $\{x \in I : \mathcal{A}(x) = ?\}$ is negligible.

Effective generic algorithm $\mathcal{A} : I \to J$ **computes** a function $f : I \to J$, if

$$\forall x \in I \ \mathcal{A}(x) \neq ? \ \Rightarrow f(x) = \mathcal{A}(x).$$

Effective generic computability $\Rightarrow$ Generic computability.
For polynomial (exponential, etc.) complexity Effective generic computability = Generic computability.

Effective generic algorithms

Algorithm $\mathcal{A} : I \to J \cup \{?\}$ is called **effective generic** if

1. $\forall x \in I \ \mathcal{A}(x) \downarrow$,
2. set $\{x \in I : \mathcal{A}(x) = ?\}$ is negligible.

Effective generic algorithm $\mathcal{A} : I \to J$ **computes** a function $f : I \to J$, if

$$\forall x \in I \ \mathcal{A}(x) \neq ? \ \Rightarrow f(x) = \mathcal{A}(x).$$

Effective generic computability $\Rightarrow$ Generic computability.
For polynomial (exponential, etc.) complexity Effective generic computability = Generic computability.
There are some perverted examples, when Effective generic computability $\neq$ Generic computability.

Part 1: Word problem in semigroups

Word problem in semigroups

Let $\mathfrak{S} = \langle A \mid R \rangle$ be a finitely defined semigroup with generators $A = \{a_1, \ldots, a_m\}$ and relations $R = \{u_1 = v_1, \ldots, u_k = v_k\}$, where $u_i, v_i, i = 1, \ldots, k$ are some words in alphabet A.

Word problem in semigroups

Let $\mathfrak{S} = \langle A \mid R \rangle$ be a finitely defined semigroup with generators $A = \{a_1, \ldots, a_m\}$ and relations $R = \{u_1 = v_1, \ldots, u_k = v_k\}$, where $u_i, v_i, i = 1, \ldots, k$ are some words in alphabet A.

Word problem in $\mathfrak{S}$

For any given words $w_1, w_2 \in A^*$ determine is $w_1 = w_2$ in $\mathfrak{S}$.

Word problem in semigroups

Let $\mathfrak{S} = \langle A \mid R \rangle$ be a finitely defined semigroup with generators $A = \{a_1, \ldots, a_m\}$ and relations $R = \{u_1 = v_1, \ldots, u_k = v_k\}$, where $u_i, v_i, i = 1, \ldots, k$ are some words in alphabet A.

Word problem in $\mathfrak{S}$

For any given words $w_1, w_2 \in A^*$ determine is $w_1 = w_2$ in $\mathfrak{S}$.

Theorem (Markov, Post, 1947)

There is a finitely defined semigroup with undecidable word problem.

Word problem in semigroups

Let $\mathfrak{S} = \langle A \mid R \rangle$ be a finitely defined semigroup with generators $A = \{a_1, \ldots, a_m\}$ and relations $R = \{u_1 = v_1, \ldots, u_k = v_k\}$, where $u_i, v_i, i = 1, \ldots, k$ are some words in alphabet A.

Word problem in $\mathfrak{S}$

For any given words $w_1, w_2 \in A^*$ determine is $w_1 = w_2$ in $\mathfrak{S}$.

Theorem (Markov, Post, 1947)

There is a finitely defined semigroup with undecidable word problem.

Theorem (Tseitin, 1958)

Semigroup $\mathfrak{T} = \langle a, b, c, d, e \mid ac = ca,\ ad = da,\ bc = cb,\ bd = db,\ ce = eca,\ de = edb,\ cca = ccae \rangle$ has undecidable word problem.

Word problem in semigroups

Let $\mathfrak{S} = \langle A \mid R \rangle$ be a finitely defined semigroup with generators $A = \{a_1, \ldots, a_m\}$ and relations $R = \{u_1 = v_1, \ldots, u_k = v_k\}$, where $u_i, v_i, i = 1, \ldots, k$ are some words in alphabet A.

Word problem in $\mathfrak{S}$

For any given words $w_1, w_2 \in A^*$ determine is $w_1 = w_2$ in $\mathfrak{S}$.

Theorem (Markov, Post, 1947)

There is a finitely defined semigroup with undecidable word problem.

Theorem (Tseitin, 1958)

Semigroup $\mathfrak{T} = \langle a, b, c, d, e \mid ac = ca,\ ad = da,\ bc = cb,\ bd = db,\ ce = eca,\ de = edb,\ cca = ccae \rangle$ has undecidable word problem.

Later Makanin and Matiyasevich constructed semigroups with lower number of defining relations

Generic-case approach to word problem for groups

Kapovich, Myasnikov, Schupp and Shpilrain in 2003 constructed a polynomial generic algorithm for the word problem in a large class finitely defined groups. This class contains many famous groups with undecidable word problem: Novikov group, Boone group, Borisov group and etc. The generic algorithm uses an idea of some approximation of a "complicated"group with undecidable word problem by a "simple"group with decidable word problem. Justification of this algorithm involves the beautiful theory of random walks on Cayley graphs, developed by Woess and Bartholdi.

Generic-case approach to word problem for semigroups

Myasnikov, Ushakov and Won in 2008 suggested a simple generic algorithm for the word problem in finitely defined semigroups, based on the notion of balanced presentation.

Generic-case approach to word problem for semigroups

Myasnikov, Ushakov and Won in 2008 suggested a simple generic algorithm for the word problem in finitely defined semigroups, based on the notion of balanced presentation.

Definition

A semigroup is called **balanced on letter** a, if for every relation of the semigroup $u = v$, the letter a is included in u and in v the same number of times.

Generic-case approach to word problem for semigroups

Myasnikov, Ushakov and Won in 2008 suggested a simple generic algorithm for the word problem in finitely defined semigroups, based on the notion of balanced presentation.

Definition

A semigroup is called **balanced on letter** a, if for every relation of the semigroup $u = v$, the letter a is included in u and in v the same number of times.

Words representing equal elements in this semigroup have the same number of letters a. It can be shown by quite elementary combinatorial computations (no any random walks or something like that) that the set of all these pairs of words is negligible.

Generic-case approach to word problem for semigroups

Algorithm

So the generic algorithm just compares the numbers of occurrences of letter a in the input words, outputs "NO if they are different and outputs "I don't know if they are equal.

Generic-case approach to word problem for semigroups

Algorithm

So the generic algorithm just compares the numbers of occurrences of letter a in the input words, outputs "NO if they are different and outputs "I don't know if they are equal.

This generic algorithm works for

- Tseitin semigroup,
- Makanin semigroup,
- its modification works for Matiyasevich semigroup.

Generic-case approach to word problem for semigroups

Algorithm

So the generic algorithm just compares the numbers of occurrences of letter a in the input words, outputs "NO if they are different and outputs "I don't know if they are equal.

This generic algorithm works for

- Tseitin semigroup,
- Makanin semigroup,
- its modification works for Matiyasevich semigroup.

But it does not work for any semigroup with one relation!

Adjan problem

Is the word problem decidable in any finitely defined semigroup with one relation?

How to improve Myasnikov-Ushakov-Won algorithm?

Let $\mathfrak{S} = \langle A \mid R \rangle$ be a finitely defined semigroup with generators $A = \{a_1, \ldots, a_m\}$ and relations R.

How to improve Myasnikov-Ushakov-Won algorithm?

Let $\mathfrak{S} = \langle A \mid R \rangle$ be a finitely defined semigroup with generators $A = \{a_1, \ldots, a_m\}$ and relations R.

Define commutative monoid $\mathfrak{S}'$ with cancellations by adding to R commutative relations $a_i a_j = a_j a_i$, $1 \leq i, j \leq m$.

How to improve Myasnikov-Ushakov-Won algorithm?

Let $\mathfrak{S} = \langle A \mid R \rangle$ be a finitely defined semigroup with generators $A = \{a_1, \ldots, a_m\}$ and relations R.

Define commutative monoid $\mathfrak{S}'$ with cancellations by adding to R commutative relations $a_i a_j = a_j a_i$, $1 \leq i, j \leq m$.

Cancellation property: $\forall x, y, z \ (xy = xz) \Rightarrow (y = z)$.

Theorem

If $\mathfrak{S}'$ is infinite, then word problem in $\mathfrak{S}$ is generically decidable in polynomial time.

How to improve Myasnikov-Ushakov-Won algorithm?

Let $\mathfrak{S} = \langle A \mid R \rangle$ be a finitely defined semigroup with generators $A = \{a_1, \ldots, a_m\}$ and relations R.

Define commutative monoid $\mathfrak{S}'$ with cancellations by adding to R commutative relations $a_i a_j = a_j a_i$, $1 \leq i, j \leq m$.

Cancellation property: $\forall x, y, z \ (xy = xz) \Rightarrow (y = z)$.

Theorem

If $\mathfrak{S}'$ is infinite, then word problem in $\mathfrak{S}$ is generically decidable in polynomial time.

Monoid $\mathfrak{S}'$ can be embedded in Abelian group G with generators $a_1, \ldots, a_m$. So we can check is $w_1 = w_2$ in G and output NO, if $w_1 \neq w_2$ in G, because then $w_1 \neq w_2$ in $\mathfrak{S}$. And output «?», if $w_1 = w_2$ in G.

Key Lemma: Sketch of the proof

Lemma

$Eq(\mathfrak{S}') = \{(w_1, w_2) \in A^* \times A^* \ : \ w_1 = w_2 \text{ in } \mathfrak{S}'\}$ is negligible.

Key Lemma: Sketch of the proof

Lemma

$Eq(\mathfrak{S}') = \{(w_1, w_2) \in A^* \times A^* \ : \ w_1 = w_2 \text{ in } \mathfrak{S}'\}$ is negligible.

1. $\mathfrak{S}'(q)$ is monoid $\mathfrak{S}'$ with additional relations $a_i^q = 1$, $i = 1, \ldots, m$. Obviously $Eq(\mathfrak{S}') \subseteq Eq(\mathfrak{S}'(q))$.

Key Lemma: Sketch of the proof

Lemma

$Eq(\mathfrak{S}') = \{(w_1, w_2) \in A^* \times A^* \ : \ w_1 = w_2 \text{ in } \mathfrak{S}'\}$ is negligible.

1. $\mathfrak{S}'(q)$ is monoid $\mathfrak{S}'$ with additional relations $a_i^q = 1$, $i = 1, \ldots, m$. Obviously $Eq(\mathfrak{S}') \subseteq Eq(\mathfrak{S}'(q))$.
2. $|\mathfrak{S}'(q)| \to \infty$ if $q \to \infty$.

Key Lemma: Sketch of the proof

Lemma

$Eq(\mathfrak{S}') = \{(w_1, w_2) \in A^* \times A^* \ : \ w_1 = w_2 \text{ in } \mathfrak{S}'\}$ is negligible.

1. $\mathfrak{S}'(q)$ is monoid $\mathfrak{S}'$ with additional relations $a_i^q = 1$, $i = 1, \ldots, m$. Obviously $Eq(\mathfrak{S}') \subseteq Eq(\mathfrak{S}'(q))$.
2. $|\mathfrak{S}'(q)| \to \infty$ if $q \to \infty$.
3. Markov chain $MC(q)$ with states $s_1, \ldots, s_r$ – elements of $\mathfrak{S}'(q)$. Transfer matrix of $MC(q)$ is $P = ||p_{ij}||$, where $p_{ij} = \frac{1}{m}$ if $s_i a_l = s_j$ for some $a_l \in A$, and $p_{ij} = 0$ otherwise.

Key Lemma: Sketch of the proof

Lemma

$Eq(\mathfrak{S}') = \{(w_1, w_2) \in A^* \times A^* \ : \ w_1 = w_2 \text{ in } \mathfrak{S}'\}$ is negligible.

1. $\mathfrak{S}'(q)$ is monoid $\mathfrak{S}'$ with additional relations $a_i^q = 1$, $i = 1, \ldots, m$. Obviously $Eq(\mathfrak{S}') \subseteq Eq(\mathfrak{S}'(q))$.
2. $|\mathfrak{S}'(q)| \to \infty$ if $q \to \infty$.
3. Markov chain $MC(q)$ with states $s_1, \ldots, s_r$ – elements of $\mathfrak{S}'(q)$. Transfer matrix of $MC(q)$ is $P = ||p_{ij}||$, where $p_{ij} = \frac{1}{m}$ if $s_i a_l = s_j$ for some $a_l \in A$, and $p_{ij} = 0$ otherwise.
4. Matrix P is doubly stochastic (sum of elements in any row and column is 1). This follows from cancellation property.

Key Lemma: Sketch of the proof

Lemma

$Eq(\mathfrak{S}') = \{(w_1, w_2) \in A^* \times A^* \ : \ w_1 = w_2 \text{ in } \mathfrak{S}'\}$ is negligible.

1. $\mathfrak{S}'(q)$ is monoid $\mathfrak{S}'$ with additional relations $a_i^q = 1$, $i = 1, \ldots, m$. Obviously $Eq(\mathfrak{S}') \subseteq Eq(\mathfrak{S}'(q))$.
2. $|\mathfrak{S}'(q)| \to \infty$ if $q \to \infty$.
3. Markov chain $MC(q)$ with states $s_1, \ldots, s_r$ – elements of $\mathfrak{S}'(q)$. Transfer matrix of $MC(q)$ is $P = ||p_{ij}||$, where $p_{ij} = \frac{1}{m}$ if $s_i a_l = s_j$ for some $a_l \in A$, and $p_{ij} = 0$ otherwise.
4. Matrix P is doubly stochastic (sum of elements in any row and column is 1). This follows from cancellation property.
5. Doubly stochastic $P \Rightarrow$ uniform ergodic distribution of $MC(q)$ $\Rightarrow$ random word w is equal s_i in $\mathfrak{S}'(q)$ with probability $\frac{1}{|\mathfrak{S}'(q)|}$.

Key Lemma: Sketch of the proof

Lemma

$Eq(\mathfrak{S}') = \{(w_1, w_2) \in A^* \times A^* \ : \ w_1 = w_2 \text{ in } \mathfrak{S}'\}$ is negligible.

1. $\mathfrak{S}'(q)$ is monoid $\mathfrak{S}'$ with additional relations $a_i^q = 1$, $i = 1, \ldots, m$. Obviously $Eq(\mathfrak{S}') \subseteq Eq(\mathfrak{S}'(q))$.
2. $|\mathfrak{S}'(q)| \to \infty$ if $q \to \infty$.
3. Markov chain $MC(q)$ with states $s_1, \ldots, s_r$ – elements of $\mathfrak{S}'(q)$. Transfer matrix of $MC(q)$ is $P = ||p_{ij}||$, where $p_{ij} = \frac{1}{m}$ if $s_i a_l = s_j$ for some $a_l \in A$, and $p_{ij} = 0$ otherwise.
4. Matrix P is doubly stochastic (sum of elements in any row and column is 1). This follows from cancellation property.
5. Doubly stochastic $P \Rightarrow$ uniform ergodic distribution of $MC(q)$ $\Rightarrow$ random word w is equal s_i in $\mathfrak{S}'(q)$ with probability $\frac{1}{|\mathfrak{S}'(q)|}$.
6. $\Rightarrow \rho_n(Eq(\mathfrak{S}'(q))) < O(\frac{1}{|\mathfrak{S}'(q)|})$.

Applications

For every letter a_i in alphabet A and for all pairs $(w_1, w_2) \in A^* \times A^*$ define $d_i(w_1, w_2)$ as the number of letter a_i in w_1 minus the number of letter a_i in w_2. Define now for all pairs $(w_1, w_2) \in A^* \times A^*$ the following vector

$$d(w_1, w_2) = \big(d_1(w_1, w_2), \ldots, d_m(w_1, w_2)\big).$$

Applications

For every letter a_i in alphabet A and for all pairs $(w_1, w_2) \in A^* \times A^*$ define $d_i(w_1, w_2)$ as the number of letter a_i in w_1 minus the number of letter a_i in w_2. Define now for all pairs $(w_1, w_2) \in A^* \times A^*$ the following vector

$$d(w_1, w_2) = \big(d_1(w_1, w_2), \ldots, d_m(w_1, w_2)\big).$$

For the semigroup $\mathfrak{S}$ denote by $V_{\mathfrak{S}}$ the subspace of vector space $\mathbb{Q}^m$, generated by vectors $d(v_i, u_i), i = 1, \ldots, k$ for all relations $u_i = v_i, i = 1, \ldots, k$.

Applications

For every letter a_i in alphabet A and for all pairs $(w_1, w_2) \in A^* \times A^*$ define $d_i(w_1, w_2)$ as the number of letter a_i in w_1 minus the number of letter a_i in w_2. Define now for all pairs $(w_1, w_2) \in A^* \times A^*$ the following vector

$$d(w_1, w_2) = \big(d_1(w_1, w_2), \ldots, d_m(w_1, w_2)\big).$$

For the semigroup $\mathfrak{S}$ denote by $V_{\mathfrak{S}}$ the subspace of vector space $\mathbb{Q}^m$, generated by vectors $d(v_i, u_i), i = 1, \ldots, k$ for all relations $u_i = v_i, i = 1, \ldots, k$.

Corollary

Let $\mathfrak{S} = \langle A \mid R \rangle$ be a finitely defined semigroup such that $V_{\mathfrak{S}}$ has dimension less than $m = |A|$. Then the word problem in $\mathfrak{S}$ is generically decidable in polynomial time.

Applications

For every letter a_i in alphabet A and for all pairs $(w_1, w_2) \in A^* \times A^*$ define $d_i(w_1, w_2)$ as the number of letter a_i in w_1 minus the number of letter a_i in w_2. Define now for all pairs $(w_1, w_2) \in A^* \times A^*$ the following vector

$$d(w_1, w_2) = \big(d_1(w_1, w_2), \ldots, d_m(w_1, w_2)\big).$$

For the semigroup $\mathfrak{S}$ denote by $V_{\mathfrak{S}}$ the subspace of vector space $\mathbb{Q}^m$, generated by vectors $d(v_i, u_i), i = 1, \ldots, k$ for all relations $u_i = v_i, i = 1, \ldots, k$.

Corollary

Let $\mathfrak{S} = \langle A \mid R \rangle$ be a finitely defined semigroup such that $V_{\mathfrak{S}}$ has dimension less than $m = |A|$. Then the word problem in $\mathfrak{S}$ is generically decidable in polynomial time.

Corollary

Then the word problem in finitely defined semigroup with one relation is generically decidable in polynomial time.

Application to semigroup with generically undecidable word problem

Let $\mathfrak{S} = \langle A \mid R \rangle$ be a semigroup with the word problem undecidable in the classical sense. Suppose $A = \{a_1, \ldots, a_m\}$ and $x \notin A$. The following semigroup

$$\mathfrak{S}_x = \langle A, x | R, x = xa_1, \ldots, x = xa_m, x = xx \rangle$$

has generically undecidable word problem (Rybalov-Myasnikov).

Application to semigroup with generically undecidable word problem

Let $\mathfrak{S} = \langle A \mid R \rangle$ be a semigroup with the word problem undecidable in the classical sense. Suppose $A = \{a_1, \ldots, a_m\}$ and $x \notin A$. The following semigroup

$$\mathfrak{S}_x = \langle A, x | R, x = xa_1, \ldots, x = xa_m, x = xx \rangle$$

has generically undecidable word problem (Rybalov-Myasnikov).

Monoid $\mathfrak{S}'_x$ is trivial!

Application to semigroup with generically undecidable word problem

Let $\mathfrak{S} = \langle A \mid R \rangle$ be a semigroup with the word problem undecidable in the classical sense. Suppose $A = \{a_1, \ldots, a_m\}$ and $x \notin A$. The following semigroup

$$\mathfrak{S}_x = \langle A, x | R, x = xa_1, \ldots, x = xa_m, x = xx \rangle$$

has generically undecidable word problem (Rybalov-Myasnikov).

Monoid $\mathfrak{S}'_x$ is trivial!

$$\begin{aligned} d(x, xa_1) &= (-1, 0, \ldots, 0, 0), \\ d(x, xa_2) &= (0, -1, \ldots, 0, 0), \\ &\ldots \\ d(x, xa_m) &= (0, 0, \ldots, -1, 0), \\ d(x, xx) &= (0, 0, \ldots, 0, -1). \end{aligned}$$

Part 2: Finite Semigroups Isomorphism problem

Graph Isomorphism problem

Problem

- INPUT: Finite graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ of size n, represented by adjacency matrices.
- OUTPUT: YES, if G_1 and G_2 are isomorphic, i.e. there is a bijection $\varphi : V_1 \to V_2$ such that for any v_1, v_2 $(v_1, v_2) \in E_1 \Leftrightarrow (\varphi(v_1), \varphi(v_2)) \in E_2$,
  NO, otherwise.

Graph Isomorphism problem

Problem

- INPUT: Finite graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ of size n, represented by adjacency matrices.
- OUTPUT: YES, if G_1 and G_2 are isomorphic, i.e. there is a bijection $\varphi : V_1 \to V_2$ such that for any v_1, v_2 $(v_1, v_2) \in E_1 \Leftrightarrow (\varphi(v_1), \varphi(v_2)) \in E_2$,
  NO, otherwise.

- Best algorithm: subexponential algorithm of Babai (2015).

Graph Isomorphism problem

Problem

- INPUT: Finite graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ of size n, represented by adjacency matrices.
- OUTPUT: YES, if G_1 and G_2 are isomorphic, i.e. there is a bijection $\varphi : V_1 \to V_2$ such that for any v_1, v_2 $(v_1, v_2) \in E_1 \Leftrightarrow (\varphi(v_1), \varphi(v_2)) \in E_2$,
  NO, otherwise.

- Best algorithm: subexponential algorithm of Babai (2015).
- If GI is NP-complete, then polynomial hierarchy collapses at 2nd level (Schoning, 1987) – something like P=NP.

Graph Isomorphism problem

Problem

- INPUT: Finite graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ of size n, represented by adjacency matrices.
- OUTPUT: YES, if G_1 and G_2 are isomorphic, i.e. there is a bijection $\varphi : V_1 \to V_2$ such that for any v_1, v_2 $(v_1, v_2) \in E_1 \Leftrightarrow (\varphi(v_1), \varphi(v_2)) \in E_2$,
  NO, otherwise.

- Best algorithm: subexponential algorithm of Babai (2015).
- If GI is NP-complete, then polynomial hierarchy collapses at 2nd level (Schoning, 1987) – something like P=NP.
- Hypothesis: GI is not in P, not NP-complete.

Generic algorithms for GI

1. Babai, Erdos and Selkow (1980): first polynomial generic algorithm for GI for $p = 1/2$. Here p is the probability that two vertices are connected by an edge.

Generic algorithms for GI

1. Babai, Erdos and Selkow (1980): first polynomial generic algorithm for GI for $p = 1/2$. Here p is the probability that two vertices are connected by an edge.
2. Ballobas (1982): polynomial algorithm for almost all sparse graphs with $p(n) \in [\frac{C_1 \ln n}{n}, \frac{C_2}{n^{11/12}}]$.

Generic algorithms for GI

1. Babai, Erdos and Selkow (1980): first polynomial generic algorithm for GI for $p = 1/2$. Here p is the probability that two vertices are connected by an edge.
2. Ballobas (1982): polynomial algorithm for almost all sparse graphs with $p(n) \in [\frac{C_1 \ln n}{n}, \frac{C_2}{n^{11/12}}]$.
3. G.A.Noskov (2016): generic polynomial algorithm for $pn \to \infty$.

Canonical labeling

Babai-Erdos-Selkow and Ballobas algorithms try to produce a canonical labeling for graphs G_1, G_2:

- $C(G) = \{c(v_1) \in \mathbb{N}, \ldots, c(v_n) \in \mathbb{N}\}$, where $v_1, \ldots, v_n$ – vertices of G,

Canonical labeling

Babai-Erdos-Selkow and Ballobas algorithms try to produce a canonical labeling for graphs G_1, G_2:

- $C(G) = \{c(v_1) \in \mathbb{N}, \ldots, c(v_n) \in \mathbb{N}\}$, where $v_1, \ldots, v_n$ – vertices of G,
- $G_1 \cong G_2 \Leftrightarrow C(G_1) = C(G_2)$,

Canonical labeling

Babai-Erdos-Selkow and Ballobas algorithms try to produce a canonical labeling for graphs G_1, G_2:

- $C(G) = \{c(v_1) \in \mathbb{N}, \ldots, c(v_n) \in \mathbb{N}\}$, where $v_1, \ldots, v_n$ – vertices of G,
- $G_1 \cong G_2 \Leftrightarrow C(G_1) = C(G_2)$,
- isomorphism maps v_i to $w_j \Leftrightarrow c(v_i) = c(w_j)$,

Canonical labeling

Babai-Erdos-Selkow and Ballobas algorithms try to produce a canonical labeling for graphs G_1, G_2:

- $C(G) = \{c(v_1) \in \mathbb{N}, \ldots, c(v_n) \in \mathbb{N}\}$, where $v_1, \ldots, v_n$ – vertices of G,
- $G_1 \cong G_2 \Leftrightarrow C(G_1) = C(G_2)$,
- isomorphism maps v_i to $w_j \Leftrightarrow c(v_i) = c(w_j)$,
- possible isomorphism is unique!

Isomorphism problem of finite semigroups

Problem

- INPUT: Finite semigroups S_1 and S_2 of order n, represented by multiplication tables.
- OUTPUT: YES, if S_1 and S_2 are isomorphic, i.e. there is a bijiction $\varphi : S_1 \to S_2$ such that for all $a_1, a_2 \in S_1$ $\varphi(a_1)\varphi(a_2) = \varphi(a_1 a_2)$,
  NO, otherwise.

Isomorphism problem of finite semigroups

Problem

- INPUT: Finite semigroups S_1 and S_2 of order n, represented by multiplication tables.
- OUTPUT: YES, if S_1 and S_2 are isomorphic, i.e. there is a bijiction $\varphi : S_1 \to S_2$ such that for all $a_1, a_2 \in S_1$ $\varphi(a_1)\varphi(a_2) = \varphi(a_1 a_2)$,
  NO, otherwise.

Zemlyachenko, Korneenko and Tyshkevich (1982): Graph isomorphism problem is polynomially reducible to Semigroup isomorphism problem.

Isomorphism problem of finite semigroups

Problem

- INPUT: Finite semigroups S_1 and S_2 of order n, represented by multiplication tables.
- OUTPUT: YES, if S_1 and S_2 are isomorphic, i.e. there is a bijiction $\varphi : S_1 \to S_2$ such that for all $a_1, a_2 \in S_1$ $\varphi(a_1)\varphi(a_2) = \varphi(a_1 a_2)$,
  NO, otherwise.

Zemlyachenko, Korneenko and Tyshkevich (1982): Graph isomorphism problem is polynomially reducible to Semigroup isomorphism problem.

Therefore Semigroup isomorphism problem is not easier!

Generic complexity of the Semigroup isomorphism problem

Theorem

Semigroup isomorphism problem is generically decidable in polynomial time.

Kleitman-Rothschild-Spencer result

Theorem (Kleitman-Rothschild-Spencer)

Almost all finite semigroups are 3-nilpotent.

Kleitman-Rothschild-Spencer result

Theorem (Kleitman-Rothschild-Spencer)

Almost all finite semigroups are 3-nilpotent.

Consider semigroups with elements from sets $\{1, 2, \ldots, n\}$. Every such semigroup S has a type $H(A, \psi, z)$, where $A \subset \{1, 2, \ldots, n\}$, $\psi : A \times A \to \bar{A}$ and $z \in \bar{A}$, and multiplication in S is defined

$$xy = \begin{cases} \psi(x, y), & \text{if } x, y \in A, \\ z, & \text{otherwise.} \end{cases}$$

Kleitman-Rothschild-Spencer result

Theorem (Kleitman-Rothschild-Spencer)

Almost all finite semigroups are 3-nilpotent.

Consider semigroups with elements from sets $\{1, 2, \ldots, n\}$. Every such semigroup S has a type $H(A, \psi, z)$, where $A \subset \{1, 2, \ldots, n\}$, $\psi : A \times A \to \bar{A}$ and $z \in \bar{A}$, and multiplication in S is defined

$$xy = \begin{cases} \psi(x, y), & \text{if } x, y \in A, \\ z, & \text{otherwise.} \end{cases}$$

Representation $S = H(A, \psi, z)$ is **minimal**, if there is no set A' such that $S = H(A', \psi', z')$ and $|A'| < |A|$.

Kleitman-Rothschild-Spencer result

Theorem (Kleitman-Rothschild-Spencer)

Almost all finite semigroups are 3-nilpotent.

Consider semigroups with elements from sets $\{1, 2, \ldots, n\}$. Every such semigroup S has a type $H(A, \psi, z)$, where $A \subset \{1, 2, \ldots, n\}$, $\psi : A \times A \to \bar{A}$ and $z \in \bar{A}$, and multiplication in S is defined

$$xy = \begin{cases} \psi(x, y), & \text{if } x, y \in A, \\ z, & \text{otherwise.} \end{cases}$$

Representation $S = H(A, \psi, z)$ is **minimal**, if there is no set A' such that $S = H(A', \psi', z')$ and $|A'| < |A|$.

Theorem (Kleitman-Rothschild-Spencer)

Almost all finite semigroups of order n (for large enough n) has type $H(A, \psi, z)$ (minimal) with $|A| \in (t_0 - 3, t_0 + 3)$, where $t_0 = n - \frac{n}{2 \ln n}$.

Isomorphic semigroups of type $H(A, \psi, z)$

Lemma

Let $S_1 = H(A_1, \psi_1, z_1)$ and $S_2 = H(A_2, \psi_2, z_2)$ are isomorphic with minimal representations and φ is an isomorphism between them. Then

1. $\varphi(z_1) = z_2$,
2. restriction φ on A_1 is a bijection τ between A_1 and A_2, whence $|A_1| = |A_2|$,
3. restriction φ on $S_1 \setminus (A_1 \cup \{z_1\})$ is a bijection π between $S_1 \setminus (A_1 \cup \{z_1\})$ and $S_2 \setminus (A_2 \cup \{z_2\})$,
4. bijection π can be computed in polynomial time by bijection τ.

Finite semigroups and graphs

For semigroup $S = H(A, \psi, z)$ define graph $G(S)$:

- Vertices of $G(S)$ is the set A.
- For $a_1, a_2 \in A$ $G(S)$ has edge $(a_1, a_2) \Leftrightarrow a_1 a_2 = z$.

Finite semigroups and graphs

For semigroup $S = H(A, \psi, z)$ define graph $G(S)$:

- Vertices of $G(S)$ is the set A.
- For $a_1, a_2 \in A$ $G(S)$ has edge $(a_1, a_2) \Leftrightarrow a_1 a_2 = z$.

Lemma

Let $S_1 = H(A_1, \psi_1, z_1)$ and $S_2 = H(A_2, \psi_2, z_2)$ be minimal and isomorphic. Then $G(S_1)$ and $G(S_2)$ are isomorphic.

Idea of generic algorithm

1. Input: semigroups S_1 and S_2 of order n.

Idea of generic algorithm

1. Input: semigroups S_1 and S_2 of order n.
2. Try to construct minimal representations $S_1 = H(A_1, \psi_1, z_1)$ and $S_2 = H(A_2, \psi_2, z_2)$. If their exist, it can be done in polynomial time. If no, output «?». Failure is negligible by Kleitman-Rothschild-Spencer result.

Idea of generic algorithm

1. Input: semigroups S_1 and S_2 of order n.
2. Try to construct minimal representations $S_1 = H(A_1, \psi_1, z_1)$ and $S_2 = H(A_2, \psi_2, z_2)$. If their exist, it can be done in polynomial time. If no, output «?». Failure is negligible by Kleitman-Rothschild-Spencer result.
3. If $|A_1| \neq |A_2|$, output NO.

Idea of generic algorithm

1. Input: semigroups S_1 and S_2 of order n.
2. Try to construct minimal representations $S_1 = H(A_1, \psi_1, z_1)$ and $S_2 = H(A_2, \psi_2, z_2)$. If their exist, it can be done in polynomial time. If no, output «?». Failure is negligible by Kleitman-Rothschild-Spencer result.
3. If $|A_1| \neq |A_2|$, output NO.
4. Now $k = |A_1|$. If $k \notin (n - \frac{n}{2 \ln n} - 3, n - \frac{n}{2 \ln n} + 3)$, output «?». Failure is negligible by Kleitman-Rothschild-Spencer result.

Idea of generic algorithm

1. Input: semigroups S_1 and S_2 of order n.
2. Try to construct minimal representations $S_1 = H(A_1, \psi_1, z_1)$ and $S_2 = H(A_2, \psi_2, z_2)$. If their exist, it can be done in polynomial time. If no, output «?». Failure is negligible by Kleitman-Rothschild-Spencer result.
3. If $|A_1| \neq |A_2|$, output NO.
4. Now $k = |A_1|$. If $k \notin (n - \frac{n}{2 \ln n} - 3, n - \frac{n}{2 \ln n} + 3)$, output «?». Failure is negligible by Kleitman-Rothschild-Spencer result.
5. Construct $G(S_1)$ and $G(S_2)$. Run Bollobas algorithm to check and find isomorphism between $G(S_1)$ and $G(S_2)$.

Idea of generic algorithm

1. Input: semigroups S_1 and S_2 of order n.
2. Try to construct minimal representations $S_1 = H(A_1, \psi_1, z_1)$ and $S_2 = H(A_2, \psi_2, z_2)$. If their exist, it can be done in polynomial time. If no, output «?». Failure is negligible by Kleitman-Rothschild-Spencer result.
3. If $|A_1| \neq |A_2|$, output NO.
4. Now $k = |A_1|$. If $k \notin (n - \frac{n}{2 \ln n} - 3, n - \frac{n}{2 \ln n} + 3)$, output «?». Failure is negligible by Kleitman-Rothschild-Spencer result.
5. Construct $G(S_1)$ and $G(S_2)$. Run Bollobas algorithm to check and find isomorphism between $G(S_1)$ and $G(S_2)$.
6. If Bollobas algorithm was failure, output «?».

Idea of generic algorithm

1. Input: semigroups S_1 and S_2 of order n.
2. Try to construct minimal representations $S_1 = H(A_1, \psi_1, z_1)$ and $S_2 = H(A_2, \psi_2, z_2)$. If their exist, it can be done in polynomial time. If no, output «?». Failure is negligible by Kleitman-Rothschild-Spencer result.
3. If $|A_1| \neq |A_2|$, output NO.
4. Now $k = |A_1|$. If $k \notin (n - \frac{n}{2 \ln n} - 3, n - \frac{n}{2 \ln n} + 3)$, output «?». Failure is negligible by Kleitman-Rothschild-Spencer result.
5. Construct $G(S_1)$ and $G(S_2)$. Run Bollobas algorithm to check and find isomorphism between $G(S_1)$ and $G(S_2)$.
6. If Bollobas algorithm was failure, output «?».
7. If Bollobas algorithm output NO, output NO.

Idea of generic algorithm

1. Input: semigroups S_1 and S_2 of order n.
2. Try to construct minimal representations $S_1 = H(A_1, \psi_1, z_1)$ and $S_2 = H(A_2, \psi_2, z_2)$. If their exist, it can be done in polynomial time. If no, output «?». Failure is negligible by Kleitman-Rothschild-Spencer result.
3. If $|A_1| \neq |A_2|$, output NO.
4. Now $k = |A_1|$. If $k \notin (n - \frac{n}{2 \ln n} - 3, n - \frac{n}{2 \ln n} + 3)$, output «?». Failure is negligible by Kleitman-Rothschild-Spencer result.
5. Construct $G(S_1)$ and $G(S_2)$. Run Bollobas algorithm to check and find isomorphism between $G(S_1)$ and $G(S_2)$.
6. If Bollobas algorithm was failure, output «?».
7. If Bollobas algorithm output NO, output NO.
8. If Bollobas algorithm output an isomorphism between $G(S_1)$ and $G(S_2)$, reconstruct an check isomorphism between S_1 and S_2.

Why the Bollobas algorithm will works almost always?

- Probability of edge in $G(S)$ is $p = \frac{1}{n-k}$.

Why the Bollobas algorithm will works almost always?

- Probability of edge in $G(S)$ is $p = \frac{1}{n-k}$.
- $k \in (n - \frac{n}{2\ln n} - 3, n - \frac{n}{2\ln n} + 3) \Rightarrow \frac{1{,}97 \cdot \ln k}{k} < p < \frac{0{,}5}{k^{11/12}}$.

Why the Bollobas algorithm will works almost always?

- Probability of edge in $G(S)$ is $p = \frac{1}{n-k}$.
- $k \in (n - \frac{n}{2\ln n} - 3, n - \frac{n}{2\ln n} + 3) \Rightarrow \frac{1{,}97 \cdot \ln k}{k} < p < \frac{0{,}5}{k^{11/12}}$.
- Bollobas algorithm works for almost all graphs with

$$\frac{C_1 \ln n}{n} \leq p(n) \leq \frac{C_2}{n^{11/12}},$$

where $C_1 > 1$, $0 < C_2 < 1$ are any fixed constants.

The end

Thank you for your attention!