Sublinear time algorithms in (semi)groups

Vladimir Shpilrain
The City College of New York

Omsk Algebraic Webinar
February 18, 2021

Appetizer

Divisibility of a decimal integer by 2, 5, or 10.

Appetizer

Divisibility of a decimal integer by 2, 5, or 10.

Read just the last digit, not the whole input.

Appetizer

Divisibility of a decimal integer by 2, 5, or 10.

Read just the last digit, not the whole input.

This is a rare instance where both “yes” and “no” answers can be given in sublinear (in fact, in constant) time.

Las Vegas algorithms

A *Las Vegas algorithm* is a randomized algorithm that never gives an incorrect results; that is, it either produces the correct result or it informs about the failure.

In contrast, a *Monte Carlo algorithm* is a randomized algorithm whose output may be incorrect with some (typically small) probability.

Las Vegas algorithms

A *Las Vegas algorithm* is a randomized algorithm that never gives an incorrect results; that is, it either produces the correct result or it informs about the failure.

In contrast, a *Monte Carlo algorithm* is a randomized algorithm whose output may be incorrect with some (typically small) probability.

Las Vegas algorithms are more useful in the sense that they can improve time complexity of "honest", "hard-working", algorithms that always give a correct answer but are slow. Specifically, by running a fast Las Vegas algorithm and a slow "honest" algorithm in parallel, one often gets an algorithm that always terminates with a correct answer and whose average-case complexity is somewhere in between.

Las Vegas algorithms

A *Las Vegas algorithm* is a randomized algorithm that never gives an incorrect results; that is, it either produces the correct result or it informs about the failure.

In contrast, a *Monte Carlo algorithm* is a randomized algorithm whose output may be incorrect with some (typically small) probability.

Las Vegas algorithms are more useful in the sense that they can improve time complexity of "honest", "hard-working", algorithms that always give a correct answer but are slow. Specifically, by running a fast Las Vegas algorithm and a slow "honest" algorithm in parallel, one often gets an algorithm that always terminates with a correct answer and whose average-case complexity is somewhere in between.

In the context of group-theoretic problems, this was well illustrated in [I. Kapovich, A. G. Myasnikov, P. Schupp, V. Shpilrain, *Generic-case complexity, decision problems in group theory and random walks*, J. Algebra **264** (2003), 665–694] and [I. Kapovich, A. G. Myasnikov, P. Schupp, V. Shpilrain, *Average-case complexity and decision problems in group theory*, Adv. Math. **190** (2005), 343–359].

From graph theory: property testing

O. Goldreich, S. Goldwasser, D. Ron, *Property Testing and its Connection to Learning and Approximation*, JACM **45** (1998), 653–750.

From graph theory: property testing

O. Goldreich, S. Goldwasser, D. Ron, *Property Testing and its Connection to Learning and Approximation*, JACM **45** (1998), 653–750.

In particular, they considered the property of k-colorability of graphs. This property is NP-complete to determine precisely but it is efficiently "testable". For example, if a graph G has a subgraph isomorphic to a complete graph K_n with $n > k$, then G is not k-colorable.

A theorem of Sanov

Denote $A(k) = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}$, $B(k) = \begin{pmatrix} 1 & 0 \\ k & 1 \end{pmatrix}$.

Theorem

The subgroup of $SL_2(\mathbb{Z})$ generated by $A(2)$ and $B(2)$ consists of all matrices of the form $\begin{pmatrix} 1+4n_1 & 2n_2 \\ 2n_3 & 1+4n_4 \end{pmatrix}$ with determinant 1, where all n_i are arbitrary integers.

A theorem of Sanov

Denote $A(k) = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}, \; B(k) = \begin{pmatrix} 1 & 0 \\ k & 1 \end{pmatrix}$.

Theorem

The subgroup of $SL_2(\mathbb{Z})$ generated by $A(2)$ and $B(2)$ consists of all matrices of the form $\begin{pmatrix} 1+4n_1 & 2n_2 \\ 2n_3 & 1+4n_4 \end{pmatrix}$ with determinant 1, where all n_i are arbitrary integers.

Corollary

The membership problem in the subgroup of $SL_2(\mathbb{Z})$ generated by $A(2)$ and $B(2)$ is solvable in constant time.

A theorem of Sanov

Denote $A(k) = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}$, $B(k) = \begin{pmatrix} 1 & 0 \\ k & 1 \end{pmatrix}$.

Theorem

The subgroup of $SL_2(\mathbb{Z})$ generated by $A(2)$ and $B(2)$ consists of all matrices of the form $\begin{pmatrix} 1+4n_1 & 2n_2 \\ 2n_3 & 1+4n_4 \end{pmatrix}$ with determinant 1, where all n_i are arbitrary integers.

Corollary

The membership problem in the subgroup of $SL_2(\mathbb{Z})$ generated by $A(2)$ and $B(2)$ is solvable in constant time.

Does Sanov's description generalize to $k > 2$?

No.

Does Sanov's description generalize to $k > 2$?

No.

[A. Chorna, K. Geller, V. Shpilrain, *On two-generator subgroups of $SL_2(\mathbb{Z})$, $SL_2(\mathbb{Q})$, and $SL_2(\mathbb{R})$*, J. Algebra **478** (2017), 367–381]:

Theorem

The subgroup of $SL_2(\mathbb{Z})$ generated by $A(k)$ and $B(k)$, $k \in \mathbb{Z},\ k \geq 3$, has infinite index in the group of all matrices of the form $\begin{pmatrix} 1+k^2m_1 & km_2 \\ km_3 & 1+k^2m_4 \end{pmatrix}$ *with determinant 1.*

Does Sanov's description generalize to $k > 2$?

No.

[A. Chorna, K. Geller, V. Shpilrain, *On two-generator subgroups of $SL_2(\mathbb{Z})$, $SL_2(\mathbb{Q})$, and $SL_2(\mathbb{R})$*, J. Algebra **478** (2017), 367–381]:

Theorem

The subgroup of $SL_2(\mathbb{Z})$ generated by $A(k)$ and $B(k)$, $k \in \mathbb{Z},\ k \geq 3$, has infinite index in the group of all matrices of the form $\begin{pmatrix} 1+k^2m_1 & km_2 \\ km_3 & 1+k^2m_4 \end{pmatrix}$ with determinant 1.

Thus, we cannot give both "yes" and "no" answers in sublinear time, but we can give the "no" answer in sublinear time if we have a sublinear time algorithm for divisibility of an integer by k.

Warning

[H.-A. Esbelin and M. Gutan, *On the membership problem for some subgroups of $SL_2(\mathbb{Z})$*, Annales mathématiques du Québec **43** (2019), 233—247].

Theorem

Let $k \in \mathbb{Z},\ k \geq 2$. A matrix of the form $\begin{pmatrix} 1+k^2m_1 & km_2 \\ km_3 & 1+k^2m_4 \end{pmatrix}$ from $SL_2(\mathbb{Z})$ belongs to the subgroup generated by $A(k)$ and $B(k)$ if and only if at least one of the rationals $p = \frac{km_2}{1+k^2m_1}$ and $q = \frac{km_3}{1+k^2m_4}$ has a continued fraction representation with all partial quotients divisible by k.

Warning

[H.-A. Esbelin and M. Gutan, *On the membership problem for some subgroups of* $SL_2(\mathbb{Z})$, Annales mathématiques du Québec **43** (2019), 233—247].

Theorem

Let $k \in \mathbb{Z},\ k \geq 2.$ *A matrix of the form* $\begin{pmatrix} 1+k^2m_1 & km_2 \\ km_3 & 1+k^2m_4 \end{pmatrix}$ *from* $SL_2(\mathbb{Z})$ *belongs to the subgroup generated by* $A(k)$ *and* $B(k)$ *if and only if at least one of the rationals* $p = \frac{km_2}{1+k^2m_1}$ *and* $q = \frac{km_3}{1+k^2m_4}$ *has a continued fraction representation with all partial quotients divisible by* k.

Theorem

Let $a \in \mathbb{Z},\ b \in \mathbb{Z},\ b \neq 0$, *and let* $g.c.d.(a,b) = 1.$ *Then the time complexity of finding a continued fraction representation of* $\frac{a}{b}$ *is* $O(\log(\max(|a|,|b|)))$.

Primitive elements of a free group

Let F_r be a free group with a free generating set $x_1, \ldots, x_r$ and let $w = w(x_1, \ldots, x_r)$. Call an element $u \in F_r$ *primitive* if u can be taken to x_1 by an automorphism of F_r.

The Whitehead graph $Wh(w)$ of w has $2r$ vertices that correspond to $x_1, \ldots, x_r, x_1^{-1}, \ldots, x_r^{-1}$. For each occurrence of a subword $x_i x_j$ in the word $w \in F_r$, there is an edge in $Wh(w)$ that connects the vertex x_i to the vertex x_j^{-1}; if w has a subword $x_i x_j^{-1}$, then there is an edge connecting x_i to x_j, etc. There is one more edge (the external edge): this is the edge that connects the vertex corresponding to the last letter of w to the vertex corresponding to the inverse of the first letter.

Primitive elements of a free group

Let F_r be a free group with a free generating set $x_1, \ldots, x_r$ and let $w = w(x_1, \ldots, x_r)$. Call an element $u \in F_r$ *primitive* if u can be taken to x_1 by an automorphism of F_r.

The Whitehead graph $Wh(w)$ of w has $2r$ vertices that correspond to $x_1, \ldots, x_r, x_1^{-1}, \ldots, x_r^{-1}$. For each occurrence of a subword $x_i x_j$ in the word $w \in F_r$, there is an edge in $Wh(w)$ that connects the vertex x_i to the vertex x_j^{-1}; if w has a subword $x_i x_j^{-1}$, then there is an edge connecting x_i to x_j, etc. There is one more edge (the external edge): this is the edge that connects the vertex corresponding to the last letter of w to the vertex corresponding to the inverse of the first letter.

It was observed by Whitehead that the Whitehead graph of any cyclically reduced primitive element w of length > 2 has either an isolated edge or a cut vertex, i.e., a vertex that, having been removed from the graph together with all incident edges, increases the number of connected components of the graph.

Primitivity-blocking words

Call a group word $w = w(x_1, \ldots, x_r)$ *primitivity-blocking* if it cannot be a subword of any cyclically reduced primitive element of F_r. For example, if the Whitehead graph of w has a Hamilton circuit, then w is primitivity-blocking because in this case, if w is a subword of u, then the Whitehead graph of u, too, has a Hamilton circuit and therefore does not have a cut vertex.

Primitivity-blocking words

Call a group word $w = w(x_1, \ldots, x_r)$ *primitivity-blocking* if it cannot be a subword of any cyclically reduced primitive element of F_r. For example, if the Whitehead graph of w has a Hamilton circuit, then w is primitivity-blocking because in this case, if w is a subword of u, then the Whitehead graph of u, too, has a Hamilton circuit and therefore does not have a cut vertex.

A fast testing algorithm T to test primitivity of an input (cyclically reduced) word u would build the Whitehead graph of u, one edge at a time, going left to right, and checking if the resulting graph is Hamiltonian. (Note that the Whitehead graph always has $2r$ vertices.) The "usual" Whitehead algorithm can run in parallel.

Primitivity-blocking words

Call a group word $w = w(x_1, \ldots, x_r)$ *primitivity-blocking* if it cannot be a subword of any cyclically reduced primitive element of F_r. For example, if the Whitehead graph of w has a Hamilton circuit, then w is primitivity-blocking because in this case, if w is a subword of u, then the Whitehead graph of u, too, has a Hamilton circuit and therefore does not have a cut vertex.

A fast testing algorithm T to test primitivity of an input (cyclically reduced) word u would build the Whitehead graph of u, one edge at a time, going left to right, and checking if the resulting graph is Hamiltonian. (Note that the Whitehead graph always has $2r$ vertices.) The "usual" Whitehead algorithm can run in parallel.

Theorem

The average-case complexity of this composite algorithm is sublinear with respect to $|u|$.

Other blocking words

Let $u \in F_r$. Consider the orbit $Orb(u) = \{\varphi(u),\ \varphi \in Aut(F_r)\}$. Call $w \in F_r$ an $Orb(u)$-blocking word if it cannot be a subword of any cyclically reduced $v \in Orb(u)$.

Other blocking words

Let $u \in F_r$. Consider the orbit $Orb(u) = \{\varphi(u),\ \varphi \in Aut(F_r)\}$. Call $w \in F_r$ an $Orb(u)$-blocking word if it cannot be a subword of any cyclically reduced $v \in Orb(u)$.

Problem. Is there an algorithm that, on input $u \in F_r$, would output at least one particular $Orb(u)$-blocking word?

Other blocking words

Let $u \in F_r$. Consider the orbit $Orb(u) = \{\varphi(u),\ \varphi \in Aut(F_r)\}$. Call $w \in F_r$ an $Orb(u)$-blocking word if it cannot be a subword of any cyclically reduced $v \in Orb(u)$.

Problem. Is there an algorithm that, on input $u \in F_r$, would output at least one particular $Orb(u)$-blocking word?

A good start would be finding an $Orb(u)$-blocking word for $u = [x_1, x_2]$. It is easy to do if $r = 2$ since, by a classical result of Nielsen, any cyclically reduced $v \in Orb([x_1, x_2])$ in this case is either $[x_1, x_2]$ or $[x_2, x_1]$.

The word problem for semigroups

Given two words g, h in generators of a semigroup G, find out whether or not $g = h$ in G.

The word problem for semigroups

Given two words g, h in generators of a semigroup G, find out whether or not $g = h$ in G.

Problem

Are there natural examples of semigroups given by generators and defining relators, where the word problem admits a sublinear time solution for "most" inputs?

The word problem for semigroups

Given two words g, h in generators of a semigroup G, find out whether or not $g = h$ in G.

Problem

Are there natural examples of semigroups given by generators and defining relators, where the word problem admits a sublinear time solution for "most" inputs?

If an algorithm for a sublinear time solution of the word problem exists, it will only give "negative" answers, i. e., $g \neq h$ in G. This is similar to results of [KMSS], where (generically) linear time solution of the word problem was offered for several large classes of groups; their solution, too, gives only "negative" answers.

The word problem for semigroups

Given two words g, h in generators of a semigroup G, find out whether or not $g = h$ in G.

Problem

Are there natural examples of semigroups given by generators and defining relators, where the word problem admits a sublinear time solution for "most" inputs?

If an algorithm for a sublinear time solution of the word problem exists, it will only give "negative" answers, i. e., $g \neq h$ in G. This is similar to results of [KMSS], where (generically) linear time solution of the word problem was offered for several large classes of groups; their solution, too, gives only "negative" answers.

One potential source of semigroups with the property in question is "positive monoids" associated with groups, i.e., monoids generated by group generators, but not their inverses. For some particular groups, e.g. for braid groups, Thompson's group, these monoids have been extensively studied.

Thompson's monoid

Thompson's group F:

$$F = \langle x_0, x_1, x_2, \ldots \mid x_k x_i = x_i x_{k+1} \ (k > i) \rangle.$$

Thompson's monoid

Thompson's group F:

$$F = \langle x_0, x_1, x_2, \ldots \mid x_k x_i = x_i x_{k+1} \ (k > i) \rangle.$$

Since all defining relators in this presentation are pairs of positive words, we can consider the positive monoid associated with this presentation; denote it by F^+.

Thompson's monoid

Thompson's group F:

$$F = \langle x_0, x_1, x_2, \ldots \mid x_k x_i = x_i x_{k+1} \ (k > i) \rangle.$$

Since all defining relators in this presentation are pairs of positive words, we can consider the positive monoid associated with this presentation; denote it by F^+.

Proposition

For any two positive words w_1 and w_2 of lengths m and n, respectively, in the alphabet $X = \{x_0, x_1, x_2, \ldots\}$, there are positive words z_1 and z_2 of lengths n and m, respectively, such that $w_1 z_1 = w_2 z_2$ in Thompson's group F.

Thompson's monoid

Thompson's group F:

$$F = \langle x_0, x_1, x_2, \ldots \mid x_k x_i = x_i x_{k+1} \ (k > i) \rangle.$$

Since all defining relators in this presentation are pairs of positive words, we can consider the positive monoid associated with this presentation; denote it by F^+.

Proposition

For any two positive words w_1 and w_2 of lengths m and n, respectively, in the alphabet $X = \{x_0, x_1, x_2, \ldots\}$, there are positive words z_1 and z_2 of lengths n and m, respectively, such that $w_1 z_1 = w_2 z_2$ in Thompson's group F.

This proposition implies, in particular, that it is impossible to tell that two positive words of length L in the alphabet $X = \{x_0, x_1, x_2, \ldots\}$ are not equal in Thompson's group F by inspecting their initial segments of length $\leq \frac{L}{2}$, i.e., there is at least no such straightforward sublinear time algorithm for detecting inequality in F^+.

Proof

Proof. (due to V.Guba) Construct the following van Kampen diagram. On a square lattice, mark one point as the origin. Starting at the origin and going to the right, write the word w_1 by marking edges of the lattice by the letters of w_1, read left to right. Then, starting at the origin and going up, write the word w_2 by marking edges of the lattice by the letters of w_2, read left to right.

Proof

Proof. (due to V.Guba) Construct the following van Kampen diagram. On a square lattice, mark one point as the origin. Starting at the origin and going to the right, write the word w_1 by marking edges of the lattice by the letters of w_1, read left to right. Then, starting at the origin and going up, write the word w_2 by marking edges of the lattice by the letters of w_2, read left to right.

Now start marking edges of the lattice inside the rectangle built on segments of length m (horizontally) and n (vertically) corresponding to the words w_1 and w_2, as follows. All horizontal edges in the lattice are directed from left to right, and all vertical edges are directed from bottom to top. Then, suppose a single square cell of the lattice has:

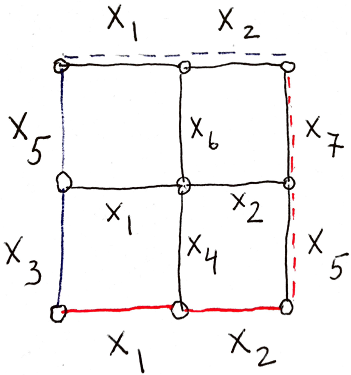
Proof

- x_i on the lower edge and x_i on the left edge. Then we mark the upper edge and the right edge of this cell with the same x_i. This cell now corresponds to the relation $x_i x_i = x_i x_i$.
- x_i on the lower edge and x_j on the left edge, where $i < j$. Then we mark the upper edge of this cell with x_i, and the right edge with x_{j+1}. This cell now corresponds to the relation $x_j x_i x_{j+1}^{-1} x_i^{-1} = 1$, or $x_j x_i = x_i x_{j+1}$.
- x_i on the lower edge and x_j on the left edge, where $i > j$. Then we mark the upper edge of this cell with x_{i+1}, and the right edge with x_j. This cell now corresponds to the relation $x_j x_{i+1} x_j^{-1} x_i^{-1} = 1$, or $x_j x_{i+1} = x_i x_j$.

After all edges of the rectangle built on segments corresponding to the words w_1 and w_2 are marked, we read a relation of the form $w_2 u_1 u_2^{-1} w_1^{-1} = 1$, or $w_2 u_1 = w_1 u_2$, off the edges of this rectangle. Here the length of u_1 is m and the length of u_2 is n. This completes the proof.

Example

If $w_1 = x_1 x_2$ and $w_2 = x_3 x_5$, this method gives $w_1 x_5 x_7 = w_2 x_1 x_2$.



Positive monoids of braid groups

We denote the braid group on n strands by B_n; this group has a standard presentation

$$\langle \sigma_1, ..., \sigma_{n-1} \,|\, \sigma_i\sigma_j = \sigma_j\sigma_i \text{ if } |i-j| > 1;\ \sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1} \text{ for } 1 \leq i \leq n-2 \rangle.$$

Positive monoids of braid groups

We denote the braid group on n strands by B_n; this group has a standard presentation

$$\langle \sigma_1, ..., \sigma_{n-1} | \; \sigma_i\sigma_j = \sigma_j\sigma_i \text{ if } |i-j| > 1; \; \sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1} \text{ for } 1 \le i \le n-2 \rangle.$$

For at least some pairs of positive words in B_n^+ there is a sublinear time test for inequality. The following proposition follows from [M. Autord, P. Dehornoy, *On the distance between the expressions of a permutation*, European J. Combin. **31** (2010), 1829–1846]; in particular, from the proof of their Proposition 2.9.

Proposition

Let $w_1 = \sigma_1\sigma_3 \cdots \sigma_{2m-1}$, $\quad w_2 = \sigma_{2m}\sigma_{2m-2} \cdots \sigma_2$. *Suppose* $w_1 u = w_2 v$ *for some* $u, v \in B_n^+$, $\; n > 2m$. *Then* $|u|, |v| = 2m^2$.

Positive monoids of braid groups

We denote the braid group on n strands by B_n; this group has a standard presentation

$$\langle \sigma_1, ..., \sigma_{n-1} \,|\, \sigma_i\sigma_j = \sigma_j\sigma_i \text{ if } |i-j| > 1;\ \sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1} \text{ for } 1 \le i \le n-2\rangle.$$

For at least some pairs of positive words in B_n^+ there is a sublinear time test for inequality. The following proposition follows from [M. Autord, P. Dehornoy, *On the distance between the expressions of a permutation*, European J. Combin. **31** (2010), 1829–1846]; in particular, from the proof of their Proposition 2.9.

Proposition

Let $w_1 = \sigma_1\sigma_3\cdots\sigma_{2m-1}$, $\quad w_2 = \sigma_{2m}\sigma_{2m-2}\cdots\sigma_2$. *Suppose* $w_1 u = w_2 v$ *for some* $u, v \in B_n^+$, $n > 2m$. *Then* $|u|, |v| = 2m^2$.

Thus, in particular, if one has two positive braid words of length L, where one of them starts with $\sigma_1\sigma_3\cdots\sigma_{2k-1}$, the other one starts with $\sigma_{2k}\sigma_{2k-2}\cdots\sigma_2$, and $k \ge \sqrt{L}$, then these braid words are not equal in B_n^+, $n > 2k$.

Positive monoids of braid groups (continued)

Of course, this is just a very special example where a sublinear time algorithm can detect inequality of two words in B_n^+, so the interesting question is whether examples of this sort are "generic". We therefore ask:

Positive monoids of braid groups (continued)

Of course, this is just a very special example where a sublinear time algorithm can detect inequality of two words in B_n^+, so the interesting question is whether examples of this sort are "generic". We therefore ask:

Problem

Is there a generic subset S of B_n^+ and a number $\epsilon > 0$ such that for any two braid words w_1, w_2 of length k representing elements of S, the minimum length of words u, v such that $w_1 u = w_2 v$, is greater than $k^{(1+\epsilon)}$?

Positive monoids of free nilpotent groups

No sublinear time algorithm for the word problem.

Positive monoids of free nilpotent groups

No sublinear time algorithm for the word problem.

[V. Shpilrain, *Sublinear time algorithms in the theory of groups and semigroups*, Illinois J. Math. **54** (2011), 187–197].

Average-case complexity of the word problem

Still, “fast checks” (e.g. considering abelianization) can be used as average-case performance boosters, even if it does not lead to a sublinear time average-case algorithm.

For example, it is known that there are algorithms for solving the word problem in nilpotent groups in polynomial time, where the degree of the polynomial grows with the nilpotency class.

Average-case complexity of the word problem

Still, “fast checks” (e.g. considering abelianization) can be used as average-case performance boosters, even if it does not lead to a sublinear time average-case algorithm.

For example, it is known that there are algorithms for solving the word problem in nilpotent groups in polynomial time, where the degree of the polynomial grows with the nilpotency class.

Problem

What is the best (over all “honest” algorithms) average-case complexity of the word problem in a free nilpotent group?

Average-case complexity of the word problem

Still, “fast checks” (e.g. considering abelianization) can be used as average-case performance boosters, even if it does not lead to a sublinear time average-case algorithm.

For example, it is known that there are algorithms for solving the word problem in nilpotent groups in polynomial time, where the degree of the polynomial grows with the nilpotency class.

Problem

What is the best (over all “honest” algorithms) average-case complexity of the word problem in a free nilpotent group?

Note: if an input is given by coordinates in a Malcev basis, then the word problem (in *any* f.g. nilpotent group) can be solved in quasi-linear time, according to [J. Macdonald, A. Myasnikov, A. Nikolaev, S. Vassileva, *Logspace and compressed-word computations in nilpotent groups*, https://arxiv.org/abs/1503.03888].

Conclusions

Conclusions

1. While the worst-case and generic-case complexity of algorithms in group theory have been well studied, this is not the case with the average-case complexity. It may be time to seriously address the average-case complexity, at least in some “smooth” groups.

Conclusions

1. While the worst-case and generic-case complexity of algorithms in group theory have been well studied, this is not the case with the average-case complexity. It may be time to seriously address the average-case complexity, at least in some “smooth” groups.

2. Those “fast checks” that are Las Vegas type algorithms can be used to boost performance of some well-established “honest” algorithms and reduce their average-case complexity when run in parallel.

Thank you