Does Locality imply Efficient Testability?

Omri Ben-Eliezer

WOLA 2019
Monotonicity testing: Yet another proof...

Consider an array of numbers. Is the array monotone increasing?

- Array: $2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 47, 53, 59$
- Array is monotone

- Array: $11, 13, 17, 19, 2, 3, 5, 7, 23, 29, 31, 37, 41, 47, 53, 59$
- Array not monotone
Monotonicity testing: Yet another proof...

Consider an array of numbers. Is the array monotone increasing?

Property Testing:
Given query access to $A: [n] \rightarrow \mathbb{R}$ that is $\epsilon$-far from being monotone increasing, how many queries needed to find (with prob. $2/3$) a “proof” that $A$ is not monotone. 

$\epsilon$-far:
Need to change $\epsilon n$ entries in $A$ to make it monotone.
Monotonicity testing: Yet another proof...

Consider an array of numbers. Is the array monotone increasing?

[2 3 5 7 11 13 17 19 23 29 31 37 41 47 53 59]

array is monotone

[11 13 17 19 2 3 5 7 23 29 31 37 41 47 53 59]

array not monotone

Monotonicity is $\varepsilon$-testable with $O(\varepsilon^{-1}\log n)$ queries.

[Ergün, Kannan, Kumar, Rubinfeld, Viswanthan '98:]

Property Testing:
Given query access to $A: [n] \rightarrow \mathbb{R}$ that is $\varepsilon$-far from being monotone increasing, how many queries needed to find (with prob. 2/3) a “proof” that $A$ is not monotone.

$\varepsilon$-far:
Need to change $\varepsilon n$ entries in $A$ to make it monotone.
Monotonicity testing: Yet another proof...
Monotonicity testing: Yet another proof...
Monotonicity testing: Yet another proof...
Monotonicity testing: Yet another proof...
Monotonicity testing: Yet another proof...

Consider a partitioning of the array into intervals.

In which intervals is the first elements larger than the last?
Monotonicity testing: Yet another proof...

Hierarchical partitioning:

each interval in level $i$ is union of two or three intervals from level $i - 1$
Monotonicity testing: Yet another proof...

Hierarchical partitioning:

*each interval in level $i$ is union of two or three intervals from level $i-1*
Monotonicity testing: Yet another proof...

Hierarchical partitioning:

Each interval in level $i$ is union of two or three intervals from level $i - 1$.
Hierarchical partitioning: each interval in level $i$ is union of two or three intervals from level $i - 1$
Monotonicity testing: Yet another proof...

Hierarchical partitioning:

Each interval in level \( i \) is the union of two or three intervals from level \( i - 1 \)
Consider only “bad” intervals that are maximal: not contained in any other “bad” one.

Claim: Suffices to edit elements within “good” intervals that are one level above maximal “bad” ones, to make array monotone.
Monotonicity testing: Yet another proof...

Consider only “bad” intervals that are maximal: not contained in any other “bad” one.

Claim: Suffices to edit elements within “good” intervals that are one level above maximal “bad” ones, to make array monotone.
Monotonicity testing: Yet another proof...

Corollary: If array is $\varepsilon$-far from monotonicity, then set of “maximal bad intervals” has size at least $\approx \varepsilon n$.

The test: Pick $\approx 1/\varepsilon$ intervals from each level, query their endpoints. Reject if any of them is bad. Total query complexity $\approx \varepsilon^{-1} \log n$. 
Local properties

A property of arrays $A: [n] \rightarrow \Sigma$ is $k$-local if it can be defined by a family of forbidden consecutive patterns of size $\leq k$.

Examples:

Monotonicity is 2-local. Forbidden patterns: “$A(i) > A(i + 1)$”

Array is monotone

Array not monotone
Local properties

A property of arrays $\Lambda: [n] \to \Sigma$ is $k$-local if it can be defined by a family of forbidden consecutive patterns of size $\leq k$.

Examples:

<table>
<thead>
<tr>
<th>Property</th>
<th>k-locality</th>
<th>Forbidden patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monotonicity</td>
<td>2-local</td>
<td>&quot;$A(i) &gt; A(i + 1)$&quot;</td>
</tr>
<tr>
<td>Lipschitz-continuity</td>
<td>2-local</td>
<td></td>
</tr>
<tr>
<td>Convexity</td>
<td>3-local</td>
<td></td>
</tr>
<tr>
<td>Properties of first $k$ discrete derivatives</td>
<td>$(k + 1)$-local</td>
<td></td>
</tr>
<tr>
<td>Pattern matching and computational biology</td>
<td>$k$-local</td>
<td>for small $k$</td>
</tr>
</tbody>
</table>
## Local properties

A property of arrays $A : [n]^d \rightarrow \Sigma$ is **$k$-local** if it can be defined by a family of **forbidden consecutive patterns** of size $\leq k \times \cdots \times k$.

### Examples:

<table>
<thead>
<tr>
<th>Property</th>
<th>LOCALITY</th>
<th>Forbidden patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Monotonicity</strong></td>
<td>2-local</td>
<td>“$A(i) &gt; A(i + 1)$”</td>
</tr>
<tr>
<td><strong>Lipschitz-continuity</strong></td>
<td>2-local</td>
<td></td>
</tr>
<tr>
<td><strong>Convexity</strong></td>
<td>3-local</td>
<td></td>
</tr>
<tr>
<td><strong>Submodularity</strong></td>
<td>2-local</td>
<td></td>
</tr>
<tr>
<td><strong>Properties of first $k$ discrete derivatives</strong></td>
<td>$(k + 1)$-local</td>
<td></td>
</tr>
<tr>
<td><strong>Pattern matching problems in computer vision</strong></td>
<td>$k$-local for small $k$</td>
<td></td>
</tr>
</tbody>
</table>
Local properties $\iff$ Local algorithms

The LOCAL model in distributed computing [Linial’87]:

Which graph properties are “locally decidable” by balls of radius $k$?

Our setting a bit different:

1. **Graph topology known in advance**: graph is the line (for $d=1$) / hypergrid ($d>1$).
2. However, each *vertex* holds a *value* (not known in advance).

**Claim**: Property is $k$-local $\iff$ has local algorithm (known topology, unknown values) with $\Theta(k)$ rounds
Generic test for local properties

**Theorem** [B., 2019]:

Any $k$-local property $\mathcal{P}$ of $[n]^d$-arrays over any finite alphabet $\Sigma$ is $\varepsilon$-testable using

$$O\left(\frac{k \log n}{\varepsilon}\right)$$
queries for $d = 1$

$$O_d\left(\frac{kn^{d-1}}{\varepsilon^{1/d}}\right)$$
queries for $d > 1$

**Property Testing:**
Given property $\mathcal{P}$, parameter $\varepsilon$, and query access to $A: [n]^d \rightarrow \Sigma$, distinguish with prob. $2/3$ between the cases:

- $A$ satisfies $\mathcal{P}$
- $A$ is $\varepsilon$-far from $\mathcal{P}$: need to change $\varepsilon n^d$ values in $A$ to satisfy $\mathcal{P}$
Generic test for local properties

**Theorem** [B., 2019]:

Any $k$-local property $\mathcal{P}$ of $[n]^d$-arrays over any finite alphabet $\Sigma$ is $\varepsilon$-testable using

\[ O \left( \frac{k \log n}{\varepsilon} \right) \text{ non-adaptive queries for } d = 1 \]

\[ O_d \left( \frac{kn^{d-1}}{\varepsilon^{1/d}} \right) \text{ non-adaptive queries for } d > 1 \]

The good news: Test is canonical (queries depend on $d, k, \varepsilon, n$, but not on $\mathcal{P}, \Sigma$);
proximity oblivious (repetitive iterations of the same “basic” test);
non-adaptive (makes all queries in advance); and has
one-sided error.

Allows “sketching for testing”.

The bad news: linear running time for $d = 1$; exponential for $d > 1$ 😞
The main idea: Unrepairability

An interval $I = \{a, a + 1, \ldots, b\} \subseteq [n]$ is unrepairable (w.r.t $A$, $\mathcal{P}$) if, no matter how we modify $A(a + 1), \ldots, A(b - 1)$, the sub-array of $A$ between $a$ and $b$ will never satisfy $\mathcal{P}$.

Observation: Enough to query only $f(a)$ and $f(b)$ to know if $I$ is unrepairable.

Example: unrepairable interval for monotonicity.
The main idea: Unrepairability

An interval $I = \{a, a + 1, \ldots, b\} \subseteq [n]$ is unrepairable (w.r.t $A, \mathcal{P}$) if, no matter how we modify $A(a + 1), \ldots, A(b - 1)$, the sub-array of $A$ between $a$ and $b$ will never satisfy $\mathcal{P}$.

Observation: Enough to query only $f(a)$ and $f(b)$ to know if $I$ is unrepairable.

Example: unrepairable interval for monotonicity.
The main idea: Unrepairability

Proof idea:
Structural result: Suppose that $A$ is $\varepsilon$-far from $\mathcal{P}$. Then there is a set of “canonical” unrepairable intervals covering $\geq \varepsilon n$ of the entries.

Algorithm: For any $i = 0, 1, \ldots, \log n$, pick $\approx 1/\varepsilon$ “canonical” intervals of length $\approx 2^i$ and query their endpoints. With good probability, one of the intervals will be unrepairable.

Extension to multiple dimensions:
Replace “intervals” by “$d$-dimensional consecutive boxes” and “endpoints” with “$(d - 1)$-dimensional boundaries”.
Non-adaptive Lower bounds

The upper bound is tight for non-adaptive algorithms, for any fixed $d \geq 1$

For $d = 1$, matches $\Theta(\log n)$ bounds for monotonicity [EKKRV’98, F’04, CS’13], convexity [PRR’04], and Lipschitz [JR’11]. Tight for monotonicity even among adaptive two-sided tests.

For $d > 1$,

**Theorem [B., 2019]:**

There exists a $k$-local property of $[n]^d$-arrays over alphabet of size $n^{O(d)}$, whose non-adaptive one-sided query complexity is $\Omega_d(k \epsilon^{-\frac{1}{d}} n^{d-1})$. 
### Non-adaptive Lower Bounds

**The upper bound**

For $d = 1$, matches $\Theta(\log n)$ [PRR’04], and Lipschitz [JR’11].

For $d > 1$,

**Theorem [B., 2019]:**

There exists a $k$-local $n^{O(d)}$, whose non-adaptive one-sided query complexity is $\Omega_d(k\varepsilon^{-\frac{1}{d}}n^{d-1})$.

<table>
<thead>
<tr>
<th>$d$</th>
<th>Lower bound</th>
<th>Upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\Theta(\log n)$</td>
<td>$\Omega_d(k\varepsilon^{-\frac{1}{d}}n^{d-1})$</td>
</tr>
<tr>
<td>$d &gt; 1$</td>
<td>$\Omega_d(k\varepsilon^{-\frac{1}{d}}n^{d-1})$</td>
<td>$\Omega_d(k\varepsilon^{-\frac{1}{d}}n^{d-1})$</td>
</tr>
</tbody>
</table>
Adaptive Lower bounds

What about the adaptive case for $d > 1$?

**Theorem [B., 2019+]:**

There exists a 2-local property of $[n]^d$-arrays, whose adaptive two-sided query complexity is $n^{\Omega(1)}$.

**Open question:** close the gaps – no known lower bounds depending on $d$. 

Adaptive Lower bounds

There exists a 2-local property of "#"-arrays, whose adaptive two-sided query complexity is $\omega(n)$. 

**Theorem [B., 2019+]:**

Open question: close the gaps – no known lower bounds depending on $d$. 

- What about the adaptive case for $\Theta^*(\cdot)$?
Questions

1. Exponential running time is undesirable.
   [S. Raskhodnikova, C. Seshadhri:] For which subclasses of local properties can we also get sublinear running time?
   [Chakrabarty, Seshadhri ‘12]: “bounded derivative” properties.

2. On which graph does “locality $\Rightarrow$ sublinear testability” hold?
   Bounded-degree graphs? Hyperfinite graphs?

3. How powerful is adaptivity?

Thank you!