Property testing of formal languages

Gabriel Bathie

STACS 2026

Property Testing

Two complementary points of view:

Property testing [Goldreich, 2017]

Study of fast, randomized, approximate decision procedures.

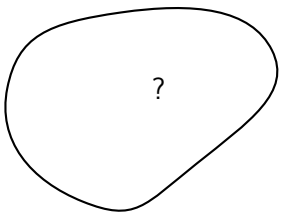
Property testing

Study of the "information complexity" of promise problems.

Property Testing (II)

Ingredients of property testing:

1. Object $\mathcal{O}$
2. Property P
3. Query mechanism: oracle that reveals local information about $\mathcal{O}$
4. Promise: either $\mathcal{O}$ has P, or $\mathcal{O}$ is ε-far from P, with $\varepsilon > 0$ (hardness level) e.g. $\forall \mathcal{X} \in P, d(\mathcal{O}, \mathcal{X}) \geq \varepsilon |\mathcal{X}|$.



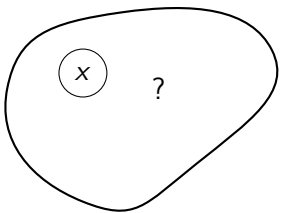
Goal: algorithm that decides the promise problem
$\rightarrow$ minimize number of queries in the worst case (*query complexity*, function of $|\mathcal{O}|, \varepsilon$)

Ideal outcome: query complexity is constant w.r.t $|\mathcal{O}|$, i.e. $f(\varepsilon)$.

Property Testing (II)

Ingredients of property testing:

1. Object $\mathcal{O}$
2. Property P
3. Query mechanism: oracle that reveals local information about $\mathcal{O}$
4. Promise: either $\mathcal{O}$ has P, or $\mathcal{O}$ is ε-far from P, with $\varepsilon > 0$ (hardness level) e.g. $\forall \mathcal{X} \in P, d(\mathcal{O}, \mathcal{X}) \geq \varepsilon |\mathcal{X}|$.



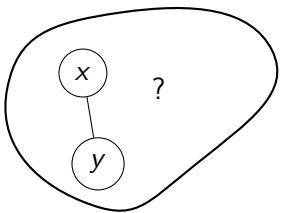
Goal: algorithm that decides the promise problem
$\rightarrow$ minimize number of queries in the worst case (*query complexity*, function of $|\mathcal{O}|, \varepsilon$)

Ideal outcome: query complexity is constant w.r.t $|\mathcal{O}|$, i.e. $f(\varepsilon)$.

Property Testing (II)

Ingredients of property testing:

1. Object $\mathcal{O}$
2. Property P
3. Query mechanism: oracle that reveals local information about $\mathcal{O}$
4. Promise: either $\mathcal{O}$ has P, or $\mathcal{O}$ is ε-far from P, with $\varepsilon > 0$ (hardness level) e.g. $\forall \mathcal{X} \in P, d(\mathcal{O}, \mathcal{X}) \geq \varepsilon |\mathcal{X}|$.



Goal: algorithm that decides the promise problem
$\rightarrow$ minimize number of queries in the worst case (*query complexity*, function of $|\mathcal{O}|, \varepsilon$)

Ideal outcome: query complexity is constant w.r.t $|\mathcal{O}|$, i.e. $f(\varepsilon)$.

Property Testing examples

Object $\mathcal{O}$	**Property P**	**Query type**	**Promise**
Graph $G = (V, E)$	Triangle-free Bipartite	$uv \in E$? ("Dense Graphs")	P vs εn^2 edges from P
	H-freeness ...	$adj(u)[i]$? ("Bounded Degree")	P vs εnd edges from P degree $\leq d$
Boolean function f	Constant Dictatorship	$f(x)$?	P vs $\varepsilon \cdot 2^n$ values in truth table differ from P
Word u	Membership in L	$u[i]$?	P vs εn letters from P

Property Testing algorithms

Usual framework:

1. **Local obstructions:** "small" sub-object that witnesses $\mathcal{O} \notin P$
   "small": can be found with few queries
   Ex: G is triangle-free $\rightarrow$ triangular subgraph
   Ex: f is constant $\rightarrow x, y$ s.t. $f(x) \neq f(y)$
2. **Lemma** "ε-far $\Rightarrow$ contains many obstructions"
   Ex: G is ε-far from triangle-free $\Rightarrow$ G contains $\Omega(\delta n^3)$ triangles,
   $\delta = f(\varepsilon)$ given by Szemerédi's Regularity Lemma
3. **Sampling algorithm:**
   If $\mathcal{O}$ is ε-far from P, finds an obstruction with large probability.
   If $\mathcal{O} \in P$, will not find an obstruction.
   Ex: G triangle-free $\rightarrow$ sample $O(\delta^{-1})$ random triples of vertices u, v, w,
   say "$G \notin P$" if any forms a triangle.

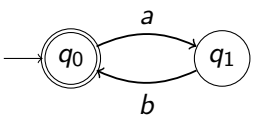
Today: Testing regular languages

Setup:

1. **Object** $\mathcal{O}$: word u of length n
2. **Property** P: is u in L? (L regular)
3. **Query mechanism:** Given $i < n$, what is $u[i]$?
4. **Promise:** either $u \in L$ or $d(u, v) \geq \varepsilon n$ for any $v \in L$.
   d: Hamming distance or edit distance, different problems.

Example

Question: is u in $L(\mathcal{A})$ or ε-far from $L(\mathcal{A})$?

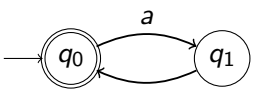


$L(\mathcal{A}) = (ab)^*$

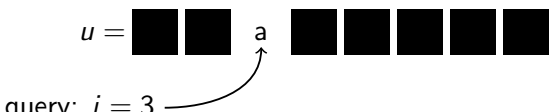
$u =$ 

Example

Question: is u in $L(\mathcal{A})$ or ε-far from $L(\mathcal{A})$?

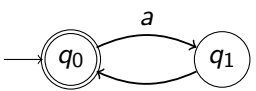


$L(\mathcal{A}) = (ab)^*$

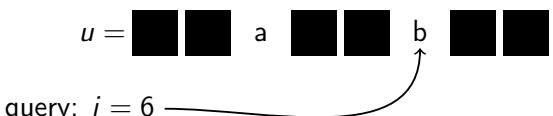


Example

Question: is u in $L(\mathcal{A})$ or ε-far from $L(\mathcal{A})$?

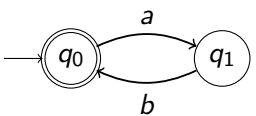


$L(\mathcal{A}) = (ab)^*$

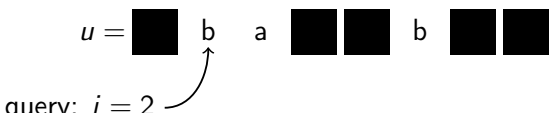


Example

Question: is u in $L(\mathcal{A})$ or ε-far from $L(\mathcal{A})$?

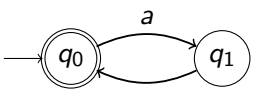


$L(\mathcal{A}) = (ab)^*$

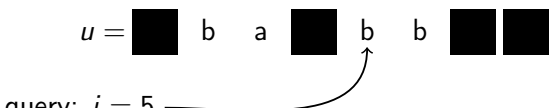


Example

Question: is u in $L(\mathcal{A})$ or ε-far from $L(\mathcal{A})$?

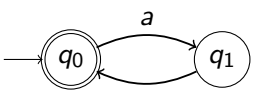


$L(\mathcal{A}) = (ab)^*$



Example

Question: is u in $L(\mathcal{A})$ or ε-far from $L(\mathcal{A})$?



$L(\mathcal{A}) = (ab)^*$

$u =$ ■ b a ■ b b ■■

A word of the form $abab\cdots$ cannot contain bb!

$\rightarrow$ **Obstructions:** every* regular language has its "bb" (blocking factors).

Today: Testing regular languages

Content:

1. Overview of existing results
2. **Local obstructions:** blocking factors
3. **Lemma** "ε-far $\Rightarrow$ contains many blocking factors"
4. **Sampling algorithm:** original by [Alon et al., 2001], improved by [François et al., 2016] and [Bathie and Starikovskaya, 2021]
5. Lower bounds, meta-questions...

Algorithmic results

Theorem [Alon et al., 2001]

All regular languages can be tested with $O(\log^3(\varepsilon^{-1})/\varepsilon)$ queries.

Corollary of [François et al., 2016]

All regular languages can be tested with $O(\log^2(\varepsilon^{-1})/\varepsilon)$ queries.

Theorem [Bathie and Starikovskaya, 2021]

All regular languages can be tested with $O(\log(\varepsilon^{-1})/\varepsilon)$ queries.
This is optimal.

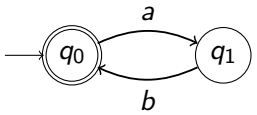
Obstructions to regular languages

Blocking factor*

A word v is a blocking factor of $\mathcal{A}$ if ($\forall u$, v is a factor of $u \Rightarrow u \notin L(\mathcal{A})$)

Lemma

v is $\mathcal{A}$-blocking iff there are no states p, q such that $p \xrightarrow{v} q$.



$L(\mathcal{A}) = (ab)^*$

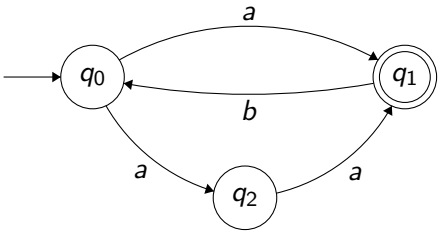
Blocking factors: $bb, aa, abba, \ldots$

Hamming distance: b at an even position is blocking: $??b??? \neq ababab$

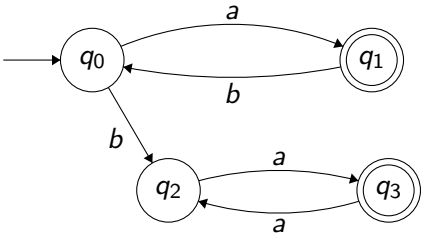
*For the edit distance.

Strongly Connected Automata

$\mathcal{A}$ is strongly connected: for any states q_i, q_j, there is a directed path from q_i to q_j



Strongly connected



Not strongly connected

Lemma

If $\mathcal{A}$ is strongly connected with m states, then for any states q_i, q_j, there exists w such that

$$q_i \xrightarrow{w} q_j \text{ and } |w| < m$$

Far implies many Blocking Factors

Lemma

If u is ε-far from $L(\mathcal{A})$, then u contains disjoint $\Omega(\varepsilon n)$ $\mathcal{A}$-blocking factors.

Proof Overview:

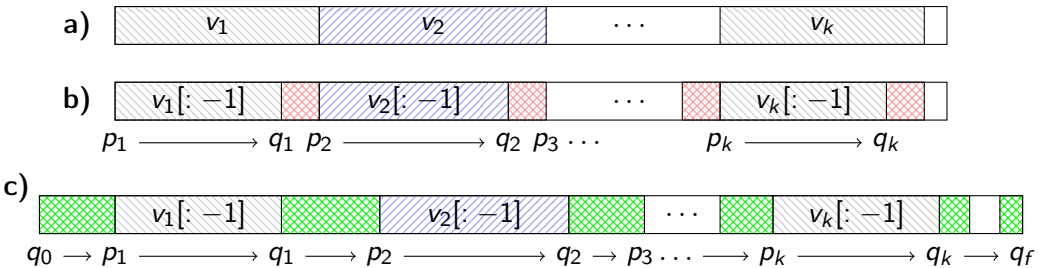
- Build a set $\mathcal{S}$ of k disjoint blocking factors of u
- From $\mathcal{S}$, build a sequence of $O(k)$ edits that turn u into $v \in L$
- Since u is ε-far, we have $k = \Omega(\varepsilon n)$.

Far implies many Blocking Factors (II)

a) Greedy decomposition into disjoint **minimal** blocking factors $\rightarrow v_1, \dots v_k$

b) Delete last letter of each $v_i \Rightarrow$ not blocking, k deletions

c) Insert factors to bridge the gaps $q_i \rightarrow p_{i+1}$: $\leq m \cdot (k+2)$ edits $\quad (\star)$



$\Rightarrow$ word in $L(A)$

Total number of edits $m \cdot (k+2) + k \geq \varepsilon n \Rightarrow k = \Omega(\varepsilon n)$.

Simple sampling algorithm

Lemma

If u is ε-far from $L(\mathcal{A})$, then u contains $c\varepsilon n$ $\mathcal{A}$-blocking factors of length at most $\ell = O(1/\varepsilon)$.

Corollary

We can test regular languages with $O(1/\varepsilon^2)$ queries.

Algorithm: Repeat $O(1/\varepsilon)$ times

- Choose a random position $i < n$
- Query $u[i : i + \ell]$
- Reject if this factor is blocking

Total: $O(1/\varepsilon) \cdot \ell = O(1/\varepsilon^2)$ queries, correct w.p. $\gg 0$ if u ε-*far* from L.

Simple sampling algorithm

Lemma

If u is ε-far from $L(\mathcal{A})$, then u contains $c\varepsilon n$ $\mathcal{A}$-blocking factors of length at most $\ell = O(1/\varepsilon)$.

Corollary

We can test regular languages with $O(1/\varepsilon^2)$ queries.

Algorithm: Repeat $O(1/\varepsilon)$ times

- Choose a random position $i < n \to p \geq c\varepsilon$ of being the start of a blocking factor
- Query $u[i : i + \ell] \to$ uses $\ell = O(1/\varepsilon)$ queries
- Reject if this factor is blocking

Total: $O(1/\varepsilon) \cdot \ell = O(1/\varepsilon^2)$ queries, correct w.p. $\gg 0$ if u ε-*far* from L.

Getting to $\log(\varepsilon^{-1})/\varepsilon$

[Alon et al., 2001]:

- Better sampling algorithm that considers $\log(\varepsilon^{-1})$ possible cases based on the length of the blocking factors.
  $\rightarrow O(\log^2(\varepsilon^{-1})/\varepsilon)$ queries.
- Strongly connected $\rightarrow$ general case adds a $\log(\varepsilon^{-1})$ factor
  $\rightarrow O(\log^3(\varepsilon^{-1})/\varepsilon)$ queries.

[François et al., 2016]: Strongly connected $\rightarrow$ general case only needs $O(1)$ factor
$\rightarrow O(\log^2(\varepsilon^{-1})/\varepsilon)$ queries.

[Bathie and Starikovskaya, 2021]: Case analysis of [Alon et al., 2001] $\rightarrow$ global analysis.
$\rightarrow O(\log(\varepsilon^{-1})/\varepsilon)$ queries.

Lower bounds

Theorem [Alon et al., 2001]

All *nontrivial* regular languages require $\Omega(1/\varepsilon)$ queries.

Trivial languages: e.g. $L = a\Sigma^*$.
Maximum distance to this language: 1.
For n large enough, no word is ε-far $\rightarrow$ can always answer "in L".

Context-free languages [Alon et al., 2001]

There is a context-free language that requires $\Omega(\sqrt{n})$ queries.

State of the art

Theorem [Alon et al., 2001]

- All regular languages: $O(\log^3(\varepsilon^{-1})/\varepsilon)$ queries.
- All "interesting" regular languages require $\Omega(1/\varepsilon)$ queries.

Theorem [B. and Starikovskaya, ICALP'21]

- All regular languages: $O(\log(\varepsilon^{-1})/\varepsilon)$ queries.
- There exists a regular language L_0 that requires $\Omega(\log(\varepsilon^{-1})/\varepsilon)$ queries.

$\rightarrow$ Problem closed?

The gap

Theorem [B. and Starikovskaya, ICALP'21]

There exists a regular language L_0 that requires $\Omega(\log(\varepsilon^{-1})/\varepsilon)$ queries.

$\rightarrow$ Applies to a single language.

There are languages with complexity $\Theta(1/\varepsilon)$, e.g. $L = a^*$ over $\Sigma = \{a, b\}$:

1. $u = aaaa \ldots aaa$ vs. u contains at least εn b's,
2. querying $O(1/\varepsilon)$ letters at random finds a "b" w.p. $\geq 2/3$.

Group by optimal query complexity:

- **"Hard":** $\Theta(\log(\varepsilon^{-1})/\varepsilon)$ queries $\to L_0$,
- **"Easy":** $\Theta(1/\varepsilon)$ queries $\to a^*$ $(\Sigma = \{a, b\})$,
- **"Trivial":** 0 queries $\to a\Sigma^*$, finite languages.

Question I

Are there other complexity classes?

Question II

Can we *characterize* the languages in each class?

Inspired by recent characterizations of:

- [Amarilli et al., 2021]: Dynamic Membership in Regular Languages,
- [Ganardi et al., 2024]: Regular Languages in Sliding Windows.

Characterizing with (minimal) blocking factors

Minimal Blocking Factor (MBF) for L

Blocking factor v with no proper factor that is blocking for L.

$\rightarrow$ for $L = a^*, MBF(L) = \{b\}$; $ba, abba, bbaba, \ldots,$ are not minimal.

Trichotomy Theorem [B., Fijalkow, Mascle, ICALP'25]

Complexity is determined by the cardinality of the set of *minimal blocking factors*.

Class	Query Complexity ($\Theta(\cdot)$)	MBF(L)
Hard	$\log(\varepsilon^{-1})/\varepsilon$	Infinite
Easy	$1/\varepsilon$	Finite, non-empty
Trivial	0	$\emptyset$

Summary

Property testing:

- "Fast approximate randomized decision procedures"
- Usual approach: try to find obstructions that show $\mathcal{O} \notin P$ using "$\mathcal{O}$ far from $P \Rightarrow \mathcal{O}$ contains many obstructions".

Testing regular languages:

- Blocking factors: central to understanding property testing of regular languages.
- Query complexity is determined by the number of *minimal blocking sequences*.

Class	Query Complexity ($\Theta(\cdot)$)	MBS(L)
Hard	$\log(\varepsilon^{-1})/\varepsilon$	Infinite
Easy	$1/\varepsilon$	Finite, non-empty
Trivial	0	$\emptyset$

Open question: test context-free language, query = membership in regular language?

References I

Alon, N., Krivelevich, M., Newman, I., and Szegedy, M. (2001).
Regular languages are testable with a constant number of queries.
SIAM Journal on Computing, 30(6):1842–1862.

Amarilli, A., Jachiet, L., and Paperman, C. (2021).
Dynamic Membership for Regular Languages.
In Bansal, N., Merelli, E., and Worrell, J., editors, *Proc. of ICALP'21*, volume 198 of *LIPIcs*, pages 116:1–116:17.

Bathie, G. and Starikovskaya, T. (2021).
Property Testing of Regular Languages with Applications to Streaming Property Testing of Visibly Pushdown Languages.
In *Proc. of ICALP*, volume 198 of *LIPIcs*, pages 119:1–119:17.

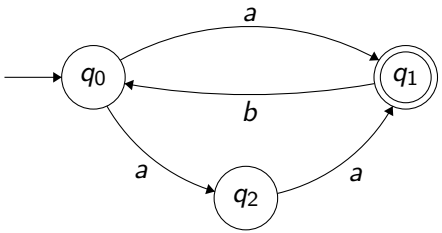
References II


François, N., Magniez, F., de Rougemont, M., and Serre, O. (2016).
Streaming property testing of visibly pushdown languages.
In *Proc. of ESA*, volume 57 of *LIPIcs*, pages 43:1–43:17.


Ganardi, M., Hucke, D., Lohrey, M., Mamouras, K., and Starikovskaya, T. (2024).
Regular languages in the sliding window model.
CoRR, abs/2402.13385.


Goldreich, O. (2017).
Introduction to property testing.
Cambridge University Press.

Finite automata and notations



- q_0 : initial state, q_1 : final (accepting) state
- $q_i \xrightarrow{w} q_j$: "w labels some transition (run) from q_i to q_j"

$$q_0 \xrightarrow{a} q_1, q_2 \xrightarrow{aba} q_2$$

- $w \in L(\mathcal{A})$: "$q_i \xrightarrow{w} q_j$ with q_i initial and q_j final"

$$aaba \in L(\mathcal{A}), abab \notin L(\mathcal{A})$$

Better sampling algorithm [Alon et al., 2001]

We have $N \geq \varepsilon n$ disjoint blocking factors of length $\leq 1/\varepsilon$.

Remark

If we have M blocking factors of length between ℓ and 2ℓ, then sampling $\simeq n/(M\ell)$ factors of length 4ℓ finds a blocking factor wp. $\geq 2/3$.

Analysis:

- M disjoint blocking factors of length $\geq \ell$ cover at least $M \cdot \ell$ positions.
- Sampling a factor v of length 4ℓ: choose a random position i, read 2ℓ letters before and after.
- Length $\leq 2\ell$: if i is inside a blocking factor, v is also blocking.
  $\rightarrow$ occurs w.p. $p \geq M\ell/n$.
- Repeat $O(1/p)$ times $\Rightarrow$ works w.p. $\geq 2/3$.
- Takes $O(n/M)$ queries.

Better sampling algorithm [Alon et al., 2001] (II)

Sampling algorithm:

- We have $N \geq \varepsilon n$ disjoint blocking factors of length $\leq 1/\varepsilon$.
- Define $\mathcal{B}_t$: blocking factors of length in $[2^t, 2^{t+1}[$, $t = 0, 1, \ldots, \log(\varepsilon^{-1})$.
- Markov inequality implies $\exists t : |\mathcal{B}_t| \geq \varepsilon n / \log(\varepsilon^{-1})$.
- For this t, sample $N \simeq \log(\varepsilon^{-1})/\varepsilon \cdot 2^{-t}$ factors of length 2^{t+2}
  $\rightarrow O(\log(\varepsilon^{-1})/\varepsilon)$ queries, find a blocking factor w.p. $\geq 2/3$.
- We don't know t: try all $\log(\varepsilon^{-1})$ values $\rightarrow O(\log^2(\varepsilon^{-1})/\varepsilon)$ queries.

About strong connectedness:

If $\mathcal{A}$ is not strongly connected, can decompose into strongly connected components (SCCs).

Lemma [François et al., 2016]

If u is ε-far from $\mathcal{A}$ with SCCs $\mathcal{C}_1, \mathcal{C}_2, \ldots$, then u contains $\Omega(\varepsilon n)$ blocking factors for $\mathcal{C}_1$, then $\Omega(\varepsilon n)$ blocking factors for $\mathcal{C}_2$, ...

- **Alon et al.:** Increase number of queries by a $\log(\varepsilon^{-1})$ factor.
- **François et al.:** We can replace with a constant factor!

$\rightarrow$ reduces complexity to $O(\log^2(\varepsilon^{-1})/\varepsilon)$ queries

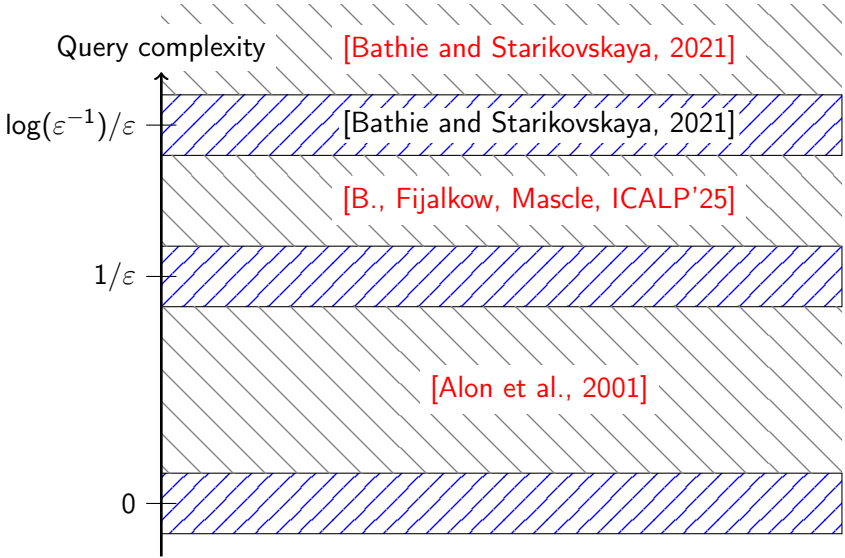
Better analysis of the sampling algorithm

[Alon et al., 2001]:

- Some bucket $\mathcal{B}_t$ contains $\varepsilon n / \log(\varepsilon^{-1})$ blocking factors of length in $[2^t, 2^{t+1}[$.
- In that bucket, $O(\log(\varepsilon^{-1})/\varepsilon)$ queries are enough.
- We don't know $\mathcal{B}_t$: try each $O(\log(\varepsilon^{-1}))$ values.
- **Total:** $O(\log^2(\varepsilon^{-1})/\varepsilon)$ queries.

[Bathie and Starikovskaya, 2021]:

- Queries aimed at $\mathcal{B}_{t+1}, \mathcal{B}_{t+2}, \ldots$ could find factors in $\mathcal{B}_t$, although with smaller probability.
- Count all factors, even those in buckets with less than $\varepsilon n / \log(\varepsilon^{-1})$ elements.
- **Total:** $O(\log(\varepsilon^{-1})/\varepsilon)$ queries are enough!


Query complexity
[Bathie and Starikovskaya, 2021]
$\log(\varepsilon^{-1})/\varepsilon$
[Bathie and Starikovskaya, 2021]
[B., Fijalkow, Mascle, ICALP'25]
$1/\varepsilon$
[Alon et al., 2001]
0

Related Results

From [Bathie, Fijalkow, Mascle, ICALP'25]:

Structural

The set MBF(L) is a regular language.

Algorithmic

Given $\mathcal{A}$, classifying $L(\mathcal{A})$ is PSPACE-complete.